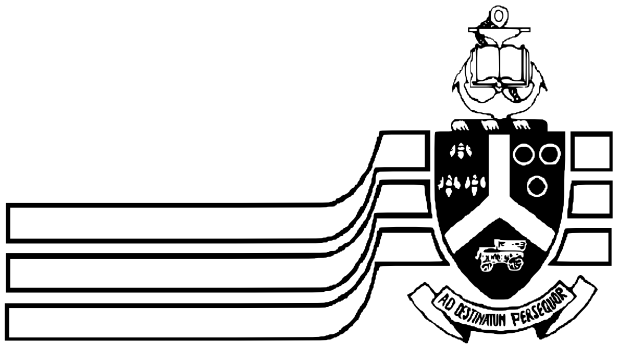


UNIVERSITY OF PRETORIA
DEPARTMENT OF COMPUTER SCIENCE

SURFACE DEFECT DETECTION BY MEANS OF A STRUCTURAL LIGHT SYSTEM

Hans Grobler
May 2006

Submitted in partial fulfilment of the requirements for the degree
Masters of Science (Computer Science)
in the
Faculty of Engineering, the Built Environment and Information Technology
UNIVERSITY OF PRETORIA

SUPERVISOR: PROF. A.P. ENGELBRECHT

Abstract

During the production of a motor vehicle, damage may be sustained by the components of the vehicle at various stages of the production process. As the vehicle being assembled progresses through the production process, the cost of correcting such damage increases significantly. This document describes a Surface Defect Detection System (SDDS) that was developed to detect damage to the outer panels of a vehicle, specifically the roof panel.

The system makes use of a structural light configuration consisting of a laser and camera, together with appropriate image processing tools. The structural light system operates and is mounted on an industrial robot, that forms part of an existing motor vehicle production plant. Although the developed system is targeted specifically towards the detection of bumps (high spots) and dents (low spots) on the surface of the roof of the BMW 3-series, the implementation is sufficiently generic as to allow other panels to be scanned. The image processing software is implemented as the Surface Defect Detection System Framework (SDDSF).

The hardware infrastructure of the production prototype developed in this project is shown in Fig. **[1]** and Fig. **[2]**. The KUKA Industrial Robot and KUKA Robot Controller were provided by BMW South Africa (Pty) Ltd. The SQ/2 Ink Jet Printing System, on loan from Goldpack (Pty) Ltd, is used to mark anomalies detected on the panel surface. The roof panel to be scanned is supported by a table manufactured for this project. Similarly, the structural light system is mounted on a "gripper" attached to the robot. In the production plant environment the table is part of the production process and supports the roof panels whilst applied sealant partially dries. The production "gripper" contains 6 suckers and is used for the placement the roof panels on the motor vehicles.

Keywords: Defect detection, image processing, high speed camera, visible-light laser, industrial robot, real-time control, ink-jet printing system, robust estimation, Kalman filtering, neural network, classification.

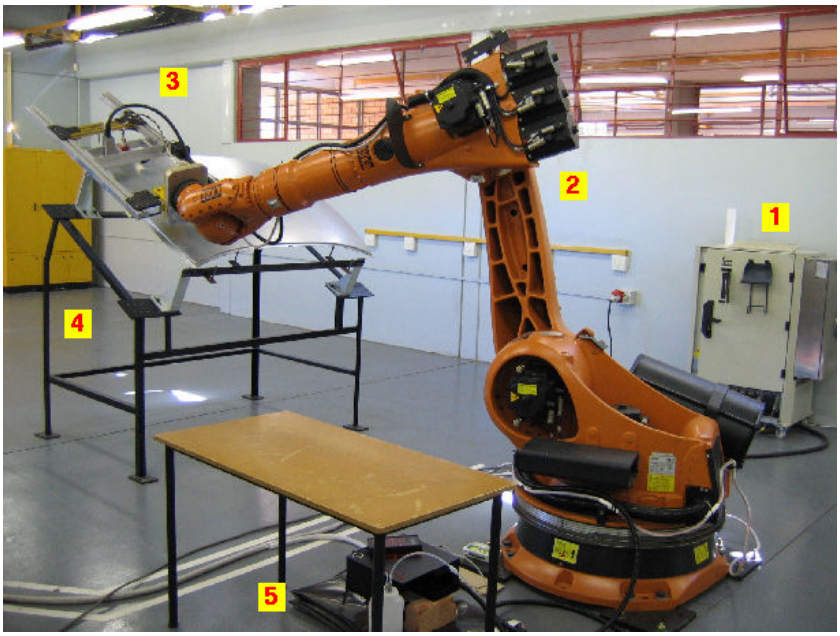


Figure 1: The Project Outcome (view 1).

Figure 2: The Project Outcome (view 2).

LEGEND

① KUKA Robot Controller

② KUKA Industrial Robot

③ Developed gripper attached to robot ($1.5\,m \times 0.5\,m$)

④ Roof panel support table

⑤ Ink-jet printing system controller

⑥ Roof panel mounted on support table

⑦ Linux server running the Surface Defect Detection System

Contents

List of Figures

List of Tables

List of Algorithms

List of Examples

Acknowledgements

Being part of the Surface Defect Detection System project has been a great opportunity. In particular, I would like to thank the following people that have supported the project and my studies in various ways:

- Professor Andries Engelbrecht, Department of Computer Science at the University of Pretoria, for establishing the project and allowing me the opportunity to work on the project.
- My family for their understanding and support of my work interests.
- Gideon Malan and Paul Wijtenburg, representatives from BMW South Africa (Pty) Ltd, for their keen interest in and support of the project, without which the project would not exist.
- Chippie Groenewald and Eugene Du Preez, the local KUKA agents Jendamark Automation (Pty) Ltd, for their help with extensions to the robot.
- Richard Sims and Clyde Challenor, Goldpack (Pty) Ltd, for providing and supporting the ink-jet printing system for experimental integration.
- Peter Hafner, KUKA Subsidiary Support (Germany), for his help with obtaining and integrating the Ethernet-RSI module.
- Kuang-Ting Liu, KUKA Senior Software Engineer (USA), for his help with development using the Ethernet-RSI module.
- Jan Brand and Nick Plomp, University of Pretoria, for designing and implementing the refined mounting system and various support activities.
- Robert Geldenhuys, Metrologist Analyst, BMW South Africa (Pty) Ltd, for providing the CAD models and associated documentation.
- Professor Roelf Sandenbergh, Dean: Faculty of Engineering, Build Environment and Information Technology at the University of Pretoria, for facilitating the acquisition of a suitable development location and associated infrastructure.
- Wessel Wessels, Gerhard le Roux and Cor Schutte, final year Industrial Engineering students, for various infrastructure support activities.
- Professor Johan Strasheim, Chair: Automotive Manufacturing, Department of Industrial and Systems Engineering, University of Pretoria, for initial management of the current stage of the project.
- Izan Crause, Operations Manager for Business Enterprises (BE) at University of Pretoria (Pty) Ltd, for handling the business component of the project.
- Doctor Frans van den Bergh and Evangelos Papacostantis, for the development of the initial prototype upon which the second stage of development is based.

CHAPTER 1

Introduction

This report describes an ongoing commercial project executed in conjunction with BMW (South Africa) (Pty) Ltd, hereafter referred to simply as BMW. This project was initiated in 2002 by representatives from BMW who posed the following question to members of the Department of Computer Science, University of Pretoria:

> "Can an automated system be developed to detect anomalies on the outer-surface panels of motor vehicles before they are spray painted?"

At the stage this question was posed, human intervention was required to detect surface anomalies. This entailed experienced personnel performing a visual and tactile examination of randomly selected panels that had been attached to the vehicle body, but before spray painting. Due to increased demands imposed by the production of a new series of motor vehicles, the desire was to eliminate this manual intervention.

The rest of this chapter will describe the problem in more detail and present an overview of an initial prototype (proof of concept). Based on the success of the initial prototype, a second development stage was entered with the aim to create a production ready implementation of the system. The rest of this document will describe this second development stage and the outcome: the Surface Defect Detection System (SDDS).

Note that the entire system, including hardware infrastructure, will be referred to as the Surface Defect Detection System (SDDS). The software implemented will in turn be referred to as the Surface Defect Detection System Framework (SDDSF). Further note that the various acronyms and symbols found in this document are defined in Appendix A and Appendix B respectively. A glossary of terms is also included in the appendices. To provide the reader with a sense of the progression and evolution of the project, Appendix H provides a summary of the project time-line and identifies milestones achieved.

1.1 Chapter Outline

The first section of this chapter will analyse the problem in more detail. The definition and nature of the various anomalies to be detected will be provided. The requirements definition of the initial prototype will be given and the possible approaches to meeting these requirements will be discussed. The implementation of the initial prototype will be described briefly. The problems encountered and results obtain from the implementation will be discussed. The chapter will conclude by supplying the requirements definition of the production prototype and discussing the research methodology applied during the development of the production prototype.

1.2 Problem Analysis

The outer-surface panels that are used during the production of a BMW motor vehicle are produced according to specification by a third party. These panels are transported by truck from the third party to the BMW production plant in *pallets* consisting of a number panels held in a support frame. The number of panels depends on the panel type, with 10 to 12 per pallet being typical. At the BMW production plant, the pallet is loaded by fork lift into a two slot vertical storage frame. This storage frame allows one pallet to be accessible to the production line whilst the next pallet is loaded. The assembly of the motor vehicle occurs almost exclusively in an automated fashion by industrial robots.

The kind of damage or defects found on the panels may be classified into two broad groups: *natural* and *artificial*. For the purposes of this project, natural defects refer to defects that result during the production of the panel and hence exist before the panel leaves the third party production unit. Natural defects are often simply referred to as defects, the meaning being clear from the context. Correspondingly, an artificial defect refers to a defect that is induced subsequent to the panel leaving the third party production unit and is also referred to as *damage*.

1.2.1 Natural Defects

Defects from these two groups are significantly different in characteristics. The defects in the first group are caused by abnormalities in the panel production process. In simple terms, a panel is produced by pressing the panel from a sheet of metal on a mold. This technique relies on the plastic deformation ability of most metals. If the mold has been damaged, or foreign objects such as dust particles are present on the mold, the resultant panel contains defects.

The presence of dust particles on the mold is the leading cause of natural defects. The resultant defects are particularly small and are difficult to see and sense by touch. However, once the panel has been spray painted, they may become visually detectable. Fortunately, the majority of these defects are sufficiently small that they are not visible once the panel has been spray painted. The defects which are larger, but still relatively small, are typically reworked by removing the resultant indentation metal by *micro-sanding*. Micro-sanding entails the removal of a few microns of surface material from the panel surface using an abrasive. The thinning of the metal that results does not significantly affect the strength of the panel.

In the cases where the natural defect is relatively large, the indentation metal is flattened by light impact (*i.e.* panel-beating) and subsequently smoothed by micro-sanding. Unfortunately, there exists no specific standard which defines what size of defect is considered small versus large. This determination is performed by a human expert who bases the decision on experience of the final nature of a particular defect once the panel has been spray painted. The effectiveness of reworking particular defects also factors into the domain knowledge of the human expert.

Another form of natural defect that occurs is metal thinning and / or tearing . If the molecular mixture of the sheet metal alloy is not correct and uniform, the sheet metal may thin and / or tear during the pressing operation. These defects may also be caused by incorrect placement of the sheet metal. Reworking these types of defects is generally not feasible and the panel must be rejected. Fortunately, this kind of natural defect occurs infrequently.

1.2.2 Artificial Defects

From the moment the panel is removed from the mold press, to the point where the panel is attached to the motor vehicle, the panel may be damaged in various ways. During the stages where robots are involved, damage typically occurs due to misalignment of the robot and panel. This misalignment is most often caused by robot calibration problems.

During the stages where the panels are transported by truck, extreme vibration or impact may cause panels in the pallet to collied with one another. Finally, when the pallet is loaded / unloaded by fork lift, operator error may cause the panels to become damaged. Reworking these defects entails flattening the defect by impact and subsequent micro-sanding. Many of these forms of damage are however too large to be effectively reworked and the damaged panel must be rejected. Rejected panels are generally returned to the supplier, where they are recycled.

1.2.3 Human Quality Control

Clearly large defects, which are generally artificial defects, are easily detectable by visual inspection. Smaller defects, such as natural defects, are more difficult to detect by cursory visual inspection. Trained experts, who also generally have natural sensitivity / talent, are able to detect smaller defects by tactile inspection.

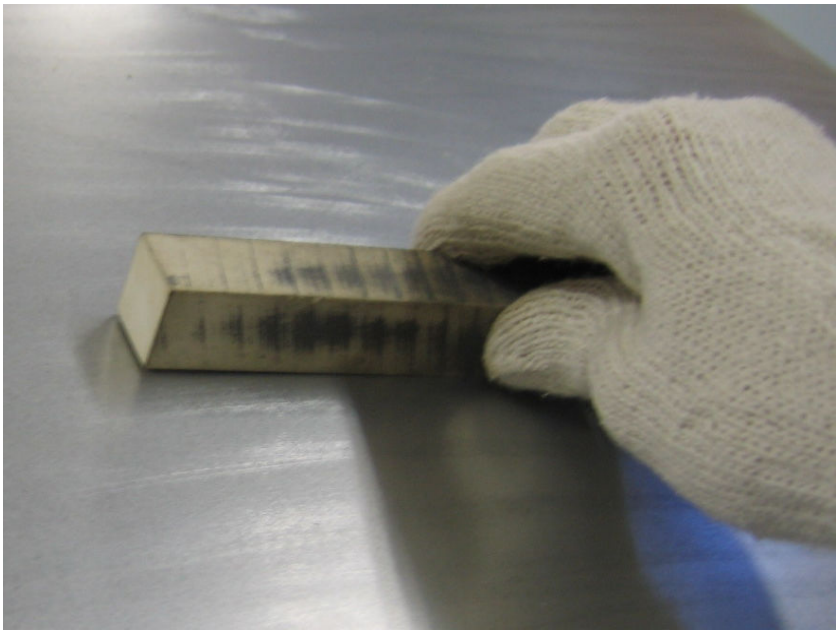
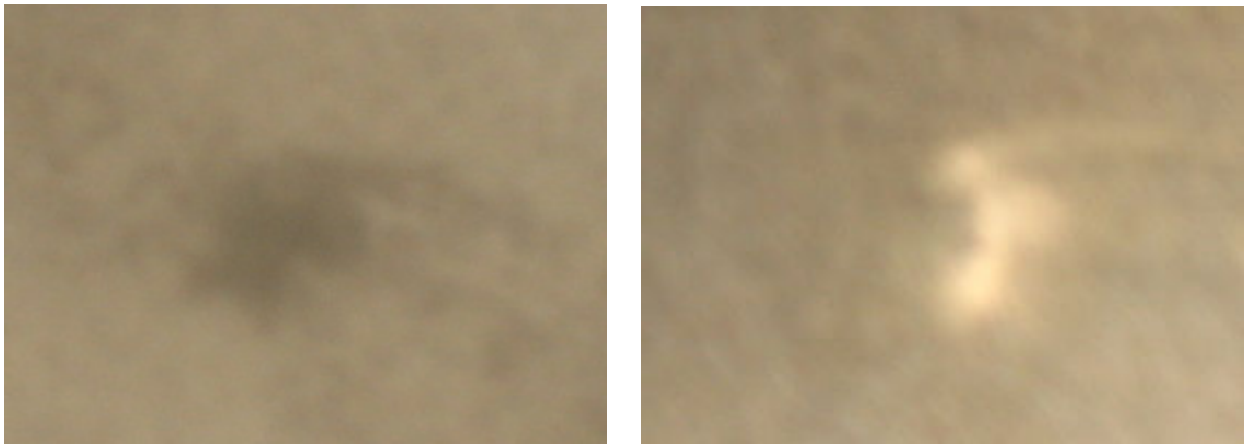


Figure 1.1: Manual defect detection using a stone.

The smallest defects, which may result in visible anomalies after spray painting, are often only detectable by means of a *stone*. Fig. **[1.1]** shows a stone being used to find defects on a panel. A "stone" is a precision engineered elongated bar made from a stone like material. The particular material is chosen to provide a fine abrasive surface. Typically a single polishing stroke of the stone will remove less than $50\,\mu m$ of material from the metal surface of the panel. When a stone is carefully moved across the surface of a panel, the effect is that the high point of any defect is polished to a higher degree than the surrounding material. Whereas the rest of the panel has a dull (diffuse) appearance,

the polished high point (spot) will be significantly more reflective and hence clearly visible. Fig. **[1.2]** shows the effect of polishing a panel with a stone. As can be seen, the defective area becomes more reflective, and depending on the angle of the incident light and positioning of the viewer, the defective area is more visible.



(a) Non-aligned light source (b) Aligned light source

Figure 1.2: Result of polishing with a stone.

Both tactile inspection and polishing with a stone are clearly time consuming and requires that the expert be in close proximity to the panel. In an automated assembly environment, this is not possible due to the presence of the robots, the arm of which may attain lethal speeds in excess of $2\,m.s^{-1}$. For human quality control, the automated process must therefore be interrupted, which leads to a loss in productivity.

1.3 The Prototype System

The prototype system was developed subsequent to the initial enquiry. The primary purpose of the prototype system was to develop an initial proof of concept, which would later be developed into a production system.

1.3.1 Requirements Definition: Prototype System

The main objective of the initial project was to develop a prototype system for the automated detection of defects on the roofs of the BMW 3-series. Specific requirements that needed to be considered were the following:

- a system was needed to scan the outer surface of roofs in order to detect defects such as bumps and dents,
- the system was required to be fast and robust, and
- the eventual implementation in a production line.

This objective was achieved in a number of sub-objectives:

- To research and to propose a model for surface anomaly detection.
- To show proof of concept, *i.e.* that the proposed model worked and had potential to be developed further as final solution.
- To propose the necessary infrastructure for the prototype system to be developed.
- To develop and test an automated anomaly detection system.

These objectives focused only on the roof panel. However, it was constantly kept in mind that it should be possible to use the same model for other vehicle panels.

1.3.2 Possible Approaches

When the original enquiry was made as to the feasibility of an automated anomaly detection system, no commercial solution was available. Initial research did not identify any similar investigations into outer surface anomaly detection on motor vehicles. However, general anomaly detection on surfaces had been investigated, *e.g.*, in the production of agricultural products [1]. In most cases, a texture based approach was used and generally the result of any anomaly detection was a binary accept / reject outcome.

In this project, the defects may not cause significant changes in the apparent (visual) texture of the panel surface. Specifically, the metal surface has a dull (diffuse) appearance, which is retained in the presence of a defect (except in the case of a tear in the metal). Clearly the primary effect of surface anomalies in this project are localised changes in the relative height of the panel surface. A detection method that is sensitive to such height changes was therefore required.

One possible approach would be to use a stereoscopic visual system. Unfortunately, multi-camera vision indirectly relies on textural information in order to select match points, which are required in order for the multi-image fusion of features [42, 89]. In the absence of significant textural information, the primary fusion of features (also known as the correspondence problem) approach makes use of correlation, which is computationally expensive and does not perform well in near uniform images.

For this project a different technique was therefore required. The implemented anomaly detection system uses as its foundation a lighting model, as described in Chapter 6. The premise is that if a straight line of light is projected onto a surface with defects, irregularities will be observed in the perceived line [36]. The objective of the prototype was therefore to develop a system to project a ray of light onto a panel surface, to capture the image and to analyse it in order to detect any irregularities in the observed line. If this approach proved successful, further refinement would be performed to produce a production ready implementation.

The prototype was researched, implemented and evaluated. The result was a proof of concept surface anomaly detection system that could successfully detect anomalies on motor vehicle roof panels (specifically for the BMW 3-series). The resulting system consists of several components, an overview of which is given in the following section (from [36]).

1.3.3 Implementation

The prototype system made use of an ABB IRB 6400 production line robot as illustrated in Fig. **[1.3]**. The task of the robot was to move the gripper system (which houses the camera and laser) over the surface of the panel such that the camera remains perpendicular to the surface of the roof and at a

Figure 1.3: ABB IRB 6400 robot [36].

constant height, and to make sure that the laser forms an approximate 8° angle with the roof surface. This is to ensure that

- the projected line remains within the field-of-view of the camera,
- the camera remains in focus, and
- the laser line is projected at a minimum angle.

The proof of concept system used a relatively simple defect detection system. Images were captured from the camera in grey-scale format. Subsequently, the images were filtered to remove noise and the image converted into a black and white image. In this instance, white pixels represent the projected laser line. Image analysis techniques were then used to detect the line in the image. A piece-wise linear approximation was obtained for the laser line by segmenting the observed line into 4 segments, and fitting a straight line through the midpoints of each segment. White pixels that deviated significantly from the fitted lines were seen as indicators of defects on the roof surface.

The gripper is the structure that houses, amongst other components, the camera and laser. Fig. **[1.4]** illustrates the gripper. The actual gripper system is shown in Fig. **[1.5]**. The gripper components are

- Shock sensor (component a): This sensor is placed at the mounting point of the gripper to the robot in order to protect the gripper from colliding with any foreign obstacles. In the case that the gripper does collide with foreign obstacles, the scanning process is interrupted immediately.
- Camera (component b): The camera is mounted perpendicular to the gripper and is perpendicular to the surface at all times during the scanning process. The images captured by this camera are analysed by the server in order to detect any anomalies on the surface.
- Defect marker (component c): The marker is used to physically mark a detected defect on the surface of the roof. The marker is mounted perpendicular to the gripper, and is spring loaded to prevent the marker to cause more defects.

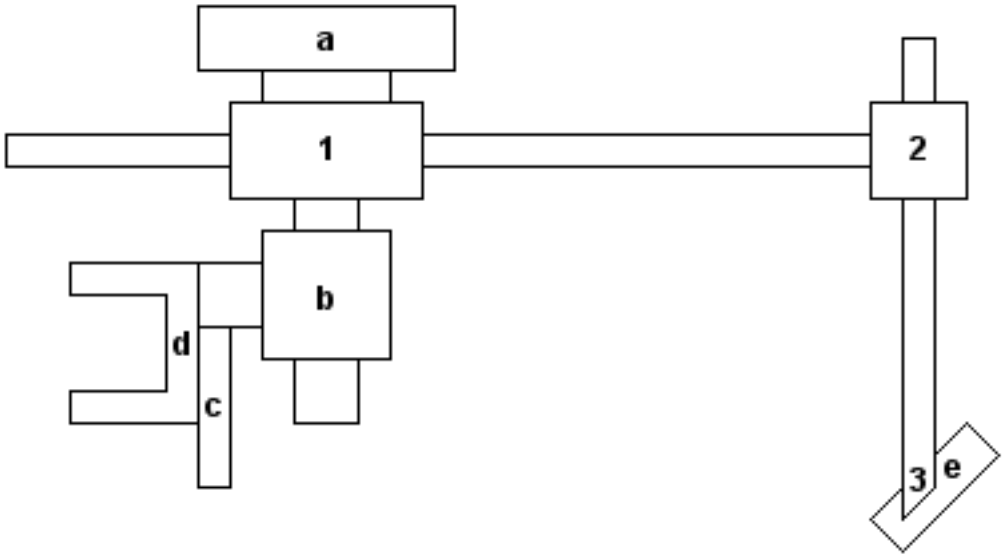


Figure 1.4: The Gripper System Structure [36].

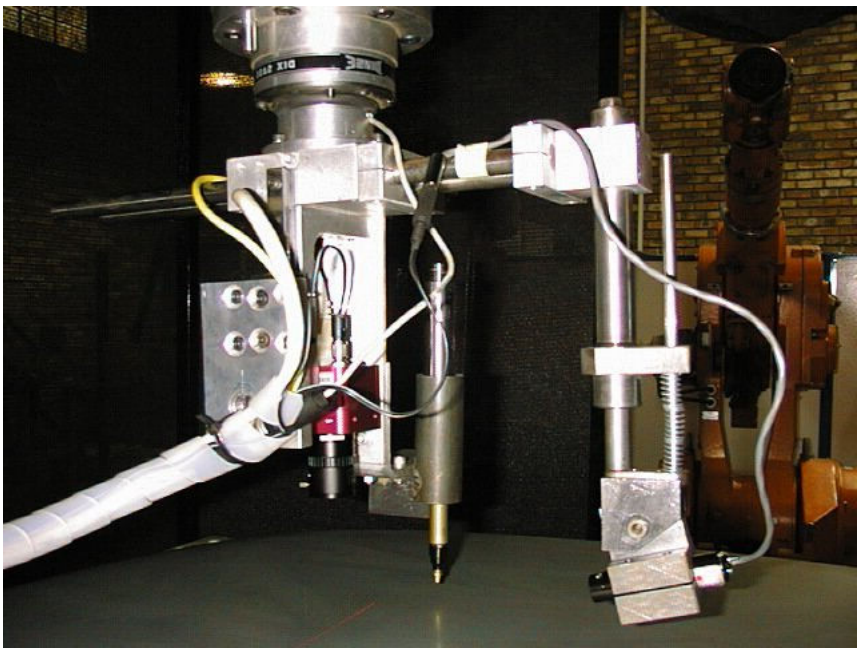


Figure 1.5: The Gripper System [36].

- Proximity sensors (component d): These sensors enable the detection of the roof edges in the X and Y directions. This is done by moving the sensors close to the roof and when triggered, the position of the sensors define the edge coordinates.
- Laser (component e): The laser projects a light ray on the surface of the roof. Disturbances in the shape of this laser line determine if there is an anomaly on the surface. The laser is mounted to form an 8° angle with the surface of the roof.

1.3.4 Problems Encountered

The following is a brief list of technical problems that were encountered during the development of the prototype. A number of these problems are inherent to the project and will be discussed in further detail in later chapters.

- The images captured by the camera contained a number of anomalies induced by the camera itself. In the prototype a simple filtering approach was used to remove these anomalies.
- Problems were experienced with the power supply to the camera. The result was that the camera produced images that were incomplete and unstable. The source of the problems was eventually traced to the $220\,V$ AC supply and solved by means of an Uninterruptable Power Supply (UPS).
- The motion of the robot was controlled by a program that initiated movements based on the CAD model of the roof panel. However, the CAD model and the actual roof panel did not correspond and deformation of the CAD model was required.
- The gripper manufactured for the project initially contained two problems. Initially, aluminium was used for various parts of the gripper which, even though it contributed to the gripper being very light, caused numerous problems in adjusting the gripper accurately as the relative soft aluminium warped when adjustments were made. The second problem was the manner in which the laser holder functioned. When one attempted to tighten the holder of the laser to keep it firm in place, this caused the laser to shift out of position.
- The camera used for the project required a PCI bus speed of $66\,MHz$. All desktop PCs have motherboards that run at PCI bus speeds of $33\,MHz$. Supporting motherboards that run at such high speeds can only be found on workstation or server motherboards and a workstation was therefore obtained.
- For the development of the prototype system, the robot was not bolted to the floor, and it was therefore not possible to safely experiment with higher scanning speeds in order to determine the shortest time to scan a roof.

1.3.5 Initial Prototype Results

The developed prototype system was tested on a number of defects. During demonstrations of the system, attendees from BMW were invited to add their own defects on the roofs, which included dents, bumps and scratches. These defects were detected successfully. The system additionally exhibited the ability to detect dust. However, by changing the detection threshold in the image analysis software, the sensitivity of the system could be adjusted to ignore small anomalies such as dust particles.

The prototype system however did not examine the following aspects introduced by conditions that are present in the production line:

- The performance of the system was sensitive to light. Tests were needed to determine at what point of environmental light intensity the system would fail to detect defects.
- Tests needed to be conducted to investigate the effects of oily residues on the panels (a side effect of the production process).
- The system needed to be tested on all kinds of defects that could occur, and the system would need to be extended in order to identify workable and non-workable defects.
- The speed of a complete scan was unknown and was to be quantified.

1.4 Requirements Definition: Production System

The second system developed, that represents the system to be deployed in the production line at BMW Rosslyn, had similar requirements to that of the prototype system. However, certain additional and / or more specific requirements were introduced:

- the system must scan the outer surface of various types of roofs in order to detect defects such as bumps and dents on the roof and any flanges,
- identified defects must be categorised as either workable or non-workable,
- if less than a predefined threshold of workable defects have been found, the roof is "accepted" and the defects are marked via a small ink dot,
- if more than the threshold of workable defects have been found, the roof is "rejected",
- the entire scanning, processing, deliberation and marking process must be completed within $30\,s$,
- if there is any doubt about the workability of a particular roof, the roof must be "rejected",
- the resultant status of "accepted" or "rejected" must be communicated with the production line Programmable Logic Controller (PLC),
- the process must function correctly in the environment as found in the production line at BMW Rosslyn, Zone 13. Specifically, the system must be robust against ambient levels of light, wind, dust, temperature and vibration as exist in the target environment.

The remainder of this document will discuss the second stage of the project, during which the requirements listed above were addressed.

1.5 Methodology

The context within which this research was performed determined the research methodology followed. As the research forms part of a commercial project with specific desired outcomes, the focus was therefore the outcomes. The commercial nature of the project limited the time available to perform the research, development and implementation. Although aspects of the project were researched in some detail, the outcome driven nature of the project essentially prohibited an exhaustive investigation of all aspects of the project.

The applied methodology could be best described as a pragmatic process of iterative refinement. The first implementation was conceptually based on the initial prototype. As specific problems or deficiencies were identified, research was performed in order to identify solutions to these problems or deficiencies. Due to the stringent performance requirements, it was invariably found that the more complex, theoretically correct, solutions were not feasible and less complex approximate solutions were eventually utilised.

1.6 Chapter Summary

This chapter has described the types of anomalies which the developed system aims to detect. The two types of anomalies targeted for identification were defined as natural defects (defects) and artificial defects (damage). However, it was highlighted that there exists no formal definition or specification that defines what constitutes a defect or damage. The lack of such specification presents significant challenges to the operational adaptability of the detection system.

The detection method utilised in the initial prototype was described and, based on the results of the initial prototype, the detection method based on a structural light system was shown to be a viable method to perform surface defect detection. Due to the commercial nature and requirements identified for the production prototype, a specific research and development methodology was necessitated for this project. This methodology, an iterative refinement based approach, impacts the nature and depth of the additional research performed during the developed of the production prototype. This methodology implies that, unless deficiencies or problems were identified, research and development was only performed until a suitable implementation was obtained.

1.7 Report Overview

Chapter 2 will provide an overview of the principles which underlie the system operation. Chapter 3 will discuss the hardware infrastructure of the second development stage. This infrastructure, although conceptually similar to the first stage hardware, differs in that a new location for the development was selected. Additionally, a different industrial robot and panel support table was used. Chapter 4 discusses miscellaneous aspects related to the developed software such as the error handling architecture. Chapter 5 discusses the control of the KUKA robot in order for the scanning of the panels to occur. The software implementing the defect detection system was redeveloped, with none of the original experimental code being incorporated. Chapter 6, Chapter 7 and Chapter 8 will discuss the theoretical and practical aspects of the image processing software. Various empirical results will also be discussed. Lastly, Chapter 9 will conclude by discussing the operational state of the system and highlight future activities to refine certain aspects of the system.

CHAPTER 2

Theoretical System Model

Before the practical implementation of the production prototype can be discussed, the theoretical system model on which it is based must be understood. The implemented anomaly detection system uses as its foundation a lighting model based on principles from the field of computer graphics and classical physics. The premise is that if a straight line of light is projected onto a surface with defects, irregularities will be observed in the perceived line [36]. This chapter will investigate the theoretical system model. The subsequent chapter will describe the implemented infrastructure in more detail.

2.1 Chapter Outline

The structural light system used to detect surface anomalies requires knowledge of the certain concepts from the field of optics. The first part of this chapter is devoted to a summary of the relevant concepts, specifically that of reflection. The particular type of reflection that is central to the structural light system, namely diffuse reflection, is subsequently discussed in more detail. A high-level description of the implementation of the structural light system is provided. The chapter concludes by providing a theoretical estimate of the accuracy and the scanning (detect) time of the conceptual system.

2.2 Modelling Reflection

The drive to achieve ever more realistic computer generated images in modern computer games, as well as modern computer generated animated movies, has promoted the development of various models of reflection and illumination. This section will give a general overview of the illumination models [41, 144]. Since the reflection of the laser light from the panel is of particular importance for the SDDS application, the aim of this overview will be to establish known models for reflection.

The general model of light reflecting from a surface is described by a *bi-directional reflectance function* (BDRF) [41, 148, 147]. This distribution describes the relative intensity of light reflected in all directions from a point on a surface. Early models of reflection made used of highly simplified BDRFs. General models of reflection were systematically developed from the basic perfect reflector model. The perfect reflector model is based on the Fresnel formulae, which is a coefficient which relates the ratio of reflected and transmitted energy as a function of the incident direction, polarisation and properties of the material.

The Fresnel formulae were in turn derived from Maxwell's wave equations [56], which describe the temporal evolution of electric and magnetic fields. Reflection from a perfect mirror surface is known as *specular reflection*, where a thin beam of light on such a surface is reflected in such a way that

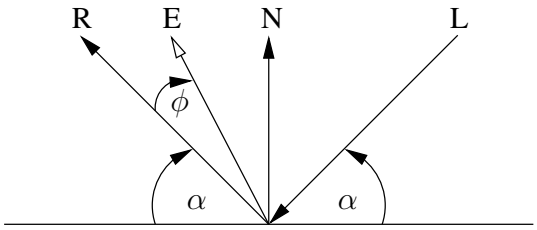


Figure 2.1: Specular reflection [36].

the outgoing angle is equal to the angle of incidence. Fig. **[2.1]** illustrates a specular reflection. The vector $\mathbf{E}$ represents the direction in which the observer lies, while $\mathbf{R}$ is the reflection of the vector $\mathbf{L}$ around the surface normal $\mathbf{N}$.

The relative intensity of the specular reflection, I_s, can be calculated using the Fresnel formulae, or the approximation equation,

$$I_s = \cos(\phi)^n = \left(\frac{\mathbf{E} \cdot \mathbf{R}}{\|\mathbf{E}\| \|\mathbf{R}\|} \right)^n$$

where n depends on the "shininess" of the surface. The maximum intensity is therefore reached when the observer views the reflected light at exactly the angle of reflection from the surface. This effect can be seen in Fig. **[1.2]**, where (b) represents light specularly reflected from the polished high spot viewed at the reflection angle.

Real surfaces are rarely perfect reflectors and the incident light is therefore not reflected in a single direction. At the extreme, the presence of imperfections results in what is known as perfectly *diffuse reflection*, where the incident light is scattered uniformly in all directions. The characteristic property of the diffuse reflection is that light is scattered around the surface normal so that the observed intensity of the light is independent of the position of the viewer. The intensity only depends on the angle between the incident light and the surface normal. This application explicitly relies on diffuse reflection since the camera is located coincident with the surface normal, whilst the angle of incident α is greater than 75°. The reason for this alignment will be discussed in the next section.

The earliest illumination studies can be traced back to the work of Pierre Bouguer (1698–1758). Between 1721 and 1729 he explored astronomical photometry and developed what is now known as Bouguer's Law. This law describes the relationship between the absorption of radiant energy and the absorbing medium. Based on the work of Bouguer, Johann Heinrich Lambert (1728–1777) subsequently developed the more well known Lambert's Cosine Law. This law states that the brightness of a diffusely radiating plane surface is proportional to the cosine of the angle formed by the line of sight and the normal to the surface. In computer graphics, the diffuse illumination equation is taken as

$$I = I_p \, k_d \, \cos\theta = I_p \, k_d \, \mathbf{N} \cdot \mathbf{L} \tag{2.2.1}$$

where I_p is the point light source's intensity; the material's *diffuse-reflection coefficient*, k_d, is a material dependant constant between 0 and 1. The angle, θ, is the angle of the incident light relative to the surface normal. Alternatively, $\mathbf{N}$ is the surface normal vector and $\mathbf{L}$ is the incident light vector. Fig. **[2.2]** illustrates a light ray, $\mathbf{L}$, striking a surface.

One of the first practical models of reflection for use in computer graphics was developed by Bui Tuong Phong (19xx–1998) in 1973 [98, 99] based on the work of Henri Gouraud (1944–) [47] . In this

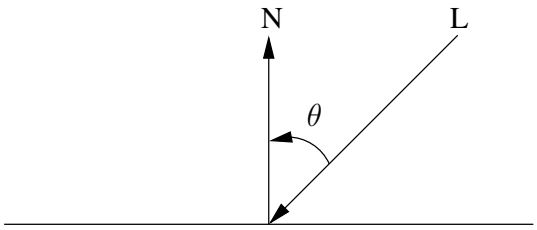


Figure 2.2: The diffuse lighting component [36].

model, the light reflected from a surface is seen as the sum of light reflected via diffuse reflection and light reflected via specular reflection. This model accounts for diffuse reflection by using Lambert's Cosine Law. The specular component is modelled using an empirical spread term.

In 1977, James Blinn [15] proposed a model of specular reflection based on physical principles. He based his model on a physical model by Kenneth E. Torrance and E. M. Sparrow (applied physicists) [137, 136]. Extending from the perfect reflector model, reflection from an imperfect surface can be modelled by incorporating the microgeometry exhibited by these surfaces. This microgeometry can be modelled as a collection of *microfacets* which are assumed to be perfect specular reflectors. The Torrance–Sparrow model included the effects of obstruction of the facets by each other.

By extending the specular model of Blinn, Robert L. Cook (1952–) and Torrance [24] developed a specular model which incorporated the spectral dependency on material type and angle of incidence of the light. Both the specular and diffuse component were calculated by integrating the contribution of surface microfacets. The orientation of the microfacets is therefore central to this model. Subsequent to the Cook and Torrance model, the He–Torrance model was developed [59] by Xiao D. He and Torrance. Incorporating aspects of most of the previous models, this model evaluates the BDRF in terms of three components, specular reflection, directional-diffuse reflection and uniform-diffuse reflection.

For many materials, diffuse reflection originates from light that enters the material. This light is absorbed and scattered within the reflecting material, where the wavelength dependant absorption determines the colour of the material. Effectively, the incident light is filtered by the material. The internal multiple scattering results in emerging light to be approximately isotropic, as well as small differences of the reflection from that expected by the Cosine Law. Pat Hanrahan and Wolfgang Kreuger [53] developed a model which incorporates the sub-surface scattering common in materials found in nature. The result of this model is that the reflected diffuse light is potentially anisotropic, exhibiting directional dependence. Algorithmically, the sub-surface scattering is calculated by solving the simplified 1D nuclear particle transportation model using a Monto Carlo (MC) approach.

For the SDDS application the primary concern is the spacial distribution of the diffuse reflection for a constant incident angle. Specifically, the BDRF for angles of reflection close to the surface normal is of interest. From the mechanical configuration and using basic trigonometry (assuming a camera height of $13\,cm$ and a lens diameter of $3\,cm$), the angles of reflection of interest can be calculated to be $\theta = \pm 7^{\circ}$. After studying the above models, the model found to be most appropriate is the Torrance–Sparrow model. Based on this model, the reflectance of roughened surfaces exhibit the distribution shown in Fig. **[2.3]**.

For the SDDS application, $\psi > 75^{\circ}$ and $|\phi| < 7^{\circ}$. From the Torrance–Sparrow model, the reflectance distribution can be considered to be a constant attenuation factor for the angles of incidence and reflection of interest (for the SDDS application). The diffuse reflection therefore is assumed to perform a constant modulation of the incident light. It should be noted that the more recent He–Torrance model

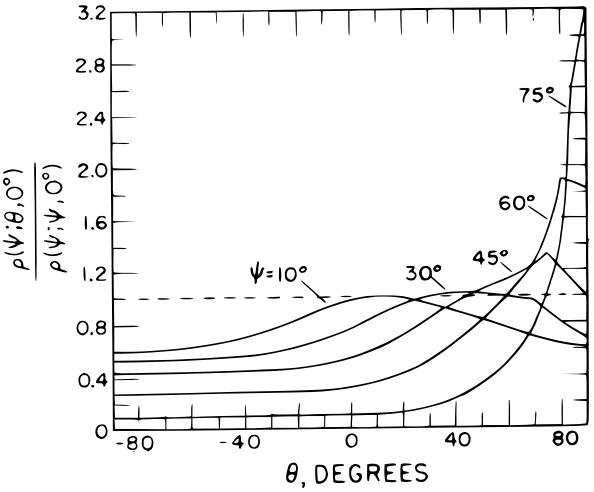
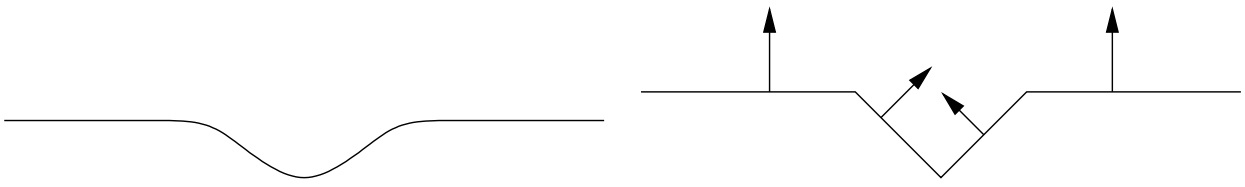


Figure 2.3: Bidirectional reflectance distributions in the plane of incidence for various angles of incidence ψ and angle of reflection θ ([136], Fig. 8.).

also predicts an approximately constant distribution for the target angle of reflection at the target angle of incidence. The true nature of the diffuse reflection must therefore correspond to the distribution of the incident light. This effect is utilised in the SDDSF and will be discussed in Section 6.7.

2.3 Visibility of Defects on Painted Surfaces

A cross-section of a typical dent in a flat surface is illustrated in Fig. **[2.4]** (a). A dent can also be represented by a simplified model, as shown in Fig. **[2.4]** (b). The arrows indicate the surface normal at various positions. The simplified model is used as it contains only three unique surface normals — the complex model contains an infinite number of continuously varying surface normals.



(a) Complex (b) Simplified

Figure 2.4: Representation of a dent [36].

Fig. **[2.5]** illustrates how specular reflection makes it possible to spot a dent in a glazed surface. The vectors with hollow arrowheads indicate the direction in which the specularly reflected light will travel. If the observer is located in the direction indicated by the vector $\mathbf{E}$, then he will notice the absence of a reflection coming from the dented region.

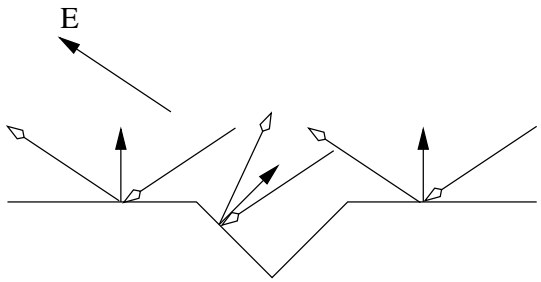


Figure 2.5: Specular reflection using the simplified representation of a dent [36].

2.4 Diffuse Translation

The goal of the defect detection system is to locate surface defects on unpainted panels in order to be repaired before painting. Since only the diffuse component of the reflection is available on an unpainted metal surface, a technique for detecting defects using only diffuse light must be considered. Fig. **[2.6]** illustrates the mechanism under consideration.

Assume three distinct incident rays, that are a distance d apart as shown in the figure, form an angle θ with the panel surface. The surface is observed from a position in the direction indicated by the vector $\mathbf{E}$. Notice that the two distances d_1 and d_2 are no longer equal to d. In the case of no defect, $d_1 = d_2 = d$. If the simplified dent model is a good approximation, then the "translation" of the rays falling on the slope of the dent will be a linear function of the position of the ray. Due to the curvature of the defective metal, a real-world defect will exhibit a non-linear, but monotonic, transformation and hence the mechanism will still be usable.

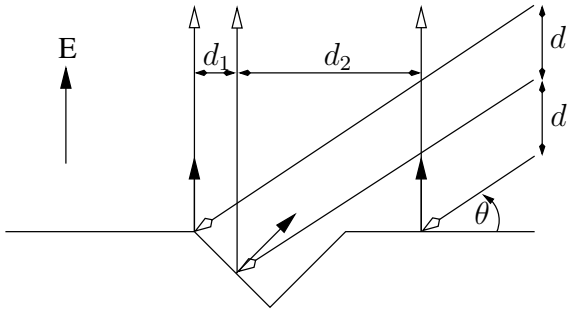


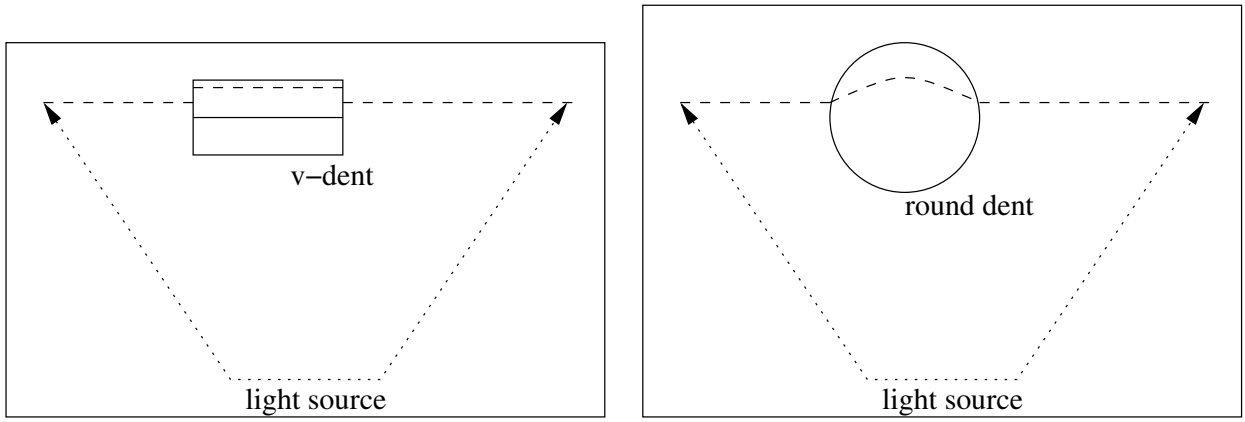
Figure 2.6: Diffuse reflection using the simplified representation of a dent [36].

Another aspect to note is that the degree of distortion is dependent on the angle θ: smaller θ values lead to greater distortion, *i.e.* the distortion is easier to detect. However, there is a trade-off associated with smaller θ-values: it is possible for the opposite edge of the dent to shadow the edge that we wish to illuminate — the physical value of d should be chosen so that self-shadowing dents can be detected.

2.5 Implemented Solution

Based on the above analysis, a potential solution therefore involves projecting a single ray from a laser light source onto the surface of a roof. A laser is selected as light source due to the monochromatic and coherent nature of the resultant light. A camera captures an image of the surface area onto which the laser is directed. If the surface area is free of any detect, the sensor pixel illuminated by the reflected ray can be calculated theoretically and represents the expected or baseline result. During defect scanning, captured images are analysed to detect any deviation from the expected pixel, which indicate defects on the surface. However, projecting a single ray along the direction of the cross-section used in the figures above requires extremely accurate positioning of the light source. Detection would therefore rely on the correlation of a number of images in order to detect defects. The reflected laser light would also need to be focused onto a single pixel in order to allow accurate detection of deviations. The equipment required to implement this approach would therefore be costly.

A more practical approach is to consider the set-up depicted in Fig. **[2.7]** (a). A v-shaped dent is used in the cross-sections. If this dent is viewed in three dimensions, it would form a v-shaped trench. Instead of a single ray or beam of light, a thin line is projected onto the surface. Viewed from above, the straight line formed by the beam will be broken where it enters the dent, *i.e.* the dent can be detected by looking for a break in the edge.



(a) Simplified dent

(b) Complex dent

Figure 2.7: Top view of the line-based solution [36].

This model can be extended to the real-world case where the dent would typically be round, such as the one depicted in Fig. **[2.7]** (b). Although there is no sharp break in the line, there will be a change in the tangent of the line as it crosses the center of the dent. By detecting these changes in the tangent, defects can be detected. The components required to implement this solution are therefore: a laser and associated optics to project a line, a high speed camera to capture images of the surface area under consideration, and a mounting system to correctly orientate the laser and camera.

2.5.1 Accuracy Estimation

The accuracy of the system is limited by three factors: the degree of dispersion of the leading edge of the light beam, the resolution of the image capture device (digital camera) used to observe the line, and the angle at which the light is projected.

The model described above assumed that all the beams were parallel. In a typical beam of light there is a degree of dispersion, resulting in slight blurring of the edge of the beam. For the proposed system the sharpness of the leading edge is more important than the actual width of the beam – only the leading edge has to be considered. However, a tightly focus beam would clearly result in a sharp leading edge. A laser beam exhibits a number of the required properties, for example, all light emitted by a laser beam is coherent and monochromatic. These two properties make it significantly simpler to obtain parallel rays.

Most lasers however produce only a thin beam of light. In order to project a line, specialised optics are required to spread the beam to form a line. Alternatively, a scanning mechanism is required that makes use of a mirror vibrating at high frequency. A simpler alternative would be to use a high-intensity white-light source, such as a halogen lamp, and filter the light to obtain a monochromatic, semi-parallel ray source.

The resolution of the image capturing device can be improved by using a zoom lens. This would clearly decrease the physical size of the area observed by the camera, which has two side effects:

1. The largest defect that can be detected must fit into the field of view of the camera, preferably with room left on either side of the defect.

2. The rate at which a large metal plate can be scanned would clearly depend on how many separate images have to be processed. Using a larger field of view would speed up the process significantly.

The magnitude of the distortion that the line could experience is limited by the diameter of the defect. Therefore, to detect a small defect would require the camera to be able to view the defect with a certain minimum resolution. For example, the camera must be able to capture an image in which the dent is visible as an area with diameter at least $10+$ pixels.

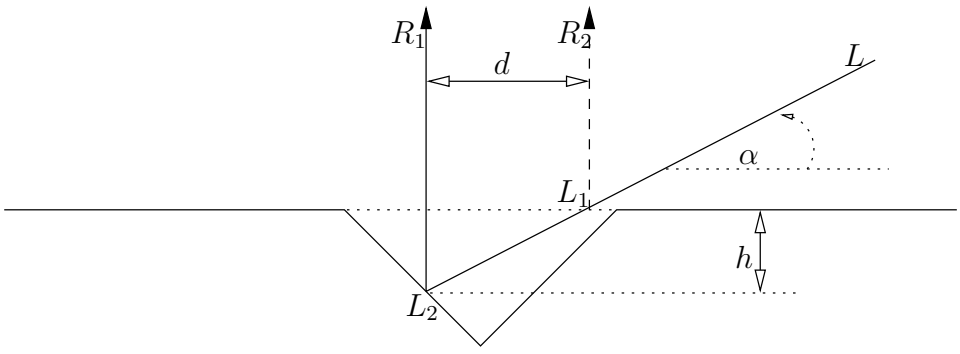


Figure 2.8: Observed displacement of ray L [36].

Fig. **[2.8]** illustrates the observed displacement of the ray **L** striking the surface at an angle α. If the surface had no dent, then the ray would have followed the path L_1R_2. Due to the dent, the ray L_2R_1 will be observed instead. The displacement of the ray, as observed from the direction of the surface normal (by the camera), is designated d in the figure. The depth of the dent, as measured from the point where the ray strikes the dented surface to the point where the surface would have been, had the dent not been there, is designated h. Then from the trigonometric relationships,

$$\tan(\alpha) = \frac{h}{d} \tag{2.5.1}$$

This implies that, given a lower limit for d, as determined by the resolution of the camera, the shallowest dent that can be detected is a function of the angle α. For the camera used in the production prototype, which has a resolution of 1024×256, the effective resolution in the direction in which deviations are detected is

$$d = 28/256 = 0.1093\, mm/pixel$$

for the active field-of-view of $115 \times 28\, mm$. Using the equation above, the minimum depth dent that can be detected is

$$h = d\tan(\alpha) = 0.023\, mm$$

for the current projection angle of $\alpha = 12°$. Given the required depth, h, that should be detected, the equation above can be used to calculate the best angle, α, at which the laser should be mounted (remember that d is determined by the camera resolution and the field-of-view).

Since most dents will be wider than they are deep, the implication is that the smallest detectable dent will be a function of the camera resolution, rather than the angle of the incident rays. If the above parameters are assumed, the smallest detectable dent (a 1-pixel deviation) will therefore have a diameter of $0.11\, mm$. However, system noise may also introduce 1-pixel deviations and hence no less that 2-pixel deviations should be considered as indicating defects. The minimum theoretical defect diameter is therefore in the region of $0.25\, mm$.

2.5.2 Scanning Speed Estimation

The roof panel has an effective sheet dimension of $1.5\, m \times 1\, m$. At the camera surface resolution specified above, this translates to approximately 120×10^6 pixels if 100% coverage is desired. Assuming perfect alignment and frame synchronisation, a total of 150×10^3 frames must therefore be captured and analysed. If one further assumes a zero image analysis time, the time required to capture the images is approximately $200\, s$, given a sample rate of $773\, fps$ (see Section 3.4.2). The estimated time required to scan a complete roof is therefore at least 3 minutes if a single laser and camera combination is used. It should be noted that the above calculations only takes into account the imaging aspects of the system. The actual scanning motion required, and the capabilities of the robot performing the scanning motion, may have a significant impact on the scanning speed.

2.6 Chapter Summary

This chapter has identified and discussed the central physical property which is used by the structural light system: diffuse reflection. The mechanism, based on diffuse reflection, which allows the detection of localised surface height changes to be detected was described. The use of a laser and camera arrangement to implement the described mechanism was motivated and a theoretical estimate of the accuracy and detection performance was provided. The theoretical accuracy of the system was conservatively estimated to allow the detection of defects with a minimum diameter of $0.25\, mm$ or a minimum depth of $0.023\, mm$. The minimum time required to scan a motor vehicle roof panel was estimated to be at least 3 minutes when a single structural light system is used.

The following chapter will describe in detail the infrastructure that forms the hardware component of the production prototype. The subsequent chapter will provide an overview of the software component of the infrastructure.

CHAPTER 3

Infrastructure

As hinted in the previous chapter, the primary components of the surface defect detection system is a laser and camera. The laser is positioned in such a way as to project a laser line onto the panel surface. The camera is mounted directly above the point at which the laser line is projected onto the panel surface, thus capturing the image of the panel surface where the laser line is visible on the panel surface.

Other components that form part of the infrastructure are: the mounting system which carries the laser and camera, the marking system which will visually mark defects for reworking, the robot which performs the scanning motion, the table on which the panel rests, and the computer which performs the image processing. These components, a number of which are shown in Fig. **[3.1]** and Fig. **[3.2]**, will be discussed in this chapter.

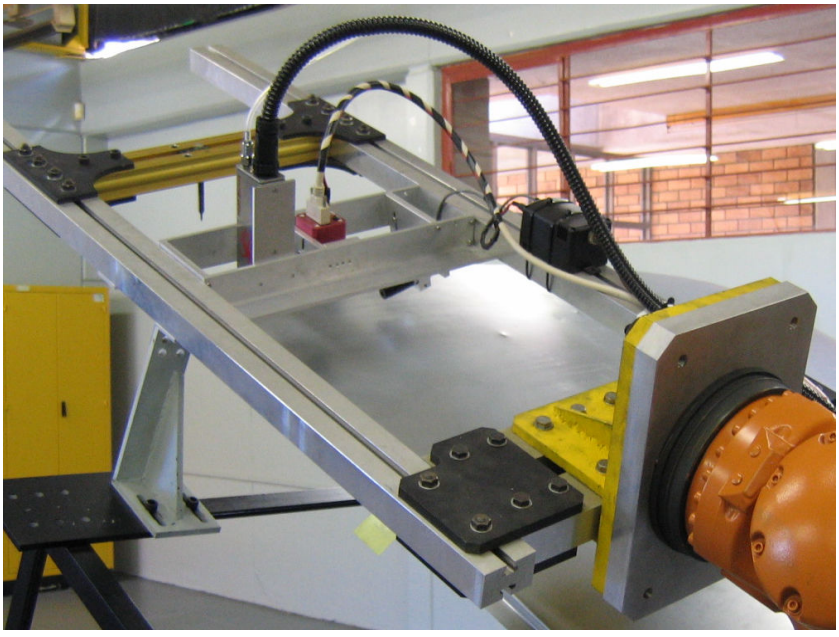


Figure 3.1: Top view of gripper.

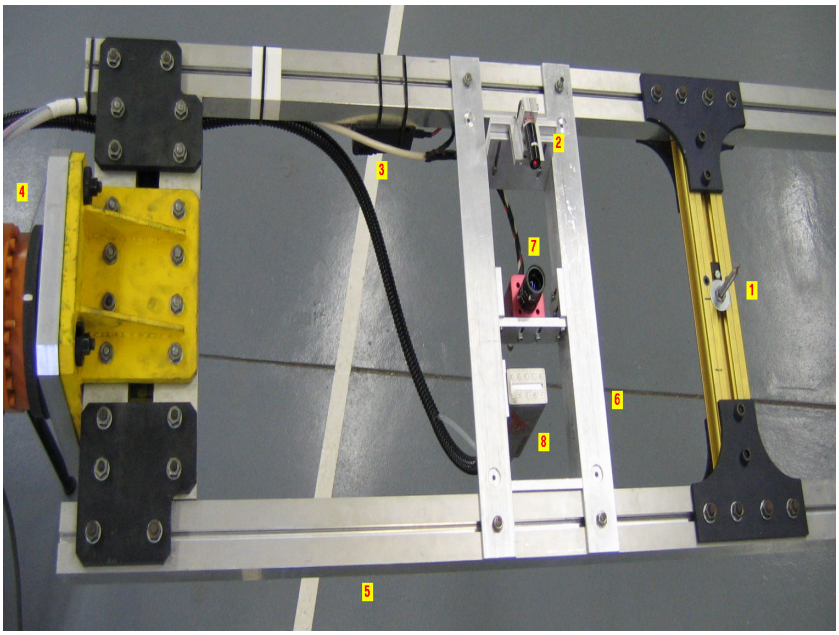


Figure 3.2: Bottom view of gripper.

LEGEND

① Calibration Point

② Laser

③ Camera and laser power supply

④ Robot flange

⑤ Gripper (pre-production version)

⑥ Camera, laser and marker mounting system

⑦ Camera

⑧ SQ/2 ink-jet printing head

3.1 Chapter Outline

The first components discussed, the laser and camera, make up the structural light system that is used for the anomaly detection. The mounting system for the laser and camera is described. The remainder of the components, the industrial robot and defect marking system are subsequently discussed.

Table 3.1: Selected technical data for Lasiris SNF Laser [130].

Diode Power	$10\,mW$
Wavelength	$670\,nm$
Fan Angle	$45°$
Operating Temperature	$-10\ °C$ to $48\ °C$ with bracket
Wavelength Drift	$0.25\,nm/\ °C$ typical
Input Voltage	$5-6\,V$ DC
Connector Type	Male phono-jack $3.5\,mm\,\varnothing$

3.2 Location

The initial prototype was developed largely at the SAR Electronic facilities, located in Montana Park, Pretoria. The production prototype, making use of the new robot, was developed in facilities at the Groenkloof Campus, University of Pretoria.

3.3 Laser

The laser used during both the first and second stages of this project is the StockerYale Lasiris ™ SNF Laser[1], model number SNF-501L-670-10-45 (serial number 030303018). The primary features of this class of laser are: red visible light source, uniform intensity distribution length-wise and Gaussian distribution width-wise, over-temperature protection, electro-static discharge (ESD) protection, reverse-voltage and over-voltage protection. An overview of the technical data for the particular laser used in this project is given in Table **{3.1}**. The particular laser was selected due to the availability of the line projection optics at a reasonable cost.

Due to refraction in the laser optics, the laser line exhibits width-wise dispersion, which has a negative impact on the projected line width. For improved sensitivity (to surface anomalies), the projected laser line should be as thin as possible. The laser optics allow for focusing (the procedure is described in Chapter 5). For a projection distance in the region of $10\,cm$, the focused laser line width is specified to be in the region of $50\,\mu m$.

Of all the components forming the system infrastructure, the laser is the component which specifically exhibits a limited operating lifetime. The operating lifetime of a laser diode is determined by the nominal diode junction temperature during operation and high junction temperatures can significantly decrease the lifetime of the diode.

The manufacturer has indicated that the lifetime of the SNF visible red laser, low power diode is roughly $60,000$ – $75,000$ hours at room temperature (ambient). The laser life time is reduced by 50% with every $10\ °C$ in temperature above room temperature ($25\ °C$). In the SDDS application, the laser is held in position by means of an aluminium mounting system, which effectively acts as a large heat

[1]For further information, see `http://www.stockeryale.com/lasers`.

sink and thereby maximising the lifetime. The laser lifetime can further be extended by duty cycling the laser. In the SDDS application, this duty cycling could be done by only activating the laser when the scanning operation is performed. This option has currently not been implemented.

3.4 Camera

To detect defects by means of the project laser line, a camera is required to capture the resultant images. Since the panel surface must be scanned in an acceptable amount of time, the camera must be able to capture images sufficiently fast. Based on this criteria, the camera selected for the SDDS application is the Photon Focus MV-D1024×128 series, which is a high-end multiline CMOS-camera with global shutter. The particular camera used during this project is the Photon Focus D1024-106-CL-8 (serial number 00271429), the technical data of which is given in Table **{3.2}**. Certain items of the technical data require elaboration and will subsequently be discussed.

3.4.1 LinLog™

When the LinLog™ mode is activated, high contrast scenes can be handled without saturation. In this mode, high intensities are logarithmically compressed whilst the low intensities retain a linear response. The transition region can be adjusted smoothly and is specifically designed to be continuously differentiable. For the SDDS application, the contrast between the laser line and the scene background should be maximised in order to optimise the signal-to-noise ratio. Since the LinLog™ mode would tend to decrease the relative contrast between the laser line pixels and the background, this mode is not suitable for use in the SDDS application.

3.4.2 Resolution

In the selected technical data the number of pixels is given as 1024×1024. The camera is however sold with the maximum resolution limited (in firmware) to 1024×128. The specified maximum capture rate is given at this resolution. Due to physical constraints, calibrating the camera / laser mounting system in order to ensure that the laser remains inside the field of view of the camera at all times is non-trivial. A firmware update was therefore obtained from Photonfocus to "unlock" the camera, allowing the full resolution of the CMOS sensor to be utilized. In practice, a resolution of 800×256 is optimal for the SDDS application (this aspect will be discussed in later chapters), with a theoretical maximum capture rate of $773\ fps$.

3.4.3 Data Interface

The CameraLink™ interface standard is a hardware specification which enables high speed data transfer between one or more cameras and frame grabbers with data rates up to $1.9\,GBit/s$. This is achieved using parallel Low Voltage Differential Signalling (LVDS) [101]. The LVDS technology makes use of a shielded twisted pair for every transmission line. Included in the CameraLink™ interface is provision for camera configuration (also over LVDS). Due to the high transmission rates, CameraLink™ cables are limited to approximately $9\,m$ worst case at maximum transfer rate [102]. CameraLink™ cables are available in various lengths; the ordering information is given in Fig. **[3.3]**.

The target deployment environment for the system places constraints on the minimum distance between the processing server (with frame-grabber) and the camera. This effectively limits the minimum

Table 3.2: Selected technical data for the MV-D1024×128 [100].

Sensor Type	analog monochrome CMOS
Number of Pixels	1024×1024, with selectable Region of Interest (ROI)
Pixel Size	$10.6\,\mu m \times 10.6\,\mu m$
Optically Active Area	$10.9\,mm \times 10.9\,mm$
Objective Mount	C-mount
Characteristic	Linear or LinLog ™ adjustable
Full Well Capacity	$200\,\mathrm{ke}^-$
Sensor Noise	< 0.5 grey-level standard deviation @ 8 bit
Fixed Pattern Noise (FPN)	< 3 grey-level standard deviation @ 8-bit
Spectral Range	400–900 nm
Data Interface	CameraLink ™
Data Resolution	8 bit
Configuration Interface	RS232 Standard, 9600 baud 8N1
Shutter Type	global shutter
Exposure Time	$1\,\mu s$ – $0.5\,s$ in steps of $25\,ns$
Capture Rate	up to $1180\,fps$ @ 1024×128
Supply Voltage	$+5\,V$ DC $\pm 10\%$
Power Requirements	$2\,W$
Dimensions	$55 \times 55 \times 46\,mm^3$ $(W \times H \times D)$
Weight	$200\,g$
Operating Temperature	$0\,^\circ C$ to $60\,^\circ C$

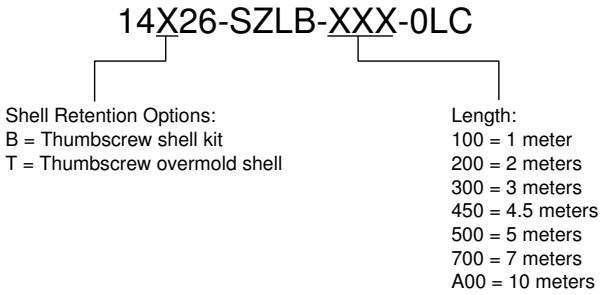


Figure 3.3: CameraLink ™ Cable ordering information [2].

total cable length to approximately $15\,m$. As indicated above, at the maximal transfer rate, the cable should be no longer than $9\,m$.

To solve this problem, a digital repeater (digipeater) is required [103] (see Fig. **[3.4]**). A digipeater consists of LVDS receivers and transmitters connected back-to-back. The digipeater effectively doubles the total distance between the framegrabber and the camera. The digipeater requires a $+5\,V$ DC $\pm 10\%$ power supply at $1\,W$. For the production prototype, two `14T26-SZLB-A00-0LC` cables were used in order to obtain the required total cable length, at the expense of additional noise.

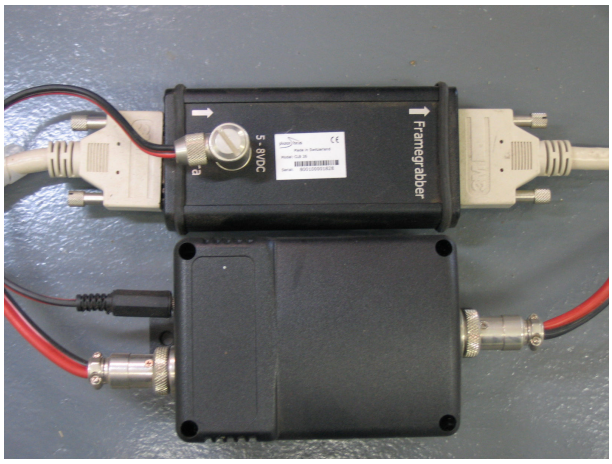


Figure 3.4: Digipeater (top) and power supply.

For debugging purposes, an additional $1\,m$ cable was also obtained. Experience has shown that under certain circumstances, the camera / framegrabber fail to synchronise. To eliminate the longer cables and digipeater as the cause and in general to diagnose camera / digipeater / cable problems, the short cable can be used as a reference. The short cable should also be carefully stored and handled (not flexed excessively) in order to ensure that it can always be considered a “known good” reference.

3.4.4 Region of Interest (ROI)

To optimise the capture rate (by minimising the required image data to transfer), the camera allows a ROI to be defined. Once defined and enabled, the camera will only transfer the pixels in this region to the framegrabber. The ROI can be defined as any rectangular area on the pixel matrix, the only restriction is that 64 pixels must be activated on both halves of the sensor in the x-direction. The

smallest ROI for the camera therefore consists of one line of 128 pixels, centered around the middle of the image. The ROI feature is used in the SDDS application to define a centered 800×256 ROI. Although not used in the SDDS application, the ROI can also be defined as a group of separate rectangular images that are joined together to form a virtual image.

3.4.5 Lens

A fairly standard lens is used on the camera, made by Yamano with focal length of 6 – $13\,mm$ and aperture $1 : 1.6$. The particular lens was selected based on its compatibility with the camera and the estimated focusing requirements. The choice of lens was not exhaustively researched and hence the chosen lens may not be the most appropriate. However, it has been found that the lens is sufficient for the purposes of this application.

3.4.6 Known Issues

A number of anomalies and issues have been encountered with the camera. Although relatively infrequent, upon occasion the camera does not initialise correctly during power-up. Two different effects are seen under these circumstances. In the first case, the camera and framegrabber are not able to synchronise, and no images can be captured. The framegrabber software typically reports this as continual timeouts.

In the second case, images can still be captured, but are incorrectly aligned or the grey-levels are shifted. A power-cycle of the camera corrects both of these problems. Generally, once the camera is running correctly, image capturing remains stable for long periods of time. No explicit measurement of mean time between failures (MTBF) has been performed.

Another issue is the noise introduced by the long cables. This noise appears as localised random fluctuations in areas of mid-level grey-scale values. This noise does not unduly effect the operation of the system. Ideally, the next lower cable lengths should be used ($7\,m$) if the relative production locations of the processing server and camera can be modified accordingly.

It should be noted that, even though the CameraLink™ cables are heavily shielded, a strong source of electro-magnetic interference (EMI) will cause corruption of the captured images. For example, it was found that when the camera cables were brought in close proximity to the switch mode power supply of the workstation, the capture process failed. Care must therefore be taken when routing the camera cables to ensure that the cables are not placed close to any source of EMI.

3.5 Peripheral Power Supply

In Section 1.3.4 mention was made of power supply problems during the development of the initial prototype. Fig. **[3.5]** illustrates the effect of power supply noise encountered during the development of the initial prototype. The design of a robust and noise free power supply for the production prototype was therefore considered a priority. The peripheral power supplies described below performed flawlessly during the execution of the project.

The camera, digipeater and laser all require a stable $+5\,V \pm 10\%$ DC power supply. The power requirements for these devices are worst-case $2\,W$, $1\,W$ and $1\,W$ respectively. In the production plant, $24\,V$ DC is readily available and is the chosen supply voltage. The approximate distance of the furthest device from the power source is $15\,m$. To minimize the probability of failure of the peripheral devices and the possibility of noise, the aim was to introduce large safety margins in the design of the

Figure 3.5: Camera noise as seen during development of initial prototype.

device power supply. The target was therefore to obtain power supply component operating regions well below the limits of the individual components.

The decision was therefore to use $2.5\,mm$ diameter flex cable to create a $24\,V$ DC supply bus. The cable is rated to carry current in excess of $10\,A$. The worst-case current requirements of the devices at $24\,V$ is approximately $0.2\,A$. The cable can thus supply the required current with a large safety margin. The resistance of the cable type selected is given as $6.8 \times 10^{-3}\,\Omega.m^{-1}$. The voltage drop is therefore $25\,mV$ worst case.

A separate $24\,V$ to $5\,V$ DC-DC converter is used to supply an isolated $5\,V$ DC supply to each device. The isolation prevents unwanted interference between devices. The camera in particular can induce high frequency noise when transmitting image data. A large $6800\,\mu F$ capacitor on the $24\,V$ side of each DC-DC converter helps to stabilise the $24\,V$ bus. The DC-DC converters and stabilising capacitor were packaged in a power supply enclosure as shown in Fig. **[3.6]**.

Linear voltage regulators are generally inefficient and EMI shielded switched mode DC-DC converters were therefore investigated. In keeping with the desire to maintain a large safety margin, the DC-DC converter power rating was standardised to be $5\,W$. Furthermore, the price difference between the $5\,W$ and $1\,W$ models was found to be insignificant. The selected DC-DC converter was the MeanWell SLW05B-05, which has the specifications given in Table **{3.3}**. From the specifications it can be seen that the selected DC-DC converters are generally robust with regards to short circuit, overload and operating temperature. In addition, the worst case input voltage drop of $25\,mV$ is not problematic due to the wide input voltage tolerance of the DC-DC converter.

The robot makes available looms and conduits in which cables are routed. The conduits primarily

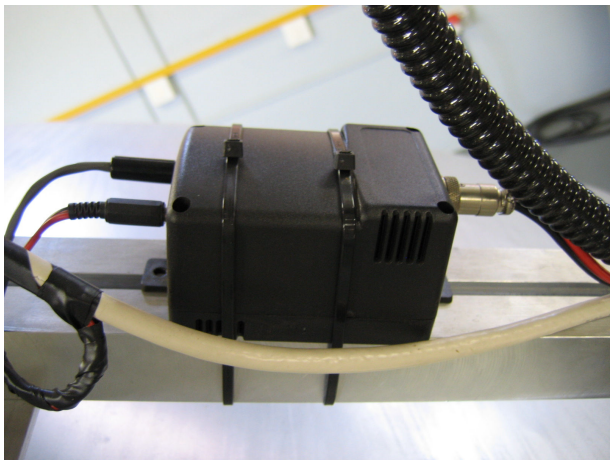


Figure 3.6: Power supply for laser and camera.

Table 3.3: MeanWell SLW05B-05 Specifications.

Input Voltage	$18 - 36\,V$ DC
Output Voltage	$5\,V$ DC
Output Current	$1000\,mA$
Ripple Voltage	$50\,mV$ p-p
Efficiency	76% (typical)
Short Circuit Protection	continuous, auto-recovery
Overload Protection	$150 - 250\%$, auto-recovery
I/O Isolation Voltage	$1000\,V$ DC (min)
I/O isolation resistance	$100\,M\Omega$ @ $500\,V$ DC
Operating Temperature	$-25\,^{\circ}C$ to $60\,^{\circ}C$ (with no derating)
EMI / RFI	Six-sided shield metal case

contain hydraulic lines, $24\,V$ power and fibre optic Interbus cables. There is therefore little chance of cable-to-cable interference. Certain sections may also contain 3 phase power cables, which may be a source of interference. The 3 phase power cables are however fully shielded. At the Groenkloof location, a variable voltage Vanson SMP-924V switch mode power supply is used to create the $24\,V$ DC power supply from the $220\,V$ AC supply. When $24\,V$ is selected as output voltage, the supply is rated at $1\,A$.

Table 3.4: EDT PCI DV Framegrabber Features [38].

Single PCI local bus slot
Supports 8- through 24-bit resolution
Direct memory access to host memory; camera resolution independent
33 or 66 MHz, 3 or 5 V PCI capable
Data rates up to 210 megabytes per second, as supported by host
Programmable mode, exposure time, gain black level and triggering
Supports AIA serial command interface over RS422 and RS232 serial lines

3.6 Mounting System

During the initial stages of the production prototype development, a mounting system was used that consisted of standard laboratory orientated components. Stability and calibration problems similar to those encountered during the initial prototype were experienced. A new mounting system was therefore required and the designers were supplied with specifications. The exact nature of the mounting system was however specifically left undefined in order to encourage an optimal design.

Comparing the new mounting system shown in Fig. **[3.1]** and Fig. **[3.2]** to the original mounting system (Fig. **[1.4]**), it is apparent that the new design differs radically from the original. The reasons for the new design are numerous. Amongst other, the spacial constraints of the gripper played a significant role in the new design. Other factors include the desired for enhanced stability and simplicity. The new design did not exhibit the calibration and stability problems encountered with the previous designs (Section 1.3.4) and generally performed flawlessly.

3.7 Frame Grabber

To capture the data from the camera, a framegrabber card is required. A framegrabber card is, as the name implies, a device which captures image frames transmitted by a camera. For the SDDS application, a high speed camera is used to capture images. The framegrabber must have a sufficiently high performance level in order to capture the resultant large volumes of data.

The framegrabber card selected for the SDDS application is the PCI DV C-Link produced by Engineering Design Team (EDT), with features as listed in Table **{3.4}**. Primary motivating factors for the selection of this card are: known integration with the selected camera and the availability of Linux drivers.

As specified, the camera captures 1024×128 images at a rate of $1180\,fps$. This translates to an effective data rate of $154\,MB/s$. Modern PCs (Pentium III or later, $500\,MHz$ or better) can generally sustain 80 – $90\,MB/s$ on a $33\,MHz$ PCI card, slot, or up to $180\,MB/s$ with a $66\,MHz$ card in a $66\,MHz$ PCI slot. The EDT PCI DV C-Link board is a $66\,MHz$ board; all others are $33\,MHz$ boards. The Dell Workstation used in this project was specifically selected as it contains two $66\,MHz$ PCI slots. Assuming the framegrabber being installed in one of them, the maximum frame rate of the

camera ($154\,MB/s$) can therefore be supported.

It should be noted that the above PCI data rates assume no other PCI bus activity. In particular, the system should not use a PCI video card. The Dell Workstation used in this project contains an AGP based video card and hence video output will not interfere with the framegrabber.

To use the framegrabber, a driver and development library are required. EDT provides[2] a RPM package for Linux Red Hat. The supplied RPM generally requires minimal user intervention during installation. When installing the RPM using, for example,

```
rpm -Uvh EDTpdv-4.0.0.2.rpm
```

the driver module for the active Linux kernel is automatically compiled and installed. During installation a script will be installed, `/etc/rc.d/init.d/edtinit`, which will load the driver module upon system startup (the appropriate run-level links will be created in `/etc/rc.d/rc3.d` and / or `/etc/rc.d/rc5.d`). The development library is installed under `/opt/EDTpdv`.

Before any application can be run that uses the framegrabber, the framegrabber must first be configured. This is done by using the `camconfig` program (assuming the configuration file `pf1024x256-80.cfg` is in the current directory,

```
/opt/EDTpdv/camconfig pf1024x256-80.cfg
```

The driver comes supplied with a large number of standard configurations for the cameras supported by the card . Due to the fact that the camera's firmware was unlocked, minor tweaks to the standard configuration file were required. The configuration file that was finally used is given in Appendix C.

3.8 KUKA Robot

The initial prototype of the surface scanning mechanism made use of the ABB IRB 6400 production line robot. The new BMW production line however makes use of KUKA robots. The particular robot targeted for this project is the KR150L110, which has a specified payload capacity of $150\,Kg$ and a repeatability factor of $\pm 0.2\,mm$. The repeatability factor gives an indication of the accuracy of the robot; the robot is able to return the tool point to a specific coordinate with an accuracy of $\pm 0.2\,mm$. The robot is controlled via a controller which consists of an embedded PC running Windows XP Embedded, and the drive electronics. Windows XP Embedded is a componentised version of Windows XP Professional that contains all of the features, functionality, and familiarity of Windows XP Professional[3].

In the plant environment, the particular robot is used to remove the roof panel from the pallet and prepare the panel to be placed onto the vehicle. From the pallet, the antenna placement hole is punched in the panel and sealant is applied to the panel edges. The sealant must be given $30\,s$ to dry and the panel is placed on the target table. Another robot then picks up the panel and places the panel on the motor vehicle. In order for the gripper to fit into the pallet in order to lift the panel, the gripper height must not exceed approximately $200\,mm$. The SDDS infrastructure attached to the gripper is therefore constrained by this dimension.

[2] available from `http://www.edt.com/`

[3] The letters "XP" stand for "eXPerience", meaning the Operating System is meant to be a new type of user experience.

Whilst the panel is being lifted by the gripper (using the orange suckers seen in Fig. **[3.8]**), the roof approaches within $60\,mm$ of the bottom of the gripper. This dimension constrains the placement of the laser, which is the lowest component of the SDDS infrastructure attached to the gripper. In the final version of the camera / laser mounting system, additional mechanical shielding will be added to prevent accidental damage to the laser.

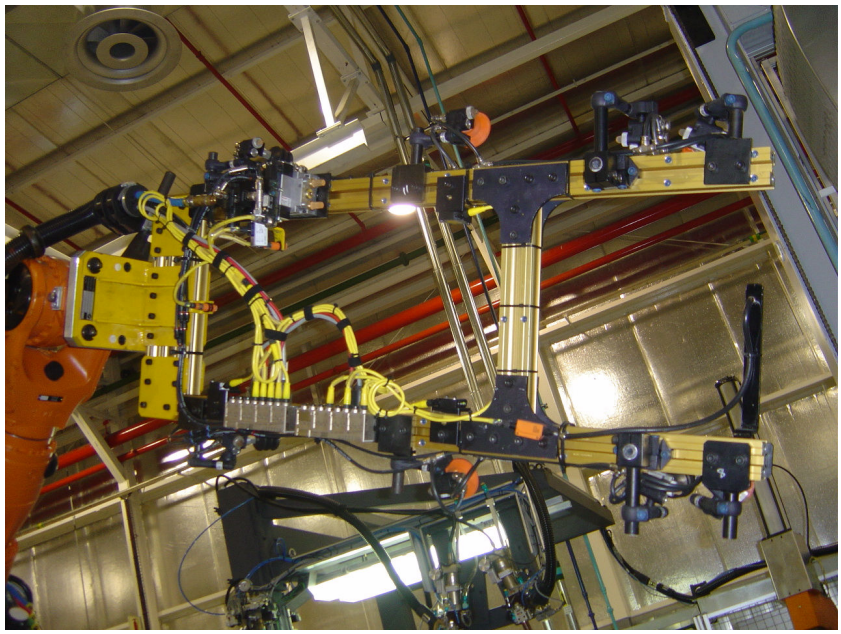


Figure 3.7: Top view of the current production gripper.

Figure 3.8: Bottom view of current production gripper.

The robot used during the development of the production prototype differs from the target robot in the production plant in one respect. The development robot has a $1.5\,m$ arm extension which is not

present in the plant robot. In addition, the plant robot is mounted on a raised platform, whereas the development robot has been mounted on the facility floor. These differences significantly alters the reachability characteristics of the two robots. The work envelopes of the two robots must therefore be carefully compared during the pre-installation stages.

3.8.1 Documentation

The documentation available was obtained on a CD from the KUKA agents. The robot specific information consists primarily of datasheets and technical specifications. The controller information consists of operating, programming and reference manuals. The operating manual describes the basic operation of the KUKA controller. The programming manual describes the KUKA Robot Language (KRL).

3.8.2 Controller Architecture

The KUKA control software executes in a VxWorks environment. The VxWorks Real-Time Operating System (RTOS) in turn runs under the Microsoft Windows XP Embedded Operating System. The system is configured so that a default startup will load Windows and will subsequently load the VxWorks environment and start the controller software. The KUKA control software supports a programming environment in which KUKA Robot Language (KRL) programs can execute.

The KRL defines many of the standard programming language constructs such as variables, execution control and subprograms. In addition, motion commands and various I/O commands are available. A particular characteristic of the KRL execution environment is that only one KRL program may be active at a time. Specifically, there is only one thread of control inside the KRL execution environment and this places significant constraints on what type of control is possible via KRL.

From an operator point of view, the controller operates in one of two user modes: “user” and “expert”. By default the controller starts up in “user” mode after a cold start. In order to program in the KRL environment and to edit system or initialisation files, the controller must be in “expert” mode. The switch to “expert” mode is done by selecting the `Configure` menu, followed by the `User group` selection. During the development described in this document, the controller was always accessed in “expert” mode. The operator interacts with robot / controller by means of a terminal, shown in Fig. **[3.9]**.

3.8.3 Startup / Shutdown

The embedded PC inside the KUKA robot controller is powered by internal batteries, allowing the controller to shut down correctly in the event of an external power failure. Switching the main power switch on the cabinet off causes the drivers to be powered down, and signals a power down / hibernation to the embedded PC. The embedded PC will perform the necessary power-down sequence, and eventually the embedded PC will be switched off. This process typically takes about a minute to complete. When the main power switch is switched on, the controller will restart. The state of the robot will be restored from that saved during the shutdown procedure.

3.8.4 Calibration

The calibration of the robot is performed using specialized equipment and must therefore be performed by trained service personal. In this case, the agents for the KUKA robots, Jendamark, are

Figure 3.9: Terminal used to interact with KUKA Robot Controller.

responsible for the calibration of the robots in the BMW plant. Once calibrated, the robot will not loose its calibration information unless there is a catastrophic failure in the robot drivers or if the cables linking the controller and robot are disconnected before the robot is properly shut down. If the robot does loose its calibration information, the controller will display a message such as "Perform mastering". During the execution of this stage of the project, no robot calibration problems were experienced.

3.8.5 Controller Instability

As reported by the local agents, apparently there has been occasions when the well known “blue screen” has occurred on the robot controllers. This has happened mostly when complex manual operations have been performed and not when under automated / program control. These “blue screen” events have not been seen during this project. There also appears to be a problem with the seating of the memory modules in the controllers. The result is spurious controller problems and such problems have been seen approximately 4 times during the execution of this project. These types of problems were resolved by removing and reinserting the memory modules.

Another problem seen during the execution of this project is what can best be described as deadlock. Under certain circumstances, which have not been clearly identified, the controller will not respond to mode changes via the control terminal. Under these conditions, one cannot switch between program execution mode and manual robot control mode. The apparent cause appears to be background activity that is in progress that blocks the mode changes. This effect was generally found to be persistent and could only be cleared by performing a complete cold start of the system.

3.8.6 Zone 13 Integration

In the production plant environment the particular robot, at which this project is targeted, forms part of Zone 13. Each zone defines default KRL software, which consists of various standard routines.

These routines are typically defined from one "home" position to another (which generally represents one operation). The principle is that if a particular operation is interrupted, the operators can halt the zone robots, fix the error, reset the robot positions to the subsequent home positions, and re-enable the automated control. The zone PLC controls the zone operations by signalling the relevant robot controllers to perform specific operations (by causing the execution of a particular program number). The PLC can also monitor the status of the robots and peripherals via the Interbus communication bus that is used at the plant.

3.9 Squid Ink SQ/2 Ink Jet Printing System

The final hardware infrastructure component is the printing system. The purpose of the printing system is to visually mark anomalies detected on the panels in order for them to be reworked. The requirements for the printing system were relatively simple: the ink used must be compliant with the vehicle manufacturing process and the mark produced must be visible. There are two possible approaches to defect marking.

A possible approach would be to attempt to mark the defects as they are detected. The main problem with this approach is that, unless the marking system is focused at the same point as the virtual camera / laser tool point, the produced mark would be relatively far away from the defect in one of the scan directions (for more details on the scanning motion, please refer to Chapter 5). If however the marking system is so focused, there is a risk that residual ink back scattered may reach the camera lens and result in system failure.

Another complication introduced by the marking system is the question of ink. Since the ink used must be compatible with the spray painting process that would be subsequently applied, the ink must be water based instead of the more typical solvent based. As the roof panel is tilted at a 30° angle (production plant requirement), the ink will run downwards to a degree. An additional disadvantage of marking whilst scanning is therefore that once a identification mark has been made, a “dead-zone” must be created in software around the defect point. In this dead-zone, no further detection is possible due to the potential presence of the defect marking ink. Due to the viscosity of the water based ink, this dead-zone would need to be relatively large.

The second, and only feasible, approach to the defect marking is to only commence marking defects once the entire panel has been scanned and the total number of defects found to be less than the acceptable threshold. The robot would then be given the exact coordinates of the defects, whilst the marker tool point is selected as the active tool point. The result would then be accurately designated defects. The disadvantage of this approach is that the total panel processing time is increased.

The chosen printing system consists of a controller cabinet, ink delivery system and a print head with mounting bracket (shown in Fig. **[3.10]**). The system allows for the printing of various messages and is targeted at printing dot matrix characters on product packaging. This feature is however not used in this application.

Alternatives to this printing system are spray painting systems. These systems however tend to deposit excessive quantities of ink / paint. From a cost perspective, both the printing system and the spray painting systems are comparable. The compact nature of the SQ/2 was however particularly attractive, as well as the ability to control the density of the printing. The printing system can be controlled via a standard RS232 serial interface and allows for various parameters to be modified. The actual marking is however triggered via another port, which is typically connected to a photocell. The trigger port has the following configuration:

- **Pin 6**: Trigger input: active low, sinking minimum $5\,mA$,

(a) Front view (b) Side view

Figure 3.10: Print head of the SQ/2 ink-jet printing system.

- **Pin 7**: Supply: $+12\,V$ DC,
- **Pin 8**: Ground.

To trigger the printing system, pin 6 must be pulsed low for a minimum of $50\,ms$, with a minimum delay of $200\,ms$ between pulses. The resultant activation of the print nozzles is however dependent on a software delay configured in the controller and the length of the umbilical. In the SDDS application, the Data Terminal Ready (DTR) signal line of the serial port used to interface with the printing system controller (Fig. **[3.11]**) was used to trigger the marking. A Sharp PC111L photocoupler was used to provided electrical isolation.

3.10 Chapter Summary

This chapter has described the components that constitute the hardware infrastructure of the project. A number of important aspects were discussed which dictate the nature of the SDDSF. Amongst other, the camera frame rate (related to the camera region of interest), robot architecture and defect marking system were highlighted.

A number of components of the initial prototype were not integrated into the production prototype. Amongst other, the shock sensor was not incorporated. The new mounting system inherently protects the SDDS infrastructure and the shock sensor would not provide significant benefit under the circumstances. Furthermore, due to the spacial characteristics of the new mounting system, the existing shock sensor would not be suitable. It was therefore decided by BMW that no shock sensor would be necessary.

It should be further noted that the robot incorporates sensors which detect joint loads. Limits are normally placed on the acceptable joint loads and when these limits are exceeded, the robot motion is immediately aborted. Excessive damage is therefore prevented in the abnormal case when the robot arm collides with a foreign object. The proximity sensors on the initial prototype were used to detect the panel edges. This was necessary as there was no supporting table available which would ensure correct alignment of the panel. The support table developed for the production prototype, and the

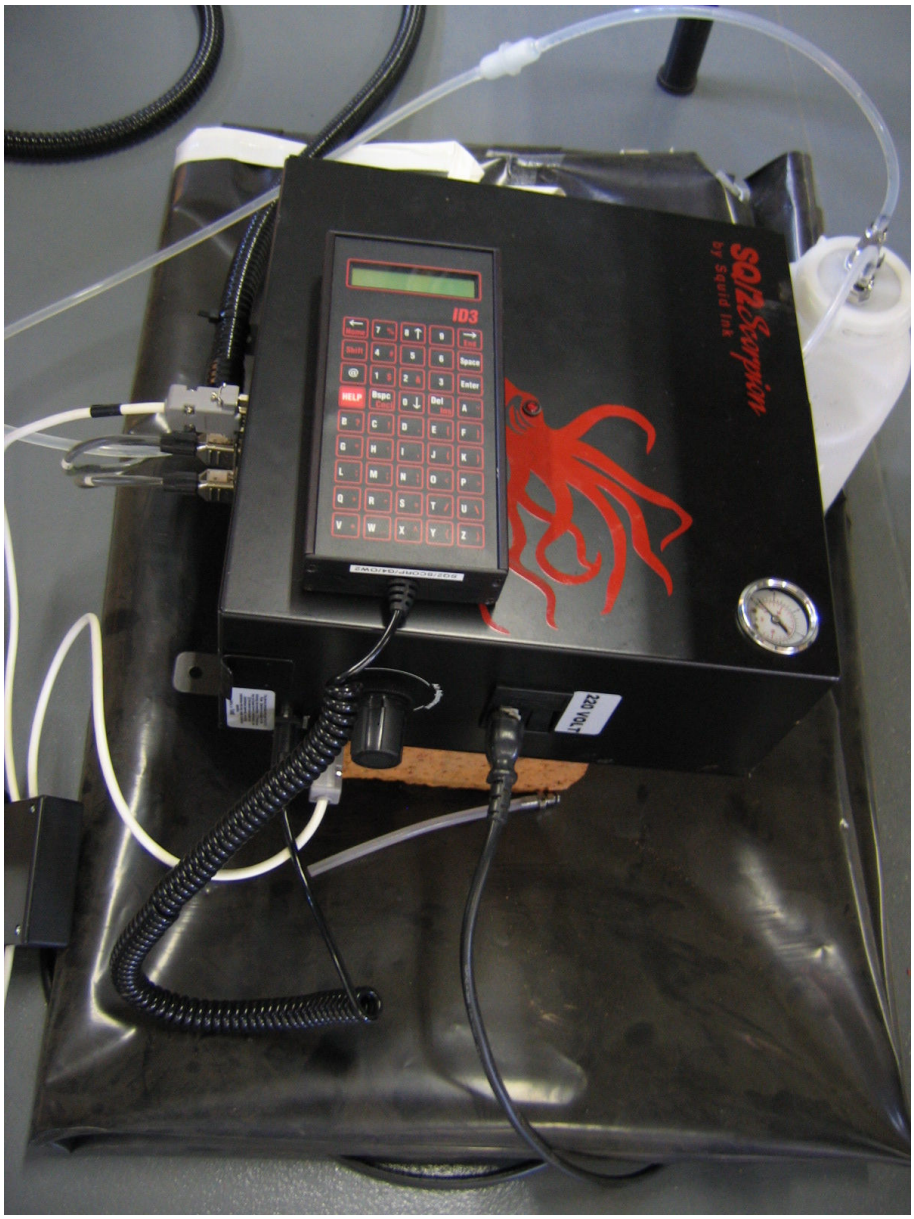


Figure 3.11: Printing system controller unit and ink pressure vessel.

support table used in the production plan environment, therefore removed the need for the proximity sensors.

The following chapter will discuss the software related infrastructure of the project.

⟵⟶

CHAPTER 4

Software Implementation

There are a number of methodologies available for the design and implementation of software, Object Oriented Programming (OOP) being one of the more recent. This chapter will not attempt to provide a detailed analysis of the relative merits of the various methodologies. The author is of the opinion that each methodology has specific advantages in particular situations. Hence methodologies, platform, programming languages, programming environments, *etc.* should simply be considered tools and one of the skills required in system design is choosing the most appropriate tool to perform a specific function. This chapter discusses the architecture, programming language, support libraries and coding conventions of the software implemented.

4.1 Chapter Outline

The first sections of this chapter will identify and motivate the platform and software development methodology used during the development of the SDDSF. Various aspects such as error handling, software layout, software build infrastructure, testing, debugging, profiling will be discussed. External support libraries and systems that are integrated with the SDDSF will be discussed. These include mathematical libraries, database systems and graphical user interface libraries.

4.2 Platform

The Fedora Linux OS was selected as implementation platform for this project. The generic Linux platform was initially selected based on its proven track record with regards to stability, security, scalability and open development environment. The fact that the Linux platform is unencumbered by software licensing costs was considered an additional advantage. The Fedora distribution was selected as a logical progression from the Red Hat distribution used during the initial prototype. The primary motivation for the selection of the Red Hat / Fedora distribution remains that the distribution is specifically supported by the framegrabber driver / library developers (EDT). Such support is crucial as the framegrabber operation is central to the system.

4.3 Structured Programming

The chosen methodology for the software implementation is the well established *structured programming* approach in the C language, applied with an objected orientated flavour. Initially a pure OOP approach in C++ was evaluated. The evaluation of the compiler (g++) and libraries indicated that

there was the potential for performance and memory utilisation problems. Furthermore, the C compiler that comes standard with the Fedora Linux distribution is generally considered to be the most mature of the available compilers and the risk of latent bugs is therefore minimized by its use.

A structured programming approach leads to several advantages [26]:

- Fewer initial mistakes and reduced time and effort in correcting them.
- Effects of modifications are localised.
- More robust and reliable programs.

Structured programming encompasses various concepts, three of which were specifically applied while developing the SDDSF software: abstraction, modularity, and consistency. Abstracting of data is performed by grouping variables related to the same process or action into an abstract data type, and this enhances readability significantly [80]. Whereas the use of C `struct` and enumerated types are examples of abstracting data in standard structured programming, abstraction was taken a step further in this case. In many cases, the data structures were designed from an object-oriented perspective.

For example, the definition of the matrix class type in the SDDSF is given in Example ⌈**4.3.1**⌉ (extract from `sddsmatrix.h`). Notice that "constructor" and "destructor" operations are defined on the generic matrix type. Vector and quaternion classes are "inherited" from the base matrix type as specialisations (aliases) of the underlying type. This allows a multiply operation to accept variable combinations of matrices and vectors, performing the necessary compatibility checks in the underlying specialisation.

Note that, unlike the `sddsmatrix` class definition below, the majority of all class definitions are not published in header files. In other words, the class type is published as an anonymous C type, which allows compiler type checking, but prevents clients from accessing any of the object members. The equivalent in C++ would be to declare all object members as `private`. The data abstraction attained in this manner is of a very high level. The final system therefore contains almost no global or static variables, but rather collections of dynamically allocated "objects". Similarly, abstraction of code involves the creation of standardised interfaces to widely varying actions, thus shielding the programmer and other software from the inner complexities of an action. This is closely coupled with modularity. The end result is a surprisingly programmer friendly environment.

Modularity is achieved by decomposing the complex characteristics of a system into collections of logically related objects and operations [20]. This technique of isolating data and functions within a module, and providing a precise specification for the interface to the module, is called *encapsulation*. In the SDDSF, the software currently consists of approximately 55 main modules, examples of which are: `sddscamera`, `sddshistogram`, `sddsimage`, `sddsmatrix`, `sddspolynomial`, *etc.* At the time of writing, the SDDSF consists of approximately $51,000$ source lines.

The structured, object orientated and modular design has several advantages that lead to higher maintainability:

- **Localisation of complexity**. The complex details related to the implementation of a module's function is localised within the module and abstracted to the rest of the system through a precise interface. This does not only enhance maintainability, but also assists with debugging of the system.
- **Information hiding**. The state (variables) of an object is contained within the object and hidden from the rest of the system. This reduced visibility protects the state from interference by other objects and results in a more robust system.

```
/** Specialisations of the basic matrix type. */
typedef enum {
  SDDS_MSTYPE_NONE,
  SDDS_MSTYPE_VECTOR2D,
  SDDS_MSTYPE_VECTOR3D,
  SDDS_MSTYPE_VECTOR3D_HOMOGENEOUS,
  SDDS_MSTYPE_QUATERNION,
} sdds_matrix_specialisation_type_t;

/** Structure representing a generic matrix (vector, quaternion, etc). */
typedef struct sdds_matrix_s {
  /** Magic number to monitor the validity of the object. */
  uint32_t magic;

  /** Identification which indicates the type of object represented. */
  sdds_matrix_specialisation_type_t type;

  /** The number of rows in the matrix (typically indexed by i) */
  size_t m;

  /** The number of columns in the matrix (typically indexed by j) */
  size_t n;

  /** The total number of elements in the data area. */
  size_t size;

  /** Pointer to the matrix data area. */
  sdds_value_p data;

  /** For large matrices, the above data area is mapped to file
   * instead of being allocated using the standard memory allocation
   * routines. */
  sdds_filemap_p fm;
} sdds_matrix_t, *sdds_matrix_p;

/**
 * Allocate a new m x n matrix and initialise elements to 0.
 * @param m the number of rows in the matrix
 * @param n the number of columns in the matrix
 * @returns the newly allocate matrix
 */
extern sdds_matrix_p sdds_matrix_new (const size_t m, const size_t n);

/**
 * Deallocate resources referenced by the given matrix.
 */
extern void sdds_matrix_free (sdds_matrix_p* mat);
```

Example 4.3.1: Code fragment illustrating object-oriented data abstraction.

- **Separate Compilation**. Modules can be programmed, tested and debugged separately, which speeds up development time.

Consistency is achieved through the use of a standard module and function structure. An example would be to define the internal representation of an object, followed by the constructor / destructor functions. These would then be followed by accessor / mutator functions, and lastly object specific functions. The use of a consistent coding style significantly enhances readability of the source code. Two forms of format consistency can be identified: naming and formatting consistency. Naming consistency refers to the use of consistent naming conventions for abstract data types, variables and functions. Included is the use of meaningful and self explanatory identifiers, as well as the utilisation of enumerated types instead of less descriptive constant values. Formatting consistency requires the use of uniform and beneficial formatting as applied to all of the source code. This includes aspects such as indenting and operator spacing. Appendix D gives an extract of the coding style used during the development of the SDDSF.

4.4 Error Handling

It is inevitable that errors will occur during the operation of a software and / or hardware system. These can be transient (communication packet loss) or permanent (the failure of a subsystem or component). The software system must implement steps to be able to deal with these, and other unanticipated situations in order to respond in an acceptable manner. This section will define the basic fault tolerance philosophy implemented in the SDDSF. Fault tolerance can only be attained if errors are adequately detected and hence the definition of undesired parameter values is central to detecting anticipated and unanticipated errors.

Application level knowledge of the correct operation is known *a priori* in the sense that the knowledge is based on known behaviour and properties of the different processes and subsystems in the system. The programming and execution environment may also provide *environmental level knowledge*: the operating system returns errors if system calls fail and the compiler may include code to check for certain errors. An example of the latter is the new GCC option `-fstack-protector`, which causes the C compiler to add run-time code to check for stack overruns during program execution.

Environmental errors are detected by the underlying operating system or the compiler generated code. Environmental errors, such as a processor exception due to a divide by zero operation, are difficult to handle during program execution. It is not clear that the program should continue to execute in such a case. Generally, prevention is better than cure, and the SDDSF has been specifically developed to minimise the possibility of these types of error. If they do occur, the system will halt and produce a diagnostic error log entry. Due to the integration of the robot, the system halt will cause the robot to immediately halt all activity. This event will in turn, subject to proper implementation of the signal in the cell PLC, cause all activity in the production cell to halt. Although this situation is undesirable, the system halt will prevent external damage due to error propagation.

Application level error detection entails explicitly added error detection code at specific points in the source code. Application level error detection therefore resembles assertions (discussed in Section 4.10.1), except that instead of aborting the system, error handling is performed with the aim to allow the system to continue operation. The exact implementation is highly dependent on the particular action or process. Since the detection code is manually added, the detection is targeted at anticipated errors. The error detection condition can however be made arbitrarily intelligent. Such error detection code is performed using the conditional `if` statement, in which one or more conditions that indicate the presence of an error is evaluated. Clearly there are a large number of possible types of tests, however, the following list constitutes the majority of error conditions that are generally detected by the application level tests:

- **Failure of a library routine**. Many atomic actions in the form of function calls can fail due to insufficient memory, corrupted data structures, attempting to exceed preset limits, *etc.* The caller of the function tests the value of a parameter returned by the function to determine if an error has occurred.

- **Invalid command or action**. The validity of a command or action often depends on the value of one or more state variables. Before a command or action is initiated, tests are performed to ensure that the command or action is allowed, based on the current object or system state.

- **Invalid parameters**. The parameters passed to a function are usually verified by the function before they are used. Note however that in certain cases, particularly internal to a module, these checks may instead be performed as precondition assertions.

- **Corruption of data**. The integrity of a data structure can be verified by utilising a checksum or

redundant copies of the data. This form of checking is often employed where data is transmitted across a communication link (for instance, between two subsystems).

Application errors can be handled more intelligently than environmental errors and normally do not require the system to halt. The exact implementation is heavily dependent on the particular error and area in the software. Every error that is detected must be handled by an error recovery routine. Placing error handling code at every error detection instance would result in a large code overhead and very complicated error handling routines. Many errors occur in subroutines or actions which are always part of a larger process. A more elegant approach would therefore be to indicate the occurrence of the error, and let the higher level process, which has a better understanding of the implications of the error, handle it. This is implemented in the SDDSF using return codes for the majority of function calls. A function in which an error is detected returns immediately with an unique error code. The calling routine checks the return value and if it is not responsible for handling the error, the calling routine in turn passes the error on to its caller. In this way, the error can be handled at the appropriate level. Note that the cleanup or action rollback local to the routine is however performed before the function returns.

In a function containing an error handling section, there are usually various points within the function where error detection is done. Performing an execution branch from these points to where the error is handled (within the same function) normally requires the setting of a flag and the use of nested `if` statements or complicated loop termination conditions. A more elegant solution is provided through the exception mechanism in languages such as ADA and C++; program execution immediately branches to the exception handler when an exception is raised. Exception handlers are not supported by C, but the effect can be simulated through the use of the `goto` statement. Example ⌈**4.4.1**⌉ illustrates the construct under discussion.

```
{
  ...

  perform_action_1

  if (some_error_1)
    goto error_1_out;

  perform_action_2

  if (some_error_2)
    goto error_2_out;

  perform_action_3

out:
  return success;

error_2_out:
  rollback_action_2

error_1_out:
  rollback_action_1

  return some_error_status;
}
```

Example 4.4.1: Abstract code fragment illustrating `goto` usage for exception handling.

Although much has been written[1] about the dangers of the `goto` statement, its use in this manner has merit. A label is defined at the start of the error handling section and `goto` statements are used at

[1]The most well known being that of Dijkstra, now published as `http://www.cs.utexas.edu/users/EWD/`

every error detection point to unconditionally force the program flow to the error handling section. It must be stressed, however, that the `goto` approach is used solely for exception handling, and then only within the same function. Furthermore, the aim should be to have no more than two function exit points. This eliminates a number of structured programming constructs which would have been needed solely to correctly guide the program flow to the error handling routine, and which would also have slowed program execution.

4.5 Directory Layout

The framework top level directory layout is given as:

- `benchmark`: Various performance testing code fragments, used to select the optimal implementation.
- `data`: Data required to support the operation of components of the framework.
- `experiments`: Contains experiments regarding specific techniques and algorithms, used to select the optimal approach.
- `fortran`: Public domain Fortran code fragments used during the development.
- `historical`: Various, now obsolete, code fragments used during the initial prototype and early framework development.
- `octave`: Octave (a Matlab-like system) scripts to help validate certain framework code.
- `src`: The SDDSF source code directory.
- `tests`: The regression test framework, discussed in more detail in Section 4.11.

The absence of a "documentation" directory should not be taken as indicative of the lack of documentation. The documentation of the SDDSF software is included in the source code, as discussed in Section 4.9. This document serves to provide the high-level documentation for the system.

4.6 Software Build Infrastructure

For C based software on the Linux platform, the `autoconf` / `automake` build system is dominant. Although a number of other systems are available, none are included by default in all Linux distributions. The `autoconf` / `automake` was therefore used to manage the software build process. The top-level build configuration files are `configure.ac` and `Makefile.am`, whilst the sub-directories contain local `Makefile.am` files. For more information regarding the `autoconf` / `automake` build system, please consult the documentation[2].

The only aspect of significance is the construction of the SDDSF library (see `src/Makefile.am`). There are in fact two versions of the library built. The debugging library, `libsddsd.a`, is build with maximal debugging enabled. In this version all assertions are enabled (see Section 4.10.1),

`ewd02xx/EWD215.PDF`. For an insightful retrospective on the goto statement as well as structured programming, please visit `http://david.tribble.com/text/goto.html`.

[2]Typically available via "`info autoconf`" and "`info automake`"

inlining of functions is disabled and the addition of compiler generated debugging information is enabled. Inlining of functions, whilst providing performance gains in certain circumstances, does make debugging code more difficult. This occurs due to the removal of the function call, which in turn removes the stack frame of the inlined call. Determining the sequence of events leading to the instructions being executed from a stack trace is therefore more difficult.

The alternate high performance library, `libsdds.a`, is compiled with debugging constructs disabled and maximal compiler optimisation enabled. These optimisations include the advance tree based optimisations only available in the most recent GCC compiler. Since both the development and target Linux servers contain Intel® Pentium IV Xeon™ processors, use of the SSE2 instruction set is enabled for both versions. Together with the tree based optimisation and loop unrolling, the latest GCC allows the possibility of concurrent execution of floating-point operations on the multiple Floating Point Units (FPUs) in the processor.

4.7 External Libraries

The use of external libraries has significant advantages with regards to development time and effort. However, due to the commercial nature of this project, external libraries can only be considered if the library licensing allows for use in a commercial product. Furthermore, if use of the library entails royalties, this would be a disadvantage. Similarly, large (commercial) developers fees would detract from the library.

Other aspects of importance when considering an external library, particularly a numerical library are ease of use, reliability, and performance [32]. The first factor has an impact on the initial development of the software. A complex, difficult to use interface can negate the benefit of development time saving provided by an existing library.

Reliability is crucial in a commercial setting, particularly where the product is mission critical. The software developed during this project can be considered mission critical: once deployed, failure of the software will result in production interruptions and have significant financial implications. The real-time nature of the software creates additional specific, consistent performance requirements.

4.7.1 LAPACK

The Linear Algebra PACKage (LAPACK) library [5] is a freely available[3] library of Fortran 77 subroutines for solving the most commonly occurring problems in numerical linear algebra. Created by recognised experts in the field, it has been designed to be efficient on a wide range of modern high-performance computers.

The library can be used in commercial applications provided that proper credit is given to the authors and reference is made to the user guide (cited above). LAPACK is built on the Basic Linear Algebra Subprograms (BLAS) library, of which a reference implementation is included. Both LAPACK and the reference BLAS are available on most Linux distributions.

For the SDDS, the LAPACK Singular Value Decomposition (SVD) routines is used in the `sddsmatrix` module. The `sddsmatrix` module also contains an interface to the LAPACK routines to solve Linear Least-Squares (LLS) problems. Other routines provided by LAPACK (but not currently used) include routines for solving systems of linear equations, and solutions to various eigenproblems.

[3] LAPACK is available from NETLIB, `http://www.netlib.org`.

With regards to reliability and performance, the LAPACK library has been extensively analysed with regards to both aspects. The details of this analysis are provided in the user's guide. A number of reputable books [45, 32, 138, 60] reference and discuss aspects of the LAPACK software. It should also be noted that many hardware vendors (such as Intel, AMD, Sun Microsystems, *etc.*) provided commercial optimised versions of the LAPACK library (or variations thereof).

The LAPACK library has therefore evolved to be a reference library for numerical linear algebra and as a result is arguably the most peer-reviewed implementation available. The library quality level is therefore considered to be very high. From a performance perspective, the performance of the LAPACK library is largely determined by the underlying BLAS implementation. The next section will discuss an optimised BLAS implementation used for this project.

4.7.2 Automatically Tuned Linear Algebra Software (ATLAS)

The BLAS reference standard defines three levels of routines: Level 1 BLAS performs vector–vector operations, Level 2 BLAS performs matrix–vector operations, and Level 3 BLAS performs matrix–matrix operations. The reference BLAS implementation is written in Fortran 77 and its performance is therefore linked to the quality of the Fortran compiler used to compile the library. Being a generic implementation, the software does not make use of specific features and performance enhancements of the target platform (in particular the CPU).

The ATLAS library [149, 150, 31] aims to provide an optimised version of BLAS, automatically tuned for maximal performance for a specific target platform. This aim is achieved by using automatic code generation based on templates. In order to generate the optimised library, the code generator can be supplied with the specifications of the target platform (CPU). These specifications include parameters such as cache sizes and arrangement, processor extensions, memory layout, *etc.* Alternatively, the code generator can perform various tests to determine the optimal configuration for the current hardware.

The ATLAS library is freely available[4] and is also included in many Linux distributions. The current Linux ATLAS packages for the target platform are installed in such a way that they automatically replace the BLAS (unoptimised) reference implementation included with LAPACK. The usage of ATLAS is therefore transparent with regards to existing software. Although no specific benchmarks were performed to validate the performance improvement, a significant improvement was noticed once the library was installed. For a discussion on the performance aspects of the library, please refer to the papers cited above.

4.8 Version Control

Version Control is part of a larger software management component, namely Software Configuration Management (SCM) [107]. SCM refers to configuration management of all aspects of the software process (development and implementation): the software itself, documentation relating to the software, and any data supplied to or produced by the software.

As discussed in Section 4.5, the software aspects (source code, documentation and data) are stored in a specific directory hierarchy. Due to the nature of the development process of this project (single developer, research driven, *etc.*), only the version control aspect of SCM is of significance. To manage the various versions of the items stored in the directory hierarchy, a version control system is used. The choice of version control system is largely dictated by its availability on the target and

[4] ATLAS is available from SourceForge, `http://math-atlas.sourceforge.net/`.

development platforms, and its functionality.

There are two version control systems that are dominant under the distribution of Linux used: Concurrent Versions System (CVS)[5] and Subversion (SVN)[6]. CVS is arguably the most widely used version control system, particularly for Open Source Software (OSS). CVS does however have specific limitations and Subversion was developed as the next evolutionary step in Open Source version control.

For this project, as there were initially no particular constraints specified, the SVN version control system was chosen. Amongst others, the fact that many large software systems switched to using SVN was a strong motivation in favour of SVN. Some of the large systems that use SVN include the Apache Software Foundation (ASF), the GNU Compiler Collection (GCC), and the K Desktop Environment (KDE). A book [23] describing SVN is freely available[7] in various formats.

4.9 Source Documentation

Source documentation can be divided into three categories: algorithmic (theoretical) documentation, interface documentation and implementation documentation. By algorithmic documentation, the description and discussion of the algorithms used are implied. Many of the algorithms used in this project are derived from a theoretical solution to specific problems. The algorithm documentation therefore entails a mathematical development and the remaining chapters of this document describe these theoretical aspects.

Interface documentation describes the Application Program Interface (API) defined by the software. In particular, the bulk of the algorithms implemented for this project have been organised into a software library. The library consists of various modules, each focusing on particular aspects of the system. To make use of the routines of a particular module, the prescribed interface should be used, as published in the corresponding header file.

To document the interface of each module, the comment blocks using `doxygen` constructs have been used. This allows interface documentation to be generated in various formats (HTML, PDF, *etc.*), by using the `doxygen` system. The `doxygen` system, included with most Linux distributions, reads source code and extracts documentation blocks that have been marked with a specific tag. The extracted information is automatically formatted into hyper-linked documentation, which allows one to browse the documentation using cross-reference links. Also generated are indexes and a table of contents.

Lastly, implementation documentation refers to documentation included in the source code of routines. The aim of this documentation is to describe what function blocks of code perform (the code implicitly describes how the function is performed).

4.10 Debugging

Invariably, errors occur during the implementation of a software system, or errors are present in external systems which interact with developed software. Much has been written about testing and debugging software. However, the fundamental principle underlying efficient debugging and testing of a software system is a logical and systematic approach, driven by experience. There are a number of tools which can significantly simplify the task of debugging a system. The particular tools used

[5]Primary CVS WWW Site: `http://www.cvs.org`

[6]Primary SVN WWW Site: `http://subversion.tigris.org`

[7]Available from `http://svnbook.red-bean.com`

during the development the SDDSF will be discussed briefly in this section.

4.10.1 Assertions

Run-time assertion checking is a "programming for validation" approach that can help ensure that a program satisfies certain constraints. Due to the simple and yet effective nature of assertions, their support have been included in the C language since the earliest versions. In C, support for assertions are provided by a header file `assert.h` in the form of a pre-processor macro. The default response to an assertion being triggered (by evaluating to a TRUE value) is to abort the program and print the filename, source line and statement that caused the assertion failure.

In the SDDSF, program output may be redirected to log files. A modified assertion macro, `SDDS_ASSERT`, has therefore been implemented in the `sddsdebug` module which makes use of the log files if they are in use. The modified macro also automatically includes a stack backtrace, which allows the programmer to quickly identify the sequence of events which led to the assertion failure. Note that if the macro `SDDS_NDEBUG` is defined, the `SDDS_ASSERT` macro will compile to a null statement and the compiler will not emit any code. The high performance version of the SDDSF library is compiled in this fashion.

Other techniques that aids error detection and isolation derive from the fields of formal methods and computational logic [88, 12, 66]: preconditions, postconditions and invariants. These constructs test for certain conditions before, after and during the activation of a function. Since the C language does not directly implement these constructs (unlike the language Eiffel), the SDDSF has these constructs implemented manually by using the `SDDS_ASSERT` macro. Example ⌈**4.10.1**⌉ illustrates the typical structure of a SDDSF library function incorporating the precondition and postcondition assertion statements. Note that invariants are seldomly used as there are almost no global variables.

4.10.2 Memory Allocation Debugging

One of the primary causes of bugs in C is incorrect memory management (memory leaks, dangling pointers, *etc.*). In order to examine the quality of the implementation, two tools were used to check the implementation for memory errors. During the development of the implementation, these tools were used regularly (incrementally) to check for implementation problems.

The first tool is part of the GNU C Library, GLIBC, and is integrated into the core memory allocation routines (malloc and friends). To enable memory tracing, an initialisation call `mtrace` is done. A shell environment variable `MALLOC_TRACE` also needs to be set as the name of the trace file. Upon program termination, the trace file will be written to contain encoded information about the memory allocation/deallocation performed during the execution of the program. Since such tracing imposes considerable overhead, it is normally only done during implementation validation and bug hunting phases. To decode and process this information the `mtrace` program is used. The output of this program gives an indication of memory management problems.

A more sophisticated memory management analysis tool is `valgrind`. Valgrind is a freely available[8], open-source tool for finding memory-management problems in Linux-x86 executables. It detects memory leaks / corruption in the program being run. This tool works similarly to `mtrace`, but provides extensive post-processing of the data. A particularly useful feature is the filtering of system/standard library labels (called *suppression*). The output is therefore more detailed than that of `mtrace`, but with all non-essential information removed (*i.e.* activity inside the standard C library).

[8] Valgrind is available from `http://www.valgrind.org/` and is also included in most Linux distributions.

```
/* This internal helper function performs a forward pass of the
 * network, i.e. supply the pattern to the network to obtain a result. */
static void
sdds_ffnn_recall_augmented (const sdds_ffnn_p nn,
    const sdds_vector_p pattern, sdds_vector_p result)
{
  /* ---[ Preconditions ]-------------------------------------------------- */

  /* Validate object pointer. */
  SDDS_ASSERT (nn);
  SDDS_ASSERT (nn->magic == SDDS_FFNN_MAGIC);

  /* Ensure that the supplied parameters match the neural network. */
  SDDS_ASSERT (sdds_vector_dim (pattern) == nn->input_size_augmented);
  SDDS_ASSERT (sdds_vector_dim (result) == nn->output_size);

  /* Ensure that the extra input unit in the augmented input pattern
   * is set to the desired constant value. */
  SDDS_ASSERT (sdds_vector_get (pattern, nn->input_size_real) == -1.0);

  /* ---[ Function Body ]-------------------------------------------------- */

  /* Perform the forward propagation on the hidden layer. */
  for (size_t j = 0; j < nn->hidden_size_real; ++j)
    ...
    /* code trimmed for example */
    ...

  /* ---[ Postconditions ]------------------------------------------------- */

  /* Ensure that the extra input unit in the augmented input pattern
   * is still  set to the desired constant value. */
  SDDS_ASSERT (sdds_vector_get (pattern, nn->input_size_real) == -1.0);

  /* Ensure that the extra hidden unit in the augmented hidden layer
   * is set to the desired constant value. */
  SDDS_ASSERT (sdds_ffnn_hu_output_get (nn, nn->hidden_size_real) == -1.0);
}
```

Example 4.10.1: Code fragment illustrating the use of preconditions and postconditions.

Common errors that `valgrind` detects include:

- Use of uninitialised memory
- Reading/writing memory after it has been freed
- Reading/writing off the end of malloc'd blocks
- Reading/writing inappropriate areas on the stack
- Memory leaks – where pointers to malloc'd blocks are lost forever
- Mismatched use of malloc/new/new[] vs free/delete/delete[]
- Some misuses of the POSIX pthreads API

The SDDS library modules have been extensively analysed using `valgrind`. The regression tests described in Section 4.11 can be executed using a batch testing script `runtest.sh`. This script will execute all the tests in the test directory using `valgrind`. To configure the operation of `valgrind`, a local configuration file `.valgrindrc` can be created in the directory where `valgrind` is executed. Example ⌈**4.10.2**⌉ gives the configuration file used. The configuration file enables complete execution elaboration, at the expense of increased running time.

```
-v
--tool=memcheck
--trace-children=yes
--track-fds=yes
--time-stamp=yes
--demangle=yes
--num-callers=8
--leak-check=full
--show-reachable=yes
--leak-resolution=high
--freelist-vol=100000000
```

Example 4.10.2: Valgrind configuration file for regression tests.

Apart from memory management problems, general algorithm efficiency was monitored by disassembling critical sections of code. This was done to ensure that the constructs used were sufficiently clear / simple in order for the compiler to generate efficient code. Program line level profiling was also done to detect such problematic constructs.

4.10.3 Numerical Debugging

Implementations where large numbers of numerical computations are performed or are numerically complex are susceptible to various numerical errors. These range from simple divide-by-zero problems to various rounding / truncation errors (due to limited precision). During the development of this implementation, the author found various instances of anomalous results were no program error was detected (*i.e.* no "core" dump).

To help detect these problems at the point at which numerical errors arise, the IEEE floating-point exception mechanism was forcefully enabled (using the `feenableexcept` library call). Once an exception is thrown (resulting in a "core" dump), the GNU debugger (`gdb`) was used to analyse the source of the problem. A number of problems were found using this approach and no further anomalous results were seen.

4.11 A Test Harness

Once developed, code must be thoroughly tested before the code can be deemed correct [13]. The `tests` sub-directory of the SDDSF contains tests for most of the modules in the SDDSF. Typically a framework module will have a corresponding test file in the tests directory, *e.g.*, `tmatrix` is the test file for the `sddsmatrix` module. The test file will contain various tests on each API function published in the module's header file. Apart from serving as an initial validation of the developed code, the tests serve to ensure that the code remains correct after subsequent changes to the code. These tests are often called *regression tests* as the patterns of failure may indicate the cause of the failures.

Note that the tests are compiled against the debugging framework library and hence all preconditions and postconditions are active. Many tests, amongst other tests of mathematical modules, make use of predefined inputs and outputs. Often these have been developed and tested using Octave, a Matlab-like environment available under Linux. Octave scripts are stored in the top-level `octave` directory. In a number of cases, a particular problem has been solved in more than one way. This situation results when the initial solutions proved to be too computationally expensive. These multiple solutions, instead of being discarded, have been retained in the framework and serve to cross-validate the

solutions on wider datasets.

4.12 Benchmarking, Profiling and Optimisation

The system developed has the characteristic that the image processing component has a strict performance target. At the operational image resolution, the images are available at a rate of at most $733\ fps$. The image processing should therefore ideally be able to process the images at the supplied rate. The developed software, making use of the theoretically optimal implementation, was initially found to allow the processing of less than $10\ fps$. This rate of processing is clearly unacceptable and general optimisation of the software was required.

The default compiler provided by the Fedora distribution, the GNU Compiler Collection (GCC), is used to compile the SDDSF. Although the latest versions (4.1.x) of GCC used in this project produces high quality code, the Intel® C++ and Fortran Compilers are likely to produce superior code given they are developed by the CPU designers. It is therefore conceivable that, if significant performance optimisation is required during the final stages of the project, the Intel compilers may need to be used. However, the Intel compilers are commercial products and must be purchased in order to be used for the SDDS project.

In addition to the optimisation required, any changes in core software requires that the software be re-validated (for correctness) and re-benchmarked (to ensure acceptable processing rates). Validation is performed in part by the regression test set. Similarly, a benchmarking framework is required that, at the very least, allows the core image processing loop to be analysed from a performance perspective.

The software developed for this project therefore has the characteristic that an iterative development, profiling, optimisation, regression testing and benchmarking process is required in order to ensure that the software meets the system performance requirements. Without systematic optimisation, it has been found that the developed software fails to complete the required processing within the required time.

Code profiling analyses a running program and determines which routines are most often executed and which contribute most to the overall processing time. Typically, the function which contributes most to the overall processing time will be the function which should be optimised first, particularly if the number of calls to this function is large. A small change in such a function will generally result in significant reduction in total processing time.

4.12.1 gprof

The GNU Profiler `gprof` augments the GNU Compiler Collection (GCC). Compiling source code using the "`-pg`" option instructs GCC to generate profiling information (the program is also linked against the profiling version of GLIBC). Executing a program compiled in this manner will result in a binary file `gmon.out` being generated in the current directory.

To analyse the collected data, `gprof` is then executing by supplying the name of the program to profile as primary option. The output of `gprof` gives the following statistics for each function call executed in the program being profiled:

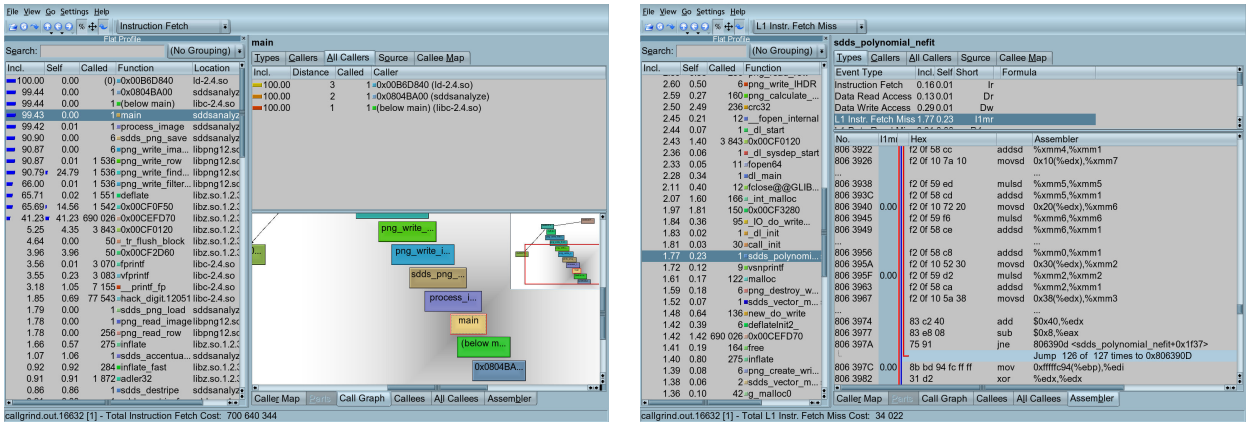
- Percentage of total execution time
- Cumulative seconds (the function plus all child functions)
- Self seconds (actual time spent in the function)

- Total number of calls
- Average time spent in function per call
- Average total time spent in this function plus descendants per call

4.12.2 Valgrind

Included in the Valgrind package is the processor cache emulator. This program `cachegrind` can be used to gather statistics regarding the cache utilisation of a program. The output generated, once processed with `vg_annotate`, shows for each source line the cache hits and misses for both instruction and data reads. The KDE application `kcachegrind` provides a graphical interface which allows the exploration of the generated profiling data. The program can read data collected by Valgrind utilities, as well as other profiling tools such as `gprof` (part of the GLIBC) and `oprofile`.

Fig. **[4.1]** shows the main window of `kcachegrind` during a profile analysis session of one of the SDDS applications. Figure (b) shows the assembler analysis, which was used to examine the cache utilisation of the normal equation-based least-squares implementation (see Section 7.2.1). As can be seen, the emitted code consists of almost exclusively SSE2 instructions and potentially a number of these would execute in parallel. Note also that `cachegrind` has determined that no L1 (level 1 cache) instruction fetch misses have occurred, which implies the code execution is never halted as a result of instruction cache flushing. Amongst other, this explains the performance advantage of the normal equation approach as compared to alternatives.



(a) Call graph view (b) Assembler view

Figure 4.1: Snapshot of KCachegrind windows.

4.13 User Interface

During the development of the SDDSF, graphical user interfaces (GUI) based on OpenGL[9] and GLUT [72] were used. Both OpenGL and GLUT are available by default on most Linux distributions. OpenGL is an environment for developing interactive 2D and 3D graphics applications. The

[9]Primary site: `http://www.opengl.org`

manufacturers of high performance graphics cards make available optimised libraries which conform to the OpenGL specification. These libraries allow the full capabilities of the graphics card to be utilised. GLUT, the OpenGL Utility Toolkit, provides utility functions such as keyboard handling to the OpenGL environment.

The most primitive GUI developed, `sddsgrab`, simply captures images from the camera and displays them in real-time. In addition, a crosshair is displayed to aid the calibration process (as described in Section 5.3.2). The more advanced GUI developed includes the entire defect detection processing pipeline. Various windows opened upon program startup shows the result of each stage of the processing pipeline in real-time. Due to the processing load added by the display and update of the windows, the effective frame processing rate is lower than the ideal.

The GUI that will be used in the production environment will resemble more conventional interfaces. Whilst selection of the widget set to be used for the development of the user interface must still be finalised, the wxWindows[10] is currently the primary candidate. wxWindows does provide integration with OpenGL and hence real-time display of processing results is possible. However, it is envisioned that under normal operation, the GUI will display real-time textual status information and potentially cumulative maps of defects found. The real-time display of all image processing steps will not be attempted due to the processing power required.

4.14 Database System

A database system was not specifically required by any client specification. However, during informal discussions it became apparent that the long term monitoring of defect tendencies would allow the identification of defect sources. Since the logical mechanism to store defect data is a database, a database interface was therefore added to the SDDSF. There are two database systems that come standard with most Linux distributions: PostgreSQL and MySQL. This selection was based on the following requirements:

- be a relational database: the data captured and to be stored is primarily relational in structure;
- must support at least a SQL2 interface: for standardisation and flexibility of database operations;
- exhibit suitable ACID characteristics: an absolute necessity for any critical data; and,
- be stable and obtainable: the database is supplied with the chosen distribution and has been proven to be reliable.

The ACID acronym above refers to Atomicity, Consistency, Isolation and Durability [112]. Atomicity implies that the results of a transaction's execution are either all committed or all rolled back, there are no partial changes allowed in the database. The consistency requirement allows only legal transactions, where legal transactions obey user-defined integrity constraints. Illegal transactions aren't allowed and, if an integrity constraint can't be satisfied the transaction is rolled back. The isolation requirement causes a transaction to be invisible to other transactions until the transaction is complete. This requirement is necessary to ensure that other transactions do not access partial or incorrect data. The durability requirement ensures that once a transaction is complete, the results are permanent and survive future system and media failures.

[10]Primary site: `http://www.wxwindows.org`

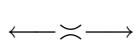
The PostgreSQL[11] database system exhibits the required characteristics listed above. Although the MySQL[12] is generally accepted to be a higher performance database, until recently the database did not comply with a number of the listed characteristics. Therefore, for the type of application under consideration, the PostgreSQL database can be consider more appropriate. From a scalability perspective, experience dictates that the system will be constrained primarily by the hardware and not the PostgreSQL system. To support a large scale database, multiple high speed storage drives are required, typically driven by a hardware RAID solution.

Although a generic database interface module, `sddsdb`, has been added to the SDDSF, the schema to be used has yet to be finalised. The possible data to be stored is largely dictated by the data available within the SDDSF context. Unfortunately, there are no serial numbers or other identification methods for particular roof panels. Hence, the only tagging method feasible is a "start-of-scan" timestamp. Other data elements that will most likely be included is the position and orientation of the gripper and the inter-frame filter state (which tracks consecutive defects). Clearly the original image which triggered the detection will also stored.

4.15 Chapter Summary

This chapter has provided a broad overview of the implementation of the SDDSF. Emphasis was placed on specific techniques, methodologies and tools that were utilised in order to ensure a high quality implementation. The quality improvement techniques employed include a structured programming methodology, an integrated test harness, and a debugging infrastructure to validate the implementation. The tools that were used include a source code version control system and code profilers. Where appropriate, high quality external libraries and software systems were integrated to complete the system. These include the LAPACK Linear Algebra Package, the PostgreSQL database and the OpenGL / GLUT graphics libraries.

The next chapters will focus in more detail on the implementation of specific components of the SDDS. The first chapter will detail the surface scanning operation. The subsequent chapters will describe the various image processing and analysis components of the SDDSF.



[11] Primary Site: `http://www.postgresql.org`.

[12] Primary Site: `http://www.mysql.org`.

CHAPTER 5

Surface Scanning

As described in Section 3.8, the operation of the KUKA robot can be controlled via a program written in the KUKA Robot Language (KRL). In order to scan the panels for anomalies, the camera and laser configuration must be moved over the panel surfaces. A KRL program must therefore be created to allow the robot to produce this type of motion. This chapter will discuss the principles of operation and the SDDSF modules that are responsible for the generation of such a KRL program.

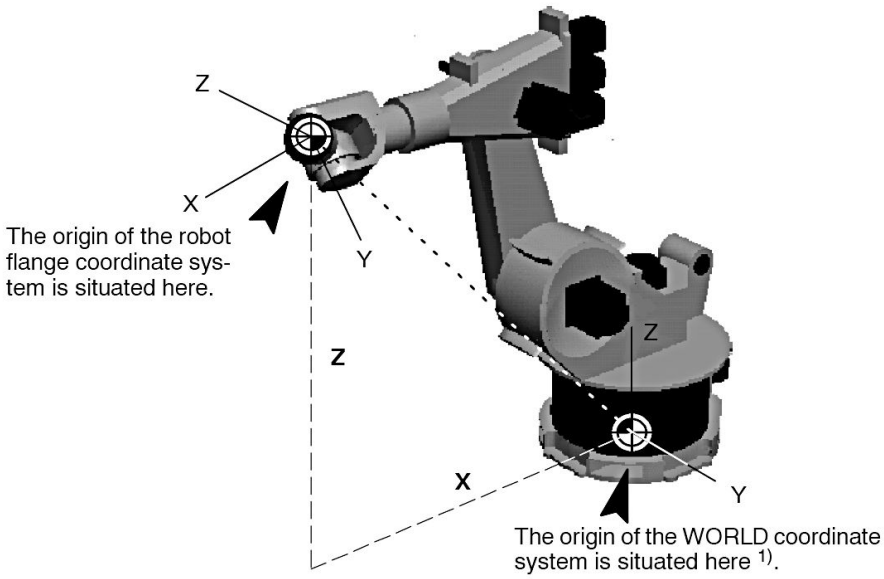
5.1 Chapter Outline

Before the program generation can be discussed, the nature and characteristics of the robot motion must be understood. The first aspect this chapter addresses is the various coordinate systems that are defined by the robot and the SDDS. The calibration of these coordinate systems is crucial to the operation of the SDDS and is discussed in detail. The KRL program generator is subsequently discussed in stages. Firstly, the structure of KRL programs is briefly described. Secondly, the profile data used to define the scanning motion is discussed, together with a description of the basic generator functionality. Thirdly, the handling of distortion effects in the roof panel when mounted on the table is discussed. The chapter concludes by describing the various methods whereby the Linux processing server can interact with the KUKA Robot Controller (KRC).

5.2 Coordinate Systems

The robot can execute various types of motion (point-to-point, linear, circular, *etc.*) relative to a number of different robot coordinate systems. Furthermore, an additional coordinate system is implicitly introduced by the CAD model (which defines the ideal form of the panel). Strictly speaking, the camera also introduces a number of coordinate systems, but these will not be discussed here as they have no direct impact on the program generation.

The system as a whole therefore contains a large number of coordinate systems. During the calibration and program generation, numerous translation and rotations are performed in the various coordinate systems. Therefore, before any discussion related to the motion can proceed, an understanding of the coordinate systems is crucial. The first coordinate systems described below are defined by the KUKA Robot system. The remainder are related to the CAD model and the table.



1) In the basic setting, the world and robot coordinate systems coincide.

Figure 5.1: Primary Coordinate Systems ([76] page 49).

5.2.1 Robot Flange Coordinate System (RFCS)

The fundamental purpose of the robot is to move a specific point, called the tool point, to various points in the accessible area around the robot. Hence, the result of all the programmatic and manual motion commands are that this tool point is moved to the desired location. By default this tool point is defined to be at the origin of the RFCS.

The RFCS is a fixed rectangular coordinate system: it is predefined during the assembly of the robot and never changes. The origin of RFCS is located at the center of the outer edge of the robot flange (which is attached to the end of the robot arm; see Fig. **[5.2]**). Various tools or grippers are typically attached to this flange.

5.2.2 Tool Coordinate System (TCS)

The TCS is a variable rectangular coordinate system and is defined relative to the RFCS. The purpose of the TCS is effectively to define the tool point. The TCS is thus calibrated once a particular tool has been attached to the robot flange (see Fig. **[5.3]**) . The positioning of the origin of the TCS is the end result of all the motion commands.

For the SDDS application, three TCS (tool points) are defined. The first is a calibration point used to perform various calibration operations. The second is the virtual focal point of the camera and laser. The last TCS is the virtual tool point of the marker system. These tool points will be discussed further in Section 5.3.1.

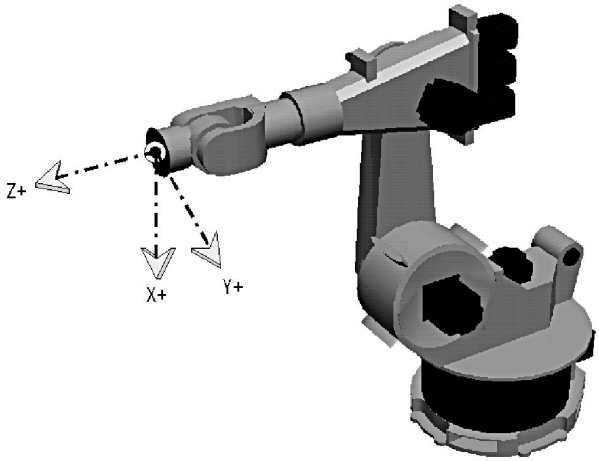


Figure 5.2: Robot Flange Coordinate System ([76] page 49).

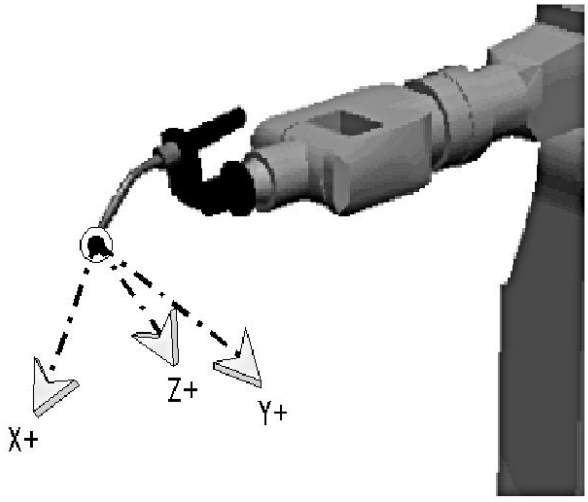


Figure 5.3: Tool Coordinate System ([76] page 49).

5.2.3 Robot Coordinate System (RCS)

This coordinate system is located in the base of the robot (at the center of the robot base, close to the bottom). See Fig. **[5.4]** for a depiction of the RCS. The RCS is a fixed rectangular coordinate system: it is predefined during the assembly of the robot and never changes. At the Groenkloof location, ground plates are used to fix the robot to the floor. Since these ground plates are approximately $1\,in$ thick, the RCS is effectively $1\,in$ above the floor. Since the RCS is inside the base of the robot, it is physically impossible to access the origin of this coordinate system.

5.2.4 World Coordinate System (WCS)

The WCS is a variable rectangular coordinate system: by default, the WCS is defined to be coincident with the RCS. Depending on the application of the robot, the WCS is typically modified to be located

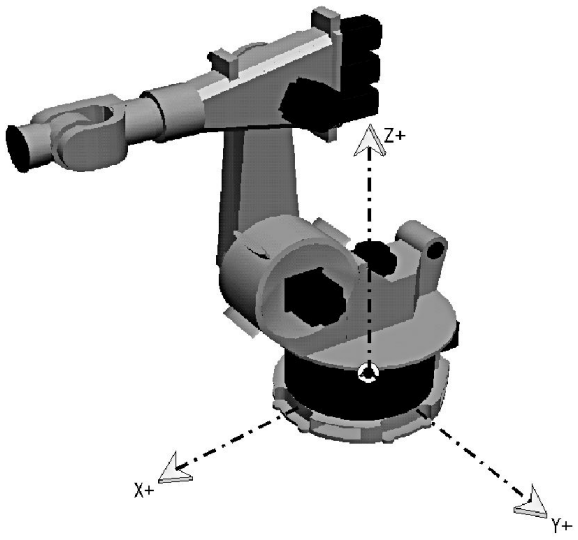


Figure 5.4: Robot Coordinate System ([76] page 47).

at a specific location in the facility. Similarly, other robots in the facility are then configured to use the single WCS definition. If this approach is used, the WCS then allows a single coordinate definition of any point in the facility. At the Groenkloof location the WCS has been left to be equal to the RCS.

5.2.5 Base Coordinate System (BCS)

The BCS is a variable rectangular coordinate system that is defined to have its origin on the workpiece that is to be processed (see Fig. **[5.5]**). For the SDDS application, the BCS is calibrated to be aligned with the roof panel. If the BCS is then selected to be the active coordinate system for programmatic motion commands, the motion commands can then use coordinates that correspond with those of the CAD model.

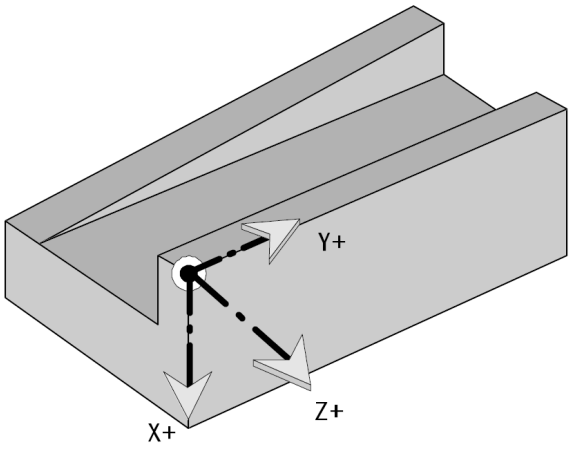


Figure 5.5: Base Coordinate System ([76] page 48).

5.2.6 Joint Coordinate System (JCS)

Each joint in the robot arm effectively has its own coordinate system (see Fig. **[5.6]**). These fixed coordinate systems can be used to move each robot axis individually in a positive or negative direction. For the purposes of the SDDS application, none of the Joint Coordinate Systems are used.

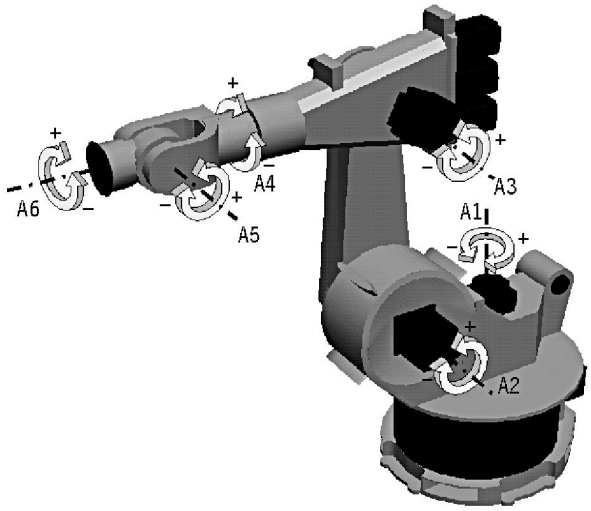


Figure 5.6: Joint Coordinate Systems ([76], page 46).

5.2.7 CAD Model Coordinate System (MCS)

The MCS is a virtual rectangular coordinate system that is defined in the CAD system. The various components of the vehicle are designed and position in the CAD system relative to what will be termed the Model Coordinate System origin. The origin of the coordinate system is defined to be in the center of the front axil of the motor vehicle and the coordinate system faces towards the back of the vehicle (x-axis). The z-axis is therefore in the vertical direction. Fig. **[5.7]** gives a representation of the panel (as seen from the top) and indicates the position of the MCS origin. Note that, although not clear from the figure, the panel is raised approximately $2\,m$ above the $x - y$ plane. It should be noted that there exists a $\pm\,0.5\,mm$ tolerance factor between the CAD model and the physical panels produced.

5.2.8 Table Coordinate System

This is a temporary rectangular coordinate system that is used during the calibration of the BCS. The origin of this coordinate system is located on one of the cross bars in the table and the $x - y$ plane lies in the plane of the table top. In this case the coordinate system is inclined at approximately $30°$ since the table top is similarly inclined.

5.2.9 Support Coordinate System

This is a temporary rectangular coordinate system that is used during the calibration of the BCS. The origin of this coordinate system is located on one (bottom left corner) of the support pillars on which

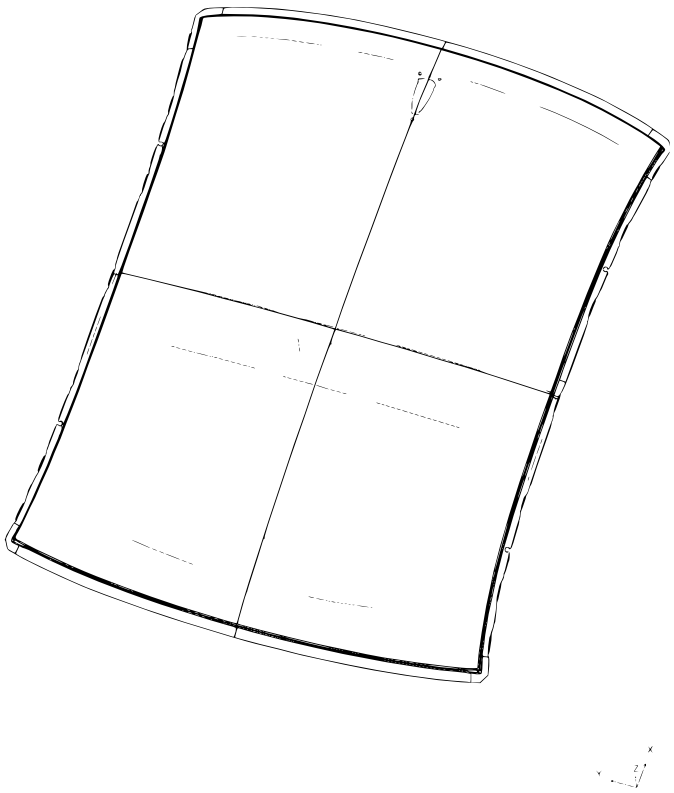


Figure 5.7: Panel with Model Coordinate System (MCS) indicated.

the roof is placed. The $x - y$ plane of the coordinate system is parallel to the $x - y$ plane of the Table Coordinate System.

5.3 Calibration of Coordinate Systems

Initially, the calibration of the variable coordinate systems was performed by manual measurements. It soon became clear that such a manual procedure would be impractical: the procedure is both exceedingly time-consuming and results in significant inaccuracies. The fact that the origin of the RCS is physically inaccessible contribute to this.

A detailed study of the KUKA robot documentation eventually resulted in the procedures below being adopted. These procedures make use of the robot itself to calibrate the various coordinate systems. After extensive experimentation, the procedures have been found to be optimal with regards to accuracy and effort. It should be noted that these procedures sill require considerable time to be performed. To calibrate all aspects of the SDDS requires approximately 6 hours. The calibration data is stored in a file on the controller. This global configuration file is `R1\System\$config.dat`.

5.3.1 Tool Point Calibration: Calibration Point

The calibration of the tool point(s) is the first step that must be performed. As mentioned in Section 5.2.2, there are three tool points used in the SDDS application. The KRC allows as many as 16 tool points to be defined. The motion commands (both programmatic and manual) can make use of any of the user defined tool points. Obviously the default tool point is also available (*i.e.* effectively

the RFCS) and is referenced as the **$NULLFRAME**.

The first tool point defined for the SDDS application is a calibration point (hence named **calpoint**) that consists of a fixed bolt with a sharp point. This tool point, shown in Fig. **[5.8]**, is used to calibrate all the physical coordinate systems used in the SDDS application. Before this can be done, the calibration tool point must itself be calibrated. This is done by a procedure called the "X Y Z - 4 Point". See pages 42-44 of [79] for a description of the procedure.

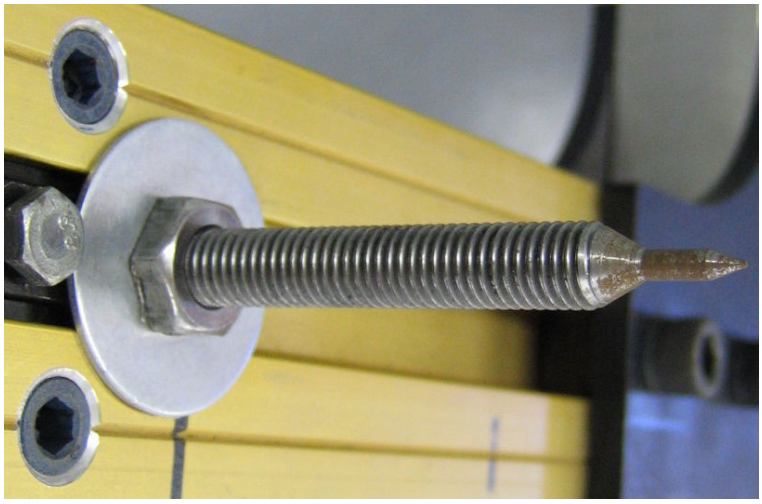


Figure 5.8: Fixed calibration point.

Briefly, the procedure entails positioning the sharp point of the calibration point to a fixed reference point from four different directions (see Fig. **[5.9]**). This produces sufficient linear equations for the controller to solve for the exact location of the calibration point. If the procedure is followed carefully and the positioning is sufficiently accurate, the resultant calibration can attain reasonable precision (typically to $\pm 0.2\,mm$).

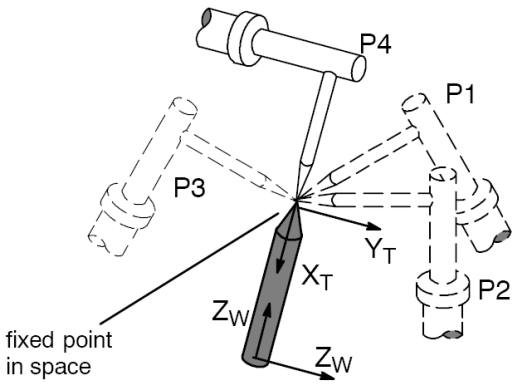


Figure 5.9: Procedure to calibrate a tool point ([78] page 43).

Note that this procedure does not calibrate the orientation of the tool point, only the position (relative to the RFCS). For the purposes of the SDDS application, the exact orientation of the calibration point is not crucial. However, it should be so orientated as to allow the calibration point to be usable.

The orientation was therefore manually specified after informally reasoning about which orientation would be appropriate. This can be done by editing the global configuration file and modifying the a, b or c Euler angles [27] defining the orientation (see Example ⌈**5.3.1**⌉).

```
TOOL_DATA[1]={x 74.1399994,y 0.0799999982,z 357.98999,a 0.0,b 45.0,c 0.0}
```

Example 5.3.1: Tool Coordinate System configuration.

5.3.2 Tool Point Calibration: Camera Assembly

The calibration of the virtual camera tool point is more complex than the procedure for the calibration bolt described above. The reason is that there is no physical point to calibrate. After experimentation, an alternative method was devised for the camera tool point.

Laser alignment

The first step is to align the laser so that the desired laser line is cast onto the target surface. The required equipment are the following:

1. To facilitate the alignment adjustment, a calibration jig was developed (shown in Fig. **[5.10]**).
2. To focus the laser, a Lasiris C-Thru focusing wrench and a small Allen wrench ($0.035\,in$) is required (supplied with the laser).
3. Various screwdrivers and Allen wrenches for adjusting the laser position and orientation.

Figure 5.10: Laser and camera calibration jig.

The procedure for the laser alignment is as follows:

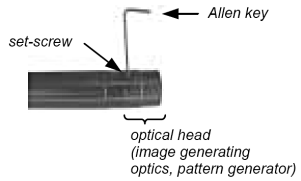
1. Attach the calibration jig to the bottom of the camera assembly.
2. Turn the laser on.

3. The laser position and angle is adjusted until a laser line of approximately $9\,cm$ is cast onto the jig.

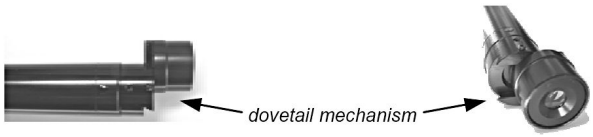
4. The angle the laser makes to the jig should be approximately $7°$.

5. The next few steps aim to focus the laser. Turn the laser off.

6. Using the small Alan wrench, loosen the forward-most set-screw a few turns. This will release the pressure on the optical head, allowing the head to be removed. Be careful not to lose the set-screw.



7. Locating the dovetail slide mechanism, grasp the laser in one hand, and the optical head in the other. Be careful not to touch the glass surfaces. Fingerprints will blur the final image. Gently slide the head across the face of the assembly.



8. Turn the laser ON. Using the supplied C-Thru focusing wrench, gently place the sprocket teeth onto the end of the laser.

   By holding the laser in one hand, and the wrench in the other, gently press and rotate the wrench until the teeth align themselves with the slots in the focusing optics assembly. Once aligned, rotate the wrench clockwise or counter-clockwise, observing the resultant spot size at your fixed target distance. The beam projected from a diode laser is elliptical in nature. Due to the laser incident angle, the resultant illumination is further dispersed. To achieve the best focus, the size of the projected elliptical pattern must be minimised.

9. Turn the laser OFF. Replace the optical head in the reverse order from the above directions by sliding it back into the dovetail mechanism. Turn the laser ON. Hold the laser housing with one hand and, with the other one, gently move the optical head until the projected pattern is well balanced (*i.e.* the optical head is correctly aligned to the laser). Re-tighten the set-screw. Be careful not to over-tighten the screw.

In the description above, the laser angle was calibrated to be approximately 7°. However, during the development of the production prototype, an angle of 12° was used. As discussed in Section 2.4, the laser angle directly determines the sensitivity of the system. Due to inaccuracies in the panel support table, the system proved to be too sensitive to calibration errors when a 7° angle was used. The observed sensitivity of the production prototype is therefore considerably less than is theoretically possible. The use of a high precision support table for the final production version would allow the system sensitivity to be optimised and is therefore strongly recommended.

Camera alignment

The next steps aim to align the camera. The procedure is as follows:

1. The camera is positioned to be approximately vertical above the laser line cast on the jig.
2. Ensure that the camera axis is perpendicular to the jig surface.
3. An online capture program (`sddsgrab`) is run on the server. This program displays the captured frames on-screen, with a cross-hair superimposed.
4. The camera captures frames of dimensions 1024×256. The camera should be orientated so that the laser line is captured along the longest dimension.
5. The camera lens is focused in such a way as to produce a focused image.
6. The camera height is now minimized until the laser line fills the captured frame (*i.e.* occupying approximately all of the 1024 pixels). Note that due to the spherical aberration introduced by the camera lens, the pixels on the edges of the frame will not be usable. Due to the dispersal of the laser light, a similar effect is produced and hence the intensity of laser light on the surface is also not uniform.
7. The jig is now removed.

Tool point calibration

The final step is to calibrate the virtual camera / laser tool point. The idea is to use a small circular dot or hole as the target point. Fig. **[5.11]** shows the particular target point used. As can be seen, the target point is a small hole on a round-head nut. The camera and laser assembly is moved in such a way that the laser line intersects the center of the cross-hair. Implicitly this effect is only possible if the virtual tool point is exactly aligned with the dot or hole. If the calibration target is placed on a pedestal, the four required measurements of the "X Y Z - 4 point" procedure can be captured (as for the calibration in the previous section). The procedure is:

1. Place the calibration pedestal on an elevated sturdy surface.
2. Ensure that the laser is on and the on-line capture program is active.
3. Activate the "X Y Z - 4 point" calibration program on the KUKA Robot Controller.
4. Move the robot so that the camera and laser assembly is aligned with the target point. As described above, the target point should be exactly at the center of the cross-hairs. Simultaneously, the laser line should intersect both the cross-hair center and the target point center. Signal the robot controller to capture the point information.

5. Repeat the above for another three different approach orientations.

6. The robot controller will now display the calibrated tool point and the point can be saved.

Figure 5.11: Camera virtual tool point calibration target.

Similar to the case for the calibration bolt, the orientation of the tool point is not captured during this process. Since the camera was positioned so that the camera axis is perpendicular to the jig surface, the orientation can be derived by reasoning and modified in the global configuration file.

5.3.3 Tool Point Calibration: Marker System

The marker system was attached to the camera mounting system in such a way as to be aligned to the camera. The implication is then that the marker system tool point has similar calibration data to that of the camera assembly, but has a fixed offset. The calibration of the marker system tool point is therefore straight forward: copy the camera system calibration data and add the offset.

5.3.4 Base Calibration

The purpose of the base calibration is to configure the BCS in such a way that the coordinate system is coincident with the resultant / effective MCS when the roof is placed to the table. The generated program can then make use of data points extracted from the CAD model to position the camera / laser assembly correctly.

Unfortunately the procedure to calibrate the BCS is complicated by the fact that the roof must first be positioned correctly on the table in order to minimize distortion of the roof. For this reason, the procedure is done in stages. The first stage is to calibrate the Table Coordinate System. This coordinate system is then used to adjust and calibrate the Support Coordinate System in such a way as to minimize distortion of the roof. Finally, the BCS is calibrated by adjusting the origin of the Support Coordinate System.

Calibrating a Coordinate System

Similar to that of tool points, it is possible to calibrate a coordinate system by manual measurements. However, this also introduces similar inaccuracies. The procedures below therefore make use of the robot itself to measure and calibrate the coordinate system. The method used is called the “3 Point”. See pages 69–74 of [79] for a description of the procedure. Briefly, the method entails positioning

a known tool point (calibrated as described above) to three specific points, which are then recorded. These three points define the origin and the orientation of the coordinate system.

Calibrating the Table Coordinate System

The first steps towards calibrating the BCS is to establish the $x - y$ plane of the coordinate system. The procedure is as follows:

1. Select suitable upper levels on the top of the table to use as a $x - y$ plane. For the given table, the L-bracket cross bars are sufficient.
2. Configure the name of the first "Base Coordinate System" in the robot as **table**.
3. Activate the "3 Point" calibration procedure in the robot controller, choosing the calibration bolt as the tool point. Select **table** as the "Base Coordinate System" to calibrate.
4. Choose any point on the left of the lower L-bracket as an origin and move the calibration point to the new origin. Save the point as the origin.
5. Now move the calibration tool point to the right on the lower L-bracket and save the point. The robot controller has now calibrated the direction and orientation of the x-axis of the new coordinate system.
6. Now move the calibration tool point to the middle of the upper L-bracket and save the point. The robot controller has now calibrated the $x - y$ plane.
7. Since the first point is the origin, and the $x - y$ plane is known, the coordinate system is now calculated.

Calibrating the Support Coordinate System

The next steps towards calibrating the BCS aim to adjust the roof supports as accurately as possible. The procedure is as follows:

1. Examine the CAD model and determine the relative position (x, y, z) of the other supports from one of the supports (the reference support). Typically the points selected should be easily measurable on the physical roof. The outer point where the roof rests was found to be ideal (see Fig. **[5.12]**).
2. On the controller, select the calibration tool point as the active tool point and select the **$NULL-FRAME** as the base coordinate system.
3. Select the Monitor / Robot option to display the current tool point parameters.
4. Move the calibration tool point to the reference support, making sure that the point of interest on the support is accurately aligned with the calibration tool point.
5. Make a note of the x, y, z coordinates of the reference point.
6. Edit the global configuration file and modify the x, y, z coordinates to the new coordinates noted above. Effectively the **table** coordinate system has now been translated so that the origin is situated on the reference support. Since the orientation is left unchanged, the $x - y$ plane will still be parallel to the top of the table.

7. On the controller, now select the active base coordinate system to be **table**.
8. Select the Monitor / Robot option to display the current tool point parameters.
9. Now for each remaining support, using the CAD measurements found in step one, move the calibration tool point to the desired position. Adjust the support so that it is aligned with the calibration tool point.
10. At this point, the Support Coordinate System has been calibrated and if the roof is placed on the supports, it should lie in a plane.

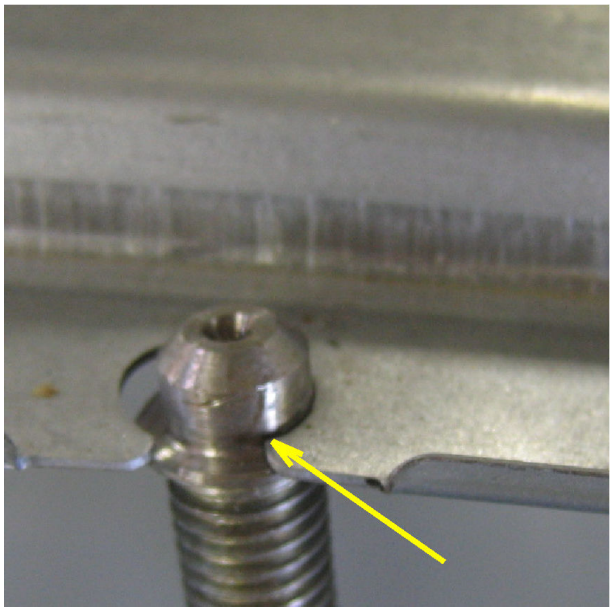


Figure 5.12: Potential panel mount calibration point.

Calibrating the Base Coordinate System

The final steps aim to calibrate the BCS itself. Typically the model coordinates used above to calibrate the Support Coordinate System were chosen at the point which the roof panel rests on the support. These may however not be accurately measurable on the roof. The final step corrects for any such inaccuracies.

1. Examine the CAD model and select a suitable origin of a base coordinate system. The point should be easily measurable on the physical roof. Symmetric points on the three opposite sides of the roof must also be available. Typically a sharp point near the side of the roof is a likely candidate.
2. Move the calibration tool point to the designated new origin on the roof.
3. On the controller, select the active base coordinate system to be the **$NULLFRAME** and the active tool point to be calibration tool point.
4. Select the Monitor / Robot option to display the current tool point parameters.
5. Make a note of the x, y, z coordinates of the reference point.

6. Edit the global configuration file and modify the x, y, z coordinates to the new coordinates noted above. Effectively the **table** coordinate system has now been translated so that the origin is situated on the reference support. Since the orientation is left unchanged, the $x - y$ plane will still be parallel to the top of the table.

7. If there is any doubt with regards to the validity of the $x - y$ plane orientation, the "3 Point" calibration procedure can be used to recalibrate the orientation.

After the above steps, all the required coordinate systems are now calibrated and the scanning of the panel is possible.

5.4 KRL Program Generator

To perform the scanning motion required in the SDDS application, a KRL program must be generated which contains the appropriate motion commands. A KRL program contains numerous components, the details of which are not important to this implementation. To develop an automated program generator, a sample KRL program was therefore created using the KRC programming environment. The resultant sources were then analysed and certain components were extracted. Details of the implementation of the generator will be given in the following sections.

5.4.1 The KRL Program Structure

KRL programs generally consist of two components: the program source file (with extension `.src`) and the program data file (with extension `.dat`). As the names indicate, the source file contains the program commands, whilst the data file contains variable definitions. Both file types generally contain a prefix section, a body section and a suffix section. The prefix section and suffix section are normally automatically generated by the KRC programming environment. User defined commands and data variables are generally added into the body section.

5.4.2 Overview of the Generator

The generator essentially creates the scanning KRL program by creating a source and data file. The predefined prefix sections (as extracted from static template files) are then appended to the start of the files. Subsequently the program generator appends the definition of numerous data variables to the data file. These variables define points in space which the robot must move the tool point to. Similarly, the program generator appends numerous point-to-point motion commands to the source file. These commands reference the data variables added to the data file. Lastly, the generator appends the suffix sections (as extracted from static template files) to the end of the source and data files.

5.4.3 The Standardized Coordinate System

The BCS is typically established in such a way that the origin is situated on one edge of the panel due to the fact that such a point is much simpler to measure than a coordinate in the center of the panel. In contrast, the origin of the MCS is situated at the front center of the panel. In order to allow for coordinates and vectors to be uniformly manipulated in the program generator, all coordinates must first be translated to the same coordinate system.

For this reason, a new coordinate system is defined internal to the SDDSF and all external coordinate

data is translated to this coordinate system during the import process. This new coordinate system, referred to as the *standardized coordinate system* (SCS), is automatically and dynamically calculated. In addition, the aim of this translation operation is to establish a new origin such that no aspect of the panel model intersects a coordinate axis. In other words, all model coordinates are positive. The inverse translation is performed when any coordinate data is exported.

5.4.4 The Profile Data

The motion the tool point must describe is a number of curves across the length of the roof. This motion allows the laser to sweep across the roof, whilst the camera captures the resultant images. The curve in turn is defined by positions on the surface of the roof. The end goal was to allow the SDDSF to directly read the CAD model and generate the desired KRL program. However, the code required to read the CAD model was complex and was not completed during the initial phases of the project. To generate the scanning programs, an alternative source of profile data was used.

The profile data was obtained by using the CATIA CAD system to place a Cartesian grid across the roof. At the grid intersections, normal vectors were placed. The intersection point coordinate and normal vector data for each point was then manually copied to a data file. Example ⌈**5.4.1**⌉ gives the data for a single profile point.

```
COORDINATES  : X =  1287.529071 Y =  -500.000000 Z =  1015.777784
VECTOR       : X = -.268654  Y = -.163832  Z =  .949202
```

Example 5.4.1: CATIA profile data point.

For each trace across the roof a separate file was created and the points related to that trace was placed in that file. The generator then reads these trace files, extracts the trace data points and stores the points in trace memory arrays. Fig. **[5.13]** gives an indication of the profile traced by the tool point given the CAD profile data points.

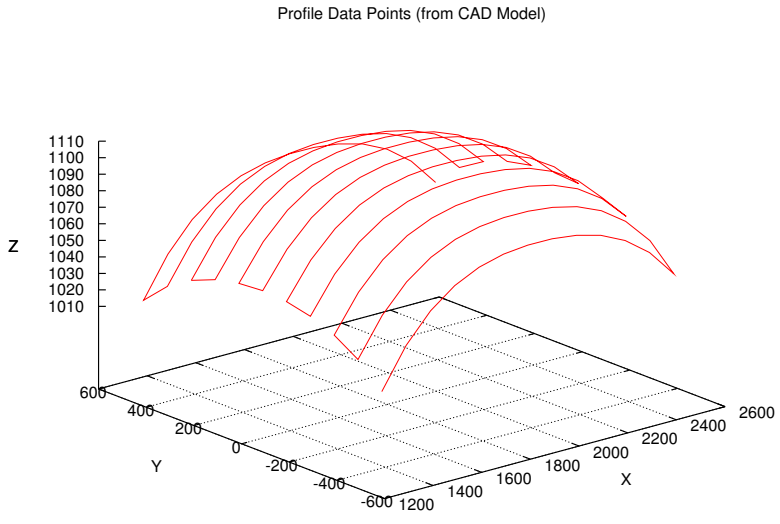


Figure 5.13: Profile data.

5.4.5 Defining the Scanning Motion

The KUKA Robot Controller defines a number of different types of motion commands. These include point-to-point (PTP), relative point-to-point (PTP_REL), and various continuous path motion commands. The ideal would be to use the continuous path motion commands as these commands ensure a constant motion velocity. Unfortunately, the type of motion required by the SDDS application is not supported, the continuous path motion commands only support linear and circular motions. The surfaces of the panel are described in the CAD model as Non-Uniform Rational B-Spline Surfaces (NURBS) [41, 144, 16]. The actual profile that needs to be traced is therefore a *parametric curve*. The term parametric curve refers to a curve for which the path is described by a mathematical function rather than a set of coordinates. A parameter within the function is varied from 0 to 1 to define all the coordinate points along the curve. Note that the parameter or parameters of the curve do not correspond to an axis in the coordinate system. In order to obtain the actual coordinate points, the parametric curve must be sampled.

Point-to-point motion is therefore the only alternative. As the name indicates, the tool point is moved from one point to another. This would however result in motion which exhibits distinctly non-uniform velocity. Typically, the tool-point would be rapidly accelerated from one point towards the next, and rapidly decelerated as the next point is reached. Since the robot is capable of velocities of $2.5\,m.s^{-1}$, the motion would be far from smooth.

The KRC therefore makes provision for modified versions of the PTP commands which make use of approximate positioning. The principle is that if the exact positioning at a point is not required, the motion can be approximated. The approximate positioning contour is automatically generated by the controller. To generate the contour, the controller pre-executes the subsequent motion commands to find future positions. The maximum number of points the controller evaluates in this fashion is 5. This unfortunately limits the ability of the controller to smooth the motion unless the supplied data points are sampled at a fairly high resolution.

For this implementation, the generated motion commands are defined as a number of `FOR` loops containing approximate positioning commands. Each loop defines a single trace (front to back or back to front) across the roof. Example ⌈**5.4.2**⌉ gives the code fragment which would be appended to the source program for the first profile trace (`SCAN0`).

```
FOR counter = 1 TO 13
  PTP SCAN0[counter] C_PTP
ENDFOR
```

Example 5.4.2: Approximate positioning command for single trace.

The `C_PTP` option to the `PTP` command activates approximate positioning. The corresponding definition of the `SCAN0` data array that is added to the data file is given in Example ⌈**5.4.3**⌉.

The `x`, `y` and `z` parameters are the target coordinates of the virtual tool point. The `a`, `b` and `c` parameters define the orientation of the virtual tool point according to the Euler angle definition [27].

5.4.6 Compensation for Profile Distortion

When a panel is placed on the supporting table, the panel is distorted to a degree from the ideal profile, as described by the CAD model. The degree of distortion depends on numerous mechanical factors and tends to be proportional to the cost of the table. Since the table manufactured for early

```
DECL E6POS SCAN0[13]
SCAN0[ 1]={x -237.23092, y 1032.21000, z -30.67259, a 0.00000, b -16.43388, c  -9.93841}
SCAN0[ 2]={x -137.23092, y 1032.21000, z  -4.82234, a 0.00000, b -11.62188, c  -9.14297}
SCAN0[ 3]={x  -37.23092, y 1032.21000, z  13.55217, a 0.00000, b  -7.97319, c  -8.65576}
SCAN0[ 4]={x   62.76907, y 1032.21000, z  26.08694, a 0.00000, b  -5.18558, c  -8.35192}
SCAN0[ 5]={x  162.76907, y 1032.21000, z  34.06584, a 0.00000, b  -3.07127, c  -8.16843}
SCAN0[ 6]={x  262.76907, y 1032.21000, z  38.51952, a 0.00000, b  -1.49212, c  -8.07226}
SCAN0[ 7]={x  362.76907, y 1032.21000, z  40.27389, a 0.00000, b  -0.34054, c  -8.04743}
SCAN0[ 8]={x  462.76907, y 1032.21000, z  39.95577, a 0.00000, b   0.55648, c  -8.09688}
SCAN0[ 9]={x  562.76907, y 1032.21000, z  37.65837, a 0.00000, b   1.59918, c  -8.27827}
SCAN0[10]={x  662.76907, y 1032.21000, z  32.85585, a 0.00000, b   3.07576, c  -8.62902}
SCAN0[11]={x  762.76907, y 1032.21000, z  24.63173, a 0.00000, b   5.15554, c  -9.14853}
SCAN0[12]={x  862.76907, y 1032.21000, z  11.84988, a 0.00000, b   7.94727, c  -9.81548}
SCAN0[13]={x  962.76907, y 1032.21000, z  -6.79249, a 0.00000, b  11.57203, c -10.61664}
```

Example 5.4.3: Positioning and coordinate data for a single profile trace.

development was intentionally designed to decrease costs, the degree of distortion was significant. The primary effect of distortion is an increase in the panel curvature. Due to the fact that the table surface is inclined at $\approx 30°$, the change in curvature is not symmetrical for the left and right sides of the panel.

To determine the nature of the curvature distortion, a number of points are measured on the panel (as it rests on the table). These feature points, often called *fiducials*, are chosen so that the points can be easily and accurately measured. In Fig. **[5.14]** points 381 and 383 are good examples as they are points where the panel flange makes a relatively sharp point. The corresponding points on the CAD model should also be readily available. The collected data is stored in a file (typically named `fiducials.cfg`) and the generator loads this file during initialisation. Example ⌈**5.4.4**⌉ gives an extract of a typical fiducial configuration file. Note that only two data points are shown for each section. The ideal would be to collect at least 20 evenly spaced points for each section in order to ensure that the distortion is correctly characterised.

```
# Fiducial Data: measured 2005/10/25
# Data Format:
#  Fiducial #, model x, model y, model z, measured x, measured y, measured z

[center]
214   2010.00     0.00  1106.37   485.249500   532.323200   59.881480
          ...       ...       ...       ...       ...
215   1198.00     0.00  1023.98  -326.547400   532.037500  -25.643080

[right]
317   1295.59  -559.07   998.00  -229.494400   -26.645520  -45.597850
          ...       ...       ...       ...       ...
393   2537.27  -554.68  1008.00  1012.627000   -21.155240  -36.491090

[left]
317   1295.59  +559.07   998.00  -229.174500  1092.374000  -47.754410
          ...       ...       ...       ...       ...
393   2537.27  +554.68  1008.00  1011.850000  1089.461000  -37.547000
```

Example 5.4.4: Fiducial Configuration File.

Once the program generator has loaded the fiducial data, quadratic polynomials are fitted (see Section 7.2.1). Fig. **[5.15]** shows typical plots of the quadratic polynomials that result. As can be seen, the left measured polynomial has greater curvature that the right side.

Clearly the quadratic polynomials are but crude approximations of what is in reality a Non-Uniform Rational B-Spline curve. It has however been found empirically that a quadratic polynomial does

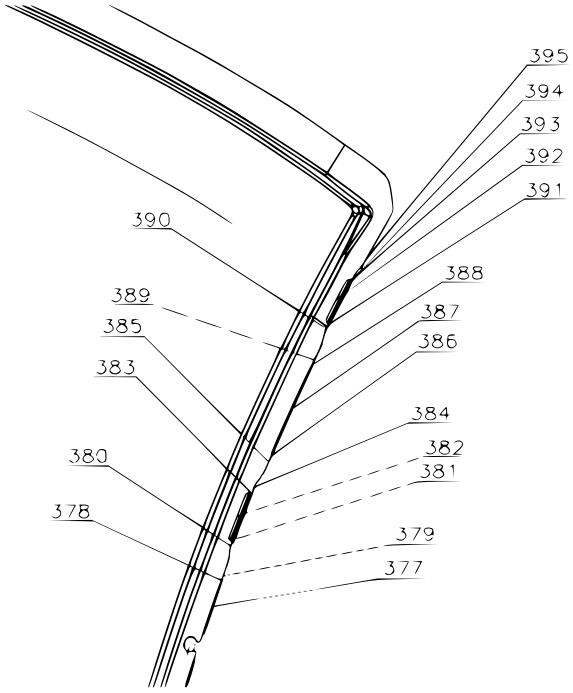


Figure 5.14: Sample of possible points where fiducials may be measured.

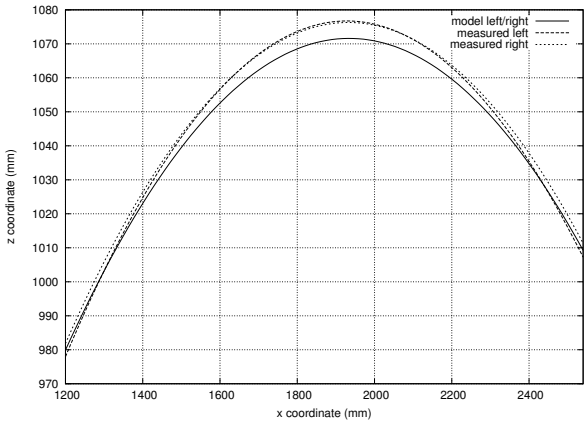


Figure 5.15: Quadratic polynomials fitted to model and fiducial data points.

fit the fiducial and model data points with acceptable residual error. Their use has therefore been deemed acceptable for this particular application. Given these curvature approximations, one can estimate the position of a point on the panel (measured and model) by obtaining the left, center and right approximations for a specific x coordinate.

By performing another quadratic approximation using these points, the panel surface (measured and model) can be estimated. The difference between these two surfaces allows one to calculate the z delta for points of interest and changes in normal vector. The calculated z delta represents the vertical change in position of the panel surface due to the distortion, compared to the expected position based on the CAD model. Note that changes in the x and y coordinates of a point are ignored due to the fact that these changes are generally at least an order of magnitude smaller than the change in z. Given that z deltas are in the region of $5 - 10\,mm$, the changes in x and y are typically below the accuracy

Algorithm 5.4.1 CAD_WARP

1. Load fiducial data from configuration file into separate matrices, one for each section
2. Translate the fiducial to the Standardized Coordinate System (Section 5.4.3)
3. Calculate curvature quadratic interpolation polynomials
4. Determine quadratic polynomial which describe the z coordinate deltas (which effectively describe the vertical deformation of the panel).
5. For each profile data point, consisting of a position coordinate and a normal vector, the following is performed:
   (a) Estimate z delta and correct position coordinate z component:
      i. Obtain the expected z delta at left, center and right of panel (for current x coordinate)
      ii. Perform quadratic interpolation to find actual z delta at current point
   (b) Estimate delta in c angle (rotation about x-axis) and generate an appropriate rotation matrix
      i. Obtain model and measured coordinates at left, center and right of panel (for current x coordinate)
      ii. Calculate the declination angle (for both model and measured coordinate) relative to center of panel
      iii. Using these declination angles, calculate the declination angle deltas at left, center and right of panel
      iv. Perform quadratic interpolation to find actual declination angle at the current coordinate
      v. Create a 3D homogeneous rotation matrix based on the declination angle delta
   (c) Estimate delta in b angle (rotation about y-axis) and generate an appropriate rotation matrix
      i. Find the coordinates of panel maximum using Brent's minimum search on the inverted panel profile
      ii. Using the z delta calculated earlier, determine the declination angles relative to the center of the panel (for both the model and measured coordinates)
      iii. Determine the expected declination angle delta (for the current x coordinate)
      iv. Create a 3D homogeneous rotation matrix based on the declination angle delta
   (d) Apply rotation matrices to normal vector

threshold of the robot. The compensation procedure implemented in the program generator, which is performed just before the data file is written, is given in Alg. $\lfloor$**5.4.1**$\rfloor$.

5.5 Robot Control

The direct control of the robot from the Linux server was also investigated in order to provide options for the marking process. Real-time information about the current robot position is also required in order to tag images captured. Such position tagging allows spacial statistical information to be collected regarding the placement of defects. Such information would allow the identification of typical causes of the defects, and may allow for a form of early warning system to identify pending process failure.

5.5.1 Interbus

Internal to the controller, the robot makes extensive use of Interbus connects. In the plant environment, Interbus is also used to connect the robots in a cell to the cell PLC. An Interbus connection would therefore be a potential means to connect the Linux server to the KRC. However, an Interbus card would be required, together with the associated driver and development libraries. A potential

problem with this approach is that, even though Linux Interbus interfaces are available, there is no local knowledge regarding their use. Another potential problem is that the Interbus interface and API are orthogonal to the control of the robot. Hence there are two isolated areas that must both be working in order for robot control to be possible.

5.5.2 Procedure 3964R

Included in the KRC installed software is a RS232 based communication system call Procedure 3964R [75]. Due to the fact that the system is available, this mechanism was investigated. The 3964R protocol is an asynchronous, bit-serial transmission procedure in which the data to transfer is packetised at the byte level. The packets are delimited by the standard `STX` (`0x02`) and `ETX` (`0x03`) control codes and data escaping is done with the `DLE` (`0x10`) control code. Fig. **[5.16]** diagrammatically shows the progression of the protocol exchange. As can be seen, the `DLE` control code is used as an acknowledgement (ACK). Negative acknowledges are indicated by the `NAK` (`0x15`) control code.

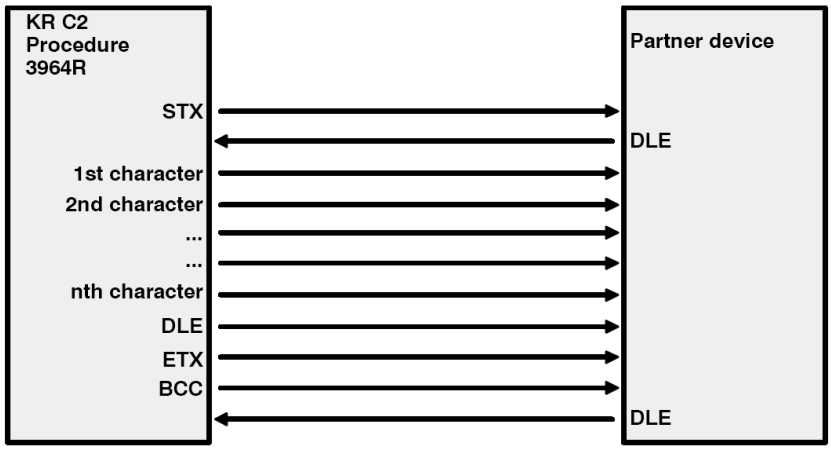


Figure 5.16: Protocol 3964R exchanges between KRC and partner device [75].

After extensive experimentation with this mechanism, two significant disadvantages were identified. Firstly, due to the RS232 nature of the communication, the baud-rate of the communication is limited. For a distance of $8\,m$ between the KRC and the Linux server, as was the case during development, the baud-rate is limited to 9600 baud. Another significant problem is the sequential nature of the KRL execution. Since the KRC can only execute a single task at any one time, any serial communication effectively halts the robot operation. This effect is unacceptable in the target application and the 3964R approach is therefore infeasible during profile scanning.

5.5.3 Ethernet-RSI

After an extensive search for alternatives to the two approaches described above, the following reference was found during a web search [1]:

> KUKA Development Labs, provider of consulting, development and integration services, today announced the availability of a direct Ethernet sensor interface for KUKA

[1] ThomasNet Industrial News Room: `http://news.thomasnet.com/fullstory/21426`

> Robots called Ethernet-RSI. Ethernet-RSI allows an external computer to communicate directly with the motion system of a KUKA Robot Controller (KR C), allowing an external computer to dynamically adjust the current motion of the robot. It also has the ability to stream data to an external computer in real-time.

The Ethernet-RSI module therefore appeared to allow the type of interaction between the KRC and an external system. The further advantage of this approach is that it makes use of Ethernet as media, an interface which comes standard on both the KRC and the Linux server. The only hardware investment would therefore be a cross-over Ethernet cable. Unfortunately, at the time of initial enquiry, the module was still undergoing field testing and was not publicly available. In addition, the documentation was available only in German. Recently, a number of months subsequent to the initial enquiry, the module was made available to this project with special permission from KUKA Roboter GmbH management.

The new system in fact comprises of two separate modules: the Robot Sensor Interface (RSI) and a special RSI component, Ethernet-RSI. The RSI makes it possible to create sensor applications in the programming language KRL (KUKA Robot Language). It contains a library of standard functions for sensor applications, such as filters, transformations, control functions, *etc.* RSI is object-oriented, modularly structured, and provides a special set of commands for standard applications [77]. The Ethernet Robot Sensor Interface (Ethernet-RSI) module enables Ethernet based sensor data acquisition within the KUKA RSI (Robot Sensor Interface) framework. Ethernet-RSI enables KRL programs to acquire and utilize real number formatted data across Ethernet using the RSI framework [74]. Fig. **[5.17]** diagrammatically describes the relationship between Ethernet-RSI and the RSI environment.

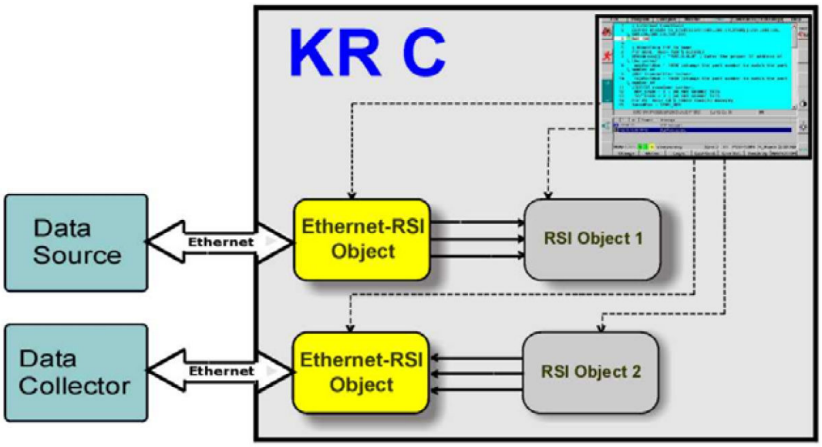


Figure 5.17: Ethernet-RSI Module ([74], Fig. 1).

The RSI approach is fundamentally object orientated. RSI objects, which execute concurrently with any KRL program, are instantiated via special calls in the KRL program. These objects are linked together in a manner that resembles the approach taken in the Hardware Description Languages (HDL) VHDL or Verilog. Similarly, once instantiated, the objects are concurrently active and the result depends on the inter linkages between the objects. The network communication performed by the Ethernet-RSI makes use of the TCP/IP protocol. The advantage of using TCP/IP is that any machine which supports the TCP/IP protocol can be used to interact with the KRC.

There is however a disadvantage to using TCP/IP over Ethernet and this disadvantage stems from the nature of Ethernet. The access method Ethernet uses to control access to the shared transmission

medium is called Carrier-Sense-Multiple-Access with Collision Detection (CSMA/CD) [134, 49]. CMSA/CD operates as follows, a station wishing to transmit a message first senses (listens to) the medium and transmits the message only if the medium is inactive. Then, as the message is being transmitted, the station monitors the actual signal on the transmission line. If this is different from the signal that is being transmitted, a collision is said to have occurred and been detected. The station then ceases transmission and tries again later.

Collisions that occur on the transmission medium result in the loss of the data packet. The data packet must therefore be resent and this operation is performed by the TCP/IP protocol. Although the data packet will arrive at the destination, the delay caused by the retransmission is however problematic, particularly in the real-time control of a robot. For this reason it is strongly recommended that the processing server and the KRC be directly connected via a full-duplex cross-over cable. Such a cable will allow simultaneous transmission in both directions (*i.e.* full-duplex) without any collisions caused by other systems attempting to send data.

At the time of writing, the position reporting component of the targeted Ethernet-RSI implementation had been integrated into the SDDSF. The RSI part of the implementation consists of a `ST_ACTPOS` object which supplies the robot's current Cartesian position (x, y, z) and orientation (a, b, c). Each output of the `ST_ACTPOS` object is then linked to `ST_P` (proportional) objects, which performs unit conversion. The results are subsequently linked to a `ST_ESIRSI`, the Ethernet-RSI object. The `ST_ESIRSI` transmits its inputs as a TCP/IP packet with specific format. The fixed message formats are defined in Fig. **[5.18]**.

Number of Bytes	Starting Index	Format	Description
4	0	4 byte size unsigned integer	Future use (type and format may change in future)
1	4	1 byte size unsigned integer.	Determines message type. Only three message types are available. 0x01 → CONNECT_REQ to request a connection. 0x02 → DISCONNECT_REQ, request for disconnection. 0x03 → DATA_MSG
1	5	1 byte size unsigned char	This byte identifies the version of the message format. Bits 0-3 indicate minor revision and bits 4-7 indicate major revision. 0 0 0 1 0 0 0 0 7 4 3 0 So for version 1.0 of the message format this byte will always be set to 0x20 or in decimal 32.
19	6	1 byte size unsigned integers.	Future use (type and format may change is future)
96	24	4 byte size real (float) values	Data values in IEEE floating point data type format.

Figure 5.18: Ethernet-RSI Message Structure [74].

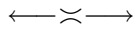
In order to interact with the KRC without disrupting the image processing being performed during a scan, a separate TCP/IP communication thread is used. Once connected to the external server, the Ethernet-RSI module emits a data packet every $12\,ms$. If a problem occurs with the reception of the packet, the robot will perform an immediate motion halt. In the current implementation, the

communication thread receives the data packet, extracts the position and orientation, and publishes the data in a thread-safe global position object within the SDDSF. The image capture object makes use of this global object to obtain the current position and orientation for image tagging purposes. Future research, described in Chapter 9, will integrate dynamic position control.

5.6 Chapter Summary

The chapter has described the generation of a KUKA Robot Language program which allows the desired scanning motion of the robot to be performed. A mechanism was explained whereby the generator can compensate for distortion effects of the roof panel due to inaccuracies in the support table. These distortions cause the laser line to be outside the field of view of the camera when scanning certain sections of the roof panel. In such cases, the SDDSF is clearly unable to detect defects on the panel. The implemented solution allows the majority of the roof panel to be scanned successfully. The possible mechanisms for communication between the Linux server and KRC have been discussed. By the creation of a control feed-back loop using these communication channels, future research (discussed in Section 9.2.1), will investigate the real-time dynamic orientation correction of the gripper. This orientation correction will allow the SDDS to automatically compensate for distortion effects in the panels.

The following chapter will introduce the first stage of the image processing pipeline, as implemented in the SDDSF.



CHAPTER 6

Stage 1 Processing

This chapter describes the first stage of the processing pipeline. The purpose of the first stage of the processing pipeline is to perform correction for various effects in the captured images and to extract the essential observed laser curve. The input to this stage is therefore a 800×256, 8-bit, grey-scale image and the source and nature of this input image are discussed. The output is an 800 element vector of extracted y coordinates which represent the best estimate of the observed laser curve. This vector is called the extracted laser curve vector.

6.1 Chapter Outline

The chapter begins with a brief overview of the relevant digital image concepts. For more information regarding digital image processing, please consult one of the general references [46, 117, 105, 21, 68, 114, 115]. Subsequent to the general overview, the various sections of the first stage of the processing pipeline will be discussed. The primary sections of the first stage are: image artefact removal, background removal and laser curve extraction.

6.2 Elementary Optics

An image is formed when light reflected from an environment is captured by an image sensor. For the captured image to be in focus, some optical mechanism such as a lens must be employed to focus a part of the received light onto the image sensor [140]. It is possible to capture images without optics. For example, an image can be captured by projecting the shadow of an object cast by a strong light source onto the image sensor. However, this form of image, known as a *silhouette*, generally only contains object boundary information and is of limited application.

Lenses are characterised by their *focal length* and their *aperture*. These two parameters determine the performance of the lens with regards to the magnification and quantity of light focused respectively. The quantity of light focused determines the effective quality or signal-to-noise ratio (SNR) of the image captured and the magnification determines the level of detail.

The focal length (measured in mm) determines the magnification of the lens and its field of view (the effective size of the visual environment focused). The required focal length f can be calculated from the basic lens equation [19]:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{6.2.1}$$

where v is the distance from the lens to the image (sensor)
u is the distance from the lens to the object (scene)
f is the focal length

By definition, the magnification factor M is:

$$M = \frac{\text{image size}}{\text{object size}} = \frac{\text{image distance}}{\text{object distance}} \tag{6.2.2}$$

Then the required focal length is:

$$f = \frac{uM}{M+1} \tag{6.2.3}$$

The lens used in the SDDS application has a focal length of $6 - 13\,mm$. From the image sensor technical data in Table **{3.2}**, the sensor dimensions can be calculated to be $10.8544\,mm$. Assume that the length of the laser line as cast on the roof is $10\,cm$. The choice of the length is dictated by the trade-off between the required number of profile traces required to scan a panel. Then, from the above equation, this gives a possible camera height range of $6.1 - 13.3\,cm$ in order for the laser line to fill the captured image.

The lens aperture determines the amount of light passing through the lens. The normalized lens aperture is given by the lens *f-number*, and is defined as the focal length divided by the diameter of the aperture. The standard scale is 1.4, 2, 2.8, 4, 5.6, 8, 11, and 16, where each successive increase decreases the amount of light passing through the lens by half. The f-number is given by

$$\text{f-number} = \frac{f}{D} \tag{6.2.4}$$

where f = focal length
D = diameter of aperture

Since this is a standard lens, images captured by means of the lens are subject to various forms of aberration (imperfect reproduction of scene on sensor). A number of the forms of aberration relate to the response of the lens to light of various wavelengths. Since the image sensor in the SDDS application is monochrome, these forms of aberration typically result in slight blurring of the captured images. This results due to the difference in the paths travelled by the light of different wavelengths. However, the magnitude of these effects in the SDDS application have been found to be negligible.

Of significance is spherical aberration, which is caused by the curvature of the optics. Spherical aberration relates to the fact that the light rays far away from the axis of the lens (*i.e.* effectively the light rays from the edges of the field of view) are not focused perfectly at the focal point of the lens. The effect of this aberration is that captured images will contain a circular distortion near the edges of the image. For this application spherical aberration is ignored by discarding the pixels at the edges of the image using the camera ROI, and is the primary reason the capture width is set to 800 pixels instead of 1024.

6.3 Sampling and Quantisation

By definition, a digital image consists of a number of discrete values that represent the original continuous image that was captured. If one considers the continuous image a function of x- and y-

coordinates, the conversion of the continuous function to a discrete representation entails the *sampling* (evaluation) of the function at various x- and y- coordinates. The sampled value (which is still a continuous value) represents the illumination and/or colour of the function at the given coordinates. This value must then be converted into one or more discrete values. This latter process is known as *quantisation* [46, 140].

The sampling and quantisation processes can be performed in various ways. Until recently, the most common image capturing sensor was the Charge-Coupled Device (CCD) [82]. A diagrammatic representation of a CCD sensor and associated support electronics is given in Fig. **[6.1]**. In simple terms, the CCD consists of a grid of *photosites*. Each photosite consists of a photodiode and an adjacent charge holding region. As incident light strikes the exposed photodiode, the photodiode converts the light into charge by means of the *photoelectric effect*. The charge is stored in the charge holding region. The captured image information is transfered by shifting out the charge in each cell, where the charge is converted into a voltage. External to the CCD sensor, the voltage is converted to digital by means of an analog-to-digital converter (ADC) in the CCD support electronics. The resultant bit stream represents the digitized form of the original image.

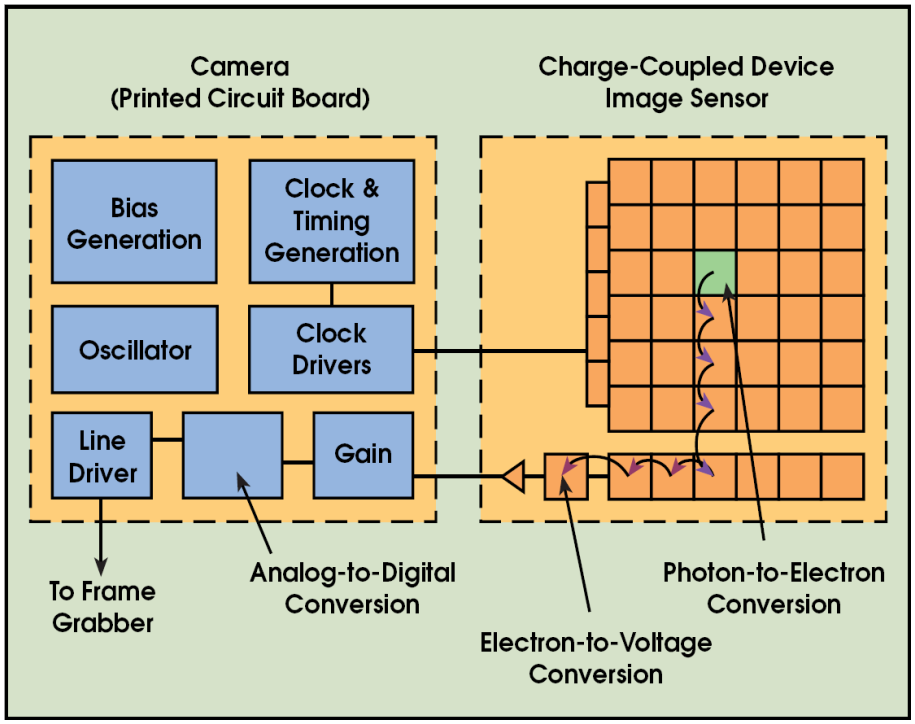


Figure 6.1: Diagrammatic representation of a CCD sensor and support electronics ([82], Fig. 1.).

The nature of the quantised values for each cell depends on the type of CCD. In a monochrome CCD, the values are typically quantized to 8-bits, hence a value from 0 to 255, which represents the light intensity (often called grey-levels). In colour CCDs, there are actually three cells for each grid point. Each cell is sensitive to a different wavelength of light, typically red, blue and green. The quantized values are then three 8-bit values, *i.e.* R (0–255), G (0–255) and B (0–255), giving 16 million colour combinations.

Each grid element sampled is called an image element, picture element or pixel. It should be clear that the amount of data captured for an image depends on the number of grid points. The number is determined by the sampling resolution and is generally fixed for specific CCD. The imager used in the SDDS application has a x- and y- resolution of 1024×1024 (also called the spacial resolution). Since

the imager is monochromatic (effectively only sampling light intensity) and since 8-bit quantisation is performed, a single captured image consists of at most $1,048,576$ bytes.

Recently the use of CMOS sensors has become widespread. A diagrammatic representation of a CMOS sensor is given in Fig. **[6.2]**. CMOS sensors differ from CCD sensors in that the photosites are arranged similar to that of a typical CMOS memory cell. CMOS sensors are therefore more flexible in that selected regions of the sensor can be addressed at will. A CMOS sensor integrates the signal amplifier section, ADC, reset, and select logic into a single chip. CMOS sensors therefore require almost no support electronics and allow considerably more compact implementations.

There are eight attributes which characterise image sensor performance [82]:

- **Responsivity**, the amount of signal the sensor delivers per unit of input optical energy. Due to the ease with which gain elements can be integrated into a CMOS sensor, CMOS sensors are marginally superior to CCDs.
- **Dynamic Range**, the ratio of a pixel's saturation level to its signal threshold. Due to the nature of the photosites, CCDs exhibit a dynamic range approximately double that of CMOS sensors. The nominal noise level on a CMOS sensor is also higher due to the larger volume of on-chip circuitry.
- **Uniformity**, the consistency of response for different pixels under identical illumination. Due to spacial wafer processing variations, particulate defects and amplifier variations, CMOS sensors tend to exhibit non-uniform response. CCDs are less susceptible to such effects and hence produce a more uniform response.
- **Shuttering**, the ability to start and stop exposure arbitrarily. CCDs generally integrate more advance shuttering mechanisms, due to the simpler cell structure. CMOS sensors can achieve the same features at the cost of a number of additional per pixel transistors.
- **Speed**. Due to the high integration of the CMOS sensors, CMOS sensors potentially outperform CCDs by a large margin. The shorter signal trace lengths result in lower inductance, capacitance and propagation delays.
- **Windowing**, the ability to selectively read out portions of the sensors contents. This ability is generally only available in CMOS sensors.
- **Anti-blooming**, the ability to gracefully drain localised overexposure without compromising the rest of the image in the sensor. CMOS devices, by their nature, provide isolation between pixel elements and hence are naturally immune to blooming. CCDs, on the other hand, must be carefully designed in order to minimise the effect of blooming. Low cost, consumer CCDs tend to exhibit significant blooming.
- **Biasing and clocking**. Due to the level of integration found on CMOS sensors, biasing and clocking are trivial as the sensor operates with a single bias voltage and clock level. CCDs generally require a number of higher-voltage biases and traditionally fairly complex clocking circuitry.

In summary, CCDs generally produce higher quality images and are therefore more suitable for high precision, scientific applications (such as optical astronomy). On the other hand, in high speed imaging applications such as this project, CMOS sensors are superior and considerably more cost effective. The image sensor found in the camera used in the SDDS application is a CMOS sensor (refer to Section 3.4)

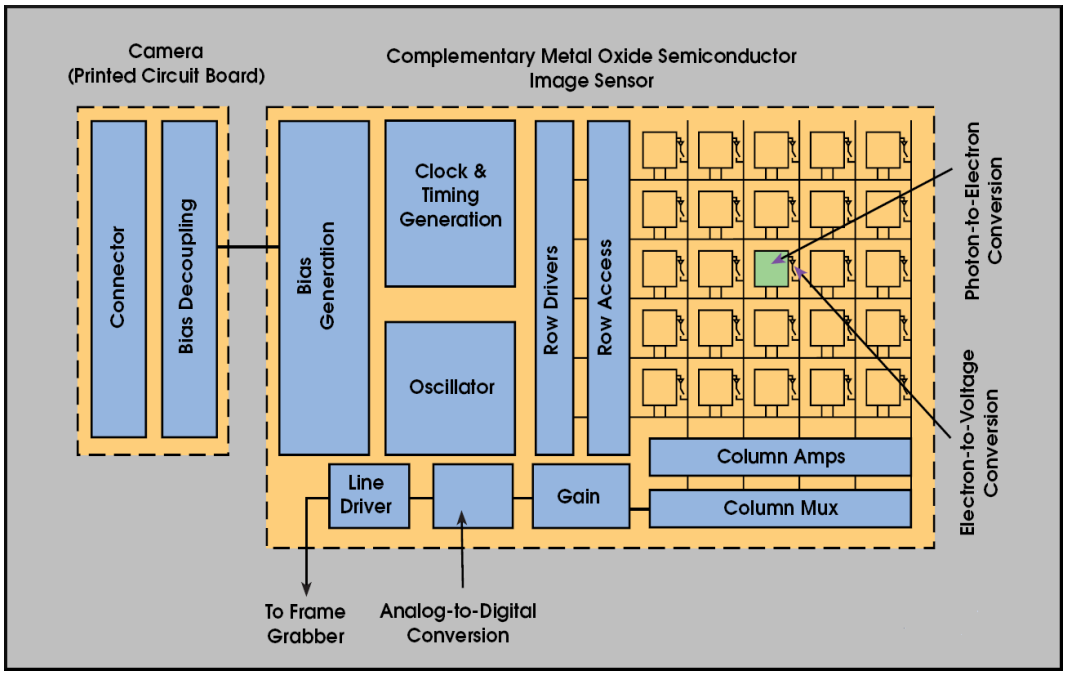


Figure 6.2: Diagrammatic representation of a CMOS sensor ([82], Fig. 2.).

6.4 Camera Software Interface

The `sddscamera` module of the SDDSF encapsulates the framegrabber camera interface. In order to capture images from the camera, a channel via the framegrabber must be opened using the framegrabber library calls. The ROI is typically also set during this initialisation procedure. For the case where a single image is to be captured, the "start capture" library call is executed and the image is captured to a buffer.

However, in the case where the camera must capture images in free running mode (as is the case in the SDDS application), the management of memory is more complex. In order to attain the high data throughput rates required to capture images at the maximum rate, the framegrabber makes use of PCI Direct Memory Access (DMA). DMA is a special memory bus mode where the CPU is isolated from the memory bus and the CPU's Bus Interface Unit (BIU) is placed in a halt state. The memory controller then allows a peripheral to transfer data to / from the memory directly, without the CPU interfering (which would slow down the transfer).

DMA is used by the framegrabber to directly transfer the incoming image to the main memory. To manage the DMA process, the framegrabber driver allocates a number of DMA buffers in main memory. The number of DMA buffers can be configured during initialisation by the application software. The memory controller is then pre-programmed with information about the buffers. As one image transfer completes, the next buffer is automatically selected and DMA transfer to that buffer is initiated. This operation occurs without any interaction from the CPU. The user application can then start processing that image whilst the next image is being transfered via DMA to the next buffer. When the last buffer has been transferred, the first buffer is automatically selected and hence the concept of a ring of buffers (or ring buffers). The ring buffer mechanism therefore operates on the first-in first-out (FIFO) basis.

During initial experimentation with the system, a problem was encountered where images captured

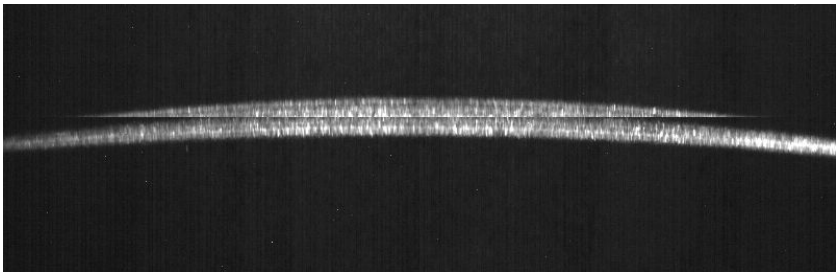


Figure 6.3: Example of a "fractured" image.

were sporadically "fractured". Fig. **[6.3]** gives an example of such an image. The source of this problem was difficult to find given the large number of possible sources: camera cable problems, interference, power supply instability, loose connections in the camera, *etc.* To complicate the debugging of this problem, the occurrence of the effect seemed to be sporadic and not correlated with any particular system event or action.

Detection of this effect in particular images was considered the first step in correlating the effect with system events and activities. This in itself proved to be problematic in that the effect, a row discontinuity, resembles the particular image construct essential to the operation of the system, the approximately horizontal laser line. Many of the obvious filters would therefore be triggered by the laser line itself. Amongst other, a detection mechanism based on image moment invariants was attempted. Since the mechanism is not integral to the final operation of the system, the details will not be described here. Appendix F provides the necessary theory in order to understand the software implementation. After considerable effort, the reliable detection of this effect proved to be surprisingly elusive.

After extensive experimentation with countless techniques from various branches of the field of image processing, an usable technique was eventually developed: 2D convolution with a Laplacian of Gaussian kernel [46], followed by a row aligned differentiation [73, 50, 3]. The implemented technique effectively monitors the second derivative of the intensity profile. As the result of this effort, the SDDSF contains extensive optimised routines to perform various types of spacial image processing [42, 46, 140, 89]: kernel based spacial filtering (Median, Laplacian, Laplacian of Gaussian, Gaussian Smooth, *etc.*), edge detection (Roberts, Sobel, Prewitt, *etc.*) and general 2D convolution. The detection approach is unfortunately a computationally expensive operation, and would be removed once the effect had been isolated and fixed. Subsequent tests indicated that the fracturing effect was occurring approximately once in every ten frames. The problem was therefore more serious than initially anticipated.

After an investigation of the expected causes, the problem was eventually traced to the DMA mechanism. The number of ring buffers proved to be insufficient and the result was that the application was busy processing the image whilst the framegrabber was already transferring the next image into the buffer. Since the DMA process has a higher bus priority than CPU accesses, a new image was copied into the buffer at the "fracture point" in the image. Based on experimentation and a detailed study of the camera and framegrabber documentation, it was found that the minimum number of ring buffers that allows stable operation in the SDDS application is 6.

In free running mode, during which the camera continuously captures images without a specific trigger, the minimum recommended number of buffers is typically 4 [38]. This number allows for one buffer to be receiving data from the camera via DMA, one buffer being processed by the CPU, one buffer being configured for the next DMA operation, and one buffer being held in reserve in case of

overlap. If however disk I/O operations are being performed by the system, as is typically the case in the SDDS application, the interrupted processing of a particular frame may result in delays. These delays sporadically cause an overlap in the buffer being processed.

Experimentation with concurrent low to medium disk I/O, as expected during normal system operation, indicated that 6 ring buffers would be sufficient to prevent buffer overruns. The memory required for the 6 buffers is slightly more than six times the image size due to overheads introduced by the framegrabber library. The 6 ring buffers would therefore require slightly more than $1.2\,MB$, which is negligible given that the server has $2\,GB$ of memory. For an additional safety margin, the SDDSF initialises the framegrabber to use 8 ring buffers, which still translates to less than $2\,MB$ of memory space. There is no particular reason why, except for memory constraints, the number of ring buffers could not be increased further if necessary. Additional software checks have also been added to monitor the ring buffers to detect and handle buffer overruns if they do occur. An appropriate error message is logged in such a case and would serve as an indication that the number of ring buffers should be increased. It would however be advisable to investigate the cause of the buffer overrun as the cause may be indicative of a problem or deficiency in the system.

6.5 Characterisation

All imaging systems exhibit a degree of imperfection. If the nature of the imperfections in a particular system is understood and quantified, steps can be taken to compensate for these imperfections. In most cases the compensation is in the form of post-processing. These imperfections are often called imaging *artifacts*.

The process whereby the imperfections of an imaging system is determined and quantified is generally known as *characterisation*. Depending on the imaging system and its application, this process can be very complex, or relatively simple. Imaging systems used in astrophysics / astronomy generally present the most significant characterisation challenges [63]. These systems must exhibit very high SNR due to the small number of photons they must capture from distant stars and hence are highly sensitive instruments. The result is that even the smallest error greatly effects the resultant output. The following is a partial list of the parameters typically measured for an imaging system:

- **Quantum Efficiency (QE)**. The QE of a photodetector is the ratio of the number of photoelectrons produced to the number of photons incident upon a detector. CCDs have QEs of 50% or greater at optical wavelengths.
- **Amplifier responsivity**. The amplifier responsivity of a sensor is measured in volts per electron and gives an indication as to how sensitive the output amplifier of the sensor is to the photoelectrons produced by the photodiode.
- **Readout noise**. The readout noise of a sensor is the signal measured for no input signal.
- **Dark signal (current)**. The dark signal of a sensor is the charge accumulated within a photosite or "well" in the absence of light.
- **Photo-Response Non-Uniformity (PRNU)**. PRNU refers to the variation in responsivity between pixels due to spacial wafer processing variations, particulate defects and amplifier variations.
- **Full Well Capacity (FWC)**. The full well capacity of a sensor is the number of electrons that can be held in one potential well (charge holding region). It is assumed that all pixels on the sensor have the same well size and that each well can hold the same number of electrons.

- **Charge Transfer Efficiency (CTE)**. As the charge in each pixel is transferred through the system, some loss occurs. The charge transfer efficiency is the fraction of electrons successfully passed to the next position during the row and column readout.

For the SDDS application, such detailed knowledge of the imaging system is not required. Their availability would not significantly improve the functionality of the system as there are other sources of noise, such as the robot motion, which place a limit on the total system signal-to-noise ratio (SNR).

6.5.1 Characterisation Test Procedures

The quantification of many of these parameters requires specialized measurement equipment and a controlled environment. The next sections will briefly discuss tests that can be performed with minimal effort and yet allow the most significant imaging artifacts to be correctly removed. These tests were performed on the camera used in the SDDS application.

Bias Images

A bias image is obtained by ensuring that no light enters the camera by externally shielding the camera, or closing a manual light-tight shutter. The image is captured with an integration time of 0 seconds. This essentially implies that the resultant image represents the influence of the camera electronics and thermal noise in the sensor. Bias images should have a mean pixel value close to 0 and be uniform if the sensor is of suitable quality. For the bias image tests performed for this application, the exposure time was set to 0, the tests were performed at night, the light-tight shutter was closed, the camera was wrapped in a thick black bag, and all artificial lights in the vicinity were switched off.

Dark Images

Similar to a bias image, a dark image is obtained by ensuring that no light enters the camera by externally shielding the camera, or closing a manual light-tight shutter. Dark images should be captured for various integration times. This test captures the effect of defective cells in the sensor matrix, which cause invalid readings. These cells are typically indicative of the presence of above average dark current in the cells. For the dark image tests performed for the SDDS application, various exposure times were selected ($10\,ms$, $100\,ms$ and $250\,ms$), the tests were performed at night, the light-tight shutter was closed, the camera was wrapped in a thick black bag, and all artificial lights in the vicinity were switched off. The exposure times were selected in order for the images to be representative of the possible exposure time range.

Flat Field Images

A flat field image is obtained by placing a diffuser in the optical light path between the camera and a stable light source. A simple diffuser can be created by placing small pieces of unmarked white paper on the camera lens (with the camera facing upwards). Ideally the intensity of the resultant light should be such that approximately $3/4$ full well capacity is obtained using and integration time of approximately 2 seconds. In other words, with an integration time of 2 seconds, the image mean pixel value should be approximately 191. These times are specified for typical digital cameras.

The camera used in the SDDS application does not allow an exposure time of 2 seconds to be set (see Table **{3.2}**). Furthermore, the camera in the SDDS application is used to perform high speed image capture. The response of the sensor with short exposure times is therefore of more importance.

Table 6.1: Camera sensor noise estimated from bias images.

Parameter	Description	Value
$\overline{pv}$	mean pixel value	8.853
σ^2	pixel value variance	19.324
σ	pixel value standard deviation	4.396
median	median	9.000
MAD	median absolute deviation	3.000

For the flat field image tests performed for the SDDS application, the manual light-tight shutter was opened, a simple diffuser was placed on the camera lens whilst the camera was orientated upward, and the artificial lights in the vicinity were switched on. An exposure time of $250\,ms$ was selected after experimentation in order to obtain the target mean pixel value of 191. The measured mean pixel value was 146.974, which was considered sufficiently close to the target.

6.5.2 Processing of Characterisation Test Results

Generally all of the following processing should be averaged over a number of sample images to ensure a representative result. For the characterisation of the camera in the SDDS application, 10 images of each image type (bias, dark and flat field) were averaged.

Sensor Noise

The standard deviation of the pixel values of the bias images represents the estimated sensor noise level. Effectively this defines the expected variation in pixel values and in further processing, deviations of less than this level can be considered noise. For the camera used in the SDDS application, the results are given in Table **{6.1}**. If large numbers of hot pixels and dark pixels are present in the sensor, the standard deviation (4.396 in this case) may not be an unbiased estimate of the actual sensor noise level. In such cases, the Median Absolute Deviation (MAD) provides a more realistic estimate of the sensor noise level (see Section 7.3.4 for a discussion of the MAD).

It is interesting to note that the MAD estimate correlates with the Fixed Pattern Noise (FPN) as specified by the camera manufacturer in Table **{3.2}**. Since the FPN provides an indication of the system noise present when a static image is being captured, there is a possibility that non-zero integration time was configured whilst the bias images were being captured. Therefore, the MAD result potentially indicates that the camera hardware or firmware may prevent a zero integration time setting, even if explicitly set to 0 as was done in this case, and clamp the integration to a small lower limit. From Table **{3.2}**, this observation is confirmed; the minimum exposure time that can be configured is $1\,\mu s$. Based on the measured noise level and the camera specifications, the calculated noise level equivalent to 3 grey-levels is however still a reasonable estimate of the camera sensor noise.

Hot Pixels

A hot pixel is a pixel that is particularly sensitive to thermal noise in the sensor and hence does not accurately respond to external light. These pixels generally appear brighter than the surrounding pixels, which is the reason they are also called bright stars or referred to as salt and pepper noise. The particular sensor used in the camera is of Grade A quality in order to minimize the number of hot pixels.

The hot pixels that are present are normally removed by software. To identify hot pixels, the dark images are post-processed to map all pixels with values which are 2σ, 3σ and 5σ above the mean dark image pixel value. A tabular list of these pixels should be generated and stored in order to allow rapid correction. In the SDDSF, the x and y coordinates of all hot pixels with pixel value larger than 3σ are listed in the file `data\characterisation\hotpixels.cfg`. Based on the characterisation results, there are 7560 pixels listed as hot pixels. The hot pixels therefore represent 0.72% of the available pixels. A map of the hot pixels is given in Fig. **[6.4]**.

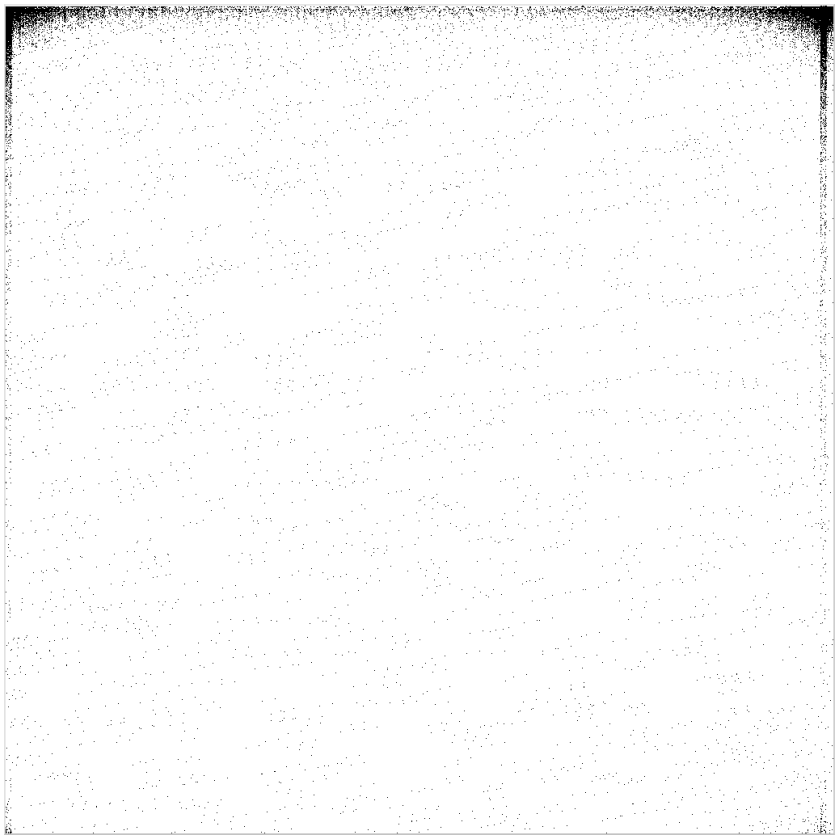


Figure 6.4: Hot pixel map based on characterisation results.

The manufacturer has indicated that future generations of the camera will include hot pixel removal, as well as fixed noise pattern removal, in the camera firmware and thus remove the need to perform this operation in the application software.

Dark Pixels

Dark pixels are produced by sensor pixel elements that do not respond sufficiently to the external light, and are also known as *point defects*. A dark pixel is generally defined to be a pixel with value 80%

below the mean pixel value of an uniformly illuminated device (as recorded in a flat field image). A map should be generated where all correct pixels are set to zero and the pixels with value 80% below the mean should be set to the maximum grey scale value. In the SDDSF, the x and y coordinates of all dark pixels are listed in the file `data\characterisation\pointdefects.cfg`. Based on the characterisation results, there are 8065 pixels listed as point defects. The point defects therefore represent 0.77% of the available pixels. A map of the point defects is given in Fig. **[6.5]**.

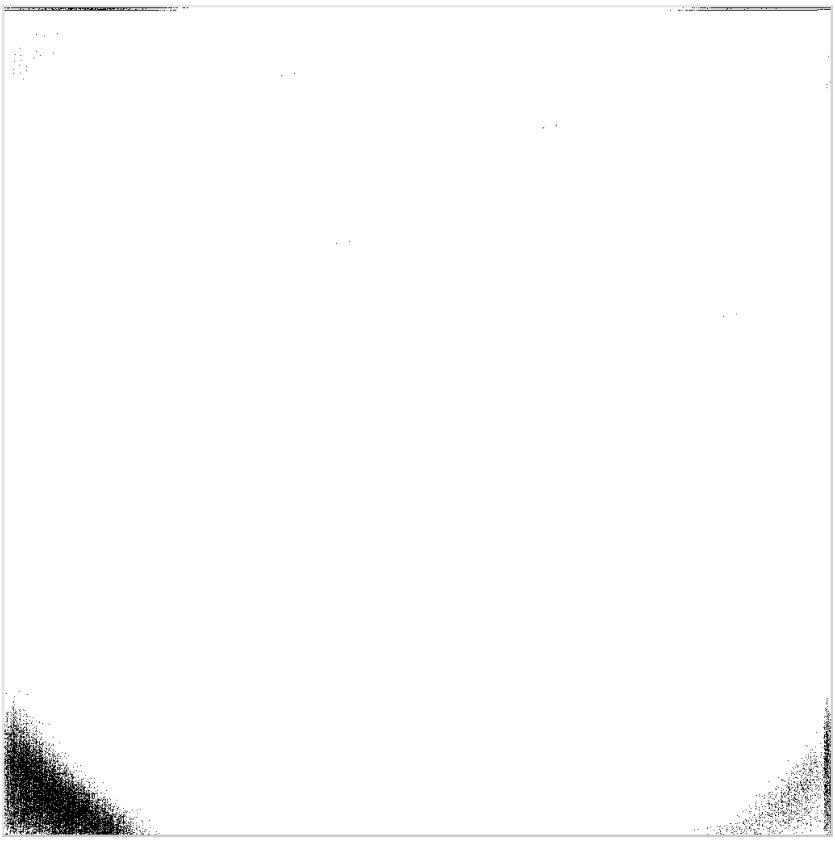


Figure 6.5: Point defect map based on characterisation results.

6.5.3 Artifact Removal

From the characterisation results given above, a total of 1.5% of the available pixels can be considered to be defective. The quantity and nature of the sensor defects correspond with the expected for the class of sensor being used. By using the results of the characterisation tests, it is possible to remove various artifacts from the images. For example, hot and dark pixels can be corrected by setting the pixel values to a predetermined value. Alternatively, the pixel value of a defective pixel element can be replaced by the mean of its neighbours. In the SDDSF implementation, both the hot and dark pixels are corrected by replacing the pixel with the mean of the neighbouring pixels.

Although the hot pixel and point defect maps indicate that the corners of the sensor are problematic, the central region of the sensor appears to be free from significant concentrations of defective pixels. These maps also indicate the presence of manufacture anomalies seen as vertical and horizontal lines near the edges of the sensor. Since the central region of the sensor is the region which is to be used in the SDDS application, the high concentrations of defective pixels in the sensor corners and sensor edges are not significant in this case.

In applications where the use of the entire image is required, the raw image can be corrected for hot pixels by subtracting the dark image of equivalent exposure time. Distortion artifacts can be similarly corrected by dividing the raw image by the flat field image, a process known as *flat fielding*. The flat field image in effect represents the distortion effects present in the imaging system. Correction by means of flat field and dark images is a standard practice in the field of astronomy [63]. Fig. **[6.6]** gives an example of a flat field image captured with the Photonfocus camera. The spherical aberration can clearly be seen in the image, as well as the hot pixels. The dark edges caused by the spherical aberration is also known as *vignetting*.

As mention in Section 6.2, the camera ROI is used to discard the spherical aberration at the edges of the image. In other words, instead of using flat field image based compensation for sensor distortion, the camera ROI mechanism is used. Empirically it was found (confirmed by a study of the laser data sheet [130]) that the projected laser line exhibited length-wise non-uniformity, particularly at the ends of the line. Flat field image processing would only compensate for camera artefacts, whereas the use of the camera ROI compensates for both camera and laser artefacts simultaneously.

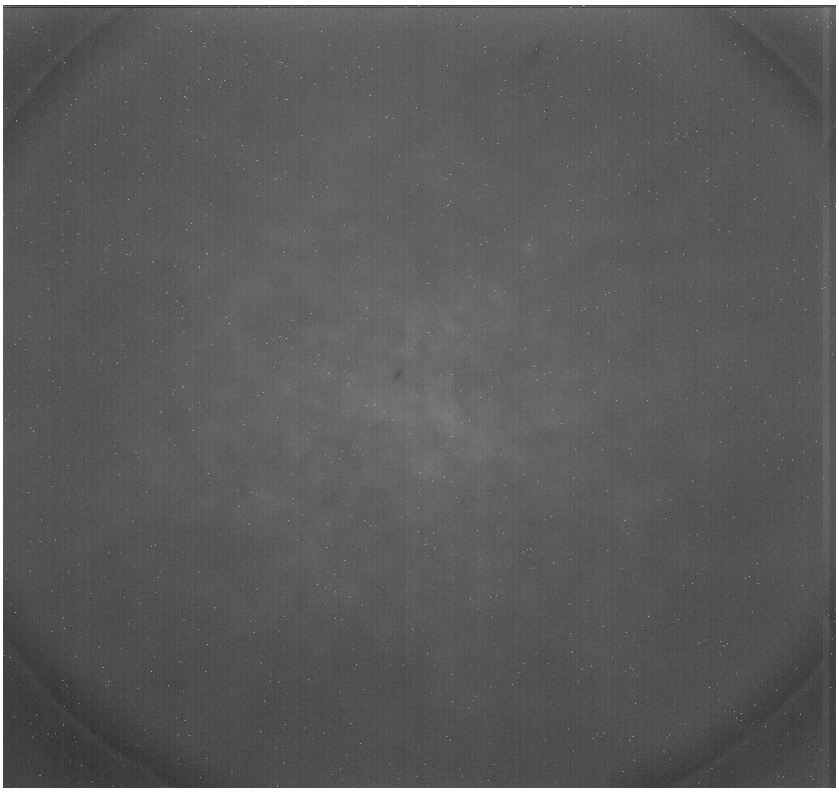


Figure 6.6: Example of a flat field image.

6.6 Background Removal

The ambient light levels in the vicinity of the operational system affects the images captured by the camera in various ways. Laser light reflected from the panel being scanned will appear as a bright line. The remainder of the image will be formed by ambient light reflected from the panel. This results in a background that appears to be dark grey, but not necessarily uniform across the image. Many of the image processing operations that could be performed are affected by this background illumination.

6.6.1 Characteristics of Captured Images

Fig. **[6.7]** gives an example of a typical captured image. The laser line can clearly be seen. For the SDDS application, the term "background" is defined to be all image aspects excluding the laser line and possibly noise. At first glance, the background appears to be black in the example image. However, examining the pixel values shows that the background pixels are not zero (black).

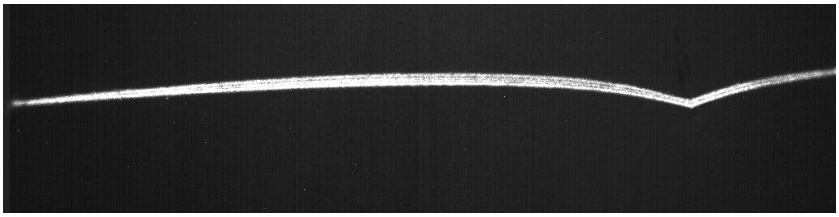


Figure 6.7: Example of typical image frame.

When studying the pixel values of an image the image histogram is an useful tool. An image histogram is a discrete function $h(r_k) = n_k$, where r_k is the k-th grey-level and n_k is the number of pixels in the image which have that particular grey-level. The histogram therefore gives an indication of the relative frequency of a particular grey-level in the image.

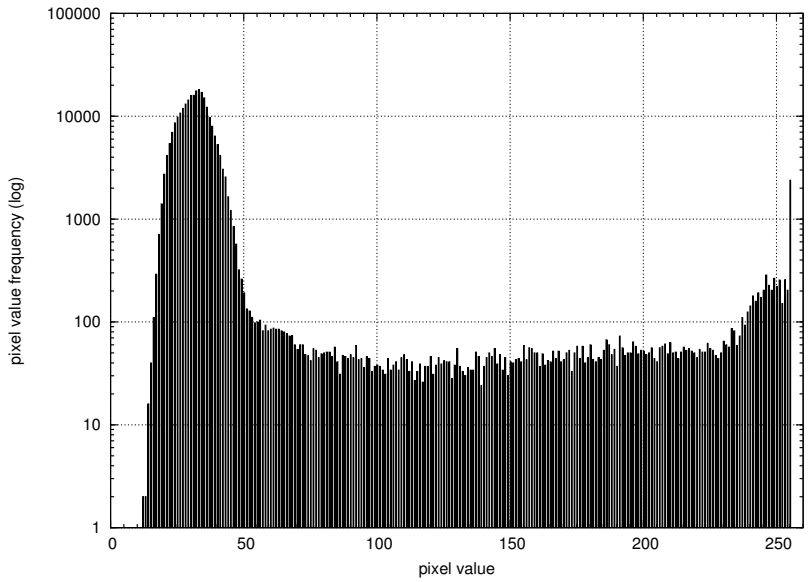


Figure 6.8: Image histogram of Fig. **[6.7]**.

For the image shown in Fig. **[6.7]**, the corresponding image histogram is given in Fig. **[6.8]**. From the histogram, three observations can be made. The first observation that can be made is that there are a large number of pixels which have a pixel value in the region of 30. These represent the bulk of the image (note that the frequency count is shown in log scale). The second observation is that a significant number of pixels have a pixel value in the region of 255. The last observation is that there are a nearly uniform distributed number of pixels which have pixel values between 30 and 255. Table **{6.2}** supplies the statistical information for the typical image (such as Fig. **[6.7]**).

Table 6.2: Statistical information from a typical image.

Parameter	Description	Value
$\overline{pv}$	mean pixel value	41.336
σ^2	pixel value variance	1607.388
σ	pixel value standard deviation	40.092
mode	most frequent pixel value	34.000
mode variation	absolute deviation from mode	13.397
median	median	33.000

The first observation identifies a large number of pixels with a relatively low intensity. A visual examination of Fig. **[6.7]** would indicate that these pixels are representative of the "black" or background pixels as these clearly represent the bulk of the image. Examining the pixel values[1] of randomly selected pixels in the background regions of Fig. **[6.7]** verifies that these pixels have pixel values in the region of 30. The second observation, pixels with value in the region of 255, would then correspond to the laser line pixels, and can be verified as for the background pixels.

The cause of the last observation is not obvious. Noting that the are practically zero pixels of values $0 - 10$, one possible cause of the third observation is noise. Noise would imply that a pixel value of 0 has a low probability of occurrence and an indication of this is the non-zero mean pixel value seen in Table **{6.1}**. Irrespective of the cause of the third observation, for the purposes of the SDDS application, the contributing pixels will be considered to be unwanted noise.

6.6.2 General Techniques for Background Removal

Background identification and / or removal is by no means unique to this problem. Over time, many techniques have been developed to deal with this issue. This section will investigate some of these techniques and discuss the possible value of each with regards to the SDDS application.

Background Subtraction

Background subtraction [8] entails the literal subtraction of an estimate of the background from an image. A background image $f_b(\mathbf{x})$ is therefore approximated and subtracted from the original image $f(\mathbf{x})$ to give a new image $f_n(\mathbf{x})$ by

$$f_n(\mathbf{x}) = f(\mathbf{x}) - f_b(\mathbf{x}) \tag{6.6.1}$$

In many applications, the background remains essentially constant and the varying foreground is the focus of the image processing. An example of such an application is the automated analysis of images captured from a security camera. The background would effectively remain constant and a template background image, $f_b(\mathbf{x})$, can be captured under controlled conditions. An image captured when a person has moved into the camera field of view would then be processed by subtracting the template

[1] This can be done by using the "colour picker" tool in most image editing software.

background. The foreground, the person, would then be clearly delimited. For applications where the background does not remain constant (static), the template approach can not be used.

In applications where the background changes over time, a dynamic estimate of the background, $f_b(\mathbf{x})$, must be generated. In such cases, the background image can potentially be obtained as a low-pass filtered version of the original image. For a comprehensive discussion of frequency based filtering, of which low-pass filtering is an example, please refer to Chapter 4 of [46]. However, such a background image would not account for local variations of the background in the original image. Such local variation in background would exhibit frequency characteristics that lie outside the pass-band of the filter and hence be rejected. Another approach that can be used to generate a dynamic estimate of the background would be to create an adaptive background image by segmenting the original image into a number of blocks. The effective background is then estimated for each block using low-pass filtering. The complete background image is then approximated as a spline interpolation of the estimated background for each block.

Due to the well known duality between frequency domain multiplication (such as performed by a low-pass filter), and convolution in the spacial or time domain [46, 93, 108, 17, 18, 73, 106], a direct method to obtain a similar result to the application of a low-pass filter is to convolve the image with a smoothing convolution kernel. Fig. **[6.9]** shows the result when a 3×3 median kernel [46] smoothed version is subtracted from the original image, Fig. **[6.7]**. Note that the image contrast has been enhanced, as discussed next. Clearly both the frequency domain and the spacial domain approaches are computationally expensive.

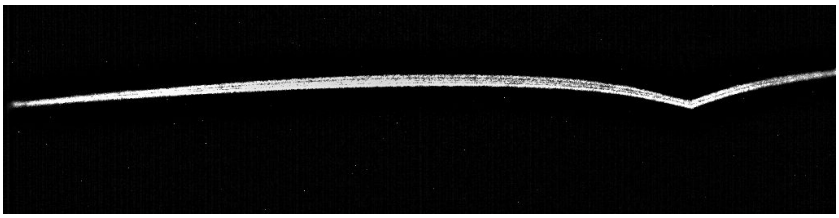


Figure 6.9: Original image with kernel smoothed version subtracted.

From Eq. **(6.6.1)** it can be seen that each pixel value is effected by the subtraction. Generally, the resultant pixel value is lower. Visually, the result is that the image brightness is decreased. From a histogram perspective, the histogram of the image is effectively shifted downwards (see Fig. **[6.10]**). In other words, lower pixel values become more frequent. The resultant histogram is now biased towards the lower pixel values and the range of the histogram is reduced (*i.e.* few pixels with high pixel values now occur). It is desirable to increase the image contrast to allow the relative intensity of the laser line to be increased. The pixel value modification performed is defined by,

$$pv_n(\mathbf{x}) = 2^d \, \frac{pv(\mathbf{x}) - pv_{min}}{pv_{max} - pv_{min}} \tag{6.6.2}$$

where $pv_n(\mathbf{x})$ is the new pixel value, $pv(\mathbf{x})$ is the current pixel value, pv_{min} and pv_{max} are the minimum and maximum pixel values respectively of the image, and d is the image depth. In the SDDS application, 8-bit grey-scale images are used exclusively, and thus $d = 8$. The corresponding histogram of the resultant image is given in Fig. **[6.10]**.

In summary, background subtraction can be used to isolate an image foreground. An issue with this approach is the computational expense of image convolution, or 2D Fourier transforms (to perform low-pass filtering). Benchmarking of image convolution with a 3×3 median kernel indicates that approximately 21 complete image convolutions per second can be performed on the reference machine.

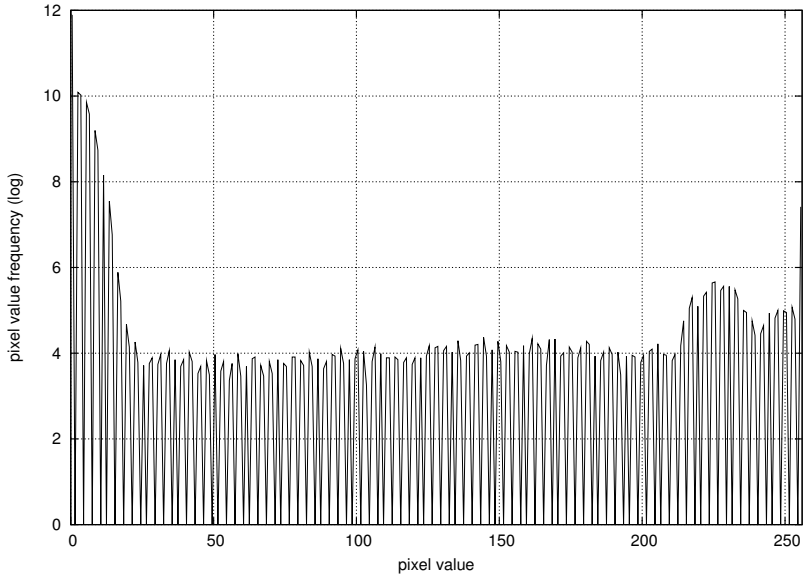


Figure 6.10: Image histogram of Fig. **[6.9]**.

For this approach, a larger kernel will be required, resulting in even lower performance. Similarly, benchmarking of image low-pass filtering indicates that approximately 5 complete image low-pass filtering operations per second can be performed on the reference machine. The image subtraction must then still be performed, contributing additional computation.

Comparing this with the target capture rate of at least 773 frames per second, it is clear that neither of these two approaches are feasible. Apart from the performance aspect, the unsatisfactory handling of noise pixels is also a disadvantage of this approach. The histogram in Fig. **[6.10]** shows that there are a significant number of pixels values with low frequency. These pixels add no information to the image and yet will typically require further processing.

Morphological Techniques

These techniques are derived from mathematical morphology [84, 121, 9], the basis of which is set theory and the associated operators. The field of image morphology is extensive and, for the purposes of this investigation, two specific operators were the focus: dilation and erosion. For a comprehensive discussion of these operators the reader is referred to Section 9.2 of [46]. Additional experiments using skeletonisation [126, 113] were also performed. These approaches however proved to aggressively (by their nature) discard or amplify boundary information from the laser curve (see Fig. **[6.11]** and Fig. **[6.12]**). In particular, it was found that the intensity information of the pixels was being ignored by these techniques. From a visual examination of the resultant images it was clear that the intensity information was crucial in order to obtain the correct estimate of the essential laser curve required by the second stage image processing.

Histogram Techniques

For this discussion, the primary aim has been to investigate the image background. As see above, the image background appears to be distinguishable based on the pixel values. The question is now: is the histogram in Fig. **[6.8]** truly representative? A crude quantitative test would be to obtain an average image histogram for a complete scan across the target panel. Another approach would to be obtain an

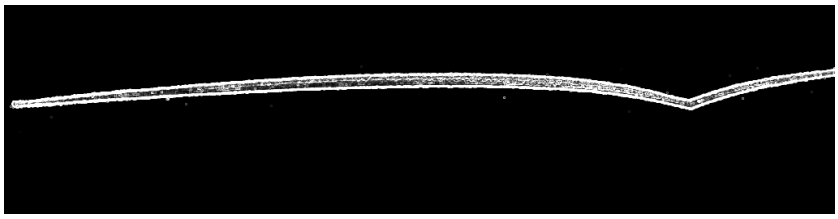


Figure 6.11: The result of Sobel edge detection applied to Fig. **[6.7]**.

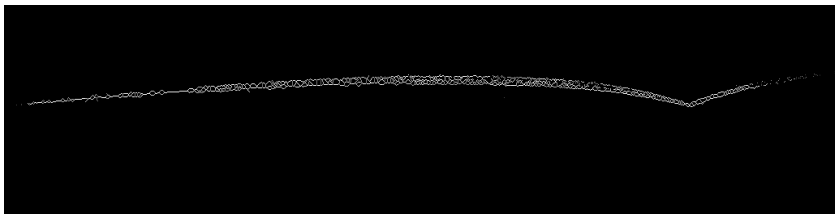


Figure 6.12: The result of skeletonisation applied to Fig. **[6.7]**.

accumulative image histogram for all the image frames captured (refer to Fig. **[6.13]**).

From this figure, it is apparent that the first two observations made in the previous histogram are still visible. The conclusion is that the image histogram does provide a view of the image in such a way that the background is distinguishable from the laser line. The ideal would be to correctly classify the histogram data based on the three classes identified: background, noise, laser. The pixel values which do not fall within the "laser" class can then be removed (made 0). The effect on the image should then be that the background and noise in the image are effectively removed.

To remove the background, a naive approach would be to simply threshold the image: Let T be a threshold value midway between the two histogram peaks. To effectively remove the background, all pixel values below the threshold T are removed. This could be done by setting these pixel values to 0. Fig. **[6.14]** shows the result of performing this operation on the image in Fig. **[6.7]** (the threshold is calculated as $T = 144$).

Histogram Clustering

The goal of clustering algorithms is to partition a set of observations into groups so that members of the same group (cluster) exhibit greater similarity as compared to members of other groups [55, 69, 35, 110]. In the SDDS application, clustering would potentially allow the background pixels to be automatically identified. Initial attempts at clustering the histogram data using the K-median clustering algorithm [71], also known as K-medoids clustering, were not successful. The choice of clustering algorithm was driven largely by simplicity of implementation and insensitivity to outliers. The initial cluster centroids were initialised to be at 25%, 50% and 75% of the pixel values, which translates to pixel values of 63, 125 and 191 respectively. The values where chosen based on the pixel value distribution seen in Fig. **[6.8]**. Fig. **[6.15]** gives the result of a histogram clustering operation. It is clear that the large differences in frequencies was causing the cluster means to be attracted to pixels of large, medium and low frequency. The laser line pixels were therefore grouped together with the noise pixels in a single class/cluster.

To mitigate the effect of the large frequencies, the histogram frequency axis was rescaled (replaced

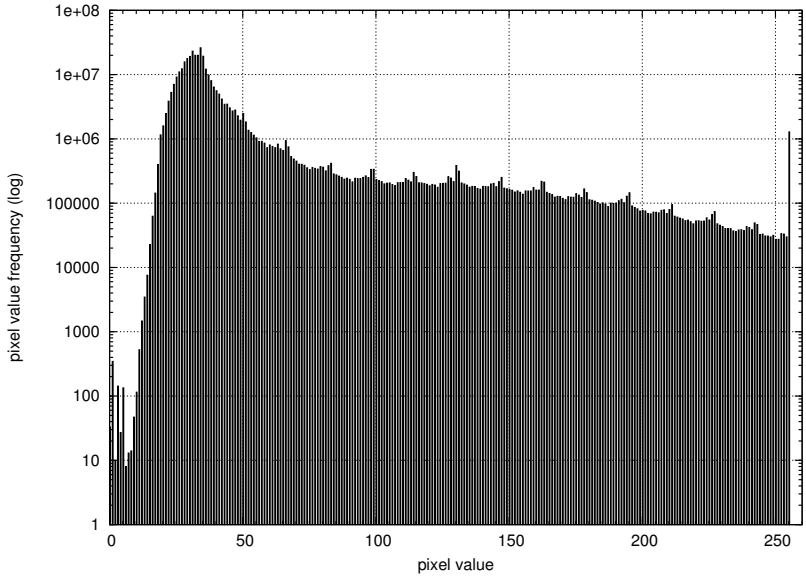


Figure 6.13: Accumulative image histogram for 1647 frames.

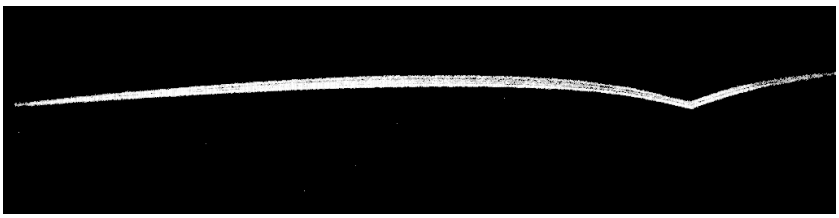
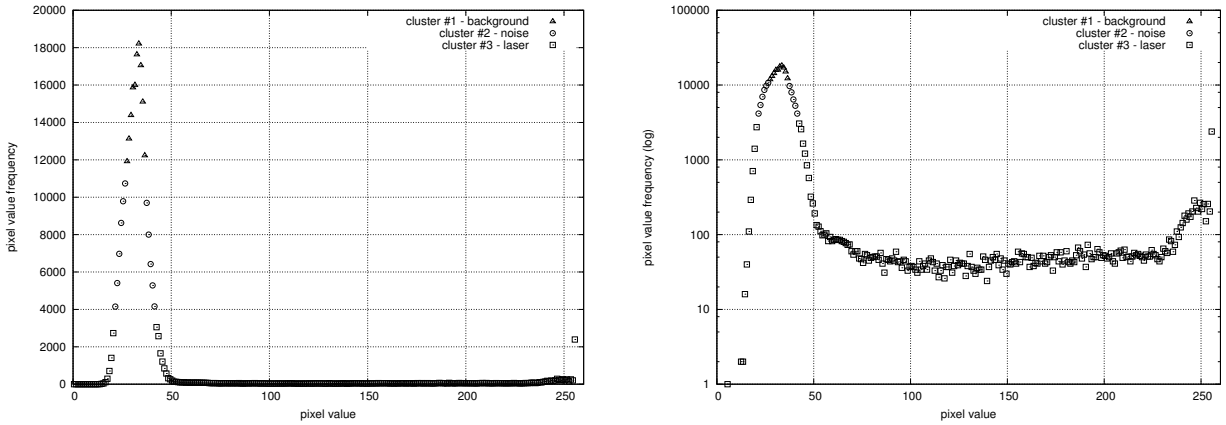


Figure 6.14: Threshold version of Fig. **[6.7]**.



(a) Normal scale

(b) Log scale

Figure 6.15: Result of histogram clustering.

by the natural logarithm), resulting in a dramatic reduction in the relative differences in frequencies. Fig. **[6.16]** gives the result of the cluster operation. The result of the rescaling has allowed the his-

togram to be clustered approximately according to the desired classes. However, the "background" and "laser" classes do contain pixel values which may be considered to be noise pixel values.

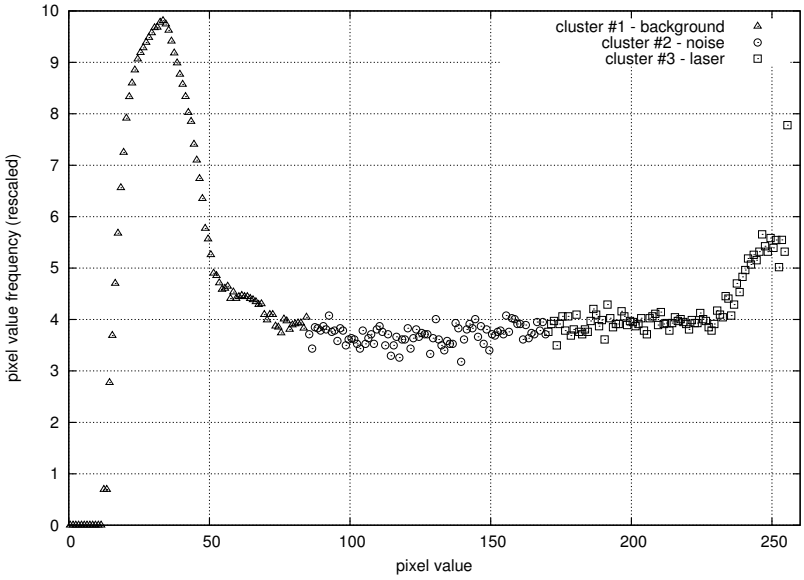


Figure 6.16: Result of histogram clustering (rescaled).

The question was now whether or not the clustering approach would be stable for a wide variety of captured images. To evaluate this aspect, a number of randomly selected anomalous images were analysed and the results studied. Amongst other, images that contained no laser line, bright highlights, *etc.* were examined. Table **{6.3}** gives an extract of the centroid data collected. The results clearly show that the clustering approach produces clusters that exhibit stable centroids. The resultant cluster members were representative of the expected and desired values. The clustering approach was therefore a promising solution to the problem of background identification.

However, during the empirical evaluation an aspect that become particularly noticeable was the computational expense of this approach. Therefore, attention was diverted to the analysis and optimisation of the implementation. It soon became clear that the computational costs would make this approach infeasible and any further investigation into a clustering based approach was abandoned.

6.6.3 Alternative Techniques for Background Removal

In complex images, the difference in texture between the foreground and background can be used to identify the foreground, using *e.g.*, a monotonic tree [123]. A monotonic tree is a rooted tree of monotonic lines, which represent digitized contours in the image. These monotonic lines represent boundaries where the image assumes higher or lower values in the pixels adjacent to the boundary from inside the boundary than those from outside the boundary. Details regarding the computational aspects of this algorithm are sparse. Intuitively it seems that locating the monotonic lines would require numerous transversals of the image, indicating a significant computational cost. Techniques based on texture were therefore not empirically evaluated.

Table 6.3: Cluster centroids.

Sample #	Cluster #1	Cluster #2	Cluster #3
1	40.5, 3.456	125.5, 7.784	213.0, 5.568
2	39.5, 3.482	121.5, 7.238	210.0, 2.555
3	39.5, 3.343	121.5, 7.257	210.0, 2.538
4	43.0, 5.056	128.5, 5.168	213.5, 3.903
5	43.0, 5.978	128.5, 5.231	213.5, 3.870
6	42.5, 5.063	128.0, 3.618	213.5, 3.688
7	42.5, 5.259	127.5, 0.232	213.0, 0.000
8	42.5, 5.147	127.5, 3.700	213.0, 4.264
9	42.5, 5.011	128.0, 4.756	213.5, 3.574
10	42.5, 4.662	127.5, 3.904	213.0, 4.462

6.6.4 The Implementation of Background Removal in the SDDSF

The final implementation of background removal that proved to be robust and exhibit acceptable performance makes use of histogram techniques. The implementation estimates the pixel value threshold which defines background pixels. The mode (most frequent pixel value) of the histogram is used to determine the threshold. The absolute difference between the mode and the rest of the pixel values is calculated to determine the spread of the histogram. The absolute difference is used instead of the normal squared difference as it is less sensitive to outliers in the data. In this case there are expected outliers at high pixel values: the laser curve pixels. The threshold is then calculated to be a weighted combination of the mode and the calculated absolute difference. The weighted combination was empirically derived to provide an estimate of the upper side of the background pixel distribution as seen in the image histograms.

Before the background removal is performed, the mode of the histogram is checked to ensure that it is within acceptable limits. For the implemented background removal algorithm, the acceptable limit has been chosen to be $\frac{2}{3}$ of the pixel value range. This limit was chosen after an examination of a typical image indicated that pixels at the edge of the laser line have pixel values as low as 200. The limit of 170 was considered to provide a sufficient margin, given that the nominal background pixel value is in the region of 30 – 40. If the mode exceeds this limit, the ambient light intensity is considered to be unacceptably high and the operating environment is considered outside specification. Under such conditions the routine flags an error condition to the caller, the particular image is rejected, and a new image is captured. If the threshold is however within limits, standard image thresholding is performed. The result for a typical image is shown in Fig. **[6.17]**. The corresponding image histogram is shown in Fig. **[6.18]**.

In order to test the adaptability of the developed background removal algorithm, the brightness (average image intensity) of the typical image (Fig. **[6.7]**) was artificially increased using an image editing application. The artificially brightened image is shown in Fig. **[6.19]**. The mean pixel value of the new

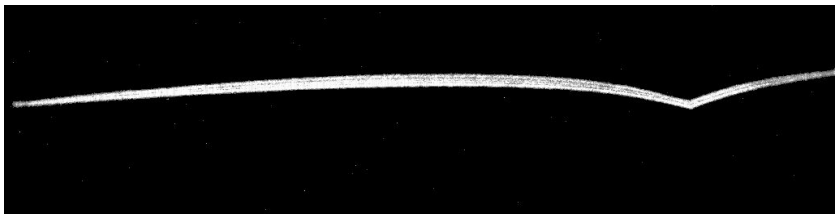


Figure 6.17: The typical image Fig. **[6.7]** after background removal.

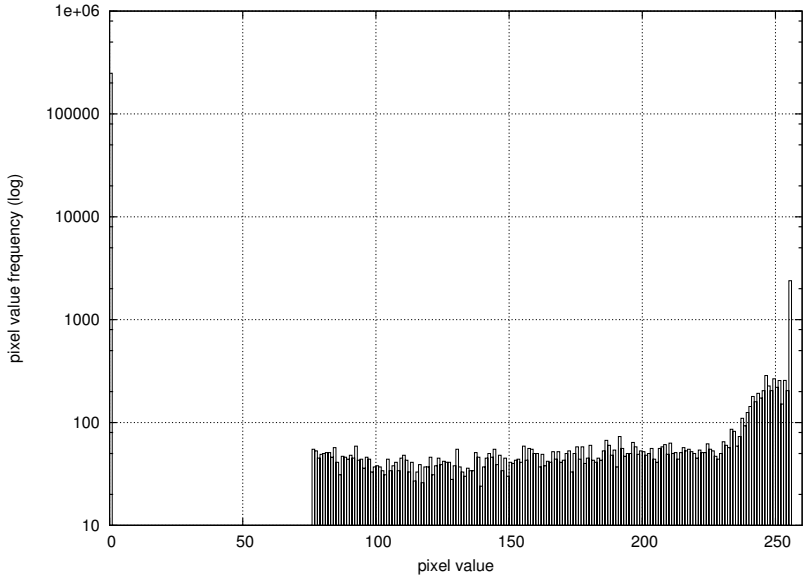


Figure 6.18: Image histogram of Fig. **[6.7]** after background removal.

image was 86.493, compared with a mean pixel value of 41.336 for the original version of the typical image. The background removal algorithm successfully removed the background to produce the image shown in Fig. **[6.20]**. Images of higher mean pixel value were rejected with a return code which indicated that the background intensity was considered unacceptably high. Based on this evaluation, the background removal algorithm is considered sufficiently robust against fairly large variations in background illumination.

6.7 Laser Curve Accentuation

Subsequent to the removal of the image background, the next step is to isolate the "essential" laser line. The primary aim is to obtain, for each image column, the best estimate of the laser line position. Depending on the light conditions during the moment the image was captured, there may be anomalies in the images, such as specular highlights. Although it would be ideal to configure the environmental conditions to avoid such interference, the system must be designed to operate under less than ideal conditions. In addition, due to the angle of the laser, large dust particles may be illuminated even when they are at a distance from the laser line on the roof panel. In such cases, the captured image will contain bright spots with a diameter of 2 or more pixels. This effect is illustrated in Fig. **[6.21]**. The assumption is therefore that the image processing should isolate the laser line whilst being insensitive

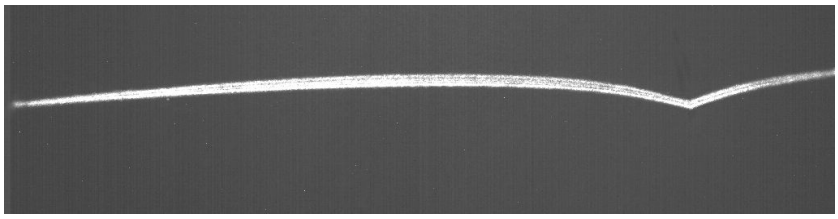


Figure 6.19: The typical image Fig. **[6.7]** after an artificial intensity increase.

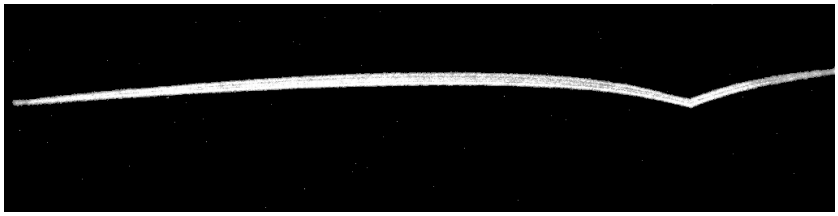


Figure 6.20: The brightened typical image Fig. **[6.19]** after background removal.

to other medium to high intensity effects in the image.

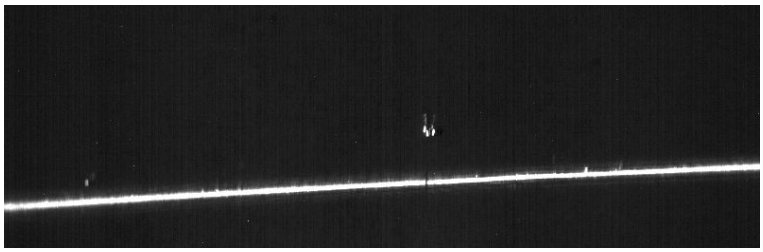


Figure 6.21: Example of an image frame containing a large dust particle.

In Section 2.2 it was shown that the effect of the reflection of the laser from the panel surface is a near uniform attenuation. Hence the intensity distribution of the captured laser curve image is determined primarily by the light emitted from the laser. The manufacturer of the laser states [130] that the projected laser line exhibits a width-wise Gaussian distribution. This characteristic can therefore be used to isolate the laser curve from other high intensity effects in the image.

Essentially, the laser curve is isolated by applying a Gaussian attenuation function to each image column with the center of the function positioned at the estimated laser line. The Gaussian attenuation function is given by,

$$w(y) = e^{-(y-y_0)^2/2\sigma^2} \tag{6.7.1}$$

where σ^2 is an estimated variance of the image and y_0 is the estimated centroid of the laser line for the particular column. These values will be discussed in more detail below. Since the attenuation operation must be performed for every pixel in the image, the computational expense of the exponential function accumulates and presents a problem. This performance bottleneck was found during profiling of the stage 1 image processing code. The exponential function above is therefore pre-sampled to an array of 256 floating-point numbers, `alut`. The index of the array represents the delta $y - y_0$.

Algorithm 6.7.1 ACCLINE

1. Calculate the block size based on the number of blocks requested per image.
2. For each block:
   (a) Calculate a normalised average column histogram of all the columns in the block.
   (b) Find the mode (maximum bin) of the histogram. The mode of each block is stored for future use.
   (c) Using the mode, calculate the average weighted absolute difference of the estimate of the laser line width. The calculation is based on the variance incremental update formulation [133] and is given as

$$w_{k+1} = w_k + h_k, \quad ad_{k+1} = ad_k + |k - m_b| \frac{h_k}{w_{k+1}} \tag{6.7.3}$$

   where w is the weight accumulator, h is the histogram, ad is the absolute difference accumulator and m_b is the mode of block b.
   (d) If the block mode is centrally situated, the global image mode and absolute difference are updated.
3. Examine the number of valid blocks (block with mode centrally situated), and if there are insufficient valid blocks, abort the algorithm with an appropriate error indication.
4. Check that the global image mode indicates that the image line is within the central region of the image. Abort with an appropriate error indication if the estimate line is near the edges.
5. Check the estimated line width (global image absolute difference) and if the width is larger that the valid line width, abort the algorithm with the appropriate error indication.
6. The sigma of the Gaussian attenuation function is estimated based on the average image absolute difference calculated above.
7. For performance reasons, the Gaussian attenuation function is pre-sampled to a lookup-table (LUT).
8. For each column in each block:
   (a) Calculate the estimated line center using bi-linear interpolation [6, 106, 3] of the mode blocks surrounding this block.
   (b) Attenuate the pixel value based on the attenuation LUT as determined by the distance of the pixel from the estimate line center.

This is the reason for the number of elements: the y resolution of the captured image is 256 and hence represents the maximum possible delta value. To apply the attenuation function to a particular pixel intensity, P_{xy}, direct multiplication is then possible to obtain the new pixel value P'_{xy}:

$$P'_{xy} = P_{xy} \cdot \texttt{alut}[|y - y0|] \tag{6.7.2}$$

Alg. $\lfloor$**6.7.1**$\rfloor$ describes the algorithm used to accentuate the laser curve in the image. The algorithm is adaptive in the sense that it processes the image in a number of vertical blocks. Gradual changes in the laser line position are therefore handled correctly. This adaptation is required in order to prevent the filtering from disrupting the laser curve when the curve is distorted by surface damage. This algorithm can be described as follows:

For each block, the algorithm determines the average mode (the y coordinate of highest light intensity). In addition, the average absolute difference of the pixels relative to the block mode is determined. This gives an indication of the width of the line. Once all the blocks have been processed, the algorithm checks that the laser line lies within an acceptable region in the image (*i.e.* not too close to the edges). The estimated average line width is also checked to ensure that the line is not overly dispersed (which effectively adds noise to the estimated laser line). A Gaussian attenuation function

is calculated for each block such that the peak is centered at the estimated line center. The pixels away from the line center are then exponentially attenuated to 1.125 times the estimated line width. The pixel values further away are forced to 0. The limiting value of 1.125 was chosen empirically to ensure that the laser line itself is not affected. The estimated line center is calculated as a bi-linear interpolation of the neighbouring average block modes [6, 106, 3]. The result of the laser curve accentuation for Fig. **[6.21]** is shown in Fig. **[6.22]**. As can be seen, the dust particle effect has been successfully removed whilst leaving the laser curve essentially unaffected.

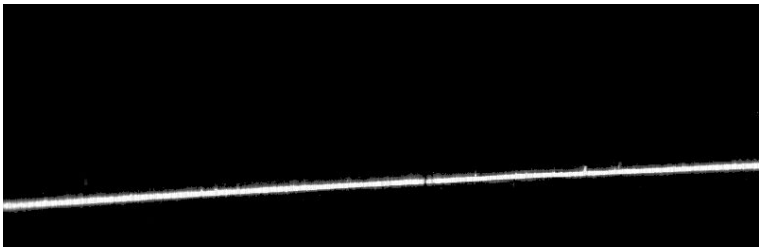


Figure 6.22: Result of the application of the ACCLINE algorithm on Fig. **[6.21]**.

6.8 Laser Curve Extraction

Once the laser curve has been isolated, for each column the best estimate of the center of the laser curve must be found. Under ideal conditions, the reflected laser curve would exhibit a width-wise Gaussian distribution function with respect to intensity. If, however, there is a defect present, the resultant curve may exhibit a distorted Gaussian intensity distribution. The intensity information is therefore important when estimating the per column laser curve position. By obtaining the pixel value centroid (first moment) [46, 133, 73] of each column as the estimate of the center of the laser curve, maximal use is made of the intensity information of the laser curve. The estimated laser curve therefore consists of an array of 800 values representing the centroids of each column. This array is the output of the first stage processing.

6.9 Chapter Summary

This chapter has discussed the first stage of the processing pipeline as implemented in the SDDSF. A brief overview of the optical mechanisms relevant to the SDDS application was provided, together with an overview of the operation of the image sensor. The characterisation of the image sensor is described, which provides information allowing the compensation for image artefacts induced by the image sensor. An adaptive mechanism was discussed which allows the SDDSF to remove the unwanted background information from capture image. The chapter was concluded with a discussion of the method used to extract the best estimate of the essential laser curve position in the captured image.

The following chapters will describe the second stage of the processing pipeline, which analyses the extracted laser curve in order to detect deviations from an ideal model of the laser curve.

CHAPTER 7

Stage 2 Processing

Once the laser curve has been extracted from the processed image, the resultant set of 800 estimated laser curve centroids must be processed in order to detect defects. This data set will be referred to as the *extracted laser curve* data set. In Chapter 2, the defect detection mechanism was defined to be the deviation of the laser curve from the expected curve when no defect is present. This chapter will discuss the *expected laser curve* model and the method used to detect defects (deviations).

7.1 Chapter Outline

The first part of this chapter will address the expected laser curve model. The first section provides a brief overview of the data model, using a simple linear model as example. The method of least-squares, commonly used for model estimation, is discussed. Subsequently, various aspects such as model fitting performance and selection of the model are investigated. Section 7.3 is devoted to an investigation into robust statistics, which allows fitting of a model to data when the data contains errors or noise. Such an investigation is necessary as the dust and defects on the roof panels distort the projected laser line. The second part of the chapter will discuss the Kalman filtering performed in the SDDSF. The Kalman filtering aims to smooth the extracted laser curve vector in order to remove system noise. Finally, the generation of the output of this stage is discussed: an 800 element vector of deviations.

7.2 The Expected Laser Curve

The expected laser curve is effectively the projection of a line segment onto a NURBS surface. The NURBS surface is defined in the CAD model of the panel and represents the outer surface of the panel under test (refer to Section 5.4.5). If the projected laser line extended across the entire panel, the best possible model of the resultant laser curve captured by the camera would be a NURBS curve obtained as the intersection of the NURBS surface and a plane coincident with the laser light.

Since only a small line segment is projected onto the panel, the resultant laser curve represents a fractional NURBS curve. This laser curve has a specific number of *degrees of freedom* (*i.e.* the number of free parameters), which is not known and cannot be directly obtained. Intuitively, a single segment of a NURBS curve would in all probability have a higher number of degrees of freedom than the resultant fractional NURBS curve. Therefore, if the extracted laser curve were to be approximated with a single NURBS curve, the result would be the *over-fitting* of the estimated laser curve. Empirically, this has been found to be the case, particularly when the extracted laser curve contained a potential defect.

7.2.1 Model Fitting of the Expected Laser Curve

Before investigating possible models for the extracted laser curve, the method of fitting the models to sample data (known as regression) will be discussed. The processes of fitting a model to data is also known as estimation, and the fitted model an estimator. There are numerous methods for fitting models to data and developing estimators [73, 143]. This section will systematically describe the particular approach followed in this project.

The Simple Linear Model

To simplify the initial discussion, a simple linear model is assumed. If a data set is available that contains a predictor variable x and a corresponding response variable y, the task is to determine the linear model which best describes the relationship between x and y for all data pairs in the data set. The data set with N samples will have the form

$$(y_1, x_1), \ldots, (y_i, x_i), \ldots, (y_N, x_N)$$

where x_i is the i-th data sample, $y_i, x_i \in \mathbb{R}^1$. If the samples data set contains no noise or errors, the samples will fit an ideal linear model exactly. The ideal linear model is given by

$$r(x) = a_0 + a_1 x$$

where the constants a_0 and a_1 are known as the model parameters. Typically, the data set contains noise or error ϵ and thus the sample response variable y is related to the sample predictor variable x, assuming the linear model, by

$$y_i = a_0 + a_1 x_i + \epsilon_i$$

for each sample i in the data set. The estimated values of the parameters a_0 and a_1 which best describe the relationship between y and x in the non-ideal data set are denoted $\hat{a}_0$ and $\hat{a}_1$. The non-ideal linear model is then

$$\hat{r}(x) = \hat{a}_0 + \hat{a}_1 x$$

The quality of the linear model is determined by how well the model predicts values in the data set (and potentially new samples). The predicted values are $\hat{y}_i = \hat{r}(x_i)$. The difference between the predicted and observed values, known as the *residuals*, approximation error or observation error, are then given by

$$\hat{\epsilon}_i(\mathbf{a}) = y_i - \hat{y}_i$$

A common measure of the "goodness of fit" is given by the *residual sums of squares* (RSS),

$$\mathrm{RSS} = \sum_{i=1}^{N} \hat{\epsilon}_i(\mathbf{a})^2 \tag{7.2.1}$$

$$= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{7.2.2}$$

where N is the number of observations or samples in the data set. To compare different estimators, the *root-mean-square* (RMS) error ϵ_{RMS} is often used, where

$$\epsilon_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\,\mathrm{RSS}} \tag{7.2.3}$$

$$= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{7.2.4}$$

Linear Least-Squares

The *linear least-squares estimator* (linear LS estimator) is obtained for the values of $\hat{a}_0$ and $\hat{a}_1$ which minimize RSS. This method, also known as the method of least-squares or ordinary least-squares (OLS), was originally developed by Karl Friedrich Gauss (1777–1855) in 1795 [124]. Formally the LS estimator is defined as

$$\underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{N} \hat{\epsilon}_i(\mathbf{a})^2 \tag{7.2.5}$$

A necessary condition for $q = \mathrm{RSS}$ to be minimum is

$$\frac{\partial q}{\partial a_0} = 0, \qquad \frac{\partial q}{\partial a_1} = 0 \tag{7.2.6}$$

Elaborating the conditions above

$$\frac{\partial q}{\partial a_0} = \frac{\partial}{\partial a_0}\sum_{i=1}^{N}\Big[y_i - a_0 - a_1 x_i\Big]^2$$

$$= -2\sum_{i=1}^{N}\Big[y_i - a_0 - a_1 x_i\Big] \tag{7.2.7}$$

and

$$\frac{\partial q}{\partial a_1} = \frac{\partial}{\partial a_1}\sum_{i=1}^{N}\Big[y_i - a_0 - a_1 x_i\Big]^2$$

$$= -2\sum_{i=1}^{N} x_i\Big[y_i - a_0 - a_1 x_i\Big] \tag{7.2.8}$$

To simplify the notation, the range on the summation will be assumed to be over i from 1 to N. Then from Eq. **(7.2.6)**, Eq. **(7.2.7)** and Eq. **(7.2.8)**,

$$-2\sum\Big[y_i - a_0 - a_1 x_i\Big] = 0$$

$$\therefore \quad \sum\Big[y_i - a_0 - a_1 x_i\Big] = 0$$

$$\therefore \quad \sum y_i - a_0 N - a_1\sum x_i = 0$$

$$\therefore \quad a_0 N + a_1\sum x_i = \sum y_i \tag{7.2.9}$$

and

$$\begin{aligned} -2\sum x_i\Big[y_j - a_0 - a_1 x_j\Big] &= 0 \\ \therefore \quad \sum x_i\Big[y_j - a_0 - a_1 x_j\Big] &= 0 \\ \therefore \quad \sum x_i y_i - a_0\sum x_i - a_1\sum x_i^2 &= 0 \\ \therefore \quad a_0\sum x_i + a_1\sum x_i^2 &= \sum x_i y_i \end{aligned} \tag{7.2.10}$$

Eq. **(7.2.9)** and Eq. **(7.2.10)** are known as the *normal equations*. In matrix form

$$\begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix} \qquad \Longleftrightarrow \qquad \mathbf{Sa} = \mathbf{t}$$

The matrix $\mathbf{S}$ is a square matrix and provided that it is not singular (depending on the samples in the data set), the matrix is invertible. The desired model parameter vector $\mathbf{a}$ is then given by

$$\mathbf{a} = \mathbf{S}^{-1}\mathbf{t} \tag{7.2.11}$$

where $\mathbf{S}^{-1}$ can be obtained from the determinant and cofactors of $\mathbf{S}$. For a 2×2 matrix $\mathbf{P}$,

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

the inverse is given in terms of the determinant and cofactors as

$$\begin{aligned} \mathbf{P}^{-1} &= \frac{1}{\det \mathbf{P}}\begin{bmatrix} P_{11} & P_{21} \\ P_{12} & P_{22} \end{bmatrix} \\ &= \frac{1}{p_{11}p_{22} - p_{12}p_{21}}\begin{bmatrix} p_{22} & -p_{12} \\ -p_{21} & p_{11} \end{bmatrix} \end{aligned}$$

The formulation for the parameter vector $\mathbf{a}$ in terms of the data set is then given by

$$\begin{aligned} \mathbf{a} &= \frac{1}{s_0 s_2 - s_1^2}\begin{bmatrix} s_2 & -s_1 \\ -s_1 & s_0 \end{bmatrix}\begin{bmatrix} t_0 \\ t_1 \end{bmatrix} \\ &= \frac{1}{s_0 s_2 - s_1^2}\begin{bmatrix} s_2 t_0 - s_1 t_1 \\ s_0 t_1 - s_1 t_0 \end{bmatrix} \end{aligned}$$

where

$$s_0 = N, \quad s_1 = \sum x_i \quad s_2 = \sum x_i^2 \quad t_0 = \sum y_i \quad t_1 = \sum x_i y_i$$

The formulation derived in this section leads to a computationally efficient implementation. Since the formulation is build around the normal equations, the use of any other model requires the derivation of the corresponding normal equations.

It should be noted that the solution of the normal equations is not always stable in the presence of rounding errors and other discrete numerical effects. For the general case, alternative methods to solving the least-squares problem are recommended. These methods will be discussed below.

Higher-Order Polynomial Models

The approach described above for the direct derivation of normal equations can be generalized to a polynomial of arbitrary order M

$$r(x) = a_0 + a_1 x + \cdots + a_M x^M \tag{7.2.12}$$

where $M \leq N - 1$. It will be understood from context that this model represents the estimator and the hat notation will be dropped. There are therefore $M + 1$ model parameters that must be estimated from the data set. Correspondingly, there are now $M + 1$ conditions for minimal q, namely,

$$\frac{\partial q}{\partial a_0} = 0, \qquad \ldots, \qquad \frac{\partial q}{\partial a_M} = 0 \tag{7.2.13}$$

which results in a system of $M + 1$ normal equations. The elaborated normal equations for the quadratic and cubic cases are given in Appendix E. Clearly the derivation of the normal equations becomes progressively more tedious for higher-order polynomials.

System of Linear Equations

An alternative formulation for obtaining the parameters of a polynomial model of order M can be derived by developing a system of linear equations (SLE) for all the available samples $(x_i, y_i), i = 1 \ldots N$ in the data set as

$$\begin{array}{l} a_0 + x_1 a_1 + \cdots + a_M x_1^M = y_1 \\ a_0 + x_2 a_1 + \cdots + a_M x_2^M = y_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_0 + x_N a_1 + \cdots + a_M x_N^M = y_N \end{array}$$

Written in matrix form

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_M \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}$$

or

$$\mathbf{X}\mathbf{a} = \mathbf{y} \tag{7.2.14}$$

The $N \times M$ matrix $\mathbf{X}$ is known as the *design matrix*. Since $\mathbf{X}$ is generally not a square matrix, it does not have a standard inverse. The system can be solved by noting that the matrix $\mathbf{X}^T\mathbf{X}$ is symmetric. If $(\mathbf{X}^T\mathbf{X})^{-1}$ exists, then

$$\begin{aligned} \mathbf{X}\mathbf{a} &= \mathbf{y} \\ \therefore \quad \mathbf{X}^T\mathbf{X}\mathbf{a} &= \mathbf{X}^T\mathbf{y} \\ \therefore \quad \mathbf{a} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{7.2.15}$$

The parameter vector $\mathbf{a}$ is therefore given as

$$\mathbf{a} = \mathbf{X}^{+}\mathbf{y} \tag{7.2.16}$$

where $\mathbf{X}^+ = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is known as the *pseudoinverse* of $\mathbf{X}$. The pseudoinverse is a generalisation of the notion of a matrix inverse to non-square matrices. Developed independently by Eliakim H. Moore [87] and Sir Roger Penrose [95], the pseudoinverse is also known as the Moore-Penrose inverse and provides a least-squares solution to a system of linear equations (SLE) [96].

Generally, the most computationally efficient way to obtain the pseudoinverse is by means of *singular value decomposition* (SVD) [45]. The singular value decomposition of a matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U}$ is the left singular matrix, $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values w_j of $\mathbf{A}$ and $\mathbf{V}$ is the right singular matrix.

The nature of the singular values gives an indication of the linear independence of the rows in the matrix $\mathbf{A}$. The *condition number* of a matrix is defined as the ratio of the largest (in magnitude) singular value to the smallest singular value [106]. A matrix is *singular* if the condition number is infinite, and is *ill-conditioned* if the condition number is large.

It can be shown [45, 131] that the pseudoinverse of $\mathbf{A}$ is given in terms of the singular value decomposition as

$$\mathbf{A}^+ = \mathbf{U}\mathbf{\Sigma}^+\mathbf{V}^T \tag{7.2.17}$$

where $\mathbf{\Sigma}^+$ is a diagonal whose entries are the reciprocals of the entries in $\mathbf{\Sigma}$ (*i.e.* the reciprocals of the singular values of $\mathbf{A}$). Although using a different syntax, this is in fact the standard inverse of a diagonal matrix. When forming $\mathbf{\Sigma}^+$, the reciprocals $1/w_j$ where the singular value w_j is 0 should be replaced by 0 (see [106] pages 59–69 and [131] pages 331–337 for a detailed discussion).

The availability of the condition number is an additional advantage of this approach to model fitting as it gives an indication of the "goodness of fit". This is due to the fact that the design matrix directly incorporates knowledge of the model and the data set. This approach therefore has the disadvantage that it is computationally less efficient than the normal equation approach, but has the advantage that it can trivially be used to fit any order polynomial and provides an indication of the "goodness of fit". The SVD approach (when correctly implemented) is also less sensitive to numerical effects (such as rounding errors) and is generally the most numerically stable method to solve a least-squares problem (see [138] pages 129–143).

For this project, the SVD routine from the LAPACK library (Section 4.7.1) was used. The LAPACK library also makes available routines (`DGELSS`, `DGELSX` and `DGELSY`) to provide a least-squares solution to a system of linear equations. Therefore, instead of forming the pseudoinverse in order to obtain the parameter vector by Eq. **(7.2.16)**, Eq. **(7.2.14)** can be solved directly to obtain the parameter vector $\mathbf{a}$. The particular LAPACK routine used, `DGELSY`, computes the minimum-norm solution to the real linear least-squares problem (using the matrix names given above)

$$\underset{\mathbf{a}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{a} - \mathbf{y}\|_2$$

using a complete orthogonal factorisation of X, where X may be rank-deficient.

Model Fitting Performance

Since the SDDS application, and in particular the image processing aspect, is time critical, the performance of image processing operations is of concern. The performance of the different model fitting techniques is therefore a primary factor determining the selection of techniques. Of the various techniques discussed, the normal equation approach is known to have the highest performance [32] (page

Table 7.1: Model fit performance (operations per second).

Technique	Order 1	Order 2	Order 3
Normal equations	330667	117308	98264
System of linear equations	8615	6043	4261
Singular value decomposition	7261	4405	4257

132), at the cost of accuracy and stability. The systems of linear equations approach has the next highest performance approach, followed by the SVD / pseudoinverse approach.

In order to select the best algorithm to perform the model fitting in the SDDS application, extensive empirical tests and benchmarks were performed. Numerical stability was evaluated by comparing all available techniques to perform model fitting across various data sets. From a numerical stability perspective, it was found that for the data sets, all model fitting techniques proved to be stable. The data sets were created as samples of real extracted laser curves captured under various conditions.

The performance on the various approaches was measured by performing the model fitting for various model orders. The data set consisted of 800 data points. For each approach and model order, the model fitting was repeated 10^6 times and the total time measured. The number of iterations was selected to ensure that the minimum measured time was no less that $60\,s$, in order to ensure a representative measurement. Furthermore, a cache flushing routine was called between each iteration to flush the Level 2 cache of the CPU. The aim of this action was to reduce the bias towards routines that operate with a small memory footprint. The results of the experiment are given in Table **{7.1}** as the number of model fitting operations per second.

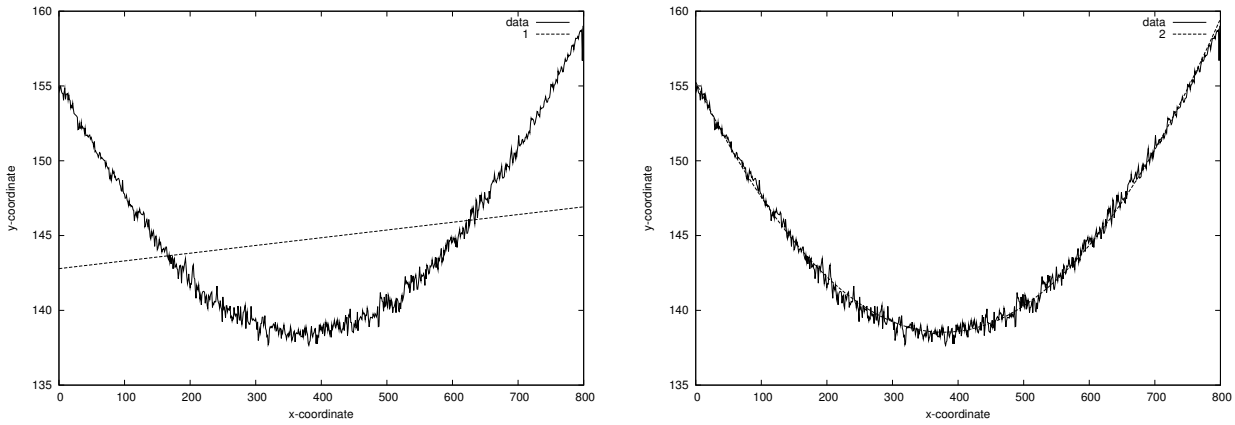
The measured performance agrees with the expected trends. It is clear that the direct solution of the normal equations is superior to the alternatives. Therefore, although the manual derivation of the direct solution to the normal equations was time consuming, the resultant performance justifies the time so spent. As a further motivation, although the result will not be discussed here, measurements of the image processing rate using the various techniques confirm that the direct solution of the normal equations approach is the only technique which allows the desired processing rate to be achieved.

As a side note, the CPUs of both the development computer and the target computer support the version 2 Intel Streaming SIMD Extensions (SSE2) instruction set. These specialised instructions allow for the efficient simultaneous execution of floating-point operations on multiple pipelines of the floating-point units (FPUs). An examination of the machine instructions generated for the implemented `sdds_polynomial_nefit` helper routines shows that the required computation is performed using almost exclusively SSE2 instructions with minimal conditional jumps. The necessary loop unrolling and data prefetching are correctly performed by the GCC compiler. Based on experience, the resultant instructions appear to be near optimal and ensure that all available computation resources are utilised during the calculation.

7.2.2 Selection of a Model

The simplest model for the laser curve would clearly be a line. However, the residuals of such a fitting would be large, depending on the nature of the laser curve. To select a suitable model, an experimental

approach was followed. A number of data sets were obtained by performing the laser curve extraction on images captured on areas of the panel under test. These areas were chosen so that no defects were suspected. The data sets so collected should therefore be representative of the expected laser curve, with the addition of measurement noise.



(a) First order polynomial

(b) Second order polynomial

Figure 7.1: Example of fitting of models to extracted laser curve from data set 2 (no defect).

For each data set (DS), polynomial models were fitted of various orders. The results of these experiments are given in Table **{7.2}** and Table **{7.3}**. Polynomial models were chosen initially as they are generally computationally simpler than alternatives such as parametric models [16]. Fig. **[7.1]** gives examples of the results of fitting a first- and second-order polynomial model to the extracted laser curve data (with no defect).

The examination of the ϵ_{RMS} and model parameters then provide an indication of the most suitable model. Intuitively, the higher-order models should result in a lower ϵ_{RMS}. However, as the model order is increased, the improvement in ϵ_{RMS} will rapidly decrease at the point where the intrinsic complexity of the data is matched by the complexity of the model. The ϵ_{RMS} versus model order curve thus provides an indication of optimal model selection.

As the model order is increased, the additional model parameters may prove to be small. If the parameters are sufficiently small (*e.g.*, $< 10^{-6}$), the parameters may well be contributing to fitting the model to the measurement noise. From Fig. **[7.2]** it is clear that the quadratic model provides a good fit for the extracted laser curve. Higher-order models do not significantly improve the fit ϵ_{RMS} for the defect free case. Note that the cause of the higher ϵ_{RMS} in data set 5 is significant noise in the extracted laser curve data.

To complete this investigation, a quadratic model was fit to a data set containing a known defect. The results are given in Fig. **[7.3]**. In addition to the quadratic model, a cubic polynomial was also fitted in order to evaluate the effect of a higher-order model. As can be seen from Fig. **[7.3]** (b), the higher-order model attempts to fit the defect, which is undesirable. The quadratic function is therefore the most suitable general model for the expected laser curve and provides the optimal balance between robustness against measurement noise / actual defects and the essential nature of the fractional NURBS curve.

Table 7.2: Model parameters for various model orders.

Order	DS	a_5	a_4	a_3	a_2	a_1	a_0
	0						+1.292814E+02
	1						+1.403373E+02
0	2						+1.448520E+02
	3						+1.642727E+02
	4						+1.322117E+02
	5						+9.012118E+01
	0					+5.038735E-03	+1.272685E+02
	1					+5.398341E-03	+1.381807E+02
1	2					+5.167133E-03	+1.427877E+02
	3					+1.962801E-03	+1.634886E+02
	4					+1.618086E-02	+1.257474E+02
	5					-5.449164E-03	+9.229812E+01
	0				+1.242551E-04	-9.424109E-02	+1.404727E+02
	1				+1.208497E-04	-9.116058E-02	+1.510230E+02
2	2				+1.173421E-04	-8.858923E-02	+1.552573E+02
	3				+1.091963E-04	-8.528502E-02	+1.750926E+02
	4				-1.531956E-03	+1.240213E+00	-3.704890E+01
	5				-1.089646E-03	+8.651778E-01	-2.349527E+01
	0			-1.333389E-08	+1.402358E-04	-9.934532E-02	+1.408115E+02
	1			-1.114279E-08	+1.342043E-04	-9.542605E-02	+1.513061E+02
3	2			-1.161389E-08	+1.312614E-04	-9.303504E-02	+1.555524E+02
	3			-2.073604E-08	+1.340484E-04	-9.322279E-02	+1.756194E+02
	4			-2.732261E-08	-1.499209E-03	+1.229754E+00	-3.635467E+01
	5			+2.594534E-07	-1.400601E-03	+9.644968E-01	-3.008757E+01
	0		-4.408382E-11	+5.711205E-08	+1.040668E-04	-9.293267E-02	+1.405567E+02
	1		-4.185204E-11	+5.573678E-08	+9.986644E-05	-8.933805E-02	+1.510643E+02
4	2		-4.548572E-11	+6.107229E-08	+9.394218E-05	-8.641847E-02	+1.552896E+02
	3		-4.204546E-11	+4.645261E-08	+9.955181E-05	-8.710666E-02	+1.753765E+02
	4		-5.109178E-09	+8.137145E-06	-5.691084E-03	+1.972960E+00	-6.587860E+01
	5		-1.317496E-09	+2.364811E-06	-2.481553E-03	+1.156146E+00	-3.770086E+01
	0	+1.512688E-13	-3.462432E-10	+2.716455E-07	+3.984733E-05	-8.562078E-02	+1.403637E+02
	1	+1.196115E-13	-2.807761E-10	+2.253730E-07	+4.908672E-05	-8.355638E-02	+1.509117E+02
5	2	+8.886602E-14	-2.229956E-10	+1.871044E-07	+5.621512E-05	-8.212294E-02	+1.551762E+02
	3	+1.326947E-13	-3.071032E-10	+2.346438E-07	+4.321777E-05	-8.069258E-02	+1.752071E+02
	4	-7.588454E-12	+1.004876E-08	-2.625004E-06	-2.469491E-03	+1.606156E+00	-5.619490E+01
	5	+1.923775E-12	-5.160236E-09	+5.093160E-06	-3.298269E-03	+1.249136E+00	-4.015581E+01

Table 7.3: ϵ_{RMS} for various model orders.

DS	$M = 0$	$M = 1$	$M = 2$	$M = 3$	$M = 4$	$M = 5$
0	+6.056806E+00	+5.913126E+00	+5.737370E+00	+5.244170E+00	+8.571892E+01	+6.206133E+01
1	+5.943974E+00	+5.780209E+00	+5.611903E+00	+5.224542E+00	+8.563743E+01	+6.204857E+01
2	+4.450331E-01	+4.211652E-01	+4.014770E-01	+4.034055E-01	+4.464682E+01	+3.388533E+01
3	+4.259218E-01	+4.071314E-01	+3.854296E-01	+3.499723E-01	+4.464603E+01	+3.379221E+01
4	+4.171527E-01	+3.988642E-01	+3.750805E-01	+3.402286E-01	+4.351971E+01	+3.369436E+01
5	+4.129158E-01	+3.960981E-01	+3.734589E-01	+3.362279E-01	+4.341790E+01	+3.368592E+01

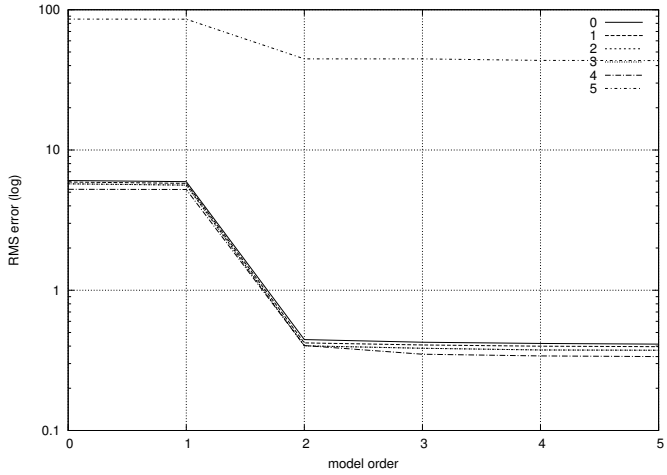
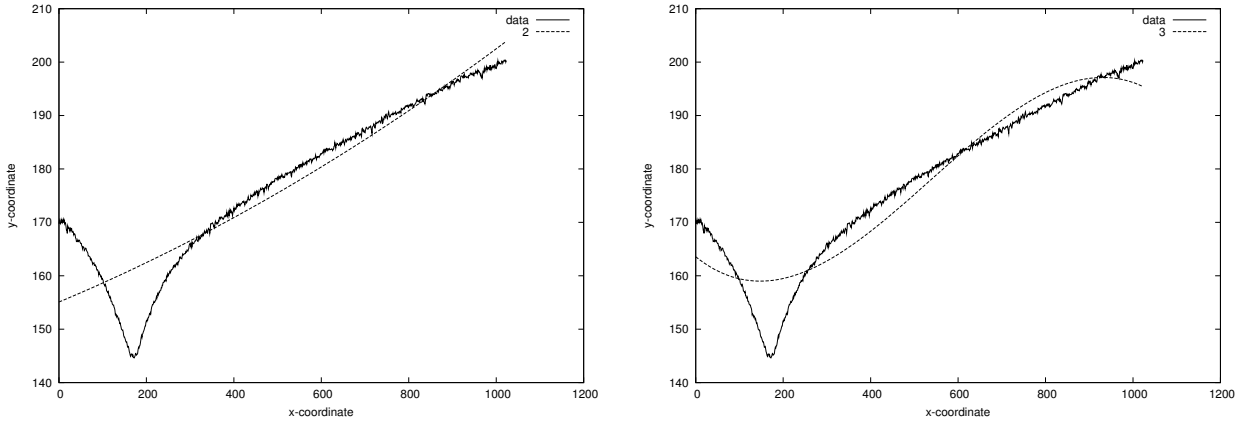


Figure 7.2: ϵ_{RMS} for various model orders.



(a) Second-order polynomial

(b) Third-order polynomial

Figure 7.3: Example of fitting of models to extracted laser curve from data set 6 (defect).

7.3 Robust Regression

The extracted laser curve data contains noise and potentially data perturbed by a defect on the roof. The question is now: In what manner is the model fitted by the least-squares methods above affected by the noise and errors in the data? The following section will investigate the effects of noise and errors in data sets on the least-squares methods.

7.3.1 Sensitivity of Least-Squares Methods

Unfortunately, models fitted by means of LS are highly sensitive to errors in both the predictor variable and the response variable. A well-known demonstration of this sensitivity is provided by performing

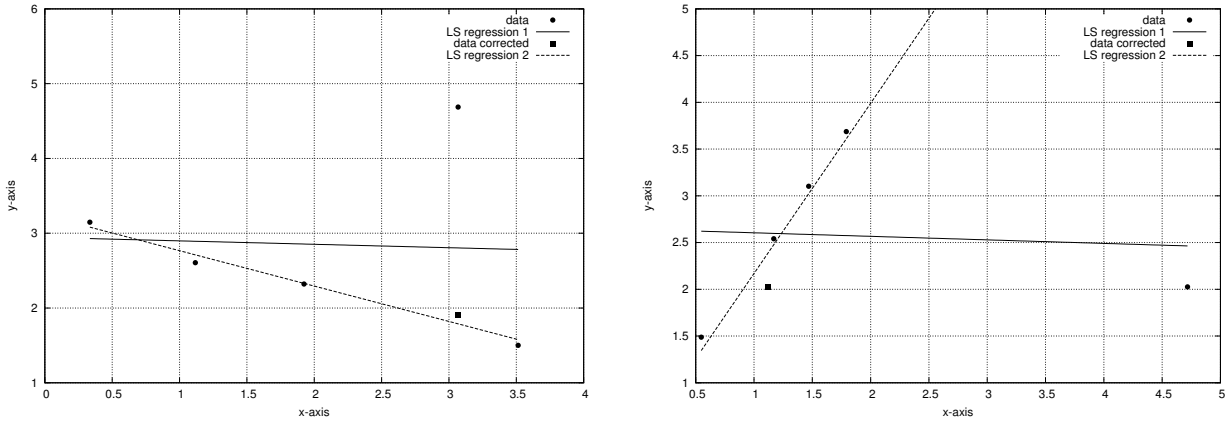
linear regression on the data sets (from [116])

$$S_A(x,y) = \{(0.3341, 3.1473), (1.1175, 2.6057), (1.9240, 2.3209), (3.0677, 4.6867), (3.5138, 1.5011)\}$$

and

$$S_B(x,y) = \{(0.5472, 1.4883), (4.7177, 2.0263), (1.1694, 2.5409), (1.4689, 3.1023), (1.7915, 3.6871)\}$$

For the first data set, Fig. **[7.4]** (a) gives the least-squares regression line. As can be seen, one data



(a) Linear regression on S_A

(b) Linear regression on S_B

Figure 7.4: Sensitivity of LS regression to data set errors.

point $(3.0677, 4.6867)$ contains an error or large deviation in the response variable. The resultant regression line is attracted by the large residual and the regression line deviates significantly from the essential linear dependence of the data. The data point that differs significantly from the bulk of the data is known as an *outlier*.

To obtain the second regression line, the outlier data point is replaced by $(3.0677, 1.9110)$ as shown by the square. The response variable was corrected (for illustration purposes) by replacing it with the mean of the surrounding data points. Clearly the resultant regression line 2 more closely describes the essential linear dependence of the data.

Fig. **[7.4]** (b) illustrates the effect of an error in the predictor variable of a data point. In both cases, a single outlier data point which lies far from the bulk of the data points results in a significant error in the regression relative to the bulk of the data set. Such a data point is known as a *leaverage point*.

The effects illustrated can severely compromise the validity of a least-squares regression and has resulted in two statistical fields of study:

- **Regression Diagnostics** [11, 7], where the outliers are specifically identified, studied and potentially modified, and
- **Robust Statistics** [65, 116] where alternatives to least-squares are devised which are not as sensitive to outliers.

In the SDDS application, isolated outliers are known *a priori* to be the result of noise or sporadic errors in the system. They are therefore not of interest and the desire is to correctly ignore such outliers, which leads to an investigation of robust statistics. Broadly speaking, the approaches followed in robust statistics entail either reducing large residuals by means of a penalty function, or simply ignoring them.

7.3.2 An Overview of Robust Statistics

In the previous section, it was seen that a single erroneous data point resulted in the break-down of the LS regression. To formalize the study of alternative regression techniques, the *breakdown point* of an estimator is said to be the ratio or percentage of outliers present in the data set which results in the estimator to break-down. For a formal mathematical definition see [116], pages 9–10. From this definition, LS estimators have a 0% breakdown point, indicating that a single outlier can cause arbitrarily large deviations of the estimator.

One of the first attempts at a robust estimator was the *least absolute values* regression estimator which is defined by

$$\underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{N} |\hat{\epsilon}_i(\mathbf{a})| \tag{7.3.1}$$

It can be shown that the least absolute value estimator, also known as the L_1 estimator (the LS estimator is correspondingly known as the L_2 estimator), is generally insensitive to outliers caused by deviations in the response variable. However, the L_1 estimator is still sensitive to deviations in the predictor variable. Over time a large number of robust estimators have been proposed and analysed [65]: *M-estimators*, *GM-estimators*, *RM-estimators*, *S-estimators*, *etc.* In the following sections, two specific robust estimators will be discussed which have proven to be both theoretically robust and allow for practical implementation.

7.3.3 Least-Median of Squares (LMedS) Estimator

The Least-Median of Squares (LMedS) Estimator is given by

$$\underset{\mathbf{a}}{\operatorname{argmin}}\ J(\mathbf{a}) \tag{7.3.2}$$

where the objective function is

$$J(\mathbf{a}) = \underset{i}{\operatorname{median}}\ \hat{\epsilon}_i(\mathbf{a})^2 \tag{7.3.3}$$

In other words, the Least-Median of Squares estimator is defined to be the parameter vector $\mathbf{a}$ which minimizes the median of the square residuals. This estimator has been shown [116] to be robust against outliers due to both the predictor and the response variables. Furthermore, the breakdown point of the LMedS estimator is 50%, the highest possible value. Note that in select literature, this estimator is called the LMS estimator, but should not be confused with the Least-Mean-Square algorithm [151].

Since the objective function $J(\mathbf{a})$ is not differentiable, there is no closed-form solution and also no iterative solution. The ideal LMedS implementation would systematically create subsets of the data set, excluding selected data points until all possible permutations of the subsets have been evaluated and the minimum median-square residual has been obtained. There are a number of alternate exact

implementations [132]. Unfortunately, these implementations are computationally expensive and cannot therefore be used in this application due to the stringent time constraints.

Most practical implementations of LMedS make use of random sampling to select a subset of the original data set. The estimator is then applied to the subset in order to obtain a candidate regression. This sampling process is repeated a predetermined number of times and the candidate that obtains the minimal median-square residual is selected as the best regression estimate.

In [116], the relationship between the number of subsets formed m, the size of the subset p, the expected fraction of outliers ε, and the probability of a "good" regression is given as

$$P = 1 - [1 - (1 - \varepsilon)^p]^m \tag{7.3.4}$$

The number of subsets (hence trial regressions) can therefore be determined as

$$m = \frac{\log(1 - P)}{\log[1 - (1 - \varepsilon)^p]} \tag{7.3.5}$$

In the SDDS application, a quadratic function is fitted, implying that $p = 3$ (assuming sample independence). If the desired probability of a good fit is $P = 0.99$ under the circumstances where at most $\varepsilon = 40\%$ outliers are present in the data, then $m = 19$. The percentage of outliers was chosen to represent a worst-case estimate of dust particle induced pixel outliers.

Alg. $\lfloor$**7.3.1**$\rfloor$ supplies an algorithm, based on the description in [116] (Chapter 5), that implements the Random Sample LMedS (`RSLMedS`) estimator. This estimator is present in the `sddspolynomial` module of the SDDSF and allows the robust fitting of a polynomial of arbitrary order to an arbitrary large data set. Note that numerous variations on the LMedS estimator algorithm exist, each with unique characteristics [116, 65]. The particular algorithm was chosen as it represents the original, reference algorithm of LMedS.

The *Random Sample Consensus* (`RANSAC`) algorithm [40] is another popular algorithm for model fitting where the objective function is not differentiable. `RANSAC` shares many of the characteristics of `RSLMedS` and in addition performs limited outlier diagnostics to guide the random selection. Hence, `RANSAC` may outperform `RSLMedS` at the cost of increased computational complexity. A performance evaluation of the `RSLMedS` (described in Section 7.3.5) indicated that the performance of the `RSLMedS` would be problematic in the SDDS application. The `RANSAC` algorithm would therefore be correspondingly more problematic and was therefore not implemented.

To ensure a statistically stable and correct source of random numbers in order to perform the sampling, the Mersenne Twister (MT19937) pseudo-random number generator [85] is used. The MT19937 generator of Makoto Matsumoto and Takuji Nishimura is a variant of the twisted generalized feedback shift-register algorithm. It has a Mersenne prime period of $2^{19937} - 1$ ($\approx 10^{6000}$). The generator is implemented in the `sddsmt19937` module and is typically seeded with the Unix epoch (time).

Clearly the result of the `RSLMedS` algorithm is not deterministic and may not represent the optimal regression. The instability of the estimate is partly due to the random sampling, and partly due to the median operator used in the objective function. For the SDDS application, the primary drawback of the LMedS approach is that the resultant regression may not be optimal. The next section will investigate an alternative approach.

Algorithm 7.3.1 RSLMedS

1. Determine the number of samples per subset p based on the order d of the polynomial used as model (*i.e.* the degrees of freedom). Let $p = d + 1$.
2. Calculate the required number of subsets m using Eq. **(7.3.5)** with $P = 0.99$ and $\varepsilon = 40\%$.
3. Ensure that the data set contains sufficient samples $N \geq mp$.
4. For each trial regression:
   (a) Construct the data subset containing p samples using uniform random sampling such that $(x_i, y_i), i \sim \mathrm{U}[1, N]$ and $i \in \mathbb{Z}$.
   (b) Perform a standard regression on the subset using, for example, SLE to obtain a candidate regression.
   (c) Using the candidate regression, form an array of square residuals for the remainder of the data set (not including the samples in the current subset). There will therefore be $N - p$ residuals.
   (d) Store the median residual and the candidate regression for future use.
5. Examine the stored median residuals to locate the minimum and select the corresponding candidate regression as the Least-Median of Squares regression.

7.3.4 Iteratively Re-weighted Least-Squares (IRLS)

M-estimators are the most commonly used robust estimators and entail the use of a penalty function, ρ, on the residuals. M-estimators are defined by

$$\underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{N} \rho(\mu_i) \tag{7.3.6}$$

where

$$\mu_i = \frac{\epsilon_i(\mathbf{a})}{\sigma_i} \tag{7.3.7}$$

In Eq. **(7.3.7)**, σ_i is the per data point robust scale factor, which is used to calculate the scale normalized residual μ_i. In certain cases, the scale factor is known *a priori* based on knowledge of the data source. In other cases, the scale factor must be estimated from the data. This can be done via a robust scale estimator such as the *median absolute deviation* (MAD) [65] (Chapter 5). For a data set containing n elements x_i,

$$\mathrm{MAD}_n = \underset{i}{\operatorname{median}} \, |x_i - M_n|,$$

where M_n is effectively a robust estimate of the mean, given by

$$M_n = \underset{i}{\operatorname{median}} \, x_i;$$

Note that the MAD essentially gives a robust estimate of the variance of a data set and is used throughout the implemented software for this purpose. Numerous robust penalty functions, ρ, have been devised [52, 65, 116, 11, 10, 61], examples of which are,

- Huber's t-function:

$$\rho_H(u) = \begin{cases} u & |u| \leq a \\ a & |u| > a \end{cases}$$

- Cauchy function:

$$\rho_C(u) = \frac{c^2}{2}\log\left[1 + (\frac{u}{c})^2\right]$$

- Tukey's bi-weight function:

$$\rho_T(u) = \begin{cases} u(1 - (\frac{u}{a})^2) & |u| \leq a \\ 0 & |u| > a \end{cases}$$

- Beaton-Tukey bi-weight function:

$$\rho_B(u) = \begin{cases} \frac{a^2}{6}\left[1 - (1 - (\frac{u}{a})^2)^3\right] & |u| \leq a \\ \frac{a^2}{6} & |u| > a \end{cases}$$

The selection of the parameters of the penalty function however remains a problem. A popular algorithm to solve Eq. **(7.3.6)** is *Iteratively Re-weighted Least-Squares* (`IRLS`) [61]. If the domain of the regression is known, the residuals can be penalised based on the domain knowledge. The resultant modified `IRLS` implementation is known as the *Domain Bound M-Estimator* [128].

An implementation of `IRLS` is given in Alg. ⌊**7.3.2**⌋. At the core of the algorithm is the weighted least-squares operation. In Section 7.2.1, the system of linear equations incorporating all the data points is given by Eq. **(7.2.14)**. The solution is in turn given by Eq. **(7.2.15)**. These equations can be modified in such a way to weight each residual using one of the weight functions above. To derive these equations, let $\mathbf{Q}$ be the matrix square root of a weight matrix $\mathbf{W}$, such that $\mathbf{W} = \mathbf{Q}^T\mathbf{Q}$. Multiplying Eq. **(7.2.14)** by $\mathbf{Q}$ and solving for the parameter vector $\mathbf{a}$,

$$\begin{aligned} \mathbf{QXa} &= \mathbf{Qy} \\ \therefore \quad (\mathbf{QX})^T\mathbf{QXa} &= (\mathbf{QX})^T\mathbf{Qy} \\ \therefore \quad \mathbf{a} &= \left((\mathbf{QX})^T\mathbf{QX}\right)^{-1}(\mathbf{QX})^T\mathbf{Qy} \\ &= \left(\mathbf{X}^T\mathbf{Q}^T\mathbf{QX}\right)^{-1}\mathbf{X}^T\mathbf{Q}^T\mathbf{Qy} \\ &= \left(\mathbf{X}^T\mathbf{WX}\right)^{-1}\mathbf{X}^T\mathbf{Wy} \end{aligned} \tag{7.3.8}$$

The `IRLS` algorithm therefore iteratively calculates a regression by successively reweighing the data points. Since `IRLS` represents a form of local search, it is susceptible to the problem of being trapped in a local minimum. If the initial estimate is therefore close to a non-optimal local minimum, typically due to using OLS to obtain the initial estimate, `IRLS` will produce stable, but non-optimal results. This deficiency can be overcome in part by using `RSLMedS` to provide the initial estimate.

7.3.5 Performance Evaluation

Similar to the procedure followed in Section 7.2.1, the performance of the robust fitting algorithms is evaluated by performing numerous fitting operations and estimating the number of fitting operations per second that can be performed. In this case, the `RSLMedS` and `IRLS` algorithms were benchmarked. The results are given in Table **{7.4}**.

During the measurement of the performance of the `RSLMedS` and `IRLS` techniques, it was soon apparent that their high computational cost would disqualify them from being used in the SDDS application (assuming the current processing resources). Thus an alternative technique was required.

Algorithm 7.3.2 IRLS

1. Form an initial parameter estimate $\hat{\mathbf{a}}_0$, using for example, regular least-squares. Alternatively, the `RSLMedS` algorithm above can be used to provide a robust initial estimate.
2. Repeat for k iterations or until the change in the parameter estimate is sufficiently small:
   (a) Calculate the residuals using the complete dataset and the current model estimate.
   (b) Update the scale estimate $\hat{\sigma}$ using [116] page 202

$$1.4826\ (1 + \frac{5}{N-p})\sqrt{\underset{i}{\text{median}}\ r_i^2}$$

   where N is the number of samples and p is the number of model parameters. The estimate assumes the presence of Gaussian noise.
   (c) Calculate the weight matrix diagonal entries as

$$w_i = \frac{\Psi(r_i/\hat{\sigma})}{r_i/\hat{\sigma}}$$

   where Ψ is the corresponding influence function (the Huber influence function is used in the implementation). Or alternatively, the diagonal weight entry can be a thresholding operation defined by

$$w_i = \begin{cases} 1 & if \epsilon_i(\mathbf{a})^2 \leq (2.5\hat{\sigma})^2 \\ 0 & otherwise \end{cases}$$

   (d) Solve the weighted least-squares Eq. **(7.3.8)** to yield the new estimate, $\hat{\mathbf{a}}_t$, for iteration number t.

Table 7.4: Robust model fit performance (operations per second).

Technique	Order 1	Order 2	Order 3
`RSLMedS`	113	63	36
`IRLS`	16	26	27
`RLSNE`	1076	1021	1001

As mentioned earlier, one method in which to handle outliers is to simply discard them. A candidate robust fitting algorithm was therefore developed, namely Robust Least-Squares Normal Equations (`RLSNE`). Alg. ⌊**7.3.3**⌋ supplies the `RLSNE` algorithm. Clearly `RLSNE` is a primitive form of `IRLS` and therefore does not allow the same level of robustness that `IRLS` provides. `RLSNE` is however less sensitive to certain forms of outliers. Empirical validation on a number of datasets indicate that the algorithm is sufficiently robust. However further tests are required under production conditions to validate the effectiveness of this approach.

7.4 Kalman Filtering

The proceeding sections discussed the expected laser curve model to which the extracted laser curve data must be compared. The extracted laser curve was obtained as the output of the stage 1 processing

Algorithm 7.3.3 RLSNE

1. Form an initial parameter estimate $\hat{\mathbf{a}}_0$, using regular least-squares based on normal equations (for performance).
2. Obtain a robust estimate of variance $\hat{\sigma}$ using MAD (see Section 7.3.4).
3. Adjust all data points that exceed a threshold of $2\hat{\sigma}$ by

   $$y_i = r_i/\hat{\sigma}$$

   effectively moving the outliers closer to the expected model.
4. Obtain a final estimate of the model parameters using regular least-squares based on normal equations, performed on the modified data set.

as discussed in Chapter 6. The extracted laser curve data is perturbed by measurement noise. In Chapter 2, the measurement noise was motivated to be primarily Gaussian in nature. Depending on the subsequent processing to be performed on the extracted laser curve data, it may be desirable to filter the data to remove the measurement noise.

Filtering of data is a well established science and hence a large number of filtering techniques are available. The selection of a suitable filtering technique and corresponding implementation should be driven by the nature of the data. Additionally, if knowledge exists about the system producing the data and the underlying noise characteristics, these can be used to further guide the selection process. During the development of the second stage filtering, various filtering techniques such as frequency domain filtering [93, 108, 17, 18], digital filters [51, 93, 108] and optimal filtering [44, 4, 127, 58] were examined. Of these, one particular technique, Kalman filtering, appeared significantly more appropriate to the SDDS application than the investigated alternatives.

Many of the alternatives required knowledge of the frequency characteristics of the data and underlying system. In this application, it is clear that some form of low pass filtering is required. However, the required frequency characteristics are not apparent and there does not appear to be an obvious way in which to estimate the required characteristics. The lack of this information results in the frequency / time domain filter design techniques being less attractive for the SDDS application.

The chosen implementation, the Kalman filter, exhibits many desirable characteristics. Amongst other, the filter is recursive in nature and allows for incremental updates. The output of the filter is an estimate of the state of a dynamic system based on the incomplete and noisy measurements. Named after Rudolf E. Kalman (1930-), the filter was developed by Kalman [70] based on the work by Peter Swerling (1920–2000).

The Kalman filter is used extensively in a wide range of applications as it works well in practice [122, 34, 62, 30, 111, 129, 120, 28]. The filter is also theoretically attractive as it can be shown that of all possible filters, the filter minimizes the variance of the estimation error [145, 4]. Hence, under specific conditions, the Kalman filter can be said to be optimal.

7.4.1 Overview of Kalman Filter Implementation

This section will briefly discuss the implementation of the Kalman filter in the SDDSF. Since the topic of optimal filtering (of which Kalman filtering is the prime example) is extensive, this section will not attempt to elaborate on the theory of Kalman filtering. Due to time constraints, the author has not explored all aspects of Kalman filtering as described in the standard references [70, 44, 4, 127, 58]. Furthermore, note that there are numerous variations on the basic Kalman filter and their study was

considered beyond the scope of this research. Furthermore, depending on the nature of the subsequent processing stages, the filtering operation may in fact not be necessary.

The implemented Kalman filter closely follows the standard Discrete Kalman Filter Algorithm (DKFA) [145]. One of the conditions under which this algorithm is applicable is that systems to which the algorithm is to be applied must be amenable to a state space representation [91, 92]. Effectively this implies that the system must exhibit an essential state described by a number of *state variables*. Furthermore, there must exist sufficient linear independent first-order differential equations which are linear combinations of the state variables in order to solve for the state variables. These equations are known as the *state equations*.

The Kalman filter is therefore applicable to systems that can be described by the following two equations:

$$\textbf{State equation}: \quad \mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_{k-1} \tag{7.4.1}$$

$$\textbf{Output equation}: \quad \mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{7.4.2}$$

In the above equations $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{H}$ are matrices; k is the time index; $\mathbf{x}$ is called the state of the system or the *state vector*; $\mathbf{u}$ is a known input to the system; and $\mathbf{z}$ is the measured output. The vector $\mathbf{w}$ is called the *process noise*, and $\mathbf{v}$ is called the *measurement noise*. Note that $\mathbf{x}$, $\mathbf{u}$, $\mathbf{z}$, $\mathbf{w}$ and $\mathbf{v}$ are generally vectors, each representing a number of variables. The vector $\mathbf{x}$ contains the information about the present state of the system. This vector can however generally not be measured directly and must be estimated (which is the essential function of the Kalman filter). To perform the estimation, measurements are taken of $\mathbf{z}$, which is a function of $\mathbf{x}$ that is corrupted by the noise $\mathbf{v}$.

Another condition which must be satisfied in order for the `DKFA` to be applicable is that the random vectors $\mathbf{w}$ and $\mathbf{v}$ are assumed to be independent and exhibit normal probability distributions which are uncorrelated in time (*white noise*), *i.e.*

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

If the noise parameters do not exhibit the above characteristics, the Kalman filter will in all probability still function correctly, but will not produce optimal results [145]. For the SDDSF implementation, the *process noise covariance* matrix $\mathbf{Q}$ and the *measurement noise covariance* matrix $\mathbf{R}$ are assumed to be constant. In practice these covariance matrices may vary at each time step or measurement. There are a number of formulations for the `DKFA` equations. Alg. ⌊**7.4.1**⌋ gives the implementation as found in the SDDSF.

The operation of the algorithm can be summarised as follows. The algorithm estimates the process state by using a predictor-corrector approach. The first part of the algorithm predicts the *a priori* process state $\hat{\mathbf{x}}_k^-$ by means of the state transition matrix (which describes the state equations) and the current input. Simultaneously, the algorithm estimates the *a priori* error covariance $\mathbf{P}_k^-$. The second part of the algorithm then corrects the predictions based on the new measurement data. The first step is to compute the Kalman gain $\mathbf{K}_k$. The new measurement data is then incorporated as a delta relative to the predicted output (based on the current *a priori* state). The result is the *a posteriori* estimate of the process state given the measured real output. The last step corrects the error covariance matrix based on the newly calculated Kalman gain.

Fig. **[7.5]** gives a diagrammatic representation of the complete filtering processing. Note that in the figure, a different notation is used to represent the various components of the model. The particular notation is based on the original work by Kalman [70] and is the notation most often used when describing the Kalman filter. The notation used elsewhere in this text is based on the introduction by

Algorithm 7.4.1 DKFA

1. Store the supplied initial state estimate $\hat{\mathbf{x}}_{k-1}$ and the initial *a posteriori* error covariance estimate $\mathbf{P}_{k-1}$.
2. Perform the Time Update or Prediction step:
   (a) Project the state ahead:
   $$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{7.4.3}$$
   (b) Project the error covariance ahead:
   $$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{7.4.4}$$
3. Perform the Measurement Update or Correction step:
   (a) Compute the Kalman gain
   $$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \tag{7.4.5}$$
   (b) Update estimate with measurement $\mathbf{z}_k$
   $$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \tag{7.4.6}$$
   (c) Update the error covariance
   $$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \tag{7.4.7}$$

Bishop [145]. The Bishop notation was used in the text and in the SDDSF as the Ψ notation is less suitable for implementation in source code.

7.4.2 System Model

Generally, the system model matrix $\mathbf{A}$ can be obtained in various ways. In this case, the system model has effectively been chosen as a quadratic function $f(x)$ of the image x coordinate (refer to Section 7.2). The filter is therefore supplied with the measured estimated laser curve y coordinate for each x coordinate ($x = 0, 1, \ldots, N-1$, where N = number of image columns). The state projection matrix can thus be obtained directly from the process equation (assuming no system noise). Since there is no controlling input, there is no $\mathbf{B}$ matrix. The system equation is given by

$$\begin{aligned}\mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \\ &= \mathbf{A}\mathbf{x}_{k-1}\end{aligned} \tag{7.4.8}$$

thus

$$\begin{aligned}\mathbf{A}\mathbf{x}_{k-1}\mathbf{x}_{k-1}^{-1} &= \mathbf{x}_k\mathbf{x}_{k-1}^{-1} \\ \therefore \quad \mathbf{A} &= \mathbf{x}_k\mathbf{x}_{k-1}^{-1}\end{aligned} \tag{7.4.9}$$

Since $f(x) \in \mathbb{R}^1$, the system model matrix $\mathbf{A}$ reduces to a single variable, A, and is then given as

$$A = \frac{f(k)}{f(k-1)} \tag{7.4.10}$$

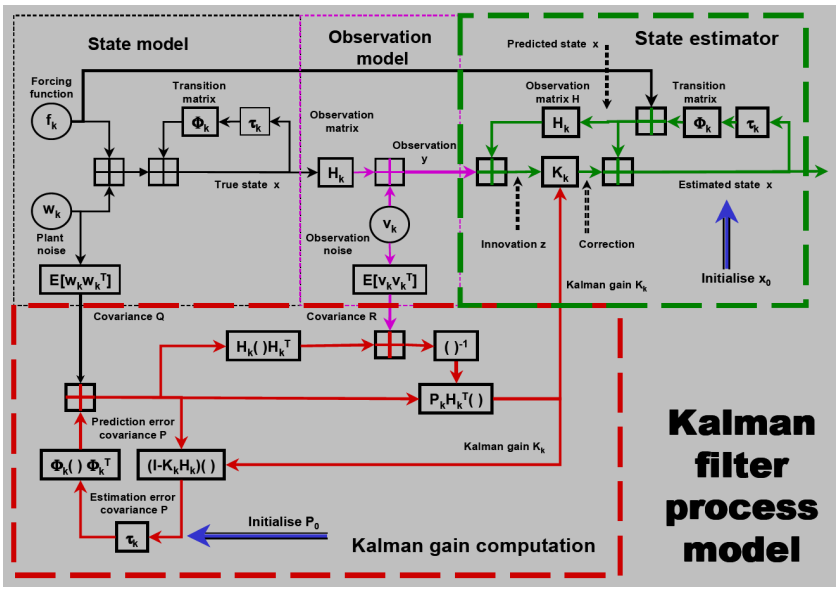


Figure 7.5: Representation of the Kalman Filter Process ([125], Fig 1).

7.4.3 System Noise

The system noise gives an indication of the accuracy of the system model as compared to the intrinsic model of the real system. If the system model is known to match the real system model exactly, the system noise parameter can be set equal to zero. In this case, the system model is known to be an approximation: the model of projected laser line should ideally be a short segment of a projected NURBS curve. In addition, the projected laser light itself exhibits a width-wise normal distribution [130]. The result is an approximately normal system noise distribution. Defects in the panel surface introduce gradual deviations of the real system model from the system model. The system noise parameter is therefore selected empirically so that the system state estimator is insensitive to the gradual defect induced deviations. The noise parameter was experimentally determined to be 0.05. The exact value is not crucial as the filter will track the system after sufficient data has been injected. The value does however impact the responsiveness to rapid deviations in the injected data and the value chosen aimed to minimise the response lag.

7.4.4 Measurement Model

For the SDDS application, the measurement is the estimated laser curve as extracted from the captured images. Hence the measurement is of the state directly and the measurement matrix is therefore unity. Since the measurement data consists of only the estimated curve y coordinate, $z \in \mathbb{R}^1$, the measurement matrix is a constant $H = 1$. Thus $z_k = x_k + v_k$.

7.4.5 Measurement Noise

For this application, the measurement noise is assumed to originate from the image capture process, as well as vibrations induced by the robot motion. The resultant measurement noise is assumed to be of a normal distribution due to the physical processes which potentially produce the noise. During the model fitting of the expected laser curve model, the algorithm makes use of the MAD estimate of

variance. This estimate is robust against large deviations (such as produced by a defect). The MAD estimate of variance is therefore a more reasonable indication of the measurement noise as compared to the sample variance. Furthermore, since the measurement noise is assumed to be due to system noise in the laser and camera, the measurement noise is taken to be time invariant. The measurement noise parameter used by the Kalman filter is therefore the MAD estimate of variance as returned by the preceding model fit operation.

7.4.6 Initial State Estimate

The initial state estimate is chosen as the average of the first two measurements. Since this represents a crude estimate of the system state at that point, the initial Kalman gain should therefore be close to unity. An initial Kalman gain of close to unity is desired in order to allow the filter to gradually track the real system. If the initial state estimate is not correct, the Kalman filter will still converge to the correct state estimate, but will require a significant number of measurements in order to do so.

7.4.7 Initial Posteriori Error Estimate

Since the initial state estimate is chosen to be close to the expected real system state, the initial posteriori error estimate can be made relatively small. In the implementation a value of 1.0^{-3} was chosen empirically.

7.4.8 Implementation and Performance

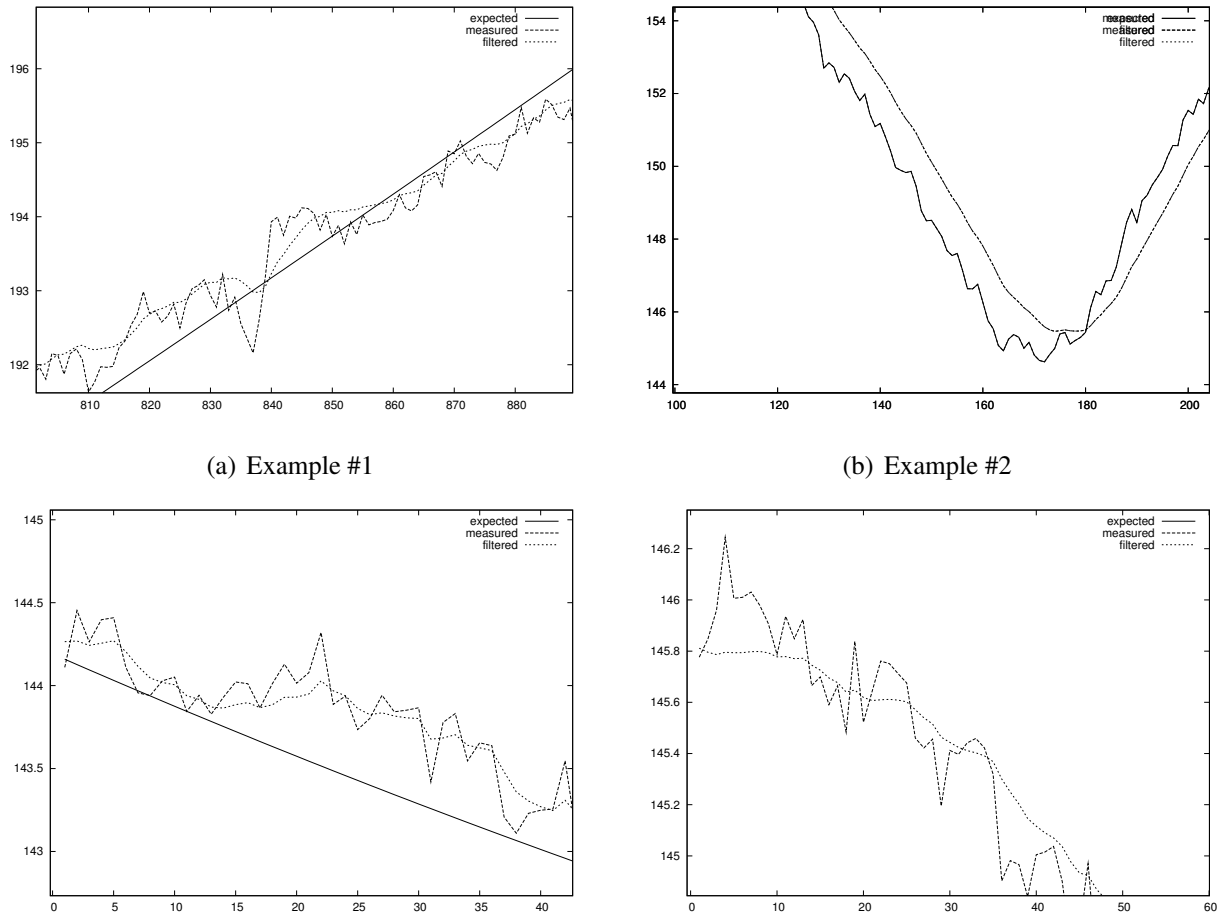
There are two versions of the `DKFA` implemented in the SDDSF. The first version is an exact implementation of the described algorithm. The second version is an optimised version, specialised for the case where there is only one state and one output variable, instead of vectors and matrices. In other words, a version optimized for the usage in the SDDS application of filtering the extracted laser curve. Fig. **[7.6]** gives an extract of a typical plot of the extracted laser curve data, together with a Kalman filtered version. The expected laser curve data is also shown. Example #2 shows the response of the filter to data representing a defect on the roof panel. Example #3 and Example #4 show the response of the filter to the initial extracted laser curve data (*i.e.* at x coordinate 0).

7.5 Deviation Vector Generation

The primary goal and desired output of the second stage processing is a vector containing the deviations of the extracted laser curve from an expected laser curve model. The first section of this chapter has focused on the generation of an expected model for the laser curve. In the previous processing stage, the measured laser curve was extracted from the captured images. The difference between the expected model and a Kalman filtered version of the extracted laser curve therefore represents a vector of delta or deviation values. Fig. **[7.7]** and Fig. **[7.8]** show plots of the deviation vectors that result from dust particles and artificial defects.

In the SDDSF implementation, instead of a simple difference calculation being used to calculate the deviations, a thresholding technique is used. Specifically, the MAD estimate of variance is used to calculate a robust standard deviation for the measured data. This threshold allows the expected deviations due to system noise to be ignored.

If the hardware infrastructure is correctly calibrated, the perceived laser curve should span the entire



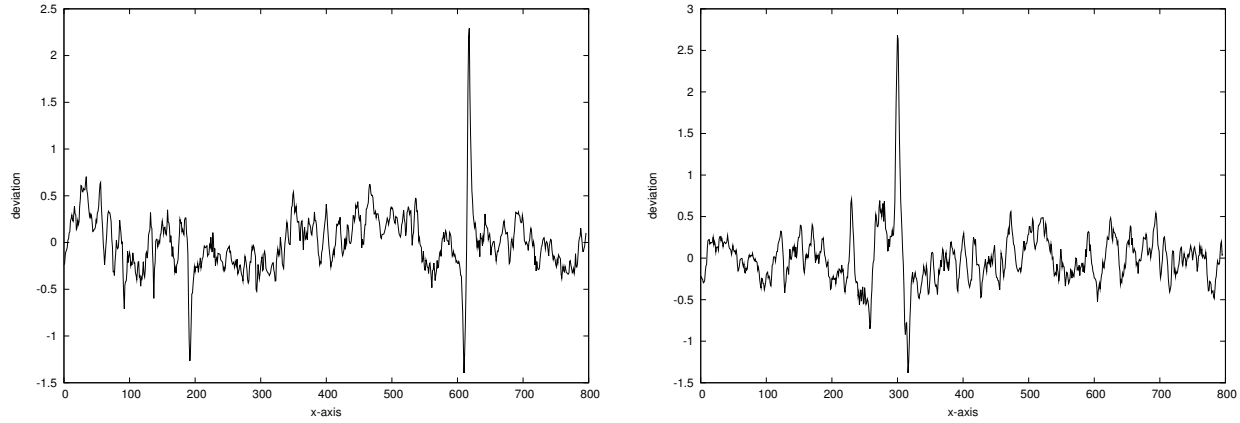
(a) Example #1

(b) Example #2

(c) Example #3

(d) Example #4

Figure 7.6: Typical results of Kalman Filtering.

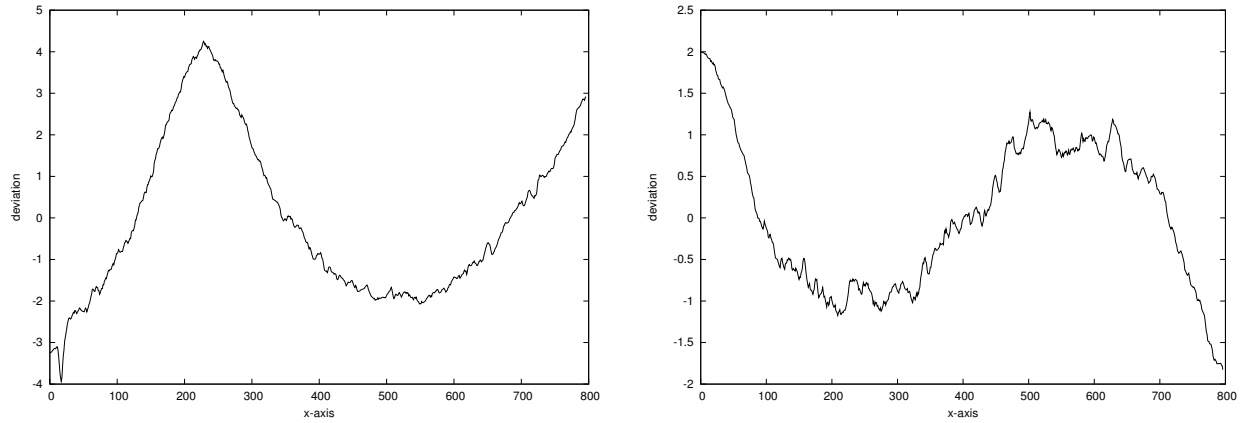


(a) Example #1

(b) Example #2

Figure 7.7: Deviations caused by dust particles.

image sensor ROI. However, if this is not the case, the deviation vector produced by the second



(a) Example #1 (b) Example #2

Figure 7.8: Deviations caused by artificial defects.

processing stage may not be valid. Due to the non-uniform light intensity distribution along the projected laser line, the projected laser line is often discontinuous at the outermost sections of the laser line and the deviations at the outermost sections are therefore invalid. To compensate for such discontinuous sections, additional filtering is performed.

The filter operates by reprocessing the captured image with the background removed (*i.e.* the output of Section 6.6). The pixel values in the image along the laser curve (as defined by the extracted laser curve vector) are evaluated. The filter scans the image from the left and locates the start of a sequence of 10 contiguous laser curve pixels which have pixel values of 200 or higher. These pixels are assumed to represent valid laser line pixels. This process is repeated from the right of the image in order to locate the end of the laser line. The two control parameters for the filter were determined by experimentation. The start and end locations found in this manner, together with the deviation vector, form the output of the stage 2 processing.

7.6 Chapter Summary

This chapter has described the model and mechanism implemented in the SDDSF to produce an expected laser curve model. The chapter has also discussed the Kalman filtering operation performed in order to remove system noise from the stage 1 output. The chapter concluded by describing the generation of the vector of deviations of the filtered data from the expected laser curve model. This vector represents the output of stage 2 of the processing pipeline.

The following chapter will investigate the analysis of the vector of deviations, which aims to determine whether or not the deviation vector represents one or more defects on the panel.

⟵⋇⟶

CHAPTER 8

Stage 3 Processing

The purpose of the third processing stage is to determine if the vector of deviations produced by the second processing stage represents one or more defects. As discussed in Chapter 1, there exists no standard which defines precisely what constitutes a defect. This lack presents a serious problem in the creation of a mechanism to automatically determine if a particular vector of deviations represents a defect. The primary task of the stage 3 processing is therefore that of classification. Although a particular approach will be described in this chapter, the topic is still under investigation and further research may result in an alternate approach being implemented.

8.1 Chapter Outline

The chapter begins by discussing possible approaches to the problem of determining whether or not a vector of deviations represents a defect or damage. The implementation of a possible approach, a neural network based classifier, is discussed. The feature extraction and data generation required to train the neural network is described. The chapter concludes with a brief discussion of ongoing work, *e.g.*, the post-processing of the neural network classification to track the presence of a particular defect as the scanning system passes over the defect.

8.2 Possible Approaches

A simplistic approach could be to use a single metric such as the square error of the deviations in the deviation vector as an indication of a defect. With such an approach, if the current deviation vector exhibits a square error above a threshold value, the deviation vector is considered to represent one or more defects. This approach does have the advantage of being independent of the nature of the defect as the square error does not incorporate any knowledge of the distribution of the deviation induced by a defect in the deviation vector. The selection of the threshold which is taken as a defect is however problematic. Some panels tend to have significant oil residue (introduced by the production process) and accumulated dust. These can result in a square error that has the same magnitude as the threshold selected to represent the presence of a defect.

The ideal approach must take into account the relative shape of the deviations in the deviation vector caused by a defect. At the same time the approach must be insensitive to position and orientation variations of the defect deviation shape. Another requirement of the final system, which has an impact on the selection of classification technique, is that of adaptability. The system is required to dynamically adjust the classification based on guidance from domain experts during operation. This requirement significantly complicates the stage 3 processing, but does suggest specific techniques that

are known to work well in such a learning environment.

The most obvious learning technique, from the field of computational intelligence (CI), is that of neural networks [118, 55, 37, 35, 57, 86, 110, 14, 54]. A possible approach is therefore to train a neural network (NN) to accept the vector of deviations and produce a classification (defect or no defect). Provided that the neural network is sufficiently complex, the neural network would allow classification that is relatively insensitive to the nature of the defect. Alternatively, the vector of deviations could be preprocessed in order to provide positioning and scale invariance.

However, in order to train the neural network, sufficient training examples are required. Apart from the number of training examples, the samples must be representative of the typical defects to be detected. This latter requirement is very problematic: no examples of real damages are available. There are a number of examples of real natural defects.

8.3 Feature Selection

Feature selection is the process of identifying the most useful (predictive) attributes in a data set. The requirement for insensitivity to the relative position of the defect in the input vector led to the decision to investigate the use of a Fourier transform of the deviation vector as the feature vector. Note that if the vector for which the Discrete Fourier Transform is to be calculated contains only real floating-point numbers, a new vector of complex numbers of the same length must be used as input to the Fourier Transform. The real components of the complex vector are initialised to the floating-point numbers in the original vector and the imaginary components of the complex vector are initialised to zero. An initial implementation that made use of a Discrete Fourier Transform (DFT) proved to be computationally expensive. The DFT implementation allowed the Fourier Transform of a vector of 1024 complex floating-point values to be calculated at a rate of 7.795 operations per second.

This is a well known result and historically led to the development of the Fast Fourier Transform (FFT) [18, 108, 17] by James W. Cooley (1926–) and John Tukey (1915–2000) in 1965 [25]. Considerable research has since been done on efficient implementations of the FFT. The basic FFT was implemented in the SDDSF and proved to be a considerable improvement over the DFT. The basic FFT implementation allowed the Fourier Transform of a vector of 1024 complex floating-point values to be calculated at a rate of 11904.053 operations per second. However, the FFT implementation requires that an input vector of radix-2 be used (integer powers of two). In other words, vector sizes of $2, 4, 8, ..., 512, 1024, ...,$*etc.* are required. For an input vector of 800 floating-point numbers, the size of the deviation vector, the vector must be padded with zeros to 1024 floating-point numbers. This results in unnecessary computation and is undesirable.

The ideal would be to use the FFTW ("Fastest Fourier Transform in the West") library[1], a free collection of fast C routines for computing the Discrete Fourier Transform in one or more dimensions. It includes complex, real, symmetric, and parallel transforms, and can handle arbitrary array sizes efficiently [43]. FFTW is generally accepted as being one of the most high performance FFT implementations available. Unfortunately, FFTW is published under the GNU General Public License (GPL). Since the GPL requires that the source code of any derivative work be released, the use of FFTW would necessitate the release of the SDDSF source code to the general public. Being a commercial venture, this is not possible and an alternative to the FFTW was required. Due to the complexity involved and time constraints, developing such an alternative would be infeasible. The implementation that was eventually used is based on public domain Fortran code by Richard C. Singleton [109]. The Singleton implementation allows arbitrary size input arrays, overcoming the computational waste

[1]Primary site: `http://www.fftw.org`.

introduced by the radix-2 FFT implementation. The Singleton implementation allowed the Fourier Transform of a vector of 1024 complex floating-point values to be calculated at a rate of 13157.029 operations per second. On a vector of 800 complex floating-point numbers, the Singleton implementation could perform 13811.296 Fourier Transforms per second.

Since the input data vector is real, and the FFT implementation expects a complex data vector, the technique described in Section 9.4.2 of [108] was used to calculate the FFT more efficiently. This technique allows one to calculate the $2N$-point FFT of a real sequence using a N-point complex FFT. When performing the FFT, an additional step is required to recover the actual complex sequence ("unscrambling"). In the SDDS application, the neural network does not attach *a priori* knowledge to its inputs. Hence the FFT result can be directly supplied to the neural network and the network will automatically handle the "unscrambling" normally required when this technique is applied. This approach saves computational costs and has been verified to work correctly. The Singleton implementation is able to perform 30118.576 Fourier Transforms per second when the technique described in this paragraph is used on a vector of 1024 real floating-point numbers. For reference, the FFTW implementation is able to perform 31248.047 Fourier Transforms per second on a vector of 1024 real floating-point numbers. The Singleton implementation therefore exhibits similar computational performance to that of the FFTW implementation.

8.4 Feed-Forward Neural Network

The network architecture chosen for this implementation is the standard Feed-Forward Neural Network (FFNN) with a single hidden layer. A FFNN is a layered network in which each layer only receives inputs from previous layers.

8.4.1 Neural Network Implementation

The neural units are stored as a vector of weights and outputs, where the network topology is created as a virtual mapping of weights and outputs. This approach allows optimisation algorithms which assume that the weights are a vector to operate directly on the weight vector. Optimisation algorithms which require knowledge of the topology of the network (*e.g.*, back-propagation) make use of the virtual view of the network as provided by the instantiation of the particular "network architecture". The use of inline functions to create the virtual view allows optimal performance given the generic nature of the network architecture implementation.

The output value of each neuron is based on the resultant output of the neuron activation function (which in turn takes as input the weighted sum of all the inputs to the neuron). During the preliminary research, the author found that most implementations of FFNNs make use of the well-known sigmoid function:

$$f_{AN}(net - \theta) = \frac{1}{1 + e^{-\lambda(net-\theta)}} \tag{8.4.1}$$

The parameter θ controls the bias of the function and the λ parameter controls the steepness of the function. It was found (as stated in [37] page 20) that the majority of researched implementations use a value of $\lambda = 1$. To decrease the complexity of this implementation, it was decided to use the sigmoid function with the recommended value, $\lambda = 1$. The bias parameter was incorporated into the input and weight vectors to create the *augmented vectors*, as described in [37], Section 2.4.1.

8.4.2 Network Optimisation Algorithms

The implementation makes provision for a number of optimisation algorithms to be used. The majority of the FFNN implementations make use of the Gradient Descent (GD) Optimisation [55, 37] to train the FFNN. The disadvantages of this method, being a local optimisation method, include sensitivity to weight initialisation, and the tendency to optimise to local minima. To improve the performance of the GD optimisation, a momentum term and variable learning rates are often added [55, 35]. The implementation in SDDS therefore makes use of GD with momentum and variable learning rate. Appendix G provides the error gradient formulation implemented in the SDDSF. Certain aspects of the implementation will be discussed in more detail below.

Weight Initialisation

Since the optimisation method chosen for this implementation is a local optimiser, the initialisation of the weight is of particular importance. Two aspects are important of with regards to the weight initialisation. Firstly, the range of the initial weights should be such that the neurons do not saturate. Secondly, the weights should be sufficiently random in nature. This is particularly important across successive trials (of a simulation) in order to ensure that the intrinsic nature of the NN configuration (NN architecture, optimisation algorithm and operational parameters) is correctly characterized. In other words, it ensures that the configuration does not by chance converge to an optimal point due to predictable weight initialisation. This would falsely improve the performance of the configuration as compared to intrinsically more appropriate configurations.

For this implementation, the first aspect was handled by following the recommendation in [146]. The randomized weights are thereby uniformly, randomly selected from the range:

$$-\frac{1}{\sqrt{fanin}} \leq w_0 \leq \frac{1}{\sqrt{fanin}} \tag{8.4.2}$$

where $fanin$ is the number of inputs to the neuron, including the bias input as applicable. To ensure the desired randomness properties, the MT19937 generator discussed in Section 7.3.3 was used.

Learning Rate and Momentum

Since a local optimisation method (*i.e.* GD) is being used, the learning rate is critical to the success of the NN training. A too large value will result in potential minima being stepped over, or can result in an oscillatory effect. A too small value can unnecessarily lengthen the training process or result in the optimisation being trapped in a local minimum. The ideal is therefore to dynamically adjust the learning rate. A number of possible adaptive learning rate techniques have been proposed [104, 22, 67, 141, 29, 81, 119, 83]. The particular technique implemented successfully in this case was a linear decay of the learning rate. The learning rate was initialised to 0.75 and decreased linearly to a final value of 0.09 after 500 training cycles.

During *stochastic learning*, the error derivatives have fluctuating changes in sign that implies positive and negative weight adjustments. In this context, stochastic learning refers to the training methodology where training patterns are randomly chosen from a training set for each pattern presentation to the neural network. A single training cycle is taken as the random presentation of all training patterns to the neural network. To ensure a smooth search path towards the minimum, a momentum factor is added [104, 39, 135]. A static value of 0.9 is typically used [35, 37]. Although the momentum factor can also be adjusted dynamically, this was not deemed necessary in this case as the optimisation process proved to be well behaved.

Stopping Conditions

The most basic stopping condition is to stop when a certain number of training cycles have been performed. In most applications this approach would however not be sufficient as it gives no indication of the state of the training process. An improved stopping condition would be to stop the training process when an optimal state has been reached. Since the goal function is the SSE, it may be used as an indication of an optimal state. A condition that gives a more holistic view (over all training patterns) is the MSE [37] (page 85). One can define a threshold and once the MSE falls below this level, the training process is terminated. There is however no obvious method whereby such a threshold can be derived. In this particular application, the optimisation process proved to be well behaved and deterministically converged to a nominal validation error. More complex stopping conditions were therefore not deemed necessary.

Scaling and Normalisation

The activation function discussed in Section 8.4 determines the range of the target values that are to be used. For the SDDS application, the architecture chosen designates an output neuron for each of the classifications. Furthermore, the sigmoid function has an active output region of $(0, 1)$. Target values must therefore be made to fall within this range and the desired output is considered to be a 'yes' value as compared to a 'no' value, where only one output should say 'yes' at any given time. The 'yes' is then taken to indicate a classification. This is generally known as *one-hot encoding* (and is the one most used when implementing state machines at the gate level) [142] (page 389). Since the sigmoid function approaches 0 or 1 asymptotically, values of 1 for 'yes' and 0 for 'no' would always result in an error. The error will cause the weights to be adjusted and hence weight adjustment will always occur. For this reason, the values 0.1 for 'no' and 0.9 for 'yes' have been chosen for the SDDSF. These values are at the points where the gradient of the sigmoid is low and sufficiently distanced from each other. The exact values chosen for the target values are not critical.

Based on the same reasoning, classification decision thresholds can be defined at a point close to the target output values of 0.1 and 0.9. For example, if all target values are less than 0.3 or greater than 0.7, the output of the NN is considered to have performed a classification. If any of the outputs are in the range $(0.3, 0.7)$, the NN is then considered to be *undecided* for the current input pattern. Generally this will be considered the same as an erroneous identification / classification.

The discussion above has focused on the scaling / normalisation of target / output values. However, the scaling / normalisation of the input values must also be performed in order to improve training performance [55, 37, 35, 14]. It has been experimentally confirmed in the SDDSF implementation that using mean centering and variance scaling (as described in [35, 14]) dramatically improves the training performance. By this method, the elements of the feature / input vector are normalised by the formula:

$$f_i' = \frac{f_i - u_i}{\sigma_i} \tag{8.4.3}$$

where f_i and f_i' are the initial and final values of the i-th feature / input vector element respectively, u_i is the mean and σ_i is the standard deviation of all the i-th elements in the available patterns.

However, once training has been completed and the resultant neural network is to be applied in an online, real-time situation, the scaling and normalisation must still be performed. To accomplish this, not only are the neural network weights saved upon the completion of training, but the scaling and normalisation parameters are also saved. This however increases the importance of a comprehensive training data set to ensure that the scaling and normalisation parameters are correctly estimated.

8.4.3 Implementation Validation

To validate that the implemented FFNN was correct, the FFNN was applied to a number of datasets from the UCI Machine Learning Repository [90]. Specifically, the `glass`, `iris`, `ionosphere`, and `thyroid` datasets were evaluated. The results obtained corresponded with previous results obtained [48] and indicated that the FFNN produced the expected results.

8.4.4 Data Set Generation and Classification Performance

In order to train the neural network, a training data set of sufficient size is required. During training, another data set is also required in order to provide a more accurate estimate of the generalisation ability of the network. As has been mentioned, the availability of suitable samples for the data sets is a significant problem. Initially the deviation vectors captured over known defects were examined. From these vectors, in a semi-manual operation, multiple polynomials were fitted in order to approximate the defect. Using these polynomials, together with suitable randomisation, the idea was to artificially generate defects that resemble the real defects found. Preliminary results indicated that the neural network was able to perform the classification learning and the generalisation error was found to be as low as 1.8%.

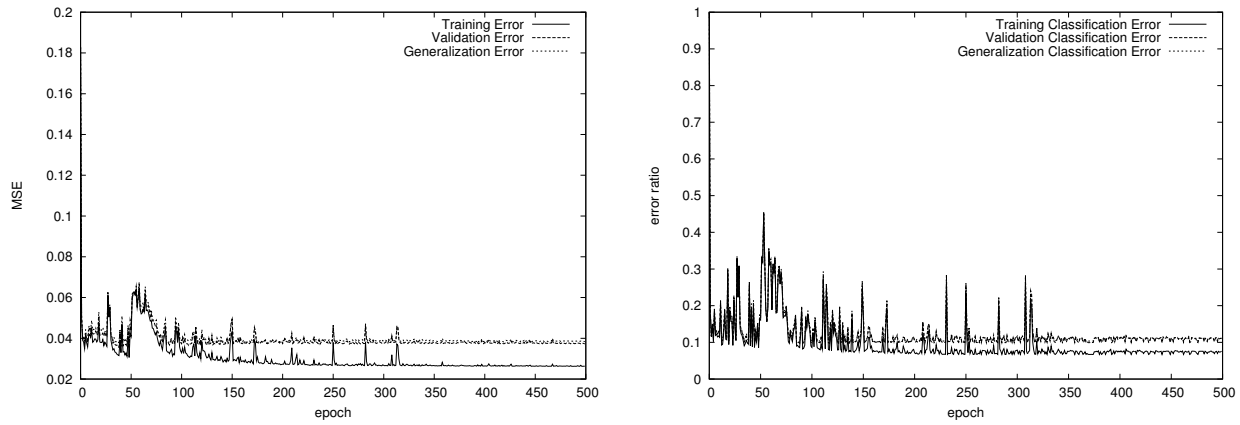
However, once the classification system was integrated into the scanning system, the system failed to detect defects correctly. After an investigation the conclusion was that the artificially generated defects were not representative of the real defects. The only option was therefore the make use of real defects for the training data. Sample images were therefore systematically collected with defects (of various sizes and types), images with dust and valid images. The total size of the collected images (93435) amounted to $10\,GB$. Once processed, the result is a pattern data set $600\,MB$ in size. The processing of the raw images to produce the pattern data set requires approximately an hour on the development machine.

Training with the resultant data set proved to be more successful. A single training run consisting of 500 training cycles requires approximately an hour to complete. The classification system appears, based on empirical evaluation under operating conditions, to correctly classify real defects. Fig. **[8.1]** gives an example of the typical training performance profiles encountered. The generalisation error was found to be in the region of 10%. The generalisation error ratio is measured by evaluating the NN classification when supplied with unseen (during training) data patterns. These unseen patterns form the generalisation data set, which contains a total of 6662 patterns (10% of the available patterns). Of these patterns, 1196 are known to represent defects.

The resulting generalisation error is expected since some of the samples can be considered borderline, they represent defects just above the system noise level. Primitive live tests with the classifier indicated that the classifier was able to correctly defect the known defects on the roof panel. Further evaluation is however required in order to formally quantify the performance of the classifier. The generated data sets are also not considered to be truly representative of all defects and damage found in practice.

8.5 Filtering of Classification

The output of the FFNN represents the classification of a single image. Clearly the classification must be correlated with previous and subsequent images in order to produce a meaningful determination whether or not a defect is currently being scanned across. In the current implementation of



(a) Mean-Square-Error

(b) Classification Error Ratio

Figure 8.1: Typical training performance of FFNN classifier.

the SDDSF, a primitive windowing filter is used to detect a sequence of 5 consecutive classifications indicating a defect. When the filter indicates that a defect is being transversed, the coordinates of the detection are stored (as obtained via Ethernet-RSI, see Section 5.5.3). The defect coordinates are stored in an object-oriented container defined by the `sddsdefect` module. The defect coordinates can be considered to be the output of the state 3 processing.

Current work on the SDDSF entails the implementation of the marking process. The method being implemented will, once a scan is complete, select the marker as tool point and move the robot to the defect coordinates (obtained from the `sddsdefect` module). The ink-jet printing system will then be triggered to produce a mark on the panel which is in close proximity to the detected defect. Described in more algorithmic terms, the planned sequence of events are:

- The server initially starts in a wait state.
- The robot, triggered by the cell PLC, initiates the scanning process by establishing a connection with the processing server after it has moved to a home position.
- Once a connection has been established, the robot moves to the "start of scan" position on the roof.
- When the start point is reached, the robot sends a start of scan command to the server via the serial communication channel.
- The server starts capturing images and processing the results whilst the robot performs the profile scan.
- When the server detects a defect (based on a sequences of "detections"), it records the defect information. At the same time, the server checks for an end of scan command from the robot.
- When the robot has reached the end of the profile scan, it sends an end of scan command to the server and the robot enters a wait state.
- The end of scan command causes the server to leave the scan state. If more than the threshold number of defects have been encountered (4), or (future addition) a non-workable defect has been identified, the server sends a "reject" command to the robot.

- If less that the threshold defects have been detected, the server sends a "move to position" command for every defect. When the robot acknowledges that the position command has completed, the server triggers the marker.
- If no defects are found, the server sends an "accept" command to the robot, which will move to a home position and terminate operation. In the plant environment, the robot will set a digital I/O in the Interbus register space, which will signify an "accept" to the cell PLC.
- If the robot receives the "reject" command, it currently does nothing except moving to the home position and terminating operation. Eventually it will set a digital I/O in the Interbus register space, which will cause the cell PLC to trigger the cell robot which normally places the panel on the car to instead discard the panel.
- Once the server has marked all the defects, it sends a termination command to the robot and enters the wait state, again waiting for a new start of scan command.
- The robot, once the termination command has been received, will move to the home position and end operation. In the plant environment, a digital I/O will be used to flag a "rework" to the cell PLC.

8.6 Chapter Summary

This chapter has described an initial attempt at classification of the output of the state 2 processing. Although initial results are promising, further refinement and evaluation is required in order to quantify the implementation. A number of other aspects must also be addressed and the following chapter will discuss these aspects in more detail.

CHAPTER 9

Conclusion

The preceding chapters have described the development of the production prototype Surface Defect Detection System. This chapter will evaluate the progress relative to the requirements specification of the production prototype. Certain developments and extensions are currently receiving attention and these will be discussed. The remaining activities required in order for the system to be production ready will subsequently be sketched.

9.1 Review of Requirements Definition

In Section 1.4 the requirements for the production system were supplied. This section will review the requirements relative to the current progress. When the development of the production prototype was initiated, it was realised that certain of the requirements would require additional research. The solutions to these requirements was therefore not known and hence the code infrastructure to produce these solutions was also unknown. As a result one of the primary goals of the SDDSF was to provide a generic image processing and mathematical environment, as compared to a specialised solution to the known and understood requirements. Such a generic framework would then simplify the development of any additional functionality required.

9.1.1 Ability to Scan Various Types of Roofs and Flanges

The requirement states that the system must scan the outer surface of various types of roofs in order to detect defects such as bumps and dents on the roof and any flanges. The various roof panels differ with regards to additional structures included in the roof. As an example, the sun roof version contains the requisite open area for the sun roof, as well as additional bracing (see Fig. **[9.1]**).

The current implementation of the SDDSF and KRL scanning program assume the standard roof panel. To support the sun roof panel, the essential requirement is for the software to be aware of the positioning of the sun roof area. Hence, knowledge extracted from the CAD model of the panel is required. It was realised that similar knowledge would be required in order to support other panel types, such as door panels. A decision was therefore made to focus on the extraction of such data from the CAD model. The CAD models are however primarily available in the proprietary CATIA format, as produced by the CATIA CAD system (developed by Dassault Systémes'). Being a closed format, there is no standard, freely available library to read models in this format. Considerable time was invested in examining the possible export formats supported by the CATIA CAD environment.

The Initial Graphics Exchange Specification (IGES) [139] format was found to be supported by CA-

Figure 9.1: Sun roof panel example.

TIA and the specification was available (for a fee) from the U.S. Product Data Association (US PRO)[1]. Due to the specification being available, this format was selected as the best candidate and the specification was purchased. The basic model reading module, `sddsiges`, was subsequently developed and could successfully read the CAD model of the standard roof panel.

The standard roof panel CAD model analysed was found to contain 4888 entities, of which 278 were Rational B-Spline Surfaces. However, it was found that sub-assemblies such as the sun roof were handled by means of associativity entities. It became evident that in order to perform the automated extraction of the effective surface to be scanned, considerable post-processing of the CAD model entities would be required. The development time of the required code was estimated to be considerable and the development was postponed.

In summary, in order to handle various panel types, the primary work required entails the completion of the IGES model handling code and this task has been planned for a subsequent project phase. It is estimated that comparatively few changes will be required in the remainder of the SDDSF in order to handle scanning of the sun roof panel.

9.1.2 Categorisation of Defects as Workable or Non-Workable

The requirement states that identified defects must be categorised as either workable or non-workable. Such categorisation relies on the stage 3 processing and preliminary support has been added. The categorisation mechanism will be finalised as part of one of the current activities.

9.1.3 Marking Criteria

The requirement states that if less than a predefined threshold of workable defects have been found, the roof is “accepted” and the defects are marked via a small ink dot. The ink-jet printing system that has been integrated can produce the desired mark based on a call to a trigger function. In Section 3.9

[1]Primary site: `https://www.uspro.org/`.

the motivation was given for separate scanning and marking phases. Tests were however performed to evaluate the "marking whilst scanning" strategy. The problems described in Section 3.9 were encountered and confirmed the need for separate phases. In order for separate phases to be possible, the activities listed in Section 9.2.2 needs to be completed.

9.1.4 Rejection Criteria

The requirement states that if more than the threshold of workable defects have been found, the roof is "rejected". This requirement translates to a simple logic decision made based on the result of the stage 3 processing and will be incorporated as part of the current activities.

9.1.5 Scanning Time Specification

The requirement states the entire scanning, processing, deliberation and marking process must be completed within $30\,s$. This requirement cannot be met with the current system without loss of system sensitivity. In Section 2.5.2 it was shown that the capturing of images to provide 100% coverage during a scanning operation requires approximately 3 minutes. Similarly, in Section 5.4.5 it was explained that the robot controller architecture places significant limits on the speed of the scanning motion. These two factors, and the resultant consequences, prohibit scanning times of less than approximately 3 minutes.

In order to achieve sub-minute scanning times, a considerable financial investment is required in order to add multiple laser and camera configurations. With proper design, this will allow the panel to be scanned in a single pass. It should be noted that additional Linux servers would be required in order to match the increase in image sources.

9.1.6 Additional Rejection Criteria

The requirement states that if there is any doubt about the workability of a particular roof, the roof must be "rejected". This requirement translates to a simple logic decision made based on the result of the stage 3 processing and will be incorporated as part of the current activities.

9.1.7 Production line integration

The requirement states that the resultant status of "accepted" or "rejected" must be communicated to the production line PLC. In the production line environment, the PLC is integrated with the cell robots via Interbus (see Section 5.5.1). In order to communicate the "accept" or "reject" information to the cell PLC, additional Interbus digital registers will be defined and utilised. The SDDSF will set the appropriate digital register in the KRC via the Ethernet-RSI interface (by means of the `ST_MAP2DIGOUT` RSI object). The KRC digital register will in turn be replicated to the cell PLC via Interbus. The addition of the Linux server to KRC signalling will be performed during the final stages of one of the current activities.

9.1.8 Operating Environment Specification

The requirement states that the process must function correctly in the environment as found in the production line at BMW Rosslyn, Zone 13. Specifically, the system must be robust against ambient levels of light, wind, dust, temperature and vibration as exist in the target environment. A number of

tours of the BMW plant was performed in order to obtain estimates of the ambient environment where the system must function. The environment was found to be of a controlled nature, air conditioning and artificial light being present. If one compares the environment at the Groenkloof location with the plant environment, the ambient levels of light, wind, dust and temperature at the Groenkloof location are of a more problematic nature. The levels of vibration experienced in the production plant are however likely to be more problematic.

Of particular note are the artificial lighting conditions at both locations. The artificial light in the plant environment is produced by powerful halogen lamps mounted high above the work area. In contrast, low hanging standard fluorescent tubes produce the artificial light at the Groenkloof location. Although measurements indicate that the ambient light levels are similar, the proximity of the lighting at the Groenkloof location results in a more challenging environment due to the greater intensity image highlights produced.

On the other hand, spot welding is performed in the plant environment and this may result in random images being captured with higher background illumination. The SDDSF however incorporates dynamic background estimation and can compensate for fairly large variations in background illumination (as discussed in Section 6.6.4). Furthermore, if the background illumination exceeds an acceptable value, the image will be rejected. The rejection of isolated images will not significantly impact the functionality of the system as a whole. There is also the possibility of using vertical light shades to shield the system's cell from the cells where spot welding is performed.

In summary, except for vibration, the production prototype system has been developed to function under environmental conditions which are considered to be more extreme than the anticipated operational conditions. The impact of vibration will however need to be investigated and it is not clear at this stage how this should be done.

9.2 Current Activities

At the time of writing, a number of development activities were being performed in order to complete the current project phase. Once the current phase is complete, the majority of the development work of the project will be completed. The subsequent phases will primarily entail evaluation and optimisation of the system.

9.2.1 Real-time Orientation Control

In Section 5.4.6 it was explained that the resultant profile of the roof when mounted on the scanning table does not match the CAD model of the roof. A mechanism was described whereby the extracted CAD profile could be algorithmically warped in order to more closely match that of the mounted roof panel. Although this warping allows the majority of the roof panel to be scanned, there are areas on the roof where the captured laser line does not appear near the centre of the image. In a number of cases, the laser line does not appear in the image at all, resulting in a failure to detect any defects in those areas.

Since the new robot interaction modules, RSI and Ethernet-RSI, have become available, there is now the possibility to dynamically correct the orientation of the gripper during scan operations. The principle is to make use of the estimated x and y centroid information extracted during stage 1 and stage 2 processing in order to dynamically calculate an orientation angle correction factor. In the KRL scanning program, a RSI `ST_PATHCORR` object will be instantiated and appropriately linked, which allows real-time access to the robot path correction system. The correction factor will thus be used

to dynamically change the gripper orientation as to ensure that the laser line captured in the images remains approximately centred.

In retrospect, this particular modification may be the single most significant change in the current system. Without such dynamic correction, the system is very sensitive to the correct placement of the panel on the scanning table.

9.2.2 Defect Marking Motion Control

As has been motivated in Section 3.9 and Section 9.1.3, it is only feasible to commence defect marking subsequent to the panel scanning being completed. The aim is therefore to make use of the RSI `ST_SKIPPTP` object to initiate direct PTP operations in order to perform the marking. There are however unanswered questions as to the implementation. Specifically, there is a question as to exactly what the interaction of an executing KRL program with motion commands with concurrent RSI motion objects are. There is no indication in the module document as to the result of such concurrent use.

Currently it is estimated that a mutually exclusive control mechanism would be needed to prevent any such concurrent usage. The principle will be that once the scan operation has been completed, a digital signal register will be used by the scanning KRL program to trigger the marking operation on the Linux server. The KRL program will then enter a "busy wait" until the Linux server sets another digital signal register to allow the KRL program to continue.

9.2.3 Stage 3 Processing Refinement

As indicated in Section 8.5, Section 9.1.2, Section 9.1.4 and Section 9.1.6, certain additions and refinements are necessary to the stage 3 processing stage. Although the modifications necessary are largely known, there is more than one possible implementation that may deliver the desired result. The modifications have therefore been postponed until the work described in Section 9.2.1 and Section 9.2.2 above has been completed in order to allow direct evaluation of the alternatives.

9.2.4 Database Integration

In Section 4.14 the selection and current integration of the database system was described. As was mentioned, the exact data to be stored in the database must be finalised. This step has been postponed until the completion of the work described in Section 9.2.3 above. Upon the completion of the above step, the system as a whole will be functional and able to produce realistic data for complete scans. At that point, the integration of the database system will be the most logical and allow the evaluation of the system operation.

9.2.5 Optical Filtering

During the investigation into the background removal (Section 6.6), it was realised that an optical band-pass filter may significantly improve the SNR of the system. The camera sensor element essentially integrates visible light of all wavelengths as the pixel values. The light of interest is however limited to a narrow band centred at $670\,nm$, the wavelength of the laser light. Clearly, the addition of a narrow band-pass optical filter with pass-band centred at $670\,nm$ would drastically increase the image contrast. The effect of sporadic highlights and spot welding flashes would also be significantly reduced. A potential source of such a filter has been located and an attempt will be made to obtain

the filter.

9.2.6 Project Milestone Demonstration

In order to officially terminate the current project phase, a formal demonstration of the system will be held. Representatives from BMW, including management, will be invited to attend. Similarly, representatives from the University of Pretoria will be invited. Based on the demonstration, the BMW representatives will signify the completion of the phase and authorise the commencement of the next phase of the project.

9.3 Future Activities

Before the commencement of the production prototype developed, a project plan was developed. As the project progressed, the progression was found to deviate from the project plan and a number of revisions were made to the plan. Before the next phase of the project is initiated, a new project plan will be developed. A large number of the activities which were initially scheduled for later project phases have already been completed. The new project plan will therefore differ considerably from the original.

9.3.1 System Evaluation and Optimisation

The evaluation and optimisation of the system was initially specified as an early project phase. The current system has however undergone significant development, and the evaluation and optimisation were therefore postponed until the system development was complete. The following list gives an indication of the aspects that need to be evaluated and optimised:

- **Evaluate minimum scanning time**. Although theoretical calculations have been done to estimate the minimum scanning time using the current system, the time should be empirically confirmed.
- **Optimize scanning speed**. There are a number of minor modifications that could be made to improve the scanning speed. For example, by decreasing the degree of coverage, the total amount of image processing required can be decreased.
- **Evaluate performance on example defects**. Originally it was envisioned that the system could be evaluated by applying it to examples provided by BMW. It has since become clear that there are no examples. The evaluation of the system will therefore require careful planning.
- **Investigate the effect of oily residue and dust**. In particular, the sensitivity of the system to various degrees of oily residue and dust should be quantified.
- **Investigate performance under different light conditions**. This is of lesser importance as the target environment is largely ambient in nature. Effects such as those produced by spot welding should be investigated, possibly by using an arc welding set, and / or a camera flash.
- **Investigate performance under vibration and different temperatures**. The evaluation of the system under various temperature conditions is fairly straight forward. The vibration aspect is however problematic and will require careful planning.

9.3.2 Support for Multiple Panel Types

In Section 9.1.1, the issue of panel types was raised. As mentioned, the reading of the SDDS IGES module is central to the solution, and hence the primary focus of this work. Additionally, detection of defects on flanges must be investigated. Due to their shape, the standard detection approach cannot be used on the flanges. Instead of using simple polynomial models, the current view is that pre-recorded models of correct flanges would provide the optimal solution. This aspect will however require further research. Clearly this aspect will also impact the required scanning time.

9.3.3 Defect Analysis, Mining and Reporting Facilities

Once the database integration is complete, functionality must be added to allow the analysis, data mining and report of the contents of the database. The exact nature of these facilities is yet to be determined.

9.3.4 Plant Integration

An integration plan must be developed that identifies all differences between the development environment (Groenkloof) and the target installation environment (Rosslyn). The plan will detail the steps required to ensure all differences are addressed and their effect evaluated. Subsequently, the plan must detail the integration procedure and provide for contingencies where necessary.

9.4 Summary

This document has described the development of a production prototype system for the detection of defects and damage on motor vehicle outer surfaces. Preliminary indications are that the developed system is considerably more robust and sensitive than the initial prototype. During the development, specific emphasis has been placed on producing a high quality implementation, suitable for use in a production environment.

The system is however not ready for integration and the steps required before integration have been discussed. The majority of the work required will however be the evaluation and optimisation of the system, for which careful planning must be done. The results of such an evaluation are eagerly anticipated as they represent the culmination of a year's development effort. There is an expectation (based on results seen during the development) that the evaluation will show the system able to match, and possibly even exceed, the ability of human operators to detect the subtle natural defects.

Appendices

APPENDIX A

Nomenclature

This appendix lists the abbreviations and acronyms found in this text and in literature that relates to the topics that are discussed.

Acronym	Description
AC	Alternating Current
ABB	Asea Brown Boveri
ACID	Atomicity, Consistency, Isolation, Durability
ACK	Handshake packet indicating a positive acknowledgement
ADC	Analog to Digital Converter
AGP	Accelerated Graphics Port
AI	Artificial Intelligence
AIA	Automated Imaging Association
API	Application Programming Interface
ATLAS	Automatically Tuned Linear Algebra Software
BCS	Base Coordinate System
BLAS	Basic Linear Algebra Subprograms
BMW	Bayerische Motoren Werke / Bavarian Motor Works
bps	Bits per second
BE	Business Enterprises
CAD	Computer-Aided Design
CATIA	Computer Aided Three-dimensional Interactive Application
CCD	Charge Coupled Device
CI	Computational Intelligence
CMOS	Complementary MOS (family of IC)
CPU	Central Processing Unit
CSMA/CD	Carrier-Sense-Multiple-Access with Collision Detection
CTE	Charge Transfer Efficiency
CVS	Concurrent Versioning System
dB	decibel
DC	Direct Current
BDRF	Bi-directional Reflectance Distribution Function
DKFA	Discrete Kalman Filter Algorithm
DMA	Direct Memory Access

continued on next page

continued from previous page

Acronym	Description
DS	Data Set
DTR	Data Terminal Ready
EDT	Engineering Design Team, Inc.
EMI	Electro-Magnetic Interference
EPROM	Erasable Programmable Read Only Memory
ESD	Electro-Static Discharge
FFNN	Feed-Forward Neural Network
FFT	Fast Fourier Transform
FIFO	First-In First-Out
FOV	Field Of View
fps	Frames per second
FPU	Floating Point Unit
FWC	Full Well Capacity
GCC	GNU Compiler Collection
GD	Gradient Descent
GL	Graphics Language
GLIBC	GNU C Library
GLUT	OpenGL Utility Toolkit
GNU	Recursive Acronym: GNU's Not Unix
GPL	General Public Licence
GUI	Graphical Users Interface
HDL	Hardware Description Language
HTML	Hyper-Text Markup Language
in	inch
I/O	Input / Output
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IGES	Initial Graphics Exchange Specification
IP	Intellectual Property
IRLS	Iteratively Re-weighted Least-Squares
IRQ	Interrupt Request
ISO	International Organisation for Standards
JCS	Joint Coordinate System
K	Kelvin
KB	Kilobyte (1024 bytes)
KRC	KUKA Robot Controller
KRL	KUKA Robot Language
LAPACK	Linear Algebra PACKage
LGPL	Lesser General Public Licence
LMedS	Least-Median of Squares
LMS	Least Mean Square
LS	Least Square
LSB	Least Significant Byte/Bit
LUT	Lookup Table
LVDS	Low Voltage Differential Signalling

continued on next page

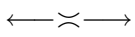
continued from previous page

Acronym	Description
MAD	Median Absolute Deviation
MB	Megabyte (1024KB = 1048576 bytes)
Mb/s	Millions of Bits per second
MB/s	Millions of Bytes per second
MC	Monte Carlo methods
MCS	Model Coordinate System
MOS	Metal-Oxide-Semiconductor
MSB	Most Significant Byte/Bit
MSE	Mean Squared Error
MTBF	Mean Time Between Failures
NAK	Handshake packet indicating a negative acknowledgement
NN	Neural Network
NURBS	Non-Uniform Rational B-Spline
OLS	Ordinary Least-Squares
OOP	Object Oriented Programming
OS	Operating System
OSS	Open Source Software
PC	Personal Computer
PCI	Peripheral Component Interconnect
PD	Public Domain
PDF	Portable Document Format
PLC	Programmable Logic Controller
PRNU	Photo Response Non Uniformity
PROM	Programmable Read Only Memory
PTP	Point-To-Point
PV	Pixel Value
QE	Quantum Efficiency
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RCS	Robot Coordinate System
RFCS	Robot Flange Coordinate System
RS232	Industry Standard Serial Transmission Protocol
RFI	Radio Frequency Interference
RMS	Root-mean-square
ROI	Region of Interest
ROM	Read Only Memory
RPM	Red Hat Package Manager
RSI	Robot Sensor Interface
RSS	residual sums of squares
RTOS	Real-Time Operating System
SCM	Software Configuration Management
SCS	Standardized Coordinate System
SDDS	Surface Defect Detection System
SDDSF	Surface Defect Detection System Framework
SIMD	Single Instruction Multiple Data

continued on next page

continued from previous page

Acronym	Description
SLE	System of Linear Equations
SNR	Signal-to-Noise Ratio
SSE	Streaming SIMD Extensions (SSE)
SSE	Sum Squared Error
SSE2	Streaming SIMD Extensions (SSE) version 2
SVD	Singular Value Decomposition
SVN	Subversion Version Control System
TCP/IP	Transmission Control Protocol/Internet Protocol
TCS	Tool Coordinate System
UP	University of Pretoria
UPS	Uninterruptable Power Supply
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WCS	World Coordinate System



APPENDIX B

List of Symbols

This chapter lists symbols found in the text and also defines the notation used [97, 58]

Mathematical Operations and Functions

$\exp$	exponential
$\log$	logarithm
var	variance operator

Scalars, Vectors and Matrices

a	Scalar
$\mathbf{a}$	Column vector
$\|\mathbf{x}\|$	Euclidean norm (length) of vector $\mathbf{x}$
$\mathbf{a}^T$	Transpose of vector $\mathbf{a}$
a_i	The ith entry of the vector $\mathbf{a}$
$\mathbf{A}$	Matrix
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^*$	Complex conjugate of matrix $\mathbf{A}$
$\mathbf{A}^H$	Hermitian (transposed and complex conjugate) of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of invertable, square matrix $\mathbf{A}$
$\mathbf{A}^+$	Moore-Penrose (pseudo) inverse of matrix $\mathbf{A}$
$\mathbf{A}^{1/2}$	The square root of a matrix (if unique), not element-wise
$A_{i,j}$	The (i, j).th entry of the matrix $\mathbf{A}$
$\det(\mathbf{A})$	Determinant of matrix $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	Trace of matrix (A)
$\|\mathbf{A}\|$	Matrix norm (subscript, if any, denotes what norm)
$\mathbf{0}$	Null matrix (zero in all entries)
$\mathbf{I}$	Identity matrix
$\mathbf{\Sigma}$	A positive definite matrix
$\mathbf{\Lambda}$	A diagonal matrix

Miscellaneous

$E[A]$	The expected value of a random variable A
$\hat{x}$	Estimate of x
$\lvert x \rvert$	Absolute value (magnitude) of x
$\underset{i}{\text{median}}$	Obtain the selection i which respresents the statistical median of the argument
$\underset{i}{\text{argmin}}$	Obtain the parameter i which results in the minimum agrument

$\longleftarrow\asymp\longrightarrow$

APPENDIX C

Camera Configuration

This appendix provides the modified camera configuration file that must be used to configure the camera before use. Failure to do so will most likely result in a system crash.

```
# camera description, for camera selection GUI and apps
# camera_class should be the manufacturer's name
camera_class:                   "Photon Focus"
camera_model:                   "pf1024x256-80CL"
camera_info:                    "1024x256 (freerun) (also works with 160)"

# actual width/height (total pixels) and depth of data from camera
# to only grab high 8-bits, set depth to 8 but leave extdepth set
# to actual depth, and adjust shift and mask accordingly
width:                          1024
height:                         256
depth:                          8
extdepth:                       8

# mode control register bits (hex):
# 0-3: on/off state of mode control lines
# 4-7: which mode control line to send expose pulse for
#      triggered exposure or pulse-width triggering.
# this directive is usually set to 00 for free-running cameras,
# or 10 for triggered or pulse-width cameras/modes
MODE_CNTL_NORM:                 00

# camera link data path register bits (hex):
# 0-3: number of bits per pixel - 1
# 4-7: number of channels - 1
CL_DATA_PATH_NORM:              17

# camera link config register bits (hex):
# 0: RGB (on for RGB color cameras only)
# 1: ignore data valid (on for most cameras though not all)
# 2: line scan
# 3: disable ROI (rarely set)
# 4: unused
# 5: data valid invert (rare)
# 6-7: undefined
CL_CFG_NORM:                    00

# region of interest start and area
# vskip/hskip is how many pixels to skip before ROI, vert and horiz
# vactv/hactv is how many pixels to DMA to memory after skip
hskip: 0
hactv: 1024
vskip: 384
vactv: 256

htaps: 2
```

APPENDIX D

Coding Standard

The definition and usage of a coding standard promotes consistent, readable and correct code. A coding standard consists of a set of rules that addresses weaknesses in the language standard and / or compiler idiosyncrasies and also defines a format or "style" used for writing code. Typical items in a coding standard could address pointer usage before initialisation, the use of recursive algorithms, dynamic memory allocation, unconditional jumps, *etc.* The coding standard should also help to improve readability.

Every person developing software has unique coding practices. Therefore, if more than one person is working on a project, a common standard should be established. Thus the coding standard is established by consensus based on the experience and knowledge of the team members. One of the reasons for developing a coding standard is to make code more readable, which will positively affect the following areas in software development:

- **Code generation:** A standard reduces the probability of coding error.
- **Code reviews:** A standard increases the efficiency of peer reviews and inspections.
- **Quality:** A standard increases the overall quality of the product.
- **Maintenance:** A standard increases the maintainability of the software product.

In this project, there was only one developer. The coding standard was therefore established based on "best practice" as found in industry and the open source community, together with knowledge of compiler operation. The aim of such a coding standard was to provide uniform source code that would ease the learning curve of any additional developers or maintainers. The items listed below have been accumulated over time, but are however not exhaustive. The source code does include other rules that were assumed subsequent to the generation of this list. It should be assumed that the top level source code directory contains the most recent version of this list.

D.1 General

General rules include:

- Every text file must end with a newline.
- Use spaces instead of tabs, except in Makefiles.
- Do not end text lines with spaces.

- Try to limit the length of source code lines to less than 80 characters.
- Try to avoid creating files with excessively long names (45 characters).
- Every program should have a "usage" message. It should be printed out if erroneous command line arguments, or a -? command line argument, are provided to the program.
- A program entry point main has the following format:

```
int main (int argc, char *argv[])
```

  `main` must also return 0 on successful termination, and non-zero otherwise.

- Avoid including the string "Error" in a source code filename. GNU Make's error messages start with "Error". So, it's much easier to search for errors if filenames don't contain "Error".
- Narrow interfaces are better than wide interfaces. If there isn't a need for an interface, leave it out. This eases maintenance, minimizes footprint, and reduces the likelihood of interference when other interfaces need to be added later.

D.2 Code Documentation

Code documentation rules include:

- Use comments and whitespace liberally. Comments should consist of complete sentences, *i.e.* start with a capital letter and end with a period.
- Code should be self documenting and the additional documentation should explain the abstract algorithm implemented, together with appropriate motivation, and not explain what the code is doing in line-by-line fashion.
- As a result of the previous statement, the primary documentation should be an API related (for header files) block and an algorithm related block (for implementation files) before each function.
- In summary, the documentation should add information that can not be deduced from the code without effort.
- Insert the standard file header into each source file.
- Be sure to follow the guidelines and restrictions for use of the documentation tools for all header files, which must follow the Doxygen[1] format requirements. The complete documentation for Doxygen is available in the Doxygen manual[2].

[1] `http://www.doxygen.org`

[2] `http://www.stack.nl/~dimitri/doxygen/download.html\#latestman`

D.3 Naming Conventions

Naming conventions include:

- Based on the naming of variables and constants, their types and usage should be obvious.
- The use of the C99 extension, which allows instantiation of variables just before their use, is recommended.
- All variables will be lowercase. Macros and enums will be upper-case.
- Be very careful with names of macros, enum values, and variables. Macros and enum values should be prefixed with `SDDS_` in order to clearly indicate that they are part of the SDDS "namespace", and to minimise clashes and aliasing with existing libraries. Similarly, functions and variables that are globally visible should be prefixed with `sdds_`.
- Do not use hard coded numbers directly in expressions. Maintainability becomes a horrendous task when the code is contaminated with these mysterious numbers. Such constants should be explicitly defined (with a suitable name) in a constant definition file or in the local header file. If the constant is only applicable to a specific routine, the constant should be defined in the routine. Any changes in the value of the constant would propagate throughout the definition scope of the software product, sparing the programmer the burden of searching the entire software product for all occurrences of the constant. A comment block describing the source of the constant value should also be added.

D.4 Pre-processor

Pre-processor rules include:

- Always follow a pre-processor `#endif` with a `/* */` C-style comment. It should correspond to the condition in the matching `#if` directive. For example,

```
#if defined(SDDS_MATRIX_DEBUG)
  #define SDDS_MATRIX_ASSERT(x)  SDDS_ASSERT(x)
#else
  #define SDDS_MATRIX_ASSERT(x)
#endif /* SDDS_MATRIX_DEBUG */
```

- Protect header files against multiple inclusion with this construct:

```
#ifndef SDDS_MATRIX_H
#define SDDS_MATRIX_H

[contents of header file]

#endif /* SDDS_MATRIX_H */
```

  Many compilers optimise the `#ifndef` construct in such a way that the files are only opened once per compilation unit. No code can appear after the final `#endif` for the optimisation to be effective and correct.

- To reduce the possibility of error when using the pre-processor, always use full parenthesising to ensure that the correct replace operation is performed. Errors caused by incorrect / missing parenthesising can result in obscure compile errors or runtime bugs. To find such a bug, the

compiler must be run in a special mode in order to examine the actual C source code being compiled (after pre-processing). For GCC, this can be done by adding the compiler option `-save-temps` to the *complete* command line used to compile the specific file. Alternatively, the pre-processor can be run separately on the file, also using the complete command line.

D.5 C Syntax and Constructs

Rules for syntax and constructs include:

- `for` loops should look like:

```
for (u_int i = 0; i < count; ++i)
  ++total;
```

  However, be careful with the contents of the loop body. It is generally recommended that the body of the loop be wrapped in braces, to avoid surprises when other code or debugging statements are added, and to maintain sanity when the body consists of a macro, such as an `SDDS_ASSERT` without a trailing semicolon:

```
for (u_int i = 0; i < count; ++i)
  {
    SDDS_ASSERT (++total < UINT_MAX;)
  }
```

  Similarly, `if` statements should have a space after the "if", and no spaces just after the opening parenthesis and just before the closing parenthesis.

- If a loop index is used after the body of the loop, it must be declared before the loop. For example,

```
size_t i = 0;
for (size_t j = 0; file_name [j] != '\0'; ++i, ++j)
  {
    if (file_name [j] == '\\' && file_name [j + 1] == '\\')
      ++j;

    file_name [i] = file_name [j];
  }

file_name [i] = '\0';
```

- Prefix operators are sometimes more efficient than postfix operators. Therefore, they are preferred over their postfix counterparts where the expression value is not used.

- Avoid unnecessary parenthesis.

- Be very careful when selecting an integer type that must be a certain size, *e.g.*,, 4 bytes. Use the `stdint.h` definitions.

- Initialisation is usually cleaner than assignment, especially in a conditional statement. Instead of code like:

```
ssize_t n_bytes;
if ((n_bytes = send (data, datalen)) == -1)
```

  rather use

```
ssize_t n_bytes = send (data, datalen);

if (n_bytes == -1)
```

But, beware if the initialisation is of a static variable. A static variable is only initialised the first time its declaration is seen. However, the use of static variables should be generally be avoided.

- It is usually clearer to write conditionals that have both branches without a negated condition. For example,

```
if (test)
  {
    /* true branch */
  }
else
  {
    /* false branch */
  }
```

is preferred over:

```
if (! test)
  {
    /* false branch */
  }
else
  {
    /* true test branch */
  }
```

- Functions should always return a status indicator type of `sdds_status_t`, if required. See `sddsstatus.h`.

D.6 I/O

Rules for I/O operations include:

- After attempting to open an existing file, always check for success. Take appropriate action if the open failed.

D.7 SDDS Library Usage Guidelines

Guidelines for SDDS library usage include:

- Be very careful with `SDDS_ASSERT`. It must only be used to check values; it may never be used to wrap a function call, or contain any other side effect. This restriction is due to the fact that the statement will disappear when `SDDS_NDEBUG` is enabled. Any side effects of the wrapped function call will then not be present, which can lead to hard to debug anomalies.
- Immediately after opening a temporary file, unlink it. For example:

```
FILE *h = fopen (filename, "w+");
unlink (filename);
```

This avoids leaving the temporary file even if the program crashes.

- Always use `$(RM)` instead of `rm` or `rm -f` in Makefiles.

D.8 Software Engineering Guidelines

General software engineering guidelines include:

- **Minimize risk:** Test all changes.
- **Revise only when necessary:** Every change has risk, hence avoid making changes unless there is a good reason for it.
- **Normalize:** Factor out commonality. For example, maintain a data value in only one place.
- **Synthesise**: Build stubs and scaffolding early to simulate the complete system. Maintain a checked-in version of the system that cleanly builds and tests at all times.

APPENDIX E

Higher-Order Models

This appendix provides the elaborated normal equations for quadratic and cubic models. The `sddspolynomial` module of the developed software framework implements these elaborations in order to provide high performance model fitting routines. See Section 7.2.1 for the background and discussion of the normal equations of a linear model.

E.1 The Quadratic Model

For the quadratic case, the model is given by

$$r(x) = a_0 + a_1 x + a_2 x^2 \tag{E.1.1}$$

and the corresponding equation for the RSS is

$$\mathrm{RSS} = \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 \right]^2 \tag{E.1.2}$$

E.1.1 Quadratic Model Normal Equations

The necessary conditions for $q = \mathrm{RSS}$ to be minimum are

$$\frac{\partial q}{\partial a_0} = 0, \qquad \frac{\partial q}{\partial a_1} = 0, \qquad \frac{\partial q}{\partial a_2} = 0 \tag{E.1.3}$$

Taking the partial derivatives of Eq. **(E.1.2)**

$$\begin{aligned} \frac{\partial q}{\partial a_0} &= \frac{\partial}{\partial a_0} \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 \right]^2 \\ &= -2 \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 \right] \end{aligned}$$

$$\begin{aligned} \frac{\partial q}{\partial a_1} &= \frac{\partial}{\partial a_1} \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 \right]^2 \\ &= -2 \sum_{i=1}^{N} x_i \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 \right] \end{aligned}$$

$$\frac{\partial q}{\partial a_2} = \frac{\partial}{\partial a_2}\sum_{i=1}^{N}\left[y_i - a_0 - a_1x_i - a_2x_i^2\right]^2$$
$$= -2\sum_{i=1}^{N} x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2\right]$$

To simplify the notation, the range on the summation will be assumed to be over i from 1 to N. Elaboration to find the normal equations then proceeds by applying the conditions in Eq. **(E.1.3)** to the partial derivatives above

$$\begin{aligned} -2\sum\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum y_i - a_0N - a_1\sum x_i - a_2\sum x_i^2 &= 0 \\ \therefore \quad a_0N + a_1\sum x_i + a_2\sum x_i^2 &= \sum y_i \end{aligned} \tag{E.1.4}$$

$$\begin{aligned} -2\sum x_i\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum x_i\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum x_iy_i - a_0\sum x_i - a_1\sum x_i^2 - a_2\sum x_i^3 &= 0 \\ \therefore \quad a_0\sum x_i + a_1\sum x_i^2 + a_2\sum x_i^3 &= \sum x_iy_i \end{aligned} \tag{E.1.5}$$

$$\begin{aligned} -2\sum x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2\right] &= 0 \\ \therefore \quad \sum x_i^2y_i - a_0\sum x_i^2 - a_1\sum x_i^3 - a_2\sum x_i^4 &= 0 \\ \therefore \quad a_0\sum x_i^2 + a_1\sum x_i^3 + a_2\sum x_i^4 &= \sum x_i^2y_i \end{aligned} \tag{E.1.6}$$

Writing the normal equations Eq. **(E.1.4)**, Eq. **(E.1.5)** and Eq. **(E.1.6)** in matrix form

$$\begin{bmatrix} N & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_iy_i \\ \sum x_i^2y_i \end{bmatrix} \tag{E.1.7}$$

To separate the summation operations and the remainder of the calculation of the parameter vector, let

$$s_0 = N, \quad s_1 = \sum x_i, \quad s_2 = \sum x_i^2, \quad s_3 = \sum x_i^3, \quad s_4 = \sum x_i^4,$$

$$t_0 = \sum y_i, \quad t_1 = \sum x_iy_i, \quad t_2 = \sum x_i^2y_i$$

which allows Eq. **(E.1.7)** to be rewritten as

$$\begin{bmatrix} s_0 & s_1 & s_2 \\ s_1 & s_2 & s_3 \\ s_2 & s_3 & s_4 \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} \qquad \Longleftrightarrow \qquad \mathbf{Sa} = \mathbf{t} \tag{E.1.8}$$

E.1.2 The Inverse of a 3×3 Matrix

To solve for $\mathbf{a}$, the inverse of a 3×3 matrix is required. For a matrix $\mathbf{P}$,

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

the inverse is given in terms of the determinant and cofactors P_{ij} as

$$\mathbf{P}^{-1} = \frac{1}{\det \mathbf{P}} \begin{bmatrix} P_{11} & P_{21} & P_{31} \\ P_{12} & P_{22} & P_{32} \\ P_{13} & P_{23} & P_{33} \end{bmatrix}$$

where

$$\begin{aligned} \det \mathbf{P} &= \begin{vmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{vmatrix} \\ &= p_{11} \begin{vmatrix} p_{22} & p_{23} \\ p_{32} & p_{33} \end{vmatrix} - p_{12} \begin{vmatrix} p_{21} & p_{23} \\ p_{31} & p_{33} \end{vmatrix} + p_{13} \begin{vmatrix} p_{21} & p_{22} \\ p_{31} & p_{32} \end{vmatrix} \\ &= p_{11}(p_{22}p_{33} - p_{23}p_{32}) - p_{12}(p_{21}p_{33} - p_{23}p_{31}) + p_{13}(p_{21}p_{32} - p_{22}p_{31}) \\ &= p_{11}p_{22}p_{33} - p_{11}p_{23}p_{32} - p_{12}p_{21}p_{33} + p_{12}p_{23}p_{31} + p_{13}p_{21}p_{32} - p_{13}p_{22}p_{31} \end{aligned}$$

and the cofactors are given by

$$\begin{aligned} P_{11} &= \begin{vmatrix} p_{22} & p_{23} \\ p_{32} & p_{33} \end{vmatrix} \\ &= p_{22}p_{33} - p_{23}p_{32} \end{aligned}$$

$$\begin{aligned} P_{12} &= - \begin{vmatrix} p_{21} & p_{23} \\ p_{31} & p_{33} \end{vmatrix} \\ &= p_{23}p_{31} - p_{21}p_{33} \end{aligned}$$

$$\begin{aligned} P_{13} &= \begin{vmatrix} p_{21} & p_{22} \\ p_{31} & p_{32} \end{vmatrix} \\ &= p_{21}p_{32} - p_{22}p_{31} \end{aligned}$$

$$\begin{aligned} P_{21} &= - \begin{vmatrix} p_{12} & p_{13} \\ p_{32} & p_{33} \end{vmatrix} \\ &= p_{13}p_{32} - p_{12}p_{33} \end{aligned}$$

$$\begin{aligned} P_{22} &= \begin{vmatrix} p_{11} & p_{13} \\ p_{31} & p_{33} \end{vmatrix} \\ &= p_{11}p_{33} - p_{13}p_{31} \end{aligned}$$

$$P_{23} = -\begin{vmatrix} p_{11} & p_{12} \\ p_{31} & p_{32} \end{vmatrix}$$
$$= p_{12}p_{31} - p_{11}p_{32}$$

$$P_{31} = \begin{vmatrix} p_{12} & p_{13} \\ p_{22} & p_{23} \end{vmatrix}$$
$$= p_{12}p_{23} - p_{13}p_{22}$$

$$P_{32} = -\begin{vmatrix} p_{11} & p_{13} \\ p_{21} & p_{23} \end{vmatrix}$$
$$= p_{13}p_{21} - p_{11}p_{23}$$

$$P_{33} = \begin{vmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{vmatrix}$$
$$= p_{11}p_{22} - p_{12}p_{21}$$

E.1.3 Quadratic Model Parameter Equations

Substituting the corresponding values from Eq. **(E.1.8)** into the equations above provides the formulation of the parameter vector $\mathbf{a}$ in terms of the data set as

$$\mathbf{a} = \frac{1}{\det \mathbf{S}} \begin{bmatrix} S_{11} & S_{21} & S_{31} \\ S_{12} & S_{22} & S_{32} \\ S_{13} & S_{23} & S_{33} \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$
$$= \frac{1}{\det \mathbf{S}} \begin{bmatrix} S_{11}t_0 + S_{21}t_1 + S_{31}t_2 \\ S_{12}t_0 + S_{22}t_1 + S_{32}t_2 \\ S_{13}t_0 + S_{23}t_1 + S_{33}t_2 \end{bmatrix}$$

where

$$\det \mathbf{S} = s_0s_2s_4 - s_0s_3s_3 - s_1s_1s_4 + s_1s_2s_3 + s_1s_2s_3 - s_2s_2s_2$$
$$= s_0s_2s_4 - s_0s_3^2 - s_1^2s_4 + 2s_1s_2s_3 - s_2^3$$

$$S_{11} = s_2s_4 - s_3s_3$$
$$= s_2s_4 - s_3^2$$

$$S_{12} = s_3s_2 - s_1s_4$$
$$= s_2s_3 - s_1s_4$$

$$S_{13} = s_1s_3 - s_2s2$$
$$= s_1s_3 - s_2^2$$

$$S_{21} = s_2s_3 - s_1s_4$$
$$= S_{12}$$

$$
\begin{aligned}
S_{22} &= s_0 s_4 - s_2 s_2 \\
&= s_0 s_4 - s_2^2
\end{aligned}
$$

$$S_{23} = s_1 s_2 - s_0 s_3$$

$$
\begin{aligned}
S_{31} &= s_1 s_3 - s_2 s_2 \\
&= s_1 s_3 - s_2^2 \\
&= S_{13}
\end{aligned}
$$

$$
\begin{aligned}
S_{32} &= s_2 s_1 - s_0 s_3 \\
&= s_1 s_2 - s_0 s_3 \\
&= S_{23}
\end{aligned}
$$

$$
\begin{aligned}
S_{33} &= s_0 s_2 - s_1 s_1 \\
&= s_0 s_2 - s_1^2
\end{aligned}
$$

The equations above are implemented in the internal `sdds_polynomial_nefit_quadatric` helper function of `sdds_polynomial_nefit`. It can be seen that as a result of the symmetry in the $\mathbf{S}$ matrix, only 6 of the 9 cofactors need to be calculated, resulting in a further saving in computation. Note that the determinant may not exist and the implementation must take appropriate action as warranted by the particular use of the function.

E.2 The Cubic Model

For the cubic case, the model is given by

$$r(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{E.2.1}$$

and the corresponding equation for the RSS is

$$\mathrm{RSS} = \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 \right]^2 \tag{E.2.2}$$

E.2.1 Cubic Model Normal Equations

The necessary conditions for $q = \mathrm{RSS}$ to be minimum are

$$\frac{\partial q}{\partial a_0} = 0, \qquad \frac{\partial q}{\partial a_1} = 0, \qquad \frac{\partial q}{\partial a_2} = 0, \qquad \frac{\partial q}{\partial a_3} = 0 \tag{E.2.3}$$

Taking the partial derivatives of Eq. **(E.2.2)**

$$
\begin{aligned}
\frac{\partial q}{\partial a_0} &= \frac{\partial}{\partial a_0} \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 \right]^2 \\
&= -2 \sum_{i=1}^{N} \left[y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 \right]
\end{aligned}
$$

$$\frac{\partial q}{\partial a_1} = \frac{\partial}{\partial a_1}\sum_{i=1}^{N}\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]^2$$
$$= -2\sum_{i=1}^{N} x_i\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]$$

$$\frac{\partial q}{\partial a_2} = \frac{\partial}{\partial a_2}\sum_{i=1}^{N}\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]^2$$
$$= -2\sum_{i=1}^{N} x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]$$

$$\frac{\partial q}{\partial a_3} = \frac{\partial}{\partial a_3}\sum_{i=1}^{N}\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]^2$$
$$= -2\sum_{i=1}^{N} x_i^3\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right]$$

To simplify the notation, the range on the summation will be assumed to be over i from 1 to N. Elaboration to find the normal equations then proceeds by applying the conditions in Eq. **(E.2.3)** to the partial derivatives above

$$-2\sum\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum y_i - a_0N - a_1\sum x_i - a_2\sum x_i^2 - a_3\sum x_i^3 = 0$$
$$\therefore \quad a_0N + a_1\sum x_i + a_2\sum x_i^2 + a_3\sum x_i^3 = \sum y_i \tag{E.2.4}$$

$$-2\sum x_i\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum x_i\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum x_iy_i - a_0\sum x_i - a_1\sum x_i^2 - a_2\sum x_i^3 - a_3\sum x_i^4 = 0$$
$$\therefore \quad a_0\sum x_i + a_1\sum x_i^2 + a_2\sum x_i^3 + a_3\sum x_i^4 = \sum x_iy_i \tag{E.2.5}$$

$$-2\sum x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum x_i^2\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] = 0$$
$$\therefore \quad \sum x_i^2y_i - a_0\sum x_i^2 - a_1\sum x_i^3 - a_2\sum x_i^4 - a_3\sum x_i^5 = 0$$
$$\therefore \quad a_0\sum x_i^2 + a_1\sum x_i^3 + a_2\sum x_i^4 + a_3\sum x_i^5 = \sum x_i^2y_i \tag{E.2.6}$$

$$
\begin{aligned}
-2\sum x_i^3\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] &= 0 \\
\therefore \quad \sum x_i^3\left[y_i - a_0 - a_1x_i - a_2x_i^2 - a_3x_i^3\right] &= 0 \\
\therefore \quad \sum x_i^3y_i - a_0\sum x_i^3 - a_1\sum x_i^4 - a_2\sum x_i^5 - a_3\sum x_i^6 &= 0 \\
\therefore \quad a_0\sum x_i^3 + a_1\sum x_i^4 + a_2\sum x_i^5 + a_3\sum x_i^6 &= \sum x_i^3y_i \qquad \text{(E.2.7)}
\end{aligned}
$$

Writing the normal equations Eq. **(E.2.4)**, Eq. **(E.2.5)**, Eq. **(E.2.6)** and Eq. **(E.2.6)** in matrix form

$$
\begin{bmatrix} N & \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \sum x_i^5 \\ \sum x_i^3 & \sum x_i^4 & \sum x_i^5 & \sum x_i^6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_iy_i \\ \sum x_i^2y_i \\ \sum x_i^3y_i \end{bmatrix} \qquad \text{(E.2.8)}
$$

To separate the summation operations and the remainder of the calculation of the parameter vector, let

$$
s_0 = N, \quad s_1 = \sum x_i, \quad s_2 = \sum x_i^2, \quad s_3 = \sum x_i^3, \quad s_4 = \sum x_i^4, \quad s_5 = \sum x_i^5,
$$

$$
s_6 = \sum x_i^6, \quad t_0 = \sum y_i, \quad t_1 = \sum x_iy_i, \quad t_2 = \sum x_i^2y_i, \quad t_3 = \sum x_i^3y_i
$$

which allows Eq. **(E.2.8)** to be rewritten as

$$
\begin{bmatrix} s_0 & s_1 & s_2 & s_3 \\ s_1 & s_2 & s_3 & s_4 \\ s_2 & s_3 & s_4 & s_5 \\ s_3 & s_4 & s_5 & s_6 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{bmatrix} \qquad \Longleftrightarrow \qquad \mathbf{Sa} = \mathbf{t} \qquad \text{(E.2.9)}
$$

E.2.2 The Inverse of a 4×4 Matrix

To solve for $\mathbf{a}$, the inverse of a 4×4 matrix is required. For a matrix $\mathbf{P}$,

$$
\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}
$$

the inverse is given in terms of the determinant and cofactors P_{ij} as

$$
\mathbf{P}^{-1} = \frac{1}{\det \mathbf{P}} \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \\ P_{14} & P_{24} & P_{34} & P_{44} \end{bmatrix}
$$

where

$$
\det \mathbf{P} = \begin{vmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{vmatrix}
$$

$$
\begin{aligned}
&= p_{11}\begin{vmatrix} p_{22} & p_{23} & p_{24} \\ p_{32} & p_{33} & p_{34} \\ p_{42} & p_{43} & p_{44} \end{vmatrix} - p_{12}\begin{vmatrix} p_{21} & p_{23} & p_{24} \\ p_{31} & p_{33} & p_{34} \\ p_{41} & p_{43} & p_{44} \end{vmatrix} + \\
&\quad p_{13}\begin{vmatrix} p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \\ p_{41} & p_{42} & p_{44} \end{vmatrix} - p_{14}\begin{vmatrix} p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \end{vmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= p_{11}\left\{ p_{22}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{23}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} \right\} - \\
&\quad p_{12}\left\{ p_{21}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{23}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} \right\} + \\
&\quad p_{13}\left\{ p_{21}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} - p_{22}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \right\} - \\
&\quad p_{14}\left\{ p_{21}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} - p_{22}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} + p_{23}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \right\}
\end{aligned}
$$

$$
\begin{aligned}
&= p_{11}\left\{ p_{22}(p_{33}p_{44} - p_{34}p_{43}) - p_{23}(p_{32}p_{44} - p_{34}p_{42}) + p_{24}(p_{32}p_{43} - p_{33}p_{42}) \right\} - \\
&\quad p_{12}\left\{ p_{21}(p_{33}p_{44} - p_{34}p_{43}) - p_{23}(p_{31}p_{44} - p_{34}p_{41}) + p_{24}(p_{31}p_{43} - p_{33}p_{41}) \right\} + \\
&\quad p_{13}\left\{ p_{21}(p_{32}p_{44} - p_{34}p_{42}) - p_{22}(p_{31}p_{44} - p_{34}p_{41}) + p_{24}(p_{31}p_{42} - p_{32}p_{41}) \right\} - \\
&\quad p_{14}\left\{ p_{21}(p_{32}p_{43} - p_{33}p_{42}) - p_{22}(p_{31}p_{43} - p_{33}p_{41}) + p_{23}(p_{31}p_{42} - p_{32}p_{41}) \right\}
\end{aligned}
$$

$$
\begin{aligned}
&= p_{11}\left\{ p_{22}p_{33}p_{44} - p_{22}p_{34}p_{43} - p_{23}p_{32}p_{44} + p_{23}p_{34}p_{42} + p_{24}p_{32}p_{43} - p_{24}p_{33}p_{42} \right\} - \\
&\quad p_{12}\left\{ p_{21}p_{33}p_{44} - p_{21}p_{34}p_{43} - p_{23}p_{31}p_{44} + p_{23}p_{34}p_{41} + p_{24}p_{31}p_{43} - p_{24}p_{33}p_{41} \right\} + \\
&\quad p_{13}\left\{ p_{21}p_{32}p_{44} - p_{21}p_{34}p_{42} - p_{22}p_{31}p_{44} + p_{22}p_{34}p_{41} + p_{24}p_{31}p_{42} - p_{24}p_{32}p_{41} \right\} - \\
&\quad p_{14}\left\{ p_{21}p_{32}p_{43} - p_{21}p_{33}p_{42} - p_{22}p_{31}p_{43} + p_{22}p_{33}p_{41} + p_{23}p_{31}p_{42} - p_{23}p_{32}p_{41} \right\}
\end{aligned}
$$

$$
\begin{aligned}
&= p_{11}p_{22}p_{33}p_{44} - p_{11}p_{22}p_{34}p_{43} - p_{11}p_{23}p_{32}p_{44} + p_{11}p_{23}p_{34}p_{42} + p_{11}p_{24}p_{32}p_{43} - \\
&\quad p_{11}p_{24}p_{33}p_{42} - p_{12}p_{21}p_{33}p_{44} + p_{12}p_{21}p_{34}p_{43} + p_{12}p_{23}p_{31}p_{44} - p_{12}p_{23}p_{34}p_{41} - \\
&\quad p_{12}p_{24}p_{31}p_{43} + p_{12}p_{24}p_{33}p_{41} + p_{13}p_{21}p_{32}p_{44} - p_{13}p_{21}p_{34}p_{42} - p_{13}p_{22}p_{31}p_{44} + \\
&\quad p_{13}p_{22}p_{34}p_{41} + p_{13}p_{24}p_{31}p_{42} - p_{13}p_{24}p_{32}p_{41} - p_{14}p_{21}p_{32}p_{43} + p_{14}p_{21}p_{33}p_{42} + \\
&\quad p_{14}p_{22}p_{31}p_{43} - p_{14}p_{22}p_{33}p_{41} - p_{14}p_{23}p_{31}p_{42} + p_{14}p_{23}p_{32}p_{41}
\end{aligned}
$$

and the cofactors are given by

$$
\begin{aligned}
P_{11} &= \begin{vmatrix} p_{22} & p_{23} & p_{24} \\ p_{32} & p_{33} & p_{34} \\ p_{42} & p_{43} & p_{44} \end{vmatrix} \\
&= p_{22}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{23}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} \\
&= p_{22}(p_{33}p_{44} - p_{34}p_{43}) - p_{23}(p_{32}p_{44} - p_{34}p_{42}) + p_{24}(p_{32}p_{43} - p_{33}p_{42}) \\
&= p_{22}p_{33}p_{44} - p_{22}p_{34}p_{43} - p_{23}p_{32}p_{44} + p_{23}p_{34}p_{42} + p_{24}p_{32}p_{43} - p_{24}p_{33}p_{42}
\end{aligned}
$$

$$
\begin{aligned}
P_{12} &= -\begin{vmatrix} p_{21} & p_{23} & p_{24} \\ p_{31} & p_{33} & p_{34} \\ p_{41} & p_{43} & p_{44} \end{vmatrix} \\
&= -\left\{ p_{21}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{23}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} \right\} \\
&= -\Big\{ p_{21}(p_{33}p_{44} - p_{34}p_{43}) - p_{23}(p_{31}p_{44} - p_{34}p_{41}) + p_{24}(p_{31}p_{43} - p_{33}p_{41}) \Big\} \\
&= -\Big\{ p_{21}p_{33}p_{44} - p_{21}p_{34}p_{43} - p_{23}p_{31}p_{44} + p_{23}p_{34}p_{41} + p_{24}p_{31}p_{43} - p_{24}p_{33}p_{41} \Big\} \\
&= -p_{21}p_{33}p_{44} + p_{21}p_{34}p_{43} + p_{23}p_{31}p_{44} - p_{23}p_{34}p_{41} - p_{24}p_{31}p_{43} + p_{24}p_{33}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{13} &= \begin{vmatrix} p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \\ p_{41} & p_{42} & p_{44} \end{vmatrix} \\
&= p_{21}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} - p_{22}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{24}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \\
&= p_{21}(p_{32}p_{44} - p_{34}p_{42}) - p_{22}(p_{31}p_{44} - p_{34}p_{41}) + p_{24}(p_{31}p_{42} - p_{32}p_{41}) \\
&= p_{21}p_{32}p_{44} - p_{21}p_{34}p_{42} - p_{22}p_{31}p_{44} + p_{22}p_{34}p_{41} + p_{24}p_{31}p_{42} - p_{24}p_{32}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{14} &= -\begin{vmatrix} p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \end{vmatrix} \\
&= -\left\{ p_{21}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} - p_{22}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} + p_{23}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \right\} \\
&= -\Big\{ p_{21}(p_{32}p_{43} - p_{33}p_{42}) - p_{22}(p_{31}p_{43} - p_{33}p_{41}) + p_{23}(p_{31}p_{42} - p_{32}p_{41}) \Big\} \\
&= -\Big\{ p_{21}p_{32}p_{43} - p_{21}p_{33}p_{42} - p_{22}p_{31}p_{43} + p_{22}p_{33}p_{41} + p_{23}p_{31}p_{42} - p_{23}p_{32}p_{41} \Big\} \\
&= -p_{21}p_{32}p_{43} + p_{21}p_{33}p_{42} + p_{22}p_{31}p_{43} - p_{22}p_{33}p_{41} - p_{23}p_{31}p_{42} + p_{23}p_{32}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{21} &= -\begin{vmatrix} p_{12} & p_{13} & p_{14} \\ p_{32} & p_{33} & p_{34} \\ p_{42} & p_{43} & p_{44} \end{vmatrix} \\
&= -\left\{ p_{12}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{13}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} \right\} \\
&= -\Big\{ p_{12}(p_{33}p_{44} - p_{34}p_{43}) - p_{13}(p_{32}p_{44} - p_{34}p_{42}) + p_{14}(p_{32}p_{43} - p_{33}p_{42}) \Big\} \\
&= -\Big\{ p_{12}p_{33}p_{44} - p_{12}p_{34}p_{43} - p_{13}p_{32}p_{44} + p_{13}p_{34}p_{42} + p_{14}p_{32}p_{43} - p_{14}p_{33}p_{42} \Big\} \\
&= -p_{12}p_{33}p_{44} + p_{12}p_{34}p_{43} + p_{13}p_{32}p_{44} - p_{13}p_{34}p_{42} - p_{14}p_{32}p_{43} + p_{14}p_{33}p_{42}
\end{aligned}
$$

$$
\begin{aligned}
P_{22} &= \begin{vmatrix} p_{11} & p_{13} & p_{14} \\ p_{31} & p_{33} & p_{34} \\ p_{41} & p_{43} & p_{44} \end{vmatrix} \\
&= p_{11}\begin{vmatrix} p_{33} & p_{34} \\ p_{43} & p_{44} \end{vmatrix} - p_{13}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} \\
&= p_{11}(p_{33}p_{44} - p_{34}p_{43}) - p_{13}(p_{31}p_{44} - p_{34}p_{41}) + p_{14}(p_{31}p_{43} - p_{33}p_{41}) \\
&= p_{11}p_{33}p_{44} - p_{11}p_{34}p_{43} - p_{13}p_{31}p_{44} + p_{13}p_{34}p_{41} + p_{14}p_{31}p_{43} - p_{14}p_{33}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{23} &= -\begin{vmatrix} p_{11} & p_{12} & p_{14} \\ p_{31} & p_{32} & p_{34} \\ p_{41} & p_{42} & p_{44} \end{vmatrix} \\
&= -\left\{ p_{11}\begin{vmatrix} p_{32} & p_{34} \\ p_{42} & p_{44} \end{vmatrix} - p_{12}\begin{vmatrix} p_{31} & p_{34} \\ p_{41} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \right\} \\
&= -\Big\{ p_{11}(p_{32}p_{44} - p_{34}p_{42}) - p_{12}(p_{31}p_{44} - p_{34}p_{41}) + p_{14}(p_{31}p_{42} - p_{32}p_{41}) \Big\} \\
&= -\Big\{ p_{11}p_{32}p_{44} - p_{11}p_{34}p_{42} - p_{12}p_{31}p_{44} + p_{12}p_{34}p_{41} + p_{14}p_{31}p_{42} - p_{14}p_{32}p_{41} \Big\} \\
&= -p_{11}p_{32}p_{44} + p_{11}p_{34}p_{42} + p_{12}p_{31}p_{44} - p_{12}p_{34}p_{41} - p_{14}p_{31}p_{42} + p_{14}p_{32}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{24} &= \begin{vmatrix} p_{11} & p_{12} & p_{13} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \end{vmatrix} \\
&= p_{11}\begin{vmatrix} p_{32} & p_{33} \\ p_{42} & p_{43} \end{vmatrix} - p_{12}\begin{vmatrix} p_{31} & p_{33} \\ p_{41} & p_{43} \end{vmatrix} + p_{13}\begin{vmatrix} p_{31} & p_{32} \\ p_{41} & p_{42} \end{vmatrix} \\
&= p_{11}(p_{32}p_{43} - p_{33}p_{42}) - p_{12}(p_{31}p_{43} - p_{33}p_{41}) + p_{13}(p_{31}p_{42} - p_{32}p_{41}) \\
&= p_{11}p_{32}p_{43} - p_{11}p_{33}p_{42} - p_{12}p_{31}p_{43} + p_{12}p_{33}p_{41} + p_{13}p_{31}p_{42} - p_{13}p_{32}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{31} &= \begin{vmatrix} p_{12} & p_{13} & p_{14} \\ p_{22} & p_{23} & p_{24} \\ p_{42} & p_{43} & p_{44} \end{vmatrix} \\
&= p_{12}\begin{vmatrix} p_{23} & p_{24} \\ p_{43} & p_{44} \end{vmatrix} - p_{13}\begin{vmatrix} p_{22} & p_{24} \\ p_{42} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{22} & p_{23} \\ p_{42} & p_{43} \end{vmatrix} \\
&= p_{12}(p_{23}p_{44} - p_{24}p_{43}) - p_{13}(p_{22}p_{44} - p_{24}p_{42}) + p_{14}(p_{22}p_{43} - p_{23}p_{42}) \\
&= p_{12}p_{23}p_{44} - p_{12}p_{24}p_{43} - p_{13}p_{22}p_{44} + p_{13}p_{24}p_{42} + p_{14}p_{22}p_{43} - p_{14}p_{23}p_{42}
\end{aligned}
$$

$$
\begin{aligned}
P_{32} &= -\begin{vmatrix} p_{11} & p_{13} & p_{14} \\ p_{21} & p_{23} & p_{24} \\ p_{41} & p_{43} & p_{44} \end{vmatrix} \\
&= -\left\{ p_{11}\begin{vmatrix} p_{23} & p_{24} \\ p_{43} & p_{44} \end{vmatrix} - p_{13}\begin{vmatrix} p_{21} & p_{24} \\ p_{41} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{21} & p_{23} \\ p_{41} & p_{43} \end{vmatrix} \right\} \\
&= -\Big\{ p_{11}(p_{23}p_{44} - p_{24}p_{43}) - p_{13}(p_{21}p_{44} - p_{24}p_{41}) + p_{14}(p_{21}p_{43} - p_{23}p_{41}) \Big\} \\
&= -\Big\{ p_{11}p_{23}p_{44} - p_{11}p_{24}p_{43} - p_{13}p_{21}p_{44} + p_{13}p_{24}p_{41} + p_{14}p_{21}p_{43} - p_{14}p_{23}p_{41} \Big\} \\
&= -p_{11}p_{23}p_{44} + p_{11}p_{24}p_{43} + p_{13}p_{21}p_{44} - p_{13}p_{24}p_{41} - p_{14}p_{21}p_{43} + p_{14}p_{23}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{33} &= \begin{vmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{41} & p_{42} & p_{44} \end{vmatrix} \\
&= p_{11}\begin{vmatrix} p_{22} & p_{24} \\ p_{42} & p_{44} \end{vmatrix} - p_{12}\begin{vmatrix} p_{21} & p_{24} \\ p_{41} & p_{44} \end{vmatrix} + p_{14}\begin{vmatrix} p_{21} & p_{22} \\ p_{41} & p_{42} \end{vmatrix} \\
&= p_{11}(p_{22}p_{44} - p_{24}p_{42}) - p_{12}(p_{21}p_{44} - p_{24}p_{41}) + p_{14}(p_{21}p_{42} - p_{22}p_{41}) \\
&= p_{11}p_{22}p_{44} - p_{11}p_{24}p_{42} - p_{12}p_{21}p_{44} + p_{12}p_{24}p_{41} + p_{14}p_{21}p_{42} - p_{14}p_{22}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{34} &= -\begin{vmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{41} & p_{42} & p_{43} \end{vmatrix} \\
&= -\left\{ p_{11}\begin{vmatrix} p_{22} & p_{23} \\ p_{42} & p_{43} \end{vmatrix} - p_{12}\begin{vmatrix} p_{21} & p_{23} \\ p_{41} & p_{43} \end{vmatrix} + p_{13}\begin{vmatrix} p_{21} & p_{22} \\ p_{41} & p_{42} \end{vmatrix} \right\} \\
&= -\Big\{ p_{11}(p_{22}p_{43} - p_{23}p_{42}) - p_{12}(p_{21}p_{43} - p_{23}p_{41}) + p_{13}(p_{21}p_{42} - p_{22}p_{41}) \Big\} \\
&= -\Big\{ p_{11}p_{22}p_{43} - p_{11}p_{23}p_{42} - p_{12}p_{21}p_{43} + p_{12}p_{23}p_{41} + p_{13}p_{21}p_{42} - p_{13}p_{22}p_{41} \Big\} \\
&= -p_{11}p_{22}p_{43} + p_{11}p_{23}p_{42} + p_{12}p_{21}p_{43} - p_{12}p_{23}p_{41} - p_{13}p_{21}p_{42} + p_{13}p_{22}p_{41}
\end{aligned}
$$

$$
\begin{aligned}
P_{41} &= -\begin{vmatrix} p_{12} & p_{13} & p_{14} \\ p_{22} & p_{23} & p_{24} \\ p_{32} & p_{33} & p_{34} \end{vmatrix} \\
&= -\left\{ p_{12}\begin{vmatrix} p_{23} & p_{24} \\ p_{33} & p_{34} \end{vmatrix} - p_{13}\begin{vmatrix} p_{22} & p_{24} \\ p_{32} & p_{34} \end{vmatrix} + p_{14}\begin{vmatrix} p_{22} & p_{23} \\ p_{32} & p_{33} \end{vmatrix} \right\} \\
&= -\Big\{ p_{12}(p_{23}p_{34} - p_{24}p_{33}) - p_{13}(p_{22}p_{34} - p_{24}p_{32}) + p_{14}(p_{22}p_{33} - p_{23}p_{32}) \Big\} \\
&= -\Big\{ p_{12}p_{23}p_{34} - p_{12}p_{24}p_{33} - p_{13}p_{22}p_{34} + p_{13}p_{24}p_{32} + p_{14}p_{22}p_{33} - p_{14}p_{23}p_{32} \Big\} \\
&= -p_{12}p_{23}p_{34} + p_{12}p_{24}p_{33} + p_{13}p_{22}p_{34} - p_{13}p_{24}p_{32} - p_{14}p_{22}p_{33} + p_{14}p_{23}p_{32}
\end{aligned}
$$

$$
\begin{aligned}
P_{42} &= \begin{vmatrix} p_{11} & p_{13} & p_{14} \\ p_{21} & p_{23} & p_{24} \\ p_{31} & p_{33} & p_{34} \end{vmatrix} \\
&= p_{11} \begin{vmatrix} p_{23} & p_{24} \\ p_{33} & p_{34} \end{vmatrix} - p_{13} \begin{vmatrix} p_{21} & p_{24} \\ p_{31} & p_{34} \end{vmatrix} + p_{14} \begin{vmatrix} p_{21} & p_{23} \\ p_{31} & p_{33} \end{vmatrix} \\
&= p_{11}(p_{23}p_{34} - p_{24}p_{33}) - p_{13}(p_{21}p_{34} - p_{24}p_{31}) + p_{14}(p_{21}p_{33} - p_{23}p_{31}) \\
&= p_{11}p_{23}p_{34} - p_{11}p_{24}p_{33} - p_{13}p_{21}p_{34} + p_{13}p_{24}p_{31} + p_{14}p_{21}p_{33} - p_{14}p_{23}p_{31}
\end{aligned}
$$

$$
\begin{aligned}
P_{43} &= -\begin{vmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{vmatrix} \\
&= -\left\{ p_{11} \begin{vmatrix} p_{22} & p_{24} \\ p_{32} & p_{34} \end{vmatrix} - p_{12} \begin{vmatrix} p_{21} & p_{24} \\ p_{31} & p_{34} \end{vmatrix} + p_{14} \begin{vmatrix} p_{21} & p_{22} \\ p_{31} & p_{32} \end{vmatrix} \right\} \\
&= -\Big\{ p_{11}(p_{22}p_{34} - p_{24}p_{32}) - p_{12}(p_{21}p_{34} - p_{24}p_{31}) + p_{14}(p_{21}p_{32} - p_{22}p_{31}) \Big\} \\
&= -\Big\{ p_{11}p_{22}p_{34} - p_{11}p_{24}p_{32} - p_{12}p_{21}p_{34} + p_{12}p_{24}p_{31} + p_{14}p_{21}p_{32} - p_{14}p_{22}p_{31} \Big\} \\
&= -p_{11}p_{22}p_{34} + p_{11}p_{24}p_{32} + p_{12}p_{21}p_{34} - p_{12}p_{24}p_{31} - p_{14}p_{21}p_{32} + p_{14}p_{22}p_{31}
\end{aligned}
$$

$$
\begin{aligned}
P_{44} &= \begin{vmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{vmatrix} \\
&= p_{11} \begin{vmatrix} p_{22} & p_{23} \\ p_{32} & p_{33} \end{vmatrix} - p_{12} \begin{vmatrix} p_{21} & p_{23} \\ p_{31} & p_{33} \end{vmatrix} + p_{13} \begin{vmatrix} p_{21} & p_{22} \\ p_{31} & p_{32} \end{vmatrix} \\
&= p_{11}(p_{22}p_{33} - p_{23}p_{32}) - p_{12}(p_{21}p_{33} - p_{23}p_{31}) + p_{13}(p_{21}p_{32} - p_{22}p_{31}) \\
&= p_{11}p_{22}p_{33} - p_{11}p_{23}p_{32} - p_{12}p_{21}p_{33} + p_{12}p_{23}p_{31} + p_{13}p_{21}p_{32} - p_{13}p_{22}p_{31}
\end{aligned}
$$

E.2.3 Cubic Model Parameter Equations

Substituting the corresponding values from Eq. **(E.2.9)** into the equations above provides the formulation of the parameter vector $\mathbf{a}$ in terms of the data set as

$$
\begin{aligned}
\mathbf{a} &= \frac{1}{\det \mathbf{S}} \begin{bmatrix} S_{11} & S_{21} & S_{31} & S_{41} \\ S_{12} & S_{22} & S_{32} & S_{42} \\ S_{13} & S_{23} & S_{33} & S_{43} \\ S_{14} & S_{24} & S_{34} & S_{44} \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{bmatrix} \\
&= \frac{1}{\det \mathbf{S}} \begin{bmatrix} S_{11}t_0 + S_{21}t_1 + S_{31}t_2 + S_{41}t_3 \\ S_{12}t_0 + S_{22}t_1 + S_{32}t_2 + S_{42}t_3 \\ S_{13}t_0 + S_{23}t_1 + S_{33}t_2 + S_{43}t_3 \\ S_{14}t_0 + S_{24}t_1 + S_{34}t_2 + S_{44}t_3 \end{bmatrix}
\end{aligned}
$$

where

$$\begin{aligned}\det \mathbf{S} &= s_0s_2s_4s_6 - s_0s_2s_5s_5 - s_0s_3s_3s_6 + s_0s_3s_5s_4 + s_0s_4s_3s_5 - \\ &\quad s_0s_4s_4s_4 - s_1s_1s_4s_6 + s_1s_1s_5s_5 + s_1s_3s_2s_6 - s_1s_3s_5s_3 - \\ &\quad s_1s_4s_2s_5 + s_1s_4s_4s_3 + s_2s_1s_3s_6 - s_2s_1s_5s_4 - s_2s_2s_2s_6 + \\ &\quad s_2s_2s_5s_3 + s_2s_4s_2s_4 - s_2s_4s_3s_3 - s_3s_1s_3s_5 + s_3s_1s_4s_4 + \\ &\quad s_3s_2s_2s_5 - s_3s_2s_4s_3 - s_3s_3s_2s_4 + s_3s_3s_3s_3 \\ &= s_0s_2s_4s_6 - s_0s_2s_5^2 - s_0s_3^2s_6 + 2s_0s_3s_4s_5 - s_0s_4^3 - s_1^2s_4s_6 + \\ &\quad s_1^2s_5^2 + 2s_1s_2s_3s_6 - 2s_1s_3^2s_5 - 2s_1s_2s_4s_5 + 2s_1s_3s_4^2 - s_2^3s_6 + \\ &\quad 2s_2^2s_3s_5 + s_2^2s_4^2 - 3s_2s_3^2s_4 + s_3^4\end{aligned}$$

$$\begin{aligned}S_{11} &= s_2s_4s_6 - s_2s_5s_5 - s_3s_3s_6 + s_3s_5s_4 + s_4s_3s_5 - s_4s_4s_4 \\ &= s_2s_4s_6 - s_2s_5^2 - s_3^2s_6 + 2s_3s_4s_5 - s_4^3\end{aligned}$$

$$\begin{aligned}S_{12} &= -s_1s_4s_6 + s_1s_5s_5 + s_3s_2s_6 - s_3s_5s_3 - s_4s_2s_5 + s_4s_4s_3 \\ &= -s_1s_4s_6 + s_1s_5^2 + s_2s_3s_6 - s_3^2s_5 - s_2s_4s_5 + s_3s_4^2\end{aligned}$$

$$\begin{aligned}S_{13} &= s_1s_3s_6 - s_1s_5s_4 - s_2s_2s_6 + s_2s_5s_3 + s_4s_2s_4 - s_4s_3s_3 \\ &= s_1s_3s_6 - s_1s_4s_5 - s_2^2s_6 + s_2s_3s_5 + s_2s_4^2 - s_3^2s_4\end{aligned}$$

$$\begin{aligned}S_{14} &= -s_1s_3s_5 + s_1s_4s_4 + s_2s_2s_5 - s_2s_4s_3 - s_3s_2s_4 + s_3s_3s_3 \\ &= -s_1s_3s_5 + s_1s_4^2 + s_2^2s_5 - 2s_2s_3s_4 + s_3^3\end{aligned}$$

$$\begin{aligned}S_{21} &= -s_1s_4s_6 + s_1s_5s_5 + s_2s_3s_6 - s_2s_5s_4 - s_3s_3s_5 + s_3s_4s_4 \\ &= -s_1s_4s_6 + s_1s_5^2 + s_2s_3s_6 - s_2s_4s_5 - s_3^2s_5 + s_3s_4^2 \\ &= S_{12}\end{aligned}$$

$$\begin{aligned}S_{22} &= s_0s_4s_6 - s_0s_5s_5 - s_2s_2s_6 + s_2s_5s_3 + s_3s_2s_5 - s_3s_4s_3 \\ &= s_0s_4s_6 - s_0s_5^2 - s_2^2s_6 + 2s_2s_3s_5 - s_3^2s_4\end{aligned}$$

$$\begin{aligned}S_{23} &= -s_0s_3s_6 + s_0s_5s_4 + s_1s_2s_6 - s_1s_5s_3 - s_3s_2s_4 + s_3s_3s_3 \\ &= -s_0s_3s_6 + s_0s_4s_5 + s_1s_2s_6 - s_1s_3s_5 - s_2s_3s_4 + s_3^3\end{aligned}$$

$$\begin{aligned}S_{24} &= s_0s_3s_5 - s_0s_4s_4 - s_1s_2s_5 + s_1s_4s_3 + s_2s_2s_4 - s_2s_3s_3 \\ &= s_0s_3s_5 - s_0s_4^2 - s_1s_2s_5 + s_1s_3s_4 + s_2^2s_4 - s_2s_3^2\end{aligned}$$

$$\begin{aligned}S_{31} &= s_1s_3s_6 - s_1s_4s_5 - s_2s_2s_6 + s_2s_4s_4 + s_3s_2s_5 - s_3s_3s_4 \\ &= s_1s_3s_6 - s_1s_4s_5 - s_2^2s_6 + s_2s_4^2 + s_2s_3s_5 - s_3^2s_4 \\ &= S_{13}\end{aligned}$$

$$\begin{aligned} S_{32} &= -s_0 s_3 s_6 + s_0 s_4 s_5 + s_2 s_1 s_6 - s_2 s_4 s_3 - s_3 s_1 s_5 + s_3 s_3 s_3 \\ &= -s_0 s_3 s_6 + s_0 s_4 s_5 + s_1 s_2 s_6 - s_2 s_3 s_4 - s_1 s_3 s_5 + s_3^3 \\ &= S_{23} \end{aligned}$$

$$\begin{aligned} S_{33} &= s_0 s_2 s_6 - s_0 s_4 s_4 - s_1 s_1 s_6 + s_1 s_4 s_3 + s_3 s_1 s_4 - s_3 s_2 s_3 \\ &= s_0 s_2 s_6 - s_0 s_4^2 - s_1^2 s_6 + 2 s_1 s_3 s_4 - s_2 s_3^2 \end{aligned}$$

$$\begin{aligned} S_{34} &= -s_0 s_2 s_5 + s_0 s_3 s_4 + s_1 s_1 s_5 - s_1 s_3 s_3 - s_2 s_1 s_4 + s_2 s_2 s_3 \\ &= -s_0 s_2 s_5 + s_0 s_3 s_4 + s_1^2 s_5 - s_1 s_3^2 - s_1 s_2 s_4 + s_2^2 s_3 \end{aligned}$$

$$\begin{aligned} S_{41} &= -s_1 s_3 s_5 + s_1 s_4 s_4 + s_2 s_2 s_5 - s_2 s_4 s_3 - s_3 s_2 s_4 + s_3 s_3 s_3 \\ &= -s_1 s_3 s_5 + s_1 s_4^2 + s_2^2 s_5 - 2 s_2 s_3 s_4 + s_3^3 \\ &= S_{14} \end{aligned}$$

$$\begin{aligned} S_{42} &= s_0 s_3 s_5 - s_0 s_4 s_4 - s_2 s_1 s_5 + s_2 s_4 s_2 + s_3 s_1 s_4 - s_3 s_3 s_2 \\ &= s_0 s_3 s_5 - s_0 s_4^2 - s_1 s_2 s_5 + s_2^2 s_4 + s_1 s_3 s_4 - s_2 s_3^2 \\ &= S_{24} \end{aligned}$$

$$\begin{aligned} S_{43} &= -s_0 s_2 s_5 + s_0 s_4 s_3 + s_1 s_1 s_5 - s_1 s_4 s_2 - s_3 s_1 s_3 + s_3 s_2 s_2 \\ &= -s_0 s_2 s_5 + s_0 s_3 s_4 + s_1^2 s_5 - s_1 s_2 s_4 - s_1 s_3^2 + s_2^2 s_3 \\ &= S_{34} \end{aligned}$$

$$\begin{aligned} S_{44} &= s_0 s_2 s_4 - s_0 s_3 s_3 - s_1 s_1 s_4 + s_1 s_3 s_2 + s_2 s_1 s_3 - s_2 s_2 s_2 \\ &= s_0 s_2 s_4 - s_0 s_3^2 - s_1^2 s_4 + 2 s_1 s_2 s_3 - s_2^3 \end{aligned}$$

The equations above are implemented in the internal `sdds_polynomial_nefit_cubic` helper function of `sdds_polynomial_nefit`. It can be seen that as a result of the symmetry in the **S** matrix, only 10 of the 16 cofactors need to be calculated, resulting in a further saving in computation.

APPENDIX F

Hu's Moment Invariants

This section summarises the theory of moments as applied to images [46, 133, 73]. The theory of moments has its primary practical application in the estimation of the center of gravity (or center of mass) of physical objects. Essentially, moments give an indication of the distribution of the "mass" of the object. The definition of moments can however be extended to various other fields, in particular digital images. The moments of an image give a compact indication of the global structural properties of an image.

F.1 Continuous Two-Dimensional Cartesian Moment

The fundamental definition of two dimensional moments is effectively the surface integral over an area. The continuous two-dimensional $(p+q)^{th}$ order Cartesian moment is defined in terms of Riemann integrals as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q \, f(x,y) \, dx \, dy \tag{F.1.1}$$

It is assumed that $f(x,y)$ is a piecewise continuous, bounded function and that it can have non-zero values only in the finite region of the $x-y$ plane. Under these conditions, an uniqueness theorem [94] states that the moment sequence (m_{pq}) is uniquely defined by $f(x,y)$. Conversely, (m_{pq}) uniquely defines $f(x,y)$.

F.2 Discrete Cartesian Moment

The discrete version of the Cartesian moment for an image consisting of pixels with pixel value, P_{xy}, replacing the integrals with summations, is:

$$m_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} x^p y^q P_{xy} \tag{F.2.1}$$

where m_{pq} is the two dimensional moment and M and N are the image dimensions. For notational convenience, the given summation range will be assumed during the subsequent discussion.

The zero order moment m_{00} is defined as the *area*, *total mass* or *power* of the image. If the image is binary $M \times N$ image, m_{00} reduces to the active pixel count, as in:

$$m_{00} = \sum_x \sum_y P_{xy} \tag{F.2.2}$$

The first two moments are also special in that they can be used to find the center of mass $(\overline{x}, \overline{y})$, also called *centroids*, of an image by:

$$\overline{x} = \frac{m_{10}}{m_{00}} \qquad \overline{y} = \frac{m_{01}}{m_{00}} \tag{F.2.3}$$

The second-order moment m_{11} is defined as the *correlation* of x and y.

F.3 Discrete Centralized Moment

The discrete Cartesian moments are however not desirable when an object needs to be characterized. Specifically, the Cartesian moments depend of the effective center of mass. This implies that they are affected by changes in the relative position of an object in the image. Therefore Hu [64] defined the *discrete centralized moment* μ_{pq} to be:

$$\mu_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q P_{xy} \tag{F.3.1}$$

This is essentially a translated discrete Cartesian moment, hence the discrete centralized moments are invariant under translation. The second-order central moments are the *variances* of x and y:

$$\mu_{20} = \sigma_x^2 \tag{F.3.2}$$
$$\mu_{02} = \sigma_y^2 \tag{F.3.3}$$

These moments given an indication of the relative size of the object with regards to the x and y dimensions. The second-order joint moment μ_{11} is the *covariance* of x and y. Lastly, the normalized second-order moment

$$\rho = \frac{\mu_{11}}{\sqrt{\mu_{20}\mu_{02}}} \qquad -1 \leq \rho \leq 1 \tag{F.3.4}$$

is known as the *correlation coefficient* of x and y.

F.4 Scaled-Normalized Centralized Moment

To obtain invariance to scaling, *scale-normalized centralized moments* η_{pq} are defined, given by:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{F.4.1}$$

where

$$\gamma = \frac{p+q}{2} + 1 \qquad \forall (p+q) \geq 2 \tag{F.4.2}$$

Of these moments, η_{30} and η_{03} are special in that they give an indication of the *skewness* of the image.

F.5 Hu's Moment Invariants

For rotational invariance, Hu [64] formulated additional moment invariants:

$$I_1 = \eta_{20} + \eta_{02} \tag{F.5.1}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{F.5.2}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{F.5.3}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{F.5.4}$$

$$\begin{aligned} I_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{F.5.5}$$

$$\begin{aligned} I_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\ & 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned} \tag{F.5.6}$$

$$\begin{aligned} I_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\ & (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{F.5.7}$$

It should be noted that these invariants are sensitive to noise. Appropriate filtering is therefore required to remove noise.

F.6 Efficient Calculation of the Centralized Moment

It should be clear that for large M and N, the calculation of the various moments becomes computationally expensive. An empirical evaluation of the direct unoptimised implementation (in the SDDSF) indicated moment calculation times of greater than a second on the target machine (to compute the Hu Invariants discussed above). Hence, instead of calculating the centralized moments via Eq. **(F.3.1)**, it is desirable to find equations for the centralized moments in terms of the already calculated moments. With this goal in mind, the relevant central moments can be written as:

$$\begin{aligned} \mu_{00} &= \sum_x \sum_y (x - \overline{x})^0 (y - \overline{y})^0 P_{xy} \\ &= \sum_x \sum_y P_{xy} \\ &= m_{00} \end{aligned} \tag{F.6.1}$$

$$\begin{aligned}
\mu_{10} &= \sum_x \sum_y (x-\overline{x})^1 (y-\overline{y})^0 P_{xy} \\
&= \sum_x \sum_y (x P_{xy} - \overline{x} P_{xy}) \\
&= \sum_x \sum_y x P_{xy} - \overline{x} \sum_x \sum_y P_{xy} \\
&= m_{10} - \overline{x}\, m_{00} \\
&= m_{10} - \frac{m_{10}}{m_{00}}\, m_{00} \\
&= 0
\end{aligned} \tag{F.6.2}$$

$$\begin{aligned}
\mu_{01} &= \sum_x \sum_y (x-\overline{x})^0 (y-\overline{y})^1 P_{xy} \\
&= \sum_x \sum_y (y P_{xy} - \overline{y} P_{xy}) \\
&= \sum_x \sum_y y P_{xy} - \overline{y} \sum_x \sum_y P_{xy} \\
&= m_{01} - \overline{y}\, m_{00} \\
&= m_{01} - \frac{m_{01}}{m_{00}}\, m_{00} \\
&= 0
\end{aligned} \tag{F.6.3}$$

$$\begin{aligned}
\mu_{11} &= \sum_x \sum_y (x-\overline{x})^1 (y-\overline{y})^1 P_{xy} \\
&= \sum_x \sum_y (xy P_{xy} - \overline{y} x P_{xy} - \overline{x} y P_{xy} + \overline{x}\,\overline{y} P_{xy}) \\
&= \sum_x \sum_y xy P_{xy} - \overline{y} \sum_x \sum_y x P_{xy} - \overline{x} \sum_x \sum_y y P_{xy} + \overline{x}\,\overline{y} \sum_x \sum_y P_{xy} \\
&= m_{11} - \overline{y}\, m_{10} - \overline{x}\, m_{01} + \overline{x}\,\overline{y}\, m_{00} \\
&= m_{11} - \frac{m_{01}}{m_{00}}\, m_{10} - \frac{m_{10}}{m_{00}}\, m_{01} + \frac{m_{10}}{m_{00}}\, \frac{m_{01}}{m_{00}}\, m_{00} \\
&= m_{11} - \frac{m_{01}\, m_{10}}{m_{00}} \\
&= m_{11} - \overline{x}\, m_{01}
\end{aligned} \tag{F.6.4}$$

$$
\begin{aligned}
\mu_{20} &= \sum_x \sum_y (x-\overline{x})^2 (y-\overline{y})^0 P_{xy} \\
&= \sum_x \sum_y (x^2 P_{xy} - 2\overline{x}\, x P_{xy} + \overline{x}^2 P_{xy}) \\
&= \sum_x \sum_y x^2 P_{xy} - 2\overline{x} \sum_x \sum_y x P_{xy} + \overline{y}^2 \sum_x \sum_y P_{xy} \\
&= m_{20} - 2\overline{x}\, m_{10} + \overline{x}^2\, m_{00} \\
&= m_{20} - 2\frac{m_{10}^2}{m_{00}} + \frac{m_{10}^2}{m_{00}} \\
&= m_{20} - \frac{m_{10}^2}{m_{00}} \\
&= m_{20} - \overline{x}\, m_{10}
\end{aligned}
\tag{F.6.5}
$$

$$
\begin{aligned}
\mu_{02} &= \sum_x \sum_y (x-\overline{x})^0 (y-\overline{y})^2 P_{xy} \\
&= \sum_x \sum_y (y^2 P_{xy} - 2\overline{y}\, y P_{xy} + \overline{y}^2 P_{xy}) \\
&= \sum_x \sum_y y^2 P_{xy} - 2\overline{y} \sum_x \sum_y y P_{xy} + \overline{y}^2 \sum_x \sum_y P_{xy} \\
&= m_{02} - 2\overline{y}\, m_{01} + \overline{y}^2\, m_{00} \\
&= m_{02} - 2\frac{m_{01}^2}{m_{00}} + \frac{m_{01}^2}{m_{00}} \\
&= m_{02} - \frac{m_{01}^2}{m_{00}} \\
&= m_{02} - \overline{y}\, m_{01}
\end{aligned}
\tag{F.6.6}
$$

$$
\begin{aligned}
\mu_{21} =& \sum_x \sum_y (x-\overline{x})^2 (y-\overline{y})^1 P_{xy} \\
=& \sum_x \sum_y (x^2 - 2\overline{x}\,x + \overline{x}^2)(y-\overline{y}) P_{xy} \\
=& \sum_x \sum_y (x^2 y P_{xy} - \overline{y}\,x^2 P_{xy} - 2\overline{x}\,xy P_{xy} + 2\overline{x}\,\overline{y}\,x P_{xy} + \overline{x}^2\,y P_{xy} - \overline{x}^2 \overline{y}\,P_{xy}) \\
=& \sum_x \sum_y x^2 y P_{xy} - \overline{y} \sum_x \sum_y x^2 P_{xy} - 2\overline{x} \sum_x \sum_y xy P_{xy} \\
& + 2\overline{x}\,\overline{y} \sum_x \sum_y x P_{xy} + \overline{x}^2 \sum_x \sum_y y P_{xy} - \overline{x}^2 \overline{y} \sum_x \sum_y P_{xy} \\
=& m_{21} - \overline{y}\,m_{20} - 2\overline{x}\,m_{11} + 2\overline{x}\,\overline{y}\,m_{10} + \overline{x}^2\,m_{01} - \overline{x}^2\overline{y}\,m_{00} \\
=& m_{21} - \frac{m_{01}}{m_{00}}\,m_{20} - 2\frac{m_{10}}{m_{00}}\,m_{11} + 2\frac{m_{10}}{m_{00}}\,\frac{m_{01}}{m_{00}}\,m_{10} + \frac{m_{10}^2}{m_{00}^2}\,m_{01} - \frac{m_{10}^2}{m_{00}^2}\frac{m_{01}}{m_{00}}\,m_{00} \\
=& m_{21} - \frac{m_{20}\,m_{01}}{m_{00}} - 2\frac{m_{10}\,m_{11}}{m_{00}} + 2\frac{m_{10}^2\,m_{01}}{m_{00}^2} + \frac{m_{10}^2\,m_{01}}{m_{00}^2} - \frac{m_{10}^2\,m_{01}}{m_{00}^2} \\
=& m_{21} - \overline{y}\,m_{20} - 2\overline{x}\,m_{11} + 2\overline{x}^2\,m_{01}
\end{aligned}
\tag{F.6.7}
$$

$$
\begin{aligned}
\mu_{12} =& \sum_x \sum_y (x-\overline{x})^1 (y-\overline{y})^2 P_{xy} \\
=& \sum_x \sum_y (x-\overline{x})(y^2 - 2\overline{y}\,y + \overline{y}^2) P_{xy} \\
=& \sum_x \sum_y (x\,y^2 P_{xy} - \overline{x}\,y^2 P_{xy} - 2\overline{y}\,xy P_{xy} + 2\overline{x}\,\overline{y}\,y P_{xy} + \overline{y}^2\,x P_{xy} - \overline{x}\,\overline{y}^2\,P_{xy}) \\
=& \sum_x \sum_y x\,y^2 P_{xy} - \overline{x} \sum_x \sum_y y^2 P_{xy} - 2\overline{y} \sum_x \sum_y xy P_{xy} \\
& + 2\overline{x}\,\overline{y} \sum_x \sum_y y P_{xy} + \overline{y}^2 \sum_x \sum_y x P_{xy} - \overline{x}, \overline{y}^2 \sum_x \sum_y P_{xy} \\
=& m_{12} - \overline{x}\,m_{02} - 2\overline{y}\,m_{11} + 2\overline{x}\,\overline{y}\,m_{01} + \overline{y}^2\,m_{10} - \overline{x}\,\overline{y}^2\,m_{00} \\
=& m_{12} - \frac{m_{10}}{m_{00}}\,m_{02} - 2\frac{m_{01}}{m_{00}}\,m_{11} + 2\frac{m_{10}}{m_{00}}\,\frac{m_{01}}{m_{00}}\,m_{01} + \frac{m_{01}^2}{m_{00}^2}\,m_{10} - \frac{m_{10}}{m_{00}}\,\frac{m_{01}^2}{m_{00}^2}\,m_{00} \\
=& m_{12} - \frac{m_{02}\,m_{10}}{m_{00}} - 2\frac{m_{01}\,m_{11}}{m_{00}^2} + 2\frac{m_{01}^2\,m_{10}}{m_{00}} + \frac{m_{01}^2\,m_{10}}{m_{00}^2} - \frac{m_{01}^2\,m_{10}}{m_{00}^2} \\
=& m_{12} - \overline{x}\,m_{02} - 2\overline{y}\,m_{11} + 2\overline{y}^2\,m_{10}
\end{aligned}
\tag{F.6.8}
$$

$$
\begin{aligned}
\mu_{30} &= \sum_x \sum_y (x-\overline{x})^3 (y-\overline{y})^0 P_{xy} \\
&= \sum_x \sum_y (x^2 - 2\overline{x}\,x + \overline{x}^2)(x-\overline{x}) P_{xy} \\
&= \sum_x \sum_y (x^3 P_{xy} - \overline{x}\,x^2 P_{xy} - 2\overline{x}\,x^2 P_{xy} + 2\overline{x}^2\,x P_{xy} + \overline{x}^2\,x P_{xy} - \overline{x}^3 P_{xy}) \\
&\quad = \sum_x \sum_y x^3 P_{xy} - \overline{x} \sum_x \sum_y x^2 P_{xy} - 2\overline{x} \sum_x \sum_y x^2 P_{xy} \\
&\qquad + 2\overline{x}^2 \sum_x \sum_y x P_{xy} + \overline{x}^2 \sum_x \sum_y x P_{xy} - \overline{x}^3 \sum_x \sum_y P_{xy} \\
&= m_{30} - \overline{x}\,m_{20} - 2\overline{x}\,m_{20} + 2\overline{x}^2\,m_{10} + \overline{x}^2\,m_{10} - \overline{x}^3\,m_{00} \\
&= m_{30} - 3\overline{x}\,m_{20} + 3\overline{x}^2\,m_{10} - \overline{x}^3\,m_{00} \\
&= m_{30} - 3\overline{x}\,m_{20} + 3\overline{x}^2\,m_{10} - \frac{m_{10}^3}{m_{00}^3}\,m_{00} \\
&= m_{30} - 3\overline{x}\,m_{20} + 3\overline{x}^2\,m_{10} - \overline{x}^2\,m_{10} \\
&= m_{30} - 3\overline{x}\,m_{20} + 2\overline{x}^2\,m_{10}
\end{aligned}
\tag{F.6.9}
$$

$$
\begin{aligned}
\mu_{03} &= \sum_x \sum_y (x-\overline{x})^0 (y-\overline{y})^3 P_{xy} \\
&= \sum_x \sum_y (y^2 - 2\overline{y}\,y + \overline{y}^2)(y-\overline{y}) P_{xy} \\
&= \sum_x \sum_y (y^3 P_{xy} - \overline{y}\,y^2 P_{xy} - 2\overline{x}\,y^2 P_{xy} + 2\overline{y}^2\,y P_{xy} + \overline{y}^2\,y P_{xy} - \overline{y}^3 P_{xy}) \\
&\quad = \sum_x \sum_y y^3 P_{xy} - \overline{y} \sum_x \sum_y y^2 P_{xy} - 2\overline{y} \sum_x \sum_y y^2 P_{xy} \\
&\qquad + 2\overline{y}^2 \sum_x \sum_y y P_{xy} + \overline{y}^2 \sum_x \sum_y y P_{xy} - \overline{y}^3 \sum_x \sum_y P_{xy} \\
&= m_{03} - \overline{y}\,m_{02} - 2\overline{y}\,m_{02} + 2\overline{y}^2\,m_{01} + \overline{y}^2\,m_{01} - \overline{y}^3\,m_{00} \\
&= m_{03} - 3\overline{y}\,m_{02} + 3\overline{y}^2\,m_{01} - \overline{y}^3\,m_{00} \\
&= m_{03} - 3\overline{y}\,m_{02} + 3\overline{y}^2\,m_{01} - \frac{m_{01}^3}{m_{00}^3}\,m_{00} \\
&= m_{03} - 3\overline{y}\,m_{02} + 3\overline{y}^2\,m_{01} - \overline{y}^2\,m_{01} \\
&= m_{03} - 3\overline{y}\,m_{02} + 2\overline{y}^2\,m_{01}
\end{aligned}
\tag{F.6.10}
$$

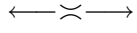
These equations therefore allow the required centralized moments to be calculated via several floating-point operations instead of a minimum of $M \times N$ floating-point operations.

F.7 Summary

The Hu Invariant Moments allow the characterisation of an image in a fashion that is invariant to translation, rotation and scale changes. From a computational point of view, the calculation of the

standard and centralized moments represents the primary source of execution time. The investigation above therefore attempted to optimise the computational cost by minimizing the costly image integration. By using the elaborated formulation given above, the subsequent versions of moments can be calculated with comparatively little cost. The source code provides routines to calculate the moments via both approaches.

To evaluate the performance enhancement offered by the elaborated formulation, an empirical test was performed. The Hu Invariant Moments were calculated for a sample image with the dimensions used in the SDDS application. The calculation was performed using the standard benchmarking mechanism (refer to Section 4.12). The original formulation allowed 0.379 calculations of the Hu Invariant Moments to be performed per second. The elaborated formulation allowed 32.746 calculations per second to be performed, translating to a speed-up factor of approximately 86. The effort required to produce the elaborated formulation therefore resulted in a significant performance gain.



APPENDIX G

Feed-Forward Neural Network Error Gradients

This appendix provides the derivations of the gradients of the error functions with respect to the weights of a single hidden layer Feed-Forward Neural Network (FFNN). These equations derived in this section are implemented in the SDDSF.

It is assumed that the sum squared error (SSE) is used as the objective function. Then for each pattern, p, the error is defined as

$$\mathcal{E}_p = \frac{1}{2}\frac{\sum_{k=1}^{K}(t_{k,p} - o_{k,p})^2}{K} \tag{G.0.1}$$

where K is the number of output units, and $t_{k,p}$ and $o_{k,p}$ are respectively the target and actual output values of the k-th output unit.

The rest of the derivations refer to an individual pattern. The pattern subscript, p, is therefore omitted for notational convenience. Similarly, the reference to time, t will be omitted. Also sigmoid activation functions in the hidden and output units are assumed, as well as the presence of bias units in the form of augmented vectors. Then,

$$o_k = f_{o_k}(net_{o_k}) = \frac{1}{1 + e^{-net_{o_k}}} \tag{G.0.2}$$

and

$$y_j = f_{y_j}(net_{y_j}) = \frac{1}{1 + e^{-net_{y_j}}} \tag{G.0.3}$$

with

$$net_{o_k} = \sum_{j=1}^{J+1} w_{kj} y_j \tag{G.0.4}$$

and

$$net_{y_j} = \sum_{i=1}^{I+1} v_{ji} z_i \tag{G.0.5}$$

where $I+1$ is the number of augmented inputs, $J+1$ is the number of augmented hidden units, w_{kj} are the output unit weights, v_{ji} hidden unit weights, z_i is the i-th input and y_j is the output of the j-th hidden unit.

For most of the back-propagation algorithms, derivations for the gradient of the error function with respect to the weights is desired, in other words, $\frac{\partial \mathcal{E}}{\partial w_{kj}}$ and $\frac{\partial \mathcal{E}}{\partial v_{ji}}$. For these derivations, the chain rule will be used.

G.1 Output Units

For the output units, the error function gradient with respect to weights will be:

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = \frac{\partial \mathcal{E}}{\partial o_k}\frac{\partial o_k}{\partial w_{kj}} \tag{G.1.1}$$

$$= \frac{\partial \mathcal{E}}{\partial o_k}\frac{\partial o_k}{\partial net_{o_k}}\frac{\partial net_{o_k}}{\partial w_{kj}} \tag{G.1.2}$$

From the objective function, Eq. **(G.0.1)**,

$$\frac{\partial \mathcal{E}}{\partial o_k} = \frac{\partial}{\partial o_k}(\frac{1}{2}\frac{\sum_{k=1}^{K}(t_k - o_k)^2}{K}) = -(t_k - o_k) \tag{G.1.3}$$

From Eq. **(G.0.2)**,

$$\frac{\partial o_k}{\partial net_{o_k}} = \frac{\partial f_{o_k}}{\partial net_{o_k}} = (1 - o_k)o_k \tag{G.1.4}$$

and from Eq. **(G.0.4)**,

$$\frac{\partial net_{o_k}}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}(\sum_{j=1}^{J+1} w_{kj}y_j) = y_j \tag{G.1.5}$$

Then from the above,

$$\frac{\partial \mathcal{E}}{\partial w_{kj}} = -(t_k - o_k)(1 - o_k)o_k y_j \tag{G.1.6}$$

In the case where there are direct connections between the inputs and the output units, with weights w_{ki}:

$$\frac{\partial \mathcal{E}}{\partial w_{ki}} = \frac{\partial \mathcal{E}}{\partial o_k}\frac{\partial o_k}{\partial w_{ki}} \tag{G.1.7}$$

$$= \frac{\partial \mathcal{E}}{\partial o_k}\frac{\partial o_k}{\partial net_{o_k}}\frac{\partial net_{o_k}}{\partial w_{ki}} \tag{G.1.8}$$

where, the network output net_{o_k} is now given by

$$net_{o_k} = \sum_{j=1}^{J+1} w_{kj} y_j + \sum_{i=1}^{I+1} w_{ki} z_i \tag{G.1.9}$$

and the derivative with respect to w_{ki} is

$$\frac{\partial net_{o_k}}{\partial w_{ki}} = \frac{\partial}{\partial w_{ki}} (\sum_{j=1}^{J+1} w_{kj} y_j + \sum_{i=1}^{I+1} w_{ki} z_i) = z_i \tag{G.1.10}$$

then

$$\frac{\partial \mathcal{E}}{\partial w_{ki}} = -(t_k - o_k)(1 - o_k) o_k z_i \tag{G.1.11}$$

G.2 Hidden Units

For the hidden units, the error function gradient with respect to weights will be:

$$\frac{\partial \mathcal{E}}{\partial v_{ji}} = \frac{\partial \mathcal{E}}{\partial y_j} \frac{\partial y_j}{\partial v_{ji}} \tag{G.2.1}$$

$$= \frac{\partial \mathcal{E}}{\partial y_j} \frac{\partial y_j}{\partial net_{y_j}} \frac{\partial net_{y_j}}{\partial v_{ji}} \tag{G.2.2}$$

From the objective function, Eq. **(G.0.1)**,

$$\frac{\partial \mathcal{E}}{\partial y_j} = \frac{\partial}{\partial y_j} (\frac{1}{2} \frac{\sum_{k=1}^{K} (t_k - o_k)^2}{K}) \tag{G.2.3}$$

$$= \sum_{k=1}^{K} \frac{\partial \mathcal{E}}{\partial o_k} \frac{\partial o_k}{\partial net_{o_k}} \frac{\partial net_{o_k}}{\partial y_j} \tag{G.2.4}$$

$$= \sum_{k=1}^{K} -(t_k - o_k)(1 - o_k) o_k w_{kj} \tag{G.2.5}$$

where $\frac{\partial net_{o_k}}{\partial y_j}$ is given as:

$$\frac{\partial net_{o_k}}{\partial y_j} = \frac{\partial}{\partial y_j} (\sum_{j=1}^{J+1} w_{kj} y_j) = w_{kj} \tag{G.2.6}$$

From Eq. **(G.0.3)**,

$$\frac{\partial y_j}{\partial net_{y_j}} = \frac{\partial f_{y_j}}{\partial net_{y_j}} = (1 - y_j)y_j \tag{G.2.7}$$

and from Eq. **(G.0.5)**,

$$\frac{\partial net_{y_j}}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}(\sum_{i=1}^{I+1} v_{ji}z_i) = z_i \tag{G.2.8}$$

Then from the above,

$$\frac{\partial \mathcal{E}}{\partial v_{ji}} = \Big[\sum_{k=1}^{K} -(t_k - o_k)(1 - o_k)o_k w_{kj}\Big]\,(1 - y_j)y_j z_i \tag{G.2.9}$$

$$\longleftarrow\asymp\longrightarrow$$

APPENDIX H

Project Timeline

To convey a sense of the progression and evolution of the project, this appendix gives a brief timeline of the project.

H.1 Initial Prototype Development

This section gives a summary of the life-time of the initial prototype. Reference to time-lines are approximate.

- February 2002: The first meeting and discussion of broad project requirements.
- March 2002: First roof samples delivered for inspection and experimentation with webcam and data projector as light source to test if a lighting model is a feasible solution.
- Mid-March 2002: Theoretical lighting model proposed.
- Mid-April 2002: First meeting with Business Enterprises (BE) at the University of Pretoria.
- Mid-April 2002: A theoretical study on the accuracy of the proposed lighting model completed.
- July 2002: Mounting table delivered at University of Pretoria (UP) for scanning roofs.
- Mid-August 2002: Final discussion on formal contract between BMW and BE@UP.
- **2 September 2002**: *First demonstration of lighting model.*
- September 2002: Attempt to find space at UP for robot.
- Mid-December 2002: Robot moves to SAR Electronic.
- 14 January 2003: Second demonstration to illustrate the system using UP bought camera and laser.
- Mid-May 2003: Received new camera and laser. New team member, Evangelos Papacostantis takes over from Dr Van den Bergh.
- July 2003: System ported to new camera and laser, and works successfully to detect dents, with a clear improvement compared to the previous equipment.
- August 2003: Design and implementation of gripper completed.

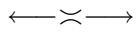
- **1 December 2003**: *Third demonstration*.
- **21 January 2004**: *Fourth demonstration*.

H.2 Production Prototype Development

The development of the production prototype commenced as a separate but continued development of the initial prototype. Note that once an activity was initiated, generally the activity continued to be performed in parallel with subsequent activities until completion. The production prototype made use of new components and equipment (except for the camera, laser and processing server). Similarly, none of the original software developed for the initial prototype was used in the development of the production prototype. The production prototype was therefore essentially a new and original development using concepts from the initial prototype.

- February 2004: New location for prototype development identified.
- March 2004: Development of project specification for the production prototype.
- April to May 2004: Development and refinement of quotation for the production prototype.
- May 2004: Published invitation for new software team member.
- June to October 2004: Establishment of new project team, with new project manager, and incorporating engineers from Department of Industrial Engineering.
- October to November 2004: Initial meetings of new project team. Planning of initial phase of production prototype.
- Early-November 2004: Installation and calibration of KUKA Industrial Robot at Groenkloof Campus of the University of Pretoria. Author joins project team.
- Mid-November 2004: Development of support table. Quotation for facilities improvement (cleaning, painting, furniture, *etc.*). Delivery of sample roof panels targeted for this project.
- End-November 2004: Development of gripper. Visit to Rosslyn Plant (BMW) in order to observe production operation. Obtained first CAD model of new roof panels.
- Early-December 2004: Delivery and installation of support table and gripper.
- Mid-January 2005: Facility improvement completed. Commencing of first phase of SDDS development (porting of initial prototype system to new robot).
- **17 February 2005**: *Milestone demonstration: Phase 1*, completion of first phase of production prototype.
- Late-February 2005: Characterisation of imaging system.
- February to March 2005: Examination of various CATIA export formats, development of CAD model reading module. Numerous visits to BMW Rosslyn plant.
- April-May 2005: Investigation of robot motion (vibration, profile following, KRL program generation).

- May 2005: Development of new software infrastructure, the SDDSF.
- Early-June 2005: Initial results from new SDDSF.
- June 2005: Measurement of ambient conditions at BMW Rosslyn Plant.
- July 2005: Investigation of marker systems. Specification of new camera, laser and marker mounting system. New cables and digipeater ordered from Germany for camera (old cable defective).
- August 2005: Development of new camera, laser and marker mounting system initiated.
- September 2005: KUKA Robot Controller to Linux server communication investigated. Installation of new camera, laser and marker mounting system. New camera cables and digipeater delivered and installed.
- September to November 2005: Development of stage 1 processing software for the SDDSF.
- November 2005 to January 2006: Development of stage 2 processing software for the SDDSF.
- January to February 2006: Development of stage 3 processing software for the SDDSF.
- Mid-March 2006: Demonstration of SQ/2 Ink Jet Printing System. Telephonic communication with KUKA Germany to solve robot communication problems and obtain Ethernet-RSI module.
- Late-March 2006: Installation and integration of ink-jet printing system. Delivery and installation of Ethernet-RSI module.
- April 2006: Communication of robot with Linux server established. Control of printing system established.



Bibliography

[1] *Proceedings of the International Conference on Sensors for Nondestructive Testing: Measuring the Quality of Fresh Fruits and Vegetables*. Natural Resource, Agriculture, and Engineering Service (NRAES), 1997.

[2] 3M™. Mini D Ribbon (MDR) Cable Assembly, 2003. (available from `http://www.3m.com/interconnects/`).

[3] ABRAMOWITZ, M., AND STEGUN, I. A., Eds. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Table*. Dover Publications, 1965.

[4] ANDERSON, B. D. O., AND MOORE, J. B. *Optimal Filtering*. Dover Publications, Inc., 1979.

[5] ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., CROZ, J. D., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. *LAPACK User's Guide*, third ed. Society for Industrial and Applied Mathematics (SIAM), 1999.

[6] ANGEL, E. *Interactive Computer Graphics: A Top-Down Approach with OpenGL*, third ed. Addison Wesley, 2002.

[7] ATKINSON, A., AND RIANI, M. *Robust Diagnostic Regression Analysis*. Springer, 2000.

[8] BALLARD, D. H., AND BROWN, C. M. *Computer Vision*, first ed. Prentice-Hall Inc., 1982.

[9] BANGHAM, J. A., AND MARSHALL, S. Image and signal processing with mathematical morphology. *Electronics & Communication Engineering Journal* (June 1998), 117–128.

[10] BEATON, A. E., AND TUKEY, J. W. The fitting of power series. *Technometrics*, 16 (1975), 147–185.

[11] BELSLEY, D. A., KUH, E., AND WELSCH, R. E. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley-Interscience, 2004.

[12] BEN-ARI, M. *Mathematical Logic for Computer Science*, second ed. Springer-Verlag, 2001.

[13] BENTLEY, J. *Programming Pearls*, second ed. Addison-Wesley, 2000.

[14] BISHOP, C. M. *Neural Networks for Pattern Recognition*, first ed. Oxford University Press, 1995.

[15] BLINN, J. F. Models of light reflection for computer synthesized pictures. *Computer Graphics 11*, 2 (1977), 40–53.

[16] BOOR, C. D. *A Practical Guide to Splines*. Springer, 2001.

[17] BRACEWELL, R. N. *The Fourier Transform and Its Applications*, third ed. McGraw-Hill Higher Education, 2000.

[18] BRIGHAM, E. O. *The Fast Fourier Transform*. Prentice Hall, 1988.

[19] BUECHE, F. J. *Introduction to Physics for Scientists and Engineers*, fourth ed. McGraw-Hill, Singapore, 1986.

[20] BURNS, A., AND WELLINGS, A. *Real-Time Systems and their Programming Languages*. Addison-Wesley, 1990.

[21] CASTLEMAN, K. R. *Digital Image Processing*, second ed. Prentice Hall, 1996.

[22] CHAN, L., AND FALLSIDE, F. An adaptive training algorithm for back propagation networks. *Computer Speech and Language 2* (1987), 205–218.

[23] COLLINS-SUSSMAN, B., FITZPATRICK, B. W., AND PILATO, C. M. *Version Control with Subversion*, first ed. O'Reilly Media, Inc., 2004.

[24] COOK, R. L., AND TORRANCE, K. E. A Reflectance Model for Computer Graphics. *Computer Graphics 15*, 3 (1982), 307–316.

[25] COOLEY, J. W., AND TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation 19* (1965), 297–301.

[26] COOLING, J. *Software Design for Real-Time Systems*. Chapman and Hall, 1991.

[27] CRAIG, J. J. *Introduction to Robotics: Mechanics and Control*, third ed. Prentice Hall, 2003.

[28] CRUMP, N. D. A Kalman filter approach to the deconvolution of seismic signals. *Geophysics 39* (1974), 1–13.

[29] DARKEN, C., AND MOODY, J. *Note on Learning Rate Schedules for Stochastic Optimization*, vol. 3. Morgan Kaufmann, 1991.

[30] DELLAERT, F., AND THORPE, C. Robust car tracking using Kalman filtering and Bayesian templates. In *Proceedings of SPIE: Intelligent Transportation Systems* (October 1997), vol. 3207.

[31] DEMMEL, J., DONGARRA, J., EIJKHOUT, V., FUENTES, E., PETITET, A., VUDUC, R., WHALEY, R. C., AND YELICK, K. Self Adapting Linear Algebra Algorithms and Software. *Proceedings of the IEEE 93*, 2 (2005), 293–312.

[32] DEMMEL, J. W. *Applied Numerical Linear Algebra*, first ed. Society for Industrial and Applied Mathematics (SIAM), 1997.

[33] DONOHO, D. L., AND HUBER, P. J. *The notion of breakdown point*. Chapman & Hall/CRC, 1985, pp. 157–184.

[34] DRÉCOURT, J.-P., AND MADSEN, H. Uncertainty estimation in groundwater modelling using Kalman filtering. In *Proceedings of the 4th International Conference on Calibration and Reliability in Groundwater Modelling* (2002), pp. 306–309.

[35] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification*, second ed. John Wiley & Sons, 2001.

[36] ENGELBRECHT, A., VAN DEN BERGH, F., AND PAPACOSTANTIS, E. Surface Anomaly Detection using a Structural Light System. Tech. rep., Department of Computer Science, University of Pretoria, February 2004.

[37] ENGELBRECHT, A. P. *Computational Intelligence*, first ed. John Wiley & Sons, 2002.

[38] ENGINEERING DESIGN TEAM, INC. (EDT). PCI DV User's Guide, 2005. (available from `http://www.edt.com`).

[39] FAHLMAN, S. E. An empirical study of learning speed in back-propogation networks. Tech. Rep. CMU-CS-88-162, Department of Computer Science, Carnegie-Mellon University, 1998.

[40] FISCHLER, M., AND BOLLES, R. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[41] FOLEY, J. D., VAN DAM, A., FEINER, S. K., AND HUGES, J. F. *Computer Graphics: Principles and Practice*, second ed. Addison-Wesley, 1997.

[42] FORSYTH, D. A., AND PONCE, J. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[43] FRIGO, M., AND JOHNSON, S. G. The Design and Implementation of FFTW3. *Proceedings of the IEEE 93*, 2 (2005), 216–231.

[44] GLEB, A., Ed. *Applied Optimal Estimation*. MIT Press, 1974.

[45] GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations*, third ed. John Hopkins University Press, 1996.

[46] GONZALEZ, R. C., AND WOODS, R. E. *Digital Image Processing*, second ed. Prentice-Hall International, 2002.

[47] GOURAUD, H. *Computer Display of Curved Surfaces*. PhD thesis, University of Utah, 1971.

[48] GROBLER, H. Neural Network Optimization Benchmark. Tech. Rep. Unpublished, Department of Computer Science, University of Pretoria, 2003.

[49] HALSALL, F. *Data Communications, Computer Networks, and Open Systems*, third ed. Addison Wesley, 1992.

[50] HAMMING, R. W. *Numerical Methods for Scientists and Engineers*, second ed. Dover Publications, 1973.

[51] HAMMING, R. W. *Digital Filters*. Dover Publications, 1998.

[52] HAMPEL, F. R., RONCHETTI, E. M., AND ROUSSEEUW, P. J. *Robust Statistics: The Approach Based on Influence Functions*. Wiley-Interscience, 1986.

[53] HANRAHAN, P., AND KREUGER, W. Reflection from layered surfaces due to sub-surface scattering. In *Proceedings of SIGGRAPH'93* (1993), pp. 165–175.

[54] HASSOUN, M. H. *Fundamentals of Artificial Neural Networks*. MIT Press, 1995.

[55] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. H. *The Elements of Statistical Learning*. Springer, 2003.

[56] HAUSI, H. A., AND MELCHER, J. R. *Electromagnetic Fields and Energy*. Prentice Hall, 1989.

[57] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*, second ed. Prentice Hall, 1998.

[58] HAYKIN, S. *Adaptive Filter Theory*, forth ed. Prentice-Hall, Inc., 2002.

[59] HE, X. D., TORRANCE, K. E., SILLION, F. X., AND GREENBERG, D. P. A Comprehensive Physical Model for Light Reflection. *Computer Graphics 25*, 4 (1991), 175–186.

[60] HIGHAM, N. J. *Accuracy and Stability of Numerical Algorithms*, first ed. Society for Industrial and Applied Mathematics (SIAM), 2002.

[61] HOLLAND, P. W., AND WELSCH, R. E. Robust regression using Iteratively Reweighted Least-Squares. *Communication Statistics (Theory and Methods) A6* (1977), 813–828.

[62] HOUTEKAMER, P. L., AND MITCHELL, H. L. A Sequential Ensemble Kalman Filter for Atmospheric Data Assimilation. *Monthly Weather Review 129*, 1 (2001), 123–137.

[63] HOWEL, S. B. *Handbook of CCD Astronomy*. Cambridge University Press, 2000.

[64] HU, M. K. Visual Pattern Recognition By Moment Invariants. *IRE Transactions on Information Theory 8* (1968), 179–187.

[65] HUBER, P. J. *Robust Statistics*. John Wiley & Sons, Inc., 2004.

[66] HUTH, M. R. A., AND RYAN, M. D. *Logic in Computer Science: Modelling and reasoning about systems*, first ed. Cambridge University Press, 2002.

[67] JACOBS, R. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1 (1988), 295–307.

[68] JAIN, A. K. *Fundementals of Digital Image Processing*. Prentice Hall, 1989.

[69] JOHNSON, R. A., AND WICHERN, D. W. *Applied Multivariate Statistical Analysis*, fifth ed. Prentice Hall, 2002.

[70] KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering 82 (Series D)* (1960), 35–45.

[71] KAUFMAN, L., AND ROUSSEEUW, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*, second ed. Wiley-Interscience, 2005.

[72] KILGARD, M. J. *The OpenGL Utility Toolkit (GLUT) Programming Interface*. Silicon Graphics, Inc., Nov 1996. API Version 3.

[73] KREYSZIG, E. *Advanced Engineering Mathematics*, sixth ed. John Wiley & Sons, Inc., 1988.

[74] KUKA ROBOTER GMBH. *Ethernet-RSI Software Module*, 2.0 ed.

[75] KUKA ROBOTER GMBH. *KR C CREAD CWRITE*, 4.1 ed.

[76] KUKA ROBOTER GMBH. *KR C Operator Control*, 4.1 ed.

[77] KUKA ROBOTER GMBH. *KR C Robot Sensor Interface (RSI)*, 2.0 ed.

[78] KUKA ROBOTER GMBH. *KR C Setup*, 4.1 ed.

[79] KUKA ROBOTER GMBH. *KR C Start-up*, 4.1 ed.

[80] LAPLANTE, P. A. *Real-Time Systems Design and Analysis: An Engineer's Handbook*, second ed. IEEE Press, 1997.

[81] LE CUN, Y., SIMARD, P. Y., AND PEARLMUTTER, B. A. *Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors*, vol. 5. Morgan Kaufmann, 1993, pp. 156–163.

[82] LITWILLER, D. CCD vs. CMOS: Facts and Fiction. *Photonics Spectra* (January 2001).

[83] MAGOULAS, G. D., VRAHATIS, M. N., AND ANDROULAKIS, G. S. Effective backpropagation training with variable stepsize. *Neural Networks 10*, 1 (1997), 69–82.

[84] MATHERON, G. *Random Sets and Integral Geometry*. John Wiley & Sons Inc., 1975.

[85] MATSUMOTO, M., AND NISHIMURA, T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation 8*, 1 (1998), 3–30.

[86] MITCHELL, T. M. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[87] MOORE, E. H. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society 26* (1920), 394–395.

[88] MORGAN, C. *Programming from Specifications*, first ed. Prentice Hall, 1998.

[89] NEVATIA, R. *Machine Perception*. Prentice Hall, 1982.

[90] NEWMAN, D., HETTICH, S., BLAKE, C., AND MERZ, C. UCI Repository of Machine Learning Databases, 1998. University of California, Irvine, Department of Information and Computer Sciences, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[91] NISE, N. S. *Control Systems Engineering*. John Wiley & Sons, 2000.

[92] OGATA, K. *Modern Control Engineering*, forth ed. Prentice Hall, 2002.

[93] OPPENHEIM, A. V., AND SCHAFER, R. W. *Discrete-Time Signal Processing*, second ed. Prentice Hall, 1999.

[94] PAPOULIS, A., AND PILLAI, S. U. *Probability, Random Variables and Stochastic Processes*, forth ed. McGraw-Hill Science/Engineering/Math, 2001.

[95] PENROSE, R. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society 51* (1955), 406–413.

[96] PENROSE, R. On best approximate solution of linear matrix equations. *Proceedings of the Cambridge Philosophical Society 52* (1956), 17–19.

[97] PETERSEN, K. B., AND PEDERSEN, M. S. The Matrix Cookbook, 2005. Version 20051003.

[98] PHONG, B. T. *Illumination for Computer Generated Images*. PhD thesis, University of Utah, 1973.

[99] PHONG, B. T. Illumination for Computer-Generated Pictures. *Communications of the ACM 18*, 6 (1975), 311–317.

[100] PHOTONFOCUS. *MV-D1024-160 Camera User's Manual*, 1.0 ed. (available from `http://www.photonfocus.com`).

[101] PHOTONFOCUS. Application Node 002: LVDS, 2004. (available from `http://www.photonfocus.com`).

[102] PHOTONFOCUS. Application Note 021: CameraLink™, 2004. (available from `http://www.photonfocus.com`).

[103] PHOTONFOCUS. Digipeater CLB26, 2004. (available from `http://www.photonfocus.com`).

[104] PLAUT, D. C., NOWLAN, S. J., AND HINTON, G. E. Experiments on Learning by Back Propagation. Tech. Rep. CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, 1986.

[105] PRATT, W. K. *Digital Image Processing*, third ed. Wiley-Interscience, 2001.

[106] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C*, second ed. Cambridge University Press, 1992.

[107] PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*, fifth ed. McGraw-Hill, 2001.

[108] PROAKIS, J. G., AND MANOLAKIS, D. G. *Digital Signal Processing*, second ed. Macmillian Publishing Company, 1992.

[109] RICHARD C. SINGLETON. An Algorithm for Computing the Mixed Radix Fast Fourier Transform. *IEEE Transactions on Audio and Electroacoustics AU-17*, 2 (June 1969), 93–103.

[110] RIPLEY, B. D. *Pattern Recognition and Neural Networks*, first ed. Cambridge University Press, 1996.

[111] RIUS, A., RUFFINI, G., AND CUCURULL, L. Improving the vertical resolution of ionospheric tomography with GPS occultations. *Geophysical Research Letters 24*, 18 (1997), 2291–2295.

[112] ROB, P., AND CORONEL, C. *Database Systems: Design, Implementation and Management*. Wadsworth Publishing Company, 1993.

[113] ROSENFELD, A. A characterization of parallel thinning algorithms. *Information and Control 29* (1975), 286–291.

[114] ROSENFIELD, R., AND KAK, A. C. *Digital Picture Processing*, second ed., vol. 1. Academic Press, 1982.

[115] ROSENFIELD, R., AND KAK, A. C. *Digital Picture Processing*, second ed., vol. 2. Academic Press, 1982.

[116] ROUSSEEUW, P. J., AND LEROY, A. M. *Robust Regression and Outlier Detection*, first ed. John Wiley & Sons, 2003.

[117] RUSS, J. C. *The Image Processing Handbook*, forth ed. CRC Press, 2002.

[118] RUSSELL, S., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*, second ed. Prentice Hall, 2003.

[119] SALOMON, R., AND VAN HEMMEN, J. L. Accelerating Backpropagation through Dynamic Self-Adaptation. *Neural Networks 9*, 4 (1996), 589–601.

[120] SCHMIDT, S. The Kalman filter: Its recognition and development for aerospace applications. *AIAA Journal of Guidance, Control and Dynamics 4*, 1 (1981), 4–7.

[121] SERRA, J. *Image Analysis and Mathematical Morphology*. Academic Press, 1988.

[122] SIMON, D., AND SIMON, D. L. Aircraft Turbofan Engine Health Estimation Using Constrained Kalman Filtering. *ASME Journal of Engineering for Gas Turbines and Power 127*, 2 (2005), 323–328.

[123] SONG, Y., AND ZHANG, A. Locating Image Background by Monotonic Tree.

[124] SORENSON, H. W. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum 7* (July 1970), 63–68.

[125] STANSFIELD, E. V. Introduction to Kalman Filters, 2001. UK Adaptive Signal Processing Club Discussion Meeting, Tutorial Session, 7 March 2001.

[126] STEFANELLI, R., AND ROSENFELD, A. Some parallel thinning algorithms for digital pictures. *Journal of the ACM 18* (1971), 255–264.

[127] STENGEL, R. F. *Optimal Control and Estimation*. Dover Publications, Inc., 1994.

[128] STEWART, C. V., BUBNA, K., AND PERERA, A. Estimating Model Parameters and Boundaries By Minimizing a Joint, Robust Objective Function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (1999), pp. 387–393.

[129] STEYN, W. H. *A Multi-mode Attitude Determination and Control System for Small Satellites*. PhD thesis, University of Stellenbosch, 1995.

[130] STOKERYALE CANADA INC. Lariris™ SNF Laser, 2003. (available from `http://www.stokeryale.com/lasers`).

[131] STRANG, G. *Linear Algebra and Its Applications*, forth ed. Thomson Brooks/Cole, 2006.

[132] STROMBERG, A. J. Computing the Exact Least Median of Squares Estimate and Stability Diagnostics in Multiple Linear Regression. *SIAM Journal on Scientific Computing 14* (1993), 1289–1299.

[133] STUART, A., AND ORD, K. *Kendall's Advanced Theory of Statistics*, sixth ed., vol. 1. Arnold (Hodder Headline Group), 1994.

[134] TANENBAUM, A. S. *Computer Networks*, forth ed. Pearson Education International, 2003.

[135] TOLLENAERE, T. SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks 3* (1990), 561–573.

[136] TORRANCE, K. E., AND SPARROW, E. M. Theory of Off-Specular Reflection from Roughened Surfaces. *Journal of the Optical Society of America 56*, 9 (1967), 1105–1114.

[137] TORRANCE, K. E., SPARROW, E. M., AND BIRKEBAK, R. C. Polarization, Directional Distribution, and Off-Specular Peak Phenomena in Light Reflected from Roughened Surfaces. *Journal of the Optical Society of America 56*, 7 (1966), 916–925.

[138] TREFETHEN, L. N., AND BAU, III, D. *Numerical Linear Algebra*, first ed. Society for Industrial and Applied Mathematics (SIAM), 1997.

[139] U.S. PRODUCT DATA ASSOCIATION. *U.S. Product Data Association*, 5.3 ed.

[140] VERNON, D. *Machine Vision*. Prentice-Hall, 1991.

[141] VOGL, T. P., MANGLIS, J. K., RIGLER, A. K., ZINK, T. W., AND ALKON, D. L. Accelerating the Convergence of the Backpropagation Method. *Biological Cybernetics 59* (1988), 257–263.

[142] WAKERLY, J. F. *Digital Design Principles and Practices*. Prentice-Hall International, Englewood Cliffs, New Jersey, 1990.

[143] WASSERMAN, L. *All of Statistics*. Springer-Verlag, 2004.

[144] WATT, A. *3D Computer Graphics*, third ed. Pearson Education Limited, 2000.

[145] WELCH, G., AND BISHOP, G. An Introduction to the Kalman Filter. In *Proceedings of SIGGRAPH'01* (2001). Course 8.

[146] WESSELS, L., AND BARNARD, E. Avoiding false local minima by proper initialisation of connections. *IEEE Transactions on Neural Networks 3*, 6 (1992), 899–905.

[147] WESTIN, S. H., LI, H., AND TORRANCE, K. E. A Comparison of Four BRDF Models. Tech. Rep. PCG-04-02, Program of Computer Graphics, Cornell University, 2004.

[148] WESTIN, S. H., LI, H., AND TORRANCE, K. E. A Field Guide to BRDF Models. Tech. Rep. PCG-04-01, Program of Computer Graphics, Cornell University, 2004.

[149] WHALEY, R. C., AND DONGARRA, J. J. Automatically Tuned Linear Algebra Software. Tech. Rep. UT-CS-97-366, University of Tennessee, Department of Computer Science, 1997.

[150] WHALEY, R. C., PETITET, A., AND DONGARRA, J. J. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing 27*, 1–2 (2001), 3–35.

[151] WIDROW, B., AND STEARNS, S. D. *Adaptive Signal Processing*, first ed. Prentice-Hall, 1985.

Glossary

A

a posteriori: dictionary definition: from effects to causes; involving reasoning thus; from what comes after. p 117

a priori: dictionary definition: from causes to effect, from general principle to particular instance; involving reasoning thus; assumed without investigation; (of knowledge) existing in the mind independently of sensory experience; from what is before. p 117

Aberration: The failure of an optical lens to produce an exact point-to-point correspondence between the object and its resulting image. Various types are chromatic, spherical, coma, astigmatism and distortion. p 77

Abrasive: Coarse, hard, sharp material which is used to wear away the surface of softer, less resistant materials. Many manufactured products are given a smooth finish using an abrasive applied by, *e.g.*, grinding wheels, sandpapers, honing stones, polishes, sandblasting, pulpstones, ball mills. p 3

Adaptive Learning Rate: Learning rate that is adjusted according to an algorithm during training to minimize training time. p 126

Alloy: A metal-like substance produced by mixing two or more metals or non-metals. Ceramics can also be mixed to form alloys. A binary alloy contains two components and a ternary alloy contains three. p 3

Ambient Light: Light which is present in the environment of the imaging front end of a vision system and generated from outside sources. This light, unless used for actual scene illumination, will be treated as background noise by the vision system. p 88

Analog-to-Digital Converter: A device which converts an analog voltage or current signal to a discrete series of digitally encoded numbers for computer processing. p 78

Angle of Incidence: When light strikes a surface it forms an angle with an imaginary line known as the *normal*, which is perpendicular to the surface. The angle created between the incident ray and the normal is referred to as the angle of incidence. p 14

Anisotropy: Anisotropy (the opposite of isotropy) is the property of being directionally dependent. Something which is anisotropic, may appear different, or have different characteristics in different directions. p 14

Aperture: Aperture is the opening of the camera lens; it is measured in f-stops where the lower the f-stop the wider the opening is. p 76

Application Program Interface: A set of routines, protocols, and tools for building software applications. p 45

Artificial Intelligence: The capability of a computer to perform functions normally attributed to human intelligence, such as learning, adapting, recognizing, classifying, reasoning, self-correction and improvement. A sub-field of Computational Intelligence. p 124

Atomicity: Results of a transaction's execution are either all committed or all rolled back. All changes take effect, or none do. p 52

B

B-Spline: B-spline (basis spline) is a kind of spline generated by so-called basis functions. The advantage of B-splines over Bezier curves (which are a special case of B-splines) is that the control points of a B-spline affect only their local region of the curve or surface. p 68

Back-propagation: A training technique which adjusts the weights of the hidden and input layers of a neural net to force the correct decision for a given feature vector data input set. p 125

Back-propagation Learning Rule: Learning rule in which weights and biases are adjusted by error-derivative (delta) vectors back-propagated through the network. Back-propagation is commonly applied to feed-forward multilayer networks. Sometimes this rule is called the generalized delta rule. p 126

Background: The part of a scene behind the object to be imaged, the areas shown behind the main subject in a picture. p 88

Backlighting: Placement of a light source behind an object so that a silhouette of that object is formed. It is used where outline information of the object and its features is important rather than surface features. p 76

Banding Noise: Banding noise is highly camera-dependent, and is noise which is introduced by the camera when it reads data from the digital sensor. Banding noise is most visible at high ISO speeds and in the shadows, or when an image has been excessively brightened. Banding noise can also increase for certain white balances, depending on camera model. p 26

Bandpass Filter: An absorbing filter which allows a known range of wavelengths to pass, blocking those of lower or higher frequency. p 136

Bezier Curve: A curve modelled using a parametric polynomial technique. Bezier curves can be defined by many vertices. Each vertex is controlled by two other points that control the endpoint tangent vectors. Bezier curves were developed by Pierre Bezier (1910–1999) for computer modelling in motor vehicle design. They are a special case of B-splines. Unlike a standard B-spline curve, the Bezier does not provide for local control, meaning that changing one control point affects the entire curve. p 68

Bi-Directional Reflectance Function: The relative angular dependence of the reflected radiance from a given region as function of both incident and viewing directions. p 12

Bias: When referring to neural networks, a parameter that is summed with the neuron's weighted inputs and passed through the neuron's transfer function to generate the neuron's output. p 125

Bias Frame or Image: An image generated from several raw CCD frames taken with no light incident upon the detector and of "zero" exposure time. p 83

Breakdown Point: The minimum fraction of points (of a data set) that can arbitrarily corrupt an estimate or regression. Also given in percentage, the worst figure is 0% (as is the case for Least Squares estimators) and the theoretical best is 50% (typical of LMedS estimators). Donoho and Huber [33] define the breakdown point as "the smallest amount of contamination that may cause an estimator to take on arbitrarily large aberrant values. p 111

Brightness: The total amount of light or incident illumination on a scene or object per unit area. Also called intensity. p 76

Brilliance: The intensity of light reflected from a surface. It is sometimes an alternative term for luminosity. p 76

C

C-mount: Short for Cine-mount, a threaded means of mounting a lens to a camera. A C-mount camera has a $17.5\,mm$ distance from the camera sensor to the edge of the threads ("face to sensor") and 32 threads/inch. p 23

Calibration: A measurement or comparison against a standard. Or, the determination of any equipment deviation from a standard source so as to ascertain the proper correction factors. p 58

CCD: A Charge Coupled Device is a photo-sensitive image sensor implemented with large scale integration technology. p 78

Centroid: Points that are, respectively, the center of a given area or midpoint of a given line segment. Can also refer to the center of mass of an object or cluster. p 98

Charge Transfer Efficiency: Pixels are read out one at a time into a register along one side of the CCD chip. The charge collected in each pixel is shifted down the CCD chip toward the register. Some charge is lost during this transfer. The ability of a CCD camera to keep as much charge as possible from being lost during readout is known as its Charge Transfer Efficiency. p 83

Classification: Assignment of image objects to one of two or more possible groups. Decisions are made by evaluating features either 1) structurally based on relationships or 2) statistically. After training the features are weighted based on significance in object identification. For multiple features, absolute values are used. p 123

Clustering: Assigning a class to a measurement, or equivalently, identifying the probabilistic source of a measurement. p 92

Coherent: A light wave whose parts are all in phase with each other. p 17

Complementary Metal-Oxide Semiconductor (CMOS): CMOS is a logic design style using combinations of PMOS and NMOS transistors. Under DC or low frequency conditions, this logic design style allows steady state current to dissipate only through leakage. Relatively large power is dissipated only during transistor switching at mid- to high-frequencies when there are brief periods were both types of devices are on simultaneously. p 23

Computer-Aided Design (CAD): Any of a number of computer-based tools which assist a designer. CAD tools are vital for a number of phases in designing vehicles because they greatly facilitate entering, verifying, tracking, and storing massive amounts of complex information required. p 53

Consistency: The database is transformed from one valid state to another valid state. A transaction is legal only if it obeys user-defined integrity constraints. Illegal transactions aren't allowed and, if an integrity constraint can't be satisfied, the transaction is rolled back. p 52

Contrast: The difference of light intensity between two adjacent regions in the image of an object. Often expressed as the difference between the lightest and darkest portion of an image. Contrast between a flaw or feature and its background is the goal of illumination. p 90

Contrast Enhancement: Stretching of the grey-level values between dark and light portions of an image to improve both visibility and feature detection. p 90

Control Points: Points used in conjunction with spline curves. These points are not part of the curve proper, but the relationship between the control points and the points on the curve is used to define the shape of the curve. p 68

Convolution: Superimposing an $m \times n$ operator (usually a 3×3 or 5×5 mask) over an area of the image, multiplying the points together, summing the results to replace the original pixel with the new value. This operation is often performed on the entire image to enhance edges, features, remove noise, and other filtering operations. p 6

Correlation: A correlation is an index of the strength of the relationship between two variables. Correlation can also refer to the process whereby the similarity of two variables or data sets is estimated. p 17

CSMA/CD: Carrier-Sense-Multiple-Access with Collision Detection is a method used to control access to a shared transmission medium to which a number of stations are connected. A station wishing to transmit a message first senses (listens to) the medium and transmits the message only if the medium is inactive. Then, as the message is being transmitted, the station monitors the actual signal on the transmission line. If this is different from the signal that is being transmitted, a collision is said to have occurred and been detected. The station then ceases transmission and tries again later. p 74

D

Dark Current: Charge detected by the photodetector without illumination. Dark current increases with increasing temperature and typically doubles for every 8 Kelvin. Dark current degrades the detection of very small optical signals since the statistical noise of dark current electrons becomes comparable to the small optical signal. p 87

Dark Frame or Image: An exposure taken with the shutter closed. Typically, the exposure time used is similar to that selected for the object frames in an observing run. Dark frames give an estimate of the background level due to dark current in a CCD. p 87

Dark Signal: Signal generated by the dark current in a specified integration time, at a given temperature. p 83

Debugging: Debugging is the process of locating and fixing or bypassing bugs (errors) in computer program code or the engineering of a hardware device. To debug a program or hardware device is to start with a problem, isolate the source of the problem, and then fix it. p 46

Defect: Flaw or imperfection. Any unintentional and undesirable irregularity in the surface of an object that could affect system performance. Examples of such defects include cracks, inclusions, blistering, dents, pits, stringers and scratches. p 2

Degrees of Freedom: Degrees of freedom is an indication of the complexity of a system or model. p 100

Derating: Term for any kind of reduction influences on the rated parameters of a device. The derating value may be a constant multiplied with the parameter. In other cases additional influencing factors have to be considered, such as high surrounding temperature and the presence of radiation. p 27

Diffuse Reflection: Light which bounces off an object surface in many different directions. Light radiated from a matte surface is highly diffused. Reflection from a rough surface, in which a single ray of light is divided up into many weaker reflected rays going in many directions. p 14

Diffused Lighting: Scattered soft lighting from a wide variety of angles used to eliminate shadows and specular glints from profiled, highly reflective surfaces. p 14

Dilation: A morphological operation which moves a probe or structuring element of a particular shape over the image, pixel by pixel. When an object boundary is contacted by the probe, a pixel is preserved in the output image. The effect is to "grow" the objects. p 91

DMA: An abbreviation for Direct Memory Access, a technique for transferring data from main memory to a device without passing through the CPU. Computers that have DMA channels can transfer data to and from devices much more quickly than computers without a DMA channel can. p 80

Durability: Once committed (completed), the results of a transaction are permanent and survive future system and media failures. p 52

E

Erosion: The converse of the morphology dilation operator. A morphological operation which moves a probe or structuring element of a particular shape over the image, pixel by pixel. When the probe fits inside an object boundary, a pixel is preserved in the output image. The effect is to "shrink or erode" objects as they appear in the output image. Any shape smaller than the probe (*i.e.* noise) disappears. p 91

Ethernet: Ethernet is a standard for connecting computers into a local area network (LAN). The most common form of Ethernet is called 100BaseT, which denotes a peak transmission speed of $100\,mbps$ using copper twisted-pair cable. p 74

Exposure Time: Image sensors integrate charge over the so called exposure or integration time. This time defines the response of the sensor to the incident radiation. The shorter the integration time, the smaller the signal the sensor produces at a given radiation intensity. p 29

F

Fast Fourier Transform: Produces a new image or data vector which represents the frequency domain content of the spatial or time domain image or data vector information. Data is represented as a series of sinusoidal waves. p 124

Feature Extraction: Determining image features by applying feature detectors to distinguish or segment them from the background. p 123

Feature Vectors: A set of features of an object (such as area, number of holes, *etc.*) that can be used for its identification or inspection. p 123

Features: Simple image data attributes such as pixel amplitudes, edge point locations and textural descriptors, center of mass, number of holes in an object with distinctive characteristics defined by boundaries or regions. p 123

Fiducial: A feature element such as a line, mark or shape used as a standard of reference for measurement or location. p 69

Field of View (FOV): The two dimensional area which can be seen through the optical imaging system. In the case of a zoom optical system, you have a varying field of view. At the highest magnification, you have a smaller field of view. At the lowest magnification, you have the largest field of view. p 76

Filter: A device or process that selectively transmits frequencies. In optics, the material either reflects or absorbs certain wavelengths of light, while passing others. Filters can also be implemented algorithmically in either time or frequency domain. p 136

Firmware: Software hard coded in non-volatile memory (ROM), usually to increase speed. p 23

Flat Field Frame or Image: A flat field is one illuminated with some uniform source. Used to determine the relative sensitivity of the elements (pixels) in a system. p 84

Flat Fielding: Flat fielding is the process of dividing by a normalised flat-field to remove the sensitivity variations of a system. p 87

Focal Length: The distance from a lens' principal point to the corresponding focal point on the object. The distance, usually given in millimetres, between the optical center of a lens and the point at which rays of light from objects at infinity are brought to focus. In general, the greater the focal length, the smaller and more magnified the part of the scene it includes in the picture frame. p 76

Focal Plane: Usually found at the image sensor, it is a plane perpendicular to the lens axis at the point of focus. p 76

Focus: The point at which rays of light converge for any given point on the object in the image. Also called the focal point. p 76

Frame: The total area scanned or captured in an image sensor. p 29

Frame Buffer: Image memory in a frame grabber. p 29

Frame Grabber: A device that interfaces with a camera and, on command, samples the image data stream, converts the sample to a digital value and stores that in a computer's memory (frame buffer). p 29

Full Well Capacity: The amount of light that can be detected before a pixel saturates at full white. This is typically measured in electrons and has implications for dynamic range, temporal noise and sensitivity. Higher full well typically gives greater dynamic range and better noise performance at the expense of sensitivity. p 83

G

Global Minimum: Lowest value of a function over the entire range of its input parameters. p 125

Gradient Descent: A technique which finds a local minimum by moving over the function surface along the direction of steepest descent, as determined by the gradient of the function at that point. p 126

GUI: An acronym for Graphical User Interface. Pronounced "gooie." A Windows based user interface screen or series of screens allowing the user to point-and-click to select icons rather than typing commands. p 51

H

HDL: Hardware Description Languages (HDLs) are used by computer and electrical engineers to model, design, simulate, and synthesize hardware systems. The two most widely accepted HDLs are VHDL and Verilog. p 73

Hidden Layer: Layer of a neural network that is not connected to the network output (for instance, the first layer of a two-layer feed-forward network). p 125

Histogram: A graphical representation of the frequency of occurrence of each intensity or range of intensities (grey-levels) of pixels in an image. The height represents the number of observations occurring in each interval. The middle of the histogram indicates the average grey value whilst the width of the histogram indicates the contrast range. p 88

Hot Pixels: Hot pixels are pixels which exhibit an excessively high dark current but otherwise behave as normal. These pixels appears as bight spots in the image and the effect is worse at longer integration times and hight temperatures. p 85

I

IGES: Acronym for Initial Graphics Exchange Specification, an indirect data exchange standard used largely in the United States to exchange CAD information between disparate systems. Originally developed by the U.S. Government and major defence contractors, it is used mainly on workstation, minicomputer, and mainframe-based CAD systems. p 132

Illumination: Normally a wavelength or range of wavelengths of light or visible light used to enhance a scene so the detector, normally a camera, can produce an image. p 14

Image: Projection of an object or scene onto a plane (*i.e.* screen or image sensor). p 76

Input Layer: Layer of neurons receiving inputs directly from outside the network. p 125

Intensity: The relative brightness of a portion of the image or illumination source. p 76

Interbus: Interbus is a high speed industrial networking technology developed by Phoenix Contact in 1984. Interbus makes use of a high speed shift register technique to propagate a number of registers to all nodes in the network (upto 256 nodes). The technology supports a distance of $400\,m$ per segment, with a total distance of $12.8\,km$. The transmission speed is $500\,Kbits/sec.$ p 72

ISO Speed: A camera's ISO setting or ISO speed is a standard which describes its absolute sensitivity to light. ISO settings are usually listed as factors of 2, such as ISO 50, ISO 100 and ISO 200, and can have a wide range of values. Higher numbers represent greater sensitivity and the ratio of two ISO numbers represents their relative sensitivity, meaning a photo at ISO 200 will take half as long to reach the same level of exposure as one taken at ISO 100 (all other settings being equal). ISO speed is analogous to ASA speed for different films, however a single digital camera can capture images at several different ISO speeds. This is accomplished by amplifying the image signal in the camera, however this also amplifies noise and so higher ISO speeds will produce progressively more noise. p 26

Isolation: The results of a transaction are invisible to other transactions until the transaction is complete. p 52

Isotropy: Isotropy (the opposite of anisotropy) is the property of being independent of direction. In the field of computer graphics, often used to indicate that the nature of the diffusely reflected light is independent of the viewing direction. p 14

K

Kernel Filters: Kernel filters are image processing algorithms applied to an image, generally to smooth or sharpen an image. A kernel is a small grid which controls the change in value of a certain pixel based on the values of neighbouring pixels. The most common types of kernel filters are low-pass ("smoothing" and "blurring" filters) and high-pass ("sharpening" filters). p 81

L

Learning : Process by which the weights of the neural network are adjusted to achieve some desired network behaviour. Also referred to as training. p 125

Learning Rate : Training parameter that controls the size of neural network weight changes during learning / training. p 125

Lens: A transparent piece of material, usually glass or plastic, with curved surfaces which either converge or diverge light rays. Often used in groups for light control and focusing. An optical device made of glass or other transparent material that forms images by bending and focusing rays of light. A lens made of a single piece of glass cannot produce very sharp or exact images, hence camera lenses are made up of a number of glass "elements" that cancel out each others' weakness and work together to give a sharp true image. The size, curvature and positioning of the elements determine the focal length and angle of view of a lens. p 76

Local Minimum: Minimum of a function over a limited range of input values. A local minimum might not be the global minimum. p 125

M

Magnification: The relationship between the length of a line or size of a feature in the object plane with the length or size of the same feature in the image plane. The ratio of the image size to the object size. p 77

Mean Squared Error: A metric used to compute, amongst other, the difference between the output of a system or process and the desired output value of the system or process given a specific input. p 126

Momentum: Technique often used to make it less likely for a back-propagation network to get caught in a shallow minimum. p 126

Momentum Constant: Training parameter that controls how much momentum is used. p 126

Morphology: Image algebra group of mathematical operations based on manipulation and recognition of shapes. Derived from mathematical morphology. Operations may be performed on either binary or grey scale images. p 91

N

Neural Network: A computing paradigm which processes information based on biological neural systems. Decisions are made based on weighted features analysed by interconnected nodes of simple processing elements using analog computer-like techniques. A configurable mapping between an input space and an output space. These networks represent arbitrary mappings by suitable adjustment of their weights (configurable parameters). p 124

Noise: Irrelevant or meaningless data resulting from various causes unrelated to the source. Random, undesired information signals. p 23

Noise: Unwanted disturbances of the signal level containing no useful information. Noise can obscure small signals which are then said to be beneath the "noise floor". p 79

NURBS: Acronym for Non-Uniform Rational B-spline, a type of free-form curve that uses rational B-splines and allows for a weighting value at each point on the surface. p 68

O

Objective Function: The function that is optimised during an optimisation process, to compute either the set of parameters yielding the minimum or the maximum function value. p 126

OOP: Object Oriented Programming (OOP) is a computer programming paradigm that has proven extremely powerful in creating complex computer systems. The concepts of Objects, Abstraction, Encapsulation, Polymorphism, and Inheritance are central to OOP. p 38

Optimisation: The process of improving the speed at which a program executes. Depending on context it may refer to the human process of making improvements at the source level or to a compiler's efforts to re-arrange code at the assembly level (or other low level). p 49

Orientation: The angle or degree of difference between the object coordinate system major axis relative to a reference axis as defined in a 3D measurement space. p 14

Output (Readout) Amplifier: A mechanism in the CCD that amplifies the electrons in the output node sufficiently to get the signal to the ADC. The output amplifier is the primary source of readout noise p 83

Over-fitting: Over-fitting occurs when the complexity of the statistical model (degrees of freedom) exceed the true (ignoring additive noise) complexity of sample data. A model with over-fitting is typically not reproducible (with new sample data from the same source) and does not predict future responses satisfactory (*i.e.* poor generalisation). p 100

P

Parametric Curves: A term used to classify curves for which the path is described by a mathematical function rather than a set of coordinates. A parameter within the function (often specified as u or v) is varied from 0 to 1 to define all the coordinate points along the curve. Note that the parameter or parameters of the curve do not correspond to an axis in the coordinate system. p 68

Photo-Response Non-Uniformity: Variation in responsivity between pixels. This a spacially correlated noise source similar to fixed pattern noise (FPN), which can be eliminated by gain-offset correction. p 83

Photoelectric Effect: Photons of sufficient energy can, upon absorption, cause the ejection of electrons from the atoms of metals and semiconductors. These so called photo-electrons can be harnessed in a photodiode to create a light to charge converter. p 78

Photometry: Photometry is the science of measuring visible light in units that are weighted by the sensitivity of the human eye. Typical photometric units include lumens, lux and candelas. p 13

Photon: The light quantum. The smallest possible "packet" of light energy at a given wavelength. p 76

Photosite: Photosites are the individual light-sensitive squares (or rectangles) of a CCD or CMOS sensor. In a sensor image, each photosite corresponds with an individual pixel in the final image. Photosites are almost always called pixels. p 78

Pixel: An acronym for "picture element". The smallest distinguishable and resolvable area in an image. The discrete location of an individual photo-sensor in a solid state camera. p 23

Profiler: A computer program that measures the performance of another program. Typically the measurement takes the form of an interval sampling of the location of the program counter or the stack. After the program being measured is finished running, the profiler collects the statistics and generates a report. Proper use of the profiler may require that the program being measured be recompiled using a special option, and/or linked with a special library. gprof and prof are two widely available profilers for Unix programs. p 49

Programmable Logic Controller (PLC): An embedded controller that allows the automated control of processes via a high-level programming environment. The control and sensing is typically

in the form of analog and digital control lines. Newer versions can also control more advanced peripherals via a communication channel such as Interbus. p 30

Q

Quantum Efficiency: The ratio of the number of photoelectrons produced to the number of photons incident upon a detector. QE is expressed as a percentage of the number of photons converted. If all the photons produced electrons, the QE would be 100%. Most amateur CCD cameras have QEs in the range of 25 – 50%. Specially cooled professional CCDs used at astronomical observatories have QEs close to 98%. p 83

R

Random Noise: Random noise is characterized by intensity and colour fluctuations above and below the actual image intensity. There will always be some random noise at any exposure length and it is most influenced by ISO speed. The pattern of random noise changes, even if the exposure settings are identical. p 26

Readout Noise: The readout noise of a sensor is the output signal measured when no input signal (light) is present. p 83

Reflection: The process by which incident light leaves the surface from the same side as it is illuminated. p 14

Refraction: Refraction is the bending of light due to a change in its velocity as it passes the boundary between two materials. The change in velocity is caused by the differences in density between the two substances. p 22

Region of Interest: A defined rectangular region in the image matrix which can be read out separately from the rest of the matrix. By reading out only this region, less data needs to be transferred and so frame rates can be dramatically increased. p 26

Regression: Regression is a method for studying the relationship between a response variable y and a covariate x. The covariate is also called a predictor variable or a feature. The term "regression" is due to Sir Francis Galton (1822–1911). p 101

Repeatability: The ability of a system to reproduce or duplicate the same measurement. The total range of variation of a dimension is called the 6-sigma repeatability. p 30

Residual: A residual is the deviation of an observation (sample) from an expected value, *i.e.* a particular observation or measurement error. p 101

S

Shutter: An electrical or mechanical device used to control the amount of time the imaging surface is exposed to light. Often used to stop blur from moving objects. p 23

Silhouette: A black and white image of an object illuminated by backlighting. p 76

Skewness: Skewness is defined as asymmetry in the distribution of the sample data values. Values on one side of the distribution tend to be further from the "middle" than values on the other side. In the case of images, the skewness provides a measure of the 2D asymmetry in the image. p 168

Specular Reflection: Light rays that are highly redirected at, or near the same angle of incidence to a surface. Observation at this angle allows the viewer to "see" the light source. p 14

Spherical Aberration: The variation in focal length of a lens from center to edge due to its spherical shape – generally all parts of the image, including its center. The effects of lens aberration usually increase with increases in aperture or in angle of field. p 77

Spline: A type of curve that is interpolated between two endpoints and two or more tangent vectors. The term dates from 1756, and derives from a thin wood or metal strip used for drafting curves in architecture and ship design. p 68

State variables: The state of a dynamic system is the smallest set of variables such that the knowledge of these variables at $t = t_0$, together with the knowledge of the inputs for $t \geq t_0$, completely determines the behaviour of the system for any time $t \geq t_0$. p 117

T

Tangent: A condition in which a straight line is in contact with a curve at only one point. p 68

Thresholding: The process of converting a grey-scale image into a binary image. If the pixel's value is above the threshold, it is converted to white. If below the threshold, the pixel value is converted to black. p 92

V

Verilog: Verilog is a HDL created by Gateway Design Automation, 1984 *c.*. Gateway was eventually acquired by Cadence who opened the language in 1991. The syntax of Verilog is similar to that of the C programming language, and is more loosely typed than VHDL. Verilog is now defined as IEEE standards 1364-1995 and 1364-2001. SystemVerilog is an extension of Verilog IEEE 1364-2001. p 73

VHDL: VHDL is a Hardware Description Language (HDL) that was developed in the US, 1980 *c.*. The language is used to model, design, simulate, and synthesize ICs and Soft IP. The language is defined by IEEE standard 1076 and its syntax is derived from the ADA software language. p 73

Vignetting: Vignetting is a darkening of the corners of an image, usually caused by light falling off toward the edge of the CCD chip due to the optical system. Vignetting is usually removed by using a flat field image through a process known as flat fielding. p 87

Viscosity: Viscosity is an internal property of a fluid that offers resistance to flow. p 34

Visible Light: The region of the electromagnetic spectrum in which the human retina is sensitive, ranging from about 380 to $780\,nm$ in wavelength. p 22

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z