UNIVERSITY OF OSLO
Department of Informatics

Augmented Reality for Safer Coronary Artery Bypass

Cand Scient Thesis

Bjørn Daniel Bengtsson

30th July 2003

Preface

This work has involved influences from many directions and persons. It has been interesting, inspiring, fun, sometimes frustrating but above all a learning adventure where no courses could have taken me.

This document has advantages by using the electronic version as every reference to figures and chapters etc. are using hyperlinks. Also most of the bibliography has links to electronic versions of the reports.

Acknowledgments

My thanks go to...

Knut Mørken, my internal advisor at the Institute for Informatics at the university of Oslo for recommending Lars Aurdal and the Interventionalcentre at the Norwegian National Hospital in Oslo as the place for my work with the thesis.

All the people and students at the Interventionalcentre for a great place to work with this thesis. Special thanks to Ole Jakob Elle, Ilangko Balasingham and Eigil Samset for always being helpful.

Many people on the OpenCV mailing list for good discussions about image processing and the OpenCV library.

My parents for their support.

All music I listened to.

And finally a big thanks to my external advisor Lars Aurdal at the Interventional Centre for discussions and all help. Especially for the push to hold presentations at both NOBIM's industrial seminar in Tønsberg October 2002 (Norwegian Society for Image Processing and Pattern Recognition) and at CARS (Computer Assisted Radiology and Surgery) International Congress June 2003 in London.

Abstract

The standard procedure for left anterior descending (LAD) coronary artery bypass requires a sternum split. The internal mammary artery (IMA), most typically the left one (LIMA), is dissected free from behind the sternum and is then anastomosed onto the heart below the occlusion of the LAD. Totally Endoscopic Coronary Artery Bypass (TECAB) is a less invasive alternative to the standard procedure. This procedure results in heavily reduced invasiveness, but also leads to loss of precision, reduced force feedback and loss of overview. The purpose of this work is to alleviate the problems related to loss of overview by generating an augmented reality for the surgeons in which they are given the impression of ’seeing through’ the tissues surrounding the LIMA thus making localization of the LIMA a simple matter. Using preoperative CT or MR data, the LIMA is located with respect to the tissues surrounding it. Intra-operatively tissues surrounding the LIMA are tracked using a stereo endoscope held by a robotic arm (AESOP). Knowing the position of the endoscope relative to the visible surface inside the chest cavity now makes it possible to calculate where in the endoscopic video stream the LIMA is located. Positioning of the endoscope is also possible using an optical tracking system. This information is then used to generate an augmented reality where the LIMA is superimposed on it’s correct position in the video stream. The thesis is using artificial models to test and experiment on the described problem.

Contents

Chapter 1

Introduction

Three fields in computer science provides the main foundation for this thesis. These fields are briefly introduced in this chapter.

1.1 Visualization

Visualization is a large field in computer science that deals with the problems of making data perceivable and understandable with the help of vision, for example using a standard monitor to display the data. Visualization has acquired more and more attention as 3D visualization has become more available due to faster hardware and a greater selection of software.

1.2 Image processing

The Merriam-Webster dictionary explains an *image* as…“*a : the optical counterpart of an object produced by an optical device (as a lens or mirror) or an electronic device b : a likeness of an object produced on a photographic material.*” An image produced by this *device* contains an equal or mostly less informative description of the object or data we want to study. The image can contain lots of useful information but also lots of redundant information that we are not interested in. The art of image processing tries to enhance the interesting information while suppressing other info. It should be noted that whatever we do with an image we can never increase the amount of information in it, the only thing we can do is to try to extract the specific information that we are interested in. Image processing does not imply that we directly use visualization to view the processed image. It can just as well be input to a computer program that makes some calculations on our processed image.

1.3 Computer vision

Computer vision is a field tightly bound to image processing. Thinking that computer vision and image processing is the same is wrong, but image processing is an important part of computer vision. Computer vision deals with seeing, to use information mediated by light or to use information from sensors that in some way examine their surroundings in order to successfully interact with the environment. Mr David Young at the University of Sussex gives some examples of problems that computer vision deals with…

- ⋆ How do people and animals see ?
- ⋆ Robotic vision
- ⋆ What computational structures underly vision ?
- ⋆ How does the environment make vision possible ?
- ⋆ How do we reconstruct a third dimension from 2D images ?
- ⋆ How can we build machines to solve specific tasks involving vision ?

Figure 1.1 is an example of computer vision used to navigate a robot. This robot uses a system with two sonars to get information of distance and position of possible obstacles in front of the robot to make it able to navigate around objects. A stereo camera is another approach commonly used to navigate robots.

This is the essence of computer vision, a computer program connected to a device that captures radiation yielding an image, and by using image processing techniques on this image, it decides what to do next.

Computer vision can be used in a huge number of applications, such as in the industry where for example computer vision can be used to decide whether the finished products are defect free or not. Until recently humans have performed these checks, but humans get tired, they take coffee breaks when they are not supposed to and so on. Computers do not suffer from these primary needs, and can keep on checking as long as they have electricity, and as long as the programmer has made a decent job in handling all sorts of

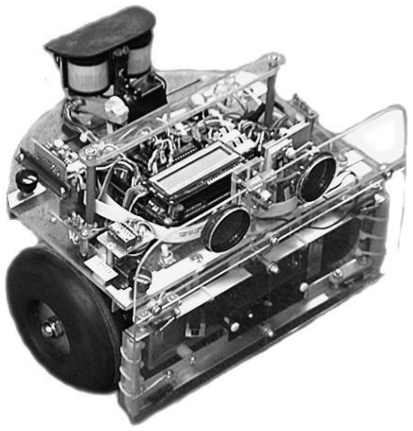


Figure 1.1: This robot uses two sonars to extract information about it's surrounding environment. Permission kindly given by Mr David P. Anderson, Southern Methodist University. `http://www.geology.smu.edu/~dpa-www/myrobots.html`

input, thus avoiding application crashes or stalls. Other examples are recognizing faces and take action based on which face that is seen. Imagis `http://www.imagistechnologies.com/` in Vancouver has a facial recognition system that is used in many environments such as casinos and airports. The company Vision IQ `http://www.vision-iq.com` has implemented a computer vision system that monitors swimming pools and warns the lifeguards of possible accidents. Dipix technologies `http://www.dipix.com/` has a system for the baking industry that monitors bake color, shape, size and other properties that bread or tortillas may have. The number of possible applications is huge.

1.4 Chapter Organization

This thesis mainly rests on the foundations of techniques from the above three mentioned fields in addition to mathematics. The thesis is divided in the following chapters.

Chapter 2 and 3 Describes the background and motivation of the thesis. First clinical and then theoretical.

Chapter 4 Notes some of the work that has been done earlier by others in relevant areas.

Chapter 5 Gives a brief descriptive and schematic overview of the project.

Chapter 6 Describes the models that have been used.

Chapter 7 Describes all theory and algorithms considered and used in the implementation and testing. This is the main part of this thesis.

Chapter 8 Describes the implemented application and its graphical user interface.

Chapter 9 Report of the results and result data.

Chapter 10 Further discussion about the results.

Chapter 11 Conclusions we can draw from the results.

Chapter 12 What can be improved and inspirations for extensions and future work.

Appendix A Report concerning this thesis that was accepted to CARS 2003 in London (Computer Assisted Radiology and Surgery)

Chapter 2

Clinical Background and Motivation

Medical imaging is an interesting combination of mathematics, medicine, and computer science. The motivation for this thesis and work starts with a medical issue occurring during heart surgery.

2.1 Heart surgery

Heart diseases are a major cause of death, especially in the developed countries. In Norway the number of deaths related to diseases in the circulatory system has been between 41% and 46% of the total number of deaths per year during the period 1991-2000 according to *Statistics Norway*, [SSB]. Of these circulatory diseases a large amount is due to ischemic[1] heart diseases, in 2000 they answered for 45% of the deaths related to circulatory diseases.

One common ischemic heart disease is arteriosclerosis where one or more of the coronary arteries on the heart are stenosed[2]. The heart with it's coronary arteries is illustrated in figure 2.1.

[1]localized tissue anemia due to obstruction of the inflow of arterial blood
[2]a narrowing or constriction of the diameter of a bodily passage or orifice

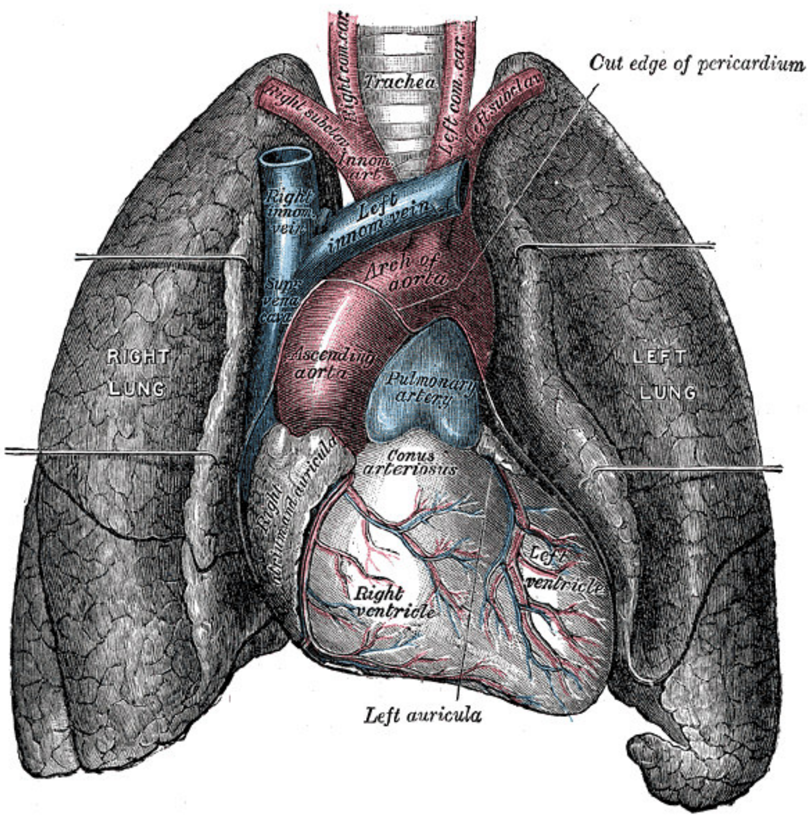


Figure 2.1: View of the human heart and lungs [GRAY-00]. The vessels that can be seen on the front of the heart are the coronary arteries of which one on the front left is LAD which is a commonly stenosed coronary artery.

The cause of the occlusion is build up of cholesterol deposits on the walls of the arteries. These deposits limit the flow of blood which in time often will result in a heart attack [MULLANY-03]. Commonly, the stenosed coronary artery is the Left Anterior Descending Artery (LAD) or *The widow maker* as it was named in the Mayo proceedings [HOLMES-00]. In cases where life style and dietary changes have no effect, surgery may be the only option left.

The operating procedures to overcome the problem are called Coronary Artery Bypass Surgery (CABG) procedures and are performed by making bypasses for the blood to get pass the occlusions. In most cases at least one of the bypasses is constructed from one of the Internal Mammary Arteries (IMA) that is located behind the sternum[3]. There are two internal mammary arteries in the human body; the left and the right, referred to as LIMA and RIMA respectively. LIMA is the most commonly used as it's located closer to the heart. A short description of

[3]The breastbone. The sternum articulates with the ribs 1 through 7 on either side of the chest

different CABG procedures follows. . .

- ⋆ *Traditional Coronary Artery Bypass Grafting (***CABG***)*
  During traditional CABG the heart is stopped and connected to a heart lung machine which does the work of both heart and lungs. There is also a possibility for off pump CABG (OPCAB) for suitable patients. Both these procedures requires a sternotomy, i.etc. a very invasive procedure which is a surgical opening through the breast bone [MULLANY-03]. This procedure can be seen in figure 2.2.

- ⋆ *Port Access Coronary Artery Bypass (***PORTCAB***) Grafting*
  This procedure eliminates the need of opening the chest required by traditional open heart surgery. The sternal incision is exchanged for three or four very small incisions between the ribs. In these small incisions which are called ports, instruments are inserted along with a camera which allows the surgeon to see inside the patients chest. The heart is stopped and connected to a heart lung machine. As the incisions are smaller, this is a less invasive procedure compared to traditional CABG and the recovery time is reduced in addition to less scarring [MORGAN-00]. Though the precision is reduced and this has been the driving force to develop surgical robots, which reduces the tremor in the instruments as well as downscaling movements [BENGTSSON-03].

- ⋆ *Minimally Invasive Direct Coronary Artery Bypass (***MIDCAB***) Grafting*
  The MIDCAB is an alternative to traditional CABG that is less invasive. Instead of a sternotomy a mini-thoracotomy is performed which is a smaller incision of the chest wall. MIDCAB is performed off pump. Reduced postoperative pain and an improvement in early mobility and functional recovery compared to traditional CABG is evaluated in [DIEGELER-99] and an overview of the procedure is given in [NATAF-97].

- ⋆ *Totally Endoscopic Coronary Artery Bypass (***TECAB***) Grafting*
  TECAB is a procedure that combines the favorable aspects of MIDCAB and PORTCAB; thus using the port access from PORTCAB while operating on a beating heart from the MIDCAB. The TECAB is a robot assisted procedure where the instruments are guided by robot arms. [MORGAN-00]

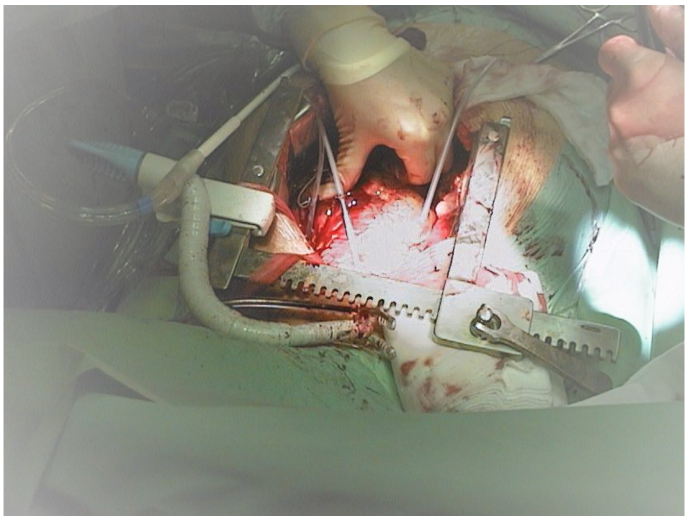


Figure 2.2: The standard CABG procedure uses a sternum split which is an opening through the breastbone to localize and dissect the Left Internal Mammary Artery (LIMA). As can be seen in this image this is a very invasive procedure.

2.2 Surgical technique of TECAB

In this thesis the focus is on the robot assisted TECAB procedure due to it's technical and minimally invasive benefits. This section describes the surgical technique in brief, for more details refer to [KAPPERT-01].

Kappert et al. has used the Da Vinci surgical system (while Rikshospitalet Oslo, Norway uses the similar ZEUS^{TM} robotic system). An article describing the surgical technique using ZEUS^{TM} is [KIAII-00]. The beating heart operating procedure uses four 1 cm chest incisions and consists mainly of two steps. The first step is to harvest the LIMA/RIMA. The second step to suture the coronary anastomoses. To minimize the movements of the chest single lung ventilation is used, which means that the left lung is collapsed during the surgery. Three of the 1 cm incisions are placed in a triangle between the third and sixth intercostal space. Humidified and warmed CO_2 is insufflated through the central port in which the endoscope thereafter is introduced. The first part of the surgery consists of exposing the LIMA from surrounding tissues.

The first third of the LIMA is quite easily located only covered by thin parietal pleura[4]. The middle third is sometimes hidden in fat and hard to identify. The distal third of the LIMA is intramuscular and therefore not visible. This can be seen in figure 2.3. At the level of the fourth and fifth intercostal spaces the LIMA

[4]the thin membrane around the lungs

is also hidden by the cardiac mass, [NATAF-96].

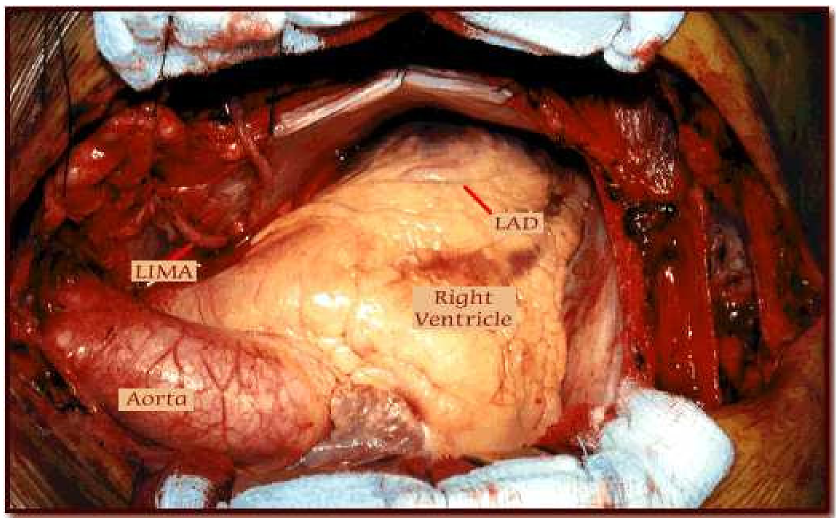


Figure 2.3: Photographic view of LIMA, LAD and heart illustrating the difficulty to locate the LIMA as most of it is hidden. The image is from an open chest surgery and therefore the overview here is much better than in the minimal invasive techniques such as the TECAB. In the TECAB procedure the view comes from the endoscope that is inserted in one of the ports. Permission kindly given by Dr. Mark Levinson and Mr Jon Golden. Image © 2003 Forum Multimedia Publishing, LLC. All rights reserved.

Locating hidden arteries beneath superficial tissue can be a difficult task in minimal invasive surgery [BEASLEY-02], and it's vital that the IMA is not injured due to failure in the localization. The localization is particularly difficult as the surgeon loses the force feedback due to the introduction of the robot. During open surgery he is able to feel where the artery is located with his fingers. The surgeon can also look for pulsations inside the parietal pleura to locate the artery, but occasionally this is not observed. In these cases gentle compression of the soft tissues can augment arterial pulsations, [KIAII-00]. The surgeons overview of the chest cavity is also greatly reduced as he only has the view from the endoscope. An illustration of the procedure is shown in figure 2.4.

2.3 Enhancements of the technique

If we review the considerations and benefits with the TECAB versus the standard procedure we have…

Benefits

⋆ Less invasive due to smaller incisions

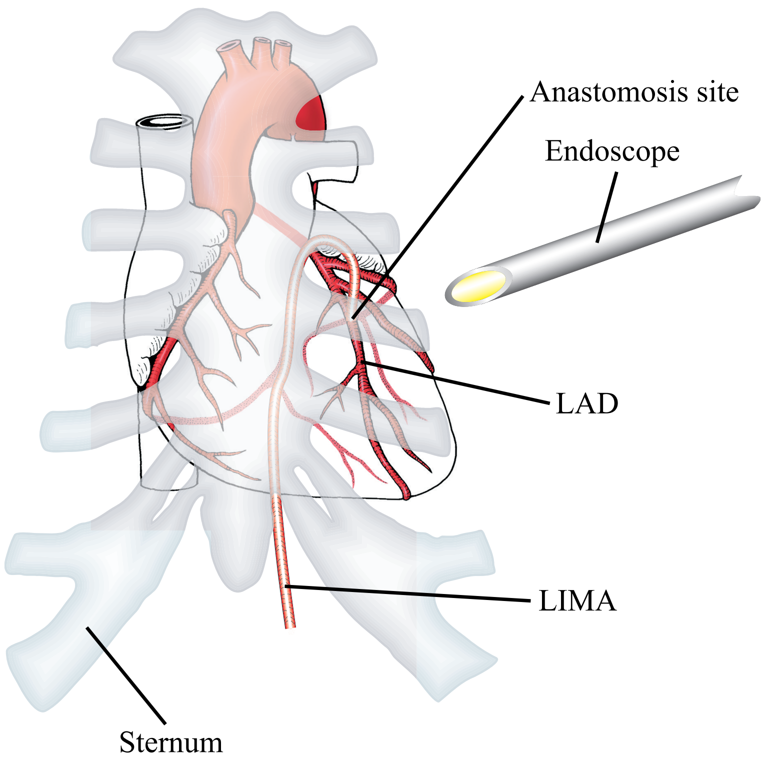


Figure 2.4: This illustration shows the heart behind the sternum and the endoscope used in the TECAB procedure. The LIMA is used as a bypass and is anastomosed to the LAD below the occlusion.

- ⋆ Reduced postoperative pain
- ⋆ Shorter recovery time and hospital stay
- ⋆ Reduced scarring and tissue damage

Considerations

- ⋆ Loss of force feedback
- ⋆ Reduced overview
- ⋆ Resulting loss of precision

As the force feedback is lost and the overview is reduced a severe loss of precision results.

Due to this difficulty in locating the LIMA (as also described in section 2.2), methods to visualize the hidden LIMA are wanted.

Localization of the LIMA relative it's surrounding tissues can be made with 3D angiography[5], Magnetic Resonance (MR) or Computerized Tomography (CT). In the rest of this thesis CT is assumed as the source. If the data obtained from the CT could be combined with the video stream from the endoscope in such a way that the LIMA could be seen inside the fat and muscles, it would be of great help for the surgeon. This augmented reality would constantly provide the surgeon with the position of the LIMA even though covered by muscles, fat, instruments and blood that also will obstruct the view as the surgeon starts to cut into the tissues.

The problem that must be solved is to find the position and orientation of the endoscope relative to the fat and muscular surface inside the chest cavity. If this correspondence is established we can project the LIMA from the postoperative CT. This assumes that the LIMA has not moved significantly relative to specific parts in the chest cavity. The addition of intraoperative imaging and localization of the LIMA relative the patient could be an extension. Two approaches are examined in this thesis, the first one exploits the stereo capability of a stereoscopic endoscope (stereoscope) to extract surface information in section 7.4. The second approach uses an optical tracker device to find the relative position and orientation of the scope in section 7.5. A combination of the techniques are discussed in chapter 7.6. Other approaches are mentioned in chapter 4.

[5]Angiography is a roentgenographic examination of blood vessels after injection of a radiopaque contrast medium

Chapter 3

Image processing background

Many aspects of image processing are in use in the thesis but the main techniques discussed are Stereo Algorithms and Image Registration. One specific part of the Image Registration called Surface Registration is of special interest in this thesis. A brief overview of these areas is given in this chapter.

3.1 Stereo Algorithms

The endoscope is stereoscopic and section 7.4 tries to take advantage of this. Stereo algorithms tries to extract depth information from a scene observed from two cameras. Two main approaches exists which is generation of dense or sparse disparity maps. Dense disparity maps have a disparity value for every pixel in one of the images while sparse disparity maps only has disparity values in certain points or areas in the image such as on edges found using a edge detector. A disparity value of d is defined as how many pixels two corresponding points differ along a scanline between the left and right frame which is further explained in section 7.4. As the stereo algorithms can be used to extract distance information about the scene the camera observes, it could be used to establish the correspondence between the camera and the observed objects.

3.2 Image Registration

The info extracted from the stereo algorithms must somehow be registered against the CT scan to establish the correspondence between the endoscope and the chest cavity.

There are three general methods for registration of images from different sources. Landmark-, surface-, and intensity-matching. In landmark matching you use corresponding landmarks in the image sources for the registration. Surface matching uses an algorithm that matches different images or data of the same surface. You are here trying to translate, rotate, and scale one of the images to match the other. Intensity matching uses mutual intensity information to register the images. The matched intensities may come from different Magnetic Resonance Imaging (MRI) scans or from different modalities such as MRI and Positron Emission Tomography (PET).

Surface matchings or Surface Registrations main motivation is to match an object observed by a range sensor (in our case a stereo camera) against an object or a library of model objects. The goal is, as described above to try to translate, rotate, and scale the model object so it fits as best as possible. Surface registration is not only used for deciding the orientation of an object, but also for object recognition where you select the best matching object among several. The matching can be global or partial, where in the global case you try to match a whole object, or organ in medicine. Partial surface registration matches only partial information against the model object. There are many methods available for surface registration. Heuristic surface registration searches a transformation that matches similarities, which can for example be points of maximum curvature. Silhouettes, or projections of 3D objects can also be examined to match surfaces. Another method uses surface normal histograms for matching convex shapes. Quaternions can also be used, for brief descriptions of more surface registration methods and references, refer to [BAREQUET-97]. Surface registration methods and the one used in this thesis are discussed in section 7.4.5.

Chapter 4

Previous work

This chapter does not explain any subjects mentioned but gives a short overview over relevant and introducing articles in these areas. Refer to later chapters, especially Chapter 7 for more in depth explanations.

Stereo images are not a new invention that came with the computers, but that came with photography in the nineteenth century. But the fascination for the concept of stereo vision can be traced back as far as around 300 B.C. when Euclid explained the principle of binocular vision. There also existed stereo drawings and paintings before the photography was invented.

A device called a steroscope was invented 1849 in which you installed two frames with images taken a small distance apart. Using it you see one image with each eye which gives a feeling of depth [SOMMERER-99]. A variant of this is the Viewmaster® in figure 4.2 that many of us has played with.

One early stereo pair are the photographs in figure 4.1. With the introduction of

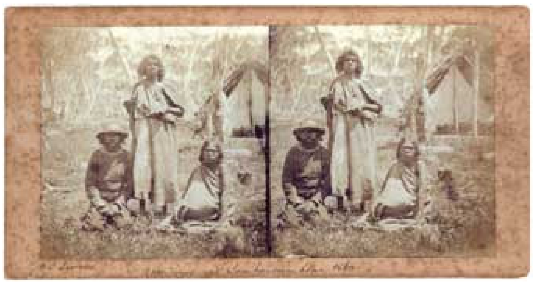


Figure 4.1: Stereo pair images from around the middle of the nineteenths century by the photographer William Stanley Jevons. © Macleay Museum, Sydney. Permission kindly given by Mr Geoff Barker.

Figure 4.2: The viewmaster® was invented in the 1930's, at that time considered a home entertainment system. It still exists though now mostly referred to as a toy.

computers and digital images there grew an interest of extracting information of depth from stereo images or more images. This has showed not to be as easy as for our brain which does beautiful depth extraction in most scenarios. Therefore stereo algorithms are often aimed at one specific use. Stereo vision has been and is still one of the most active research areas in computer vision. A survey and comparison between existing stereo algorithms can be found in [SCHARSTEIN-02].

Stereo algorithms that has been tried can be found in [ROJAS-97], [KOSCHAN-96] and [BIRCHFIELD-98].

Another problem discussed in this thesis is that of matching a partial surface against a 3D model. This is a problem not uncommon in medical situations. A short overview over surface based image registration methods can be found in [FITZPATRICK-00] which also discusses the image registration techniques in general. Methods doing surface registration can be found in [BAREQUET-97], [BESL-92], [ZHANG-92], [AUDETTE-00] and [JOHNSON-97]. This is discussed in section 7.4.5 of this thesis.

Camera calibration is also a field that has been looked into by many people, the problem here has been to develop an easy technique with few input parameters. The technique that has been used in the calibration tool in this thesis is based on the article [ZHANG-00] which do not need any other input than some images of a pattern in different angles. An excellent historical overview of the calibration field is in [CLARKE-98].

Other works that especially address the problem of locating the internal mammary artery during minimally invasive coronary artery bypass exists. Tactile sensors has been tested by Biorobotics Lab of Harvard University, `http://biorobotics.harvard.edu`. A tactile sensor is a sensor that measures pressure changes in it's environment, this sensor is placed on the superficial tissue and by the use of signal processing algorithms it's able to approximate the location of the artery. The considerations of this project is that you have to insert this sensor into the chest cavity and place it onto the superficial tissue. It sometimes has problems to follow the artery correctly and is using backtracking algorithms to get back on track again. According to the report the measured errors in an experimental setup was 2 mm with a standard deviation of 3 mm, while it was able to follow the artery for 3 cm. This project is described in [BEASLEY-02].

And finally we have the field of mathematics that is in use in this thesis and in almost every article cited. Most of the mathematics can be explained in any university level introducing book in linear algebra or calculus.

Chapter 5

Project overview

In figure 5.1 there is an overview of the whole project as planned. This section provides a short schematicly bound description.

We have two sources at the top, *Stereo endoscope* and *CT*. The endoscope gives the stereoscopic video stream while the CT gives image slices that can be used to build up a 3D representation of a scanned object.

Following the endoscope path, we first let some images of a calibration pattern go through *Peru*, which is the implemented tool which includes the functionality to calibrate a camera. This gives camera specific parameters that can be used to undistort the video stream. The undistorted video stream is going through stereo algorithms to extract depth information. This enters the *Surface registration* together with the orientation vectors of the endoscope retrieved by a tracking system, and a segmented surface.

The segmented surface comes from the CT path. The scanned CT slices are represented as a 3D dataset which is segmented with two labels, the exterior surface (the visible surface) and the interior surface (the surface of the hidden parts that we want to superimpose onto the video stream).

The *Surface matching/registration* takes the three earlier mentioned inputs and determines the correct orientation of the dataset relative the camera. Thus we can overlay the video stream with the hidden objects(lima).

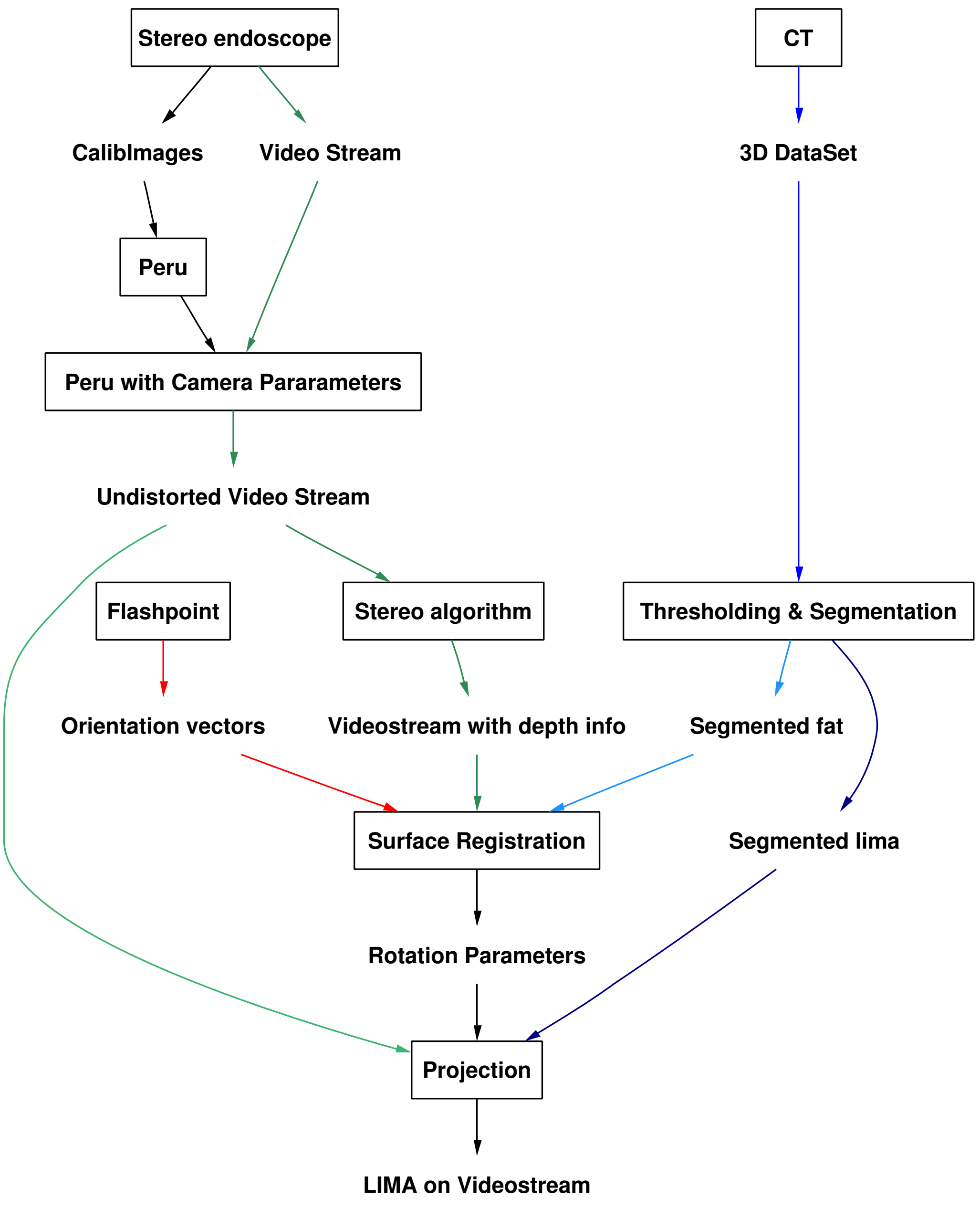


Figure 5.1: Flow diagram of the project. Greenish arrows indicates the flow of the video stream while bluish arrows indicates the flow of the model data set. This figure is explained in depth in the chapter text.

Chapter 6

The Model

6.1 Purpose

Working with stereo images from inside the chest cavity is not an easy task. The intensity is evenly distributed and it's not easy to find corresponding points between the left and right image. There are also instruments partly blocking the view. So it seems like a good idea to start with something simpler.

A simple model was built in lego. The model consists of one large house with a smaller house hidden inside. There are also two objects outside the larger house that can be used if needed. The small house inside the larger house will here represent the hidden object that we want to visualize, and relating to the TECAB procedure the little house would represent the LIMA and the large house would represent the visible surface inside the chest cavity. The interesting parts of the dataset are the surfaces of the mentioned parts, as that is what is seen by the eye or a camera.

The model can be seen in figure 6.1, and figure 6.2.

The model was scanned in a CT-scanner and one slice can be seen in figure 6.3.

One screenshot from the developed application where a part of the model is visualized can be seen in figure 6.4.

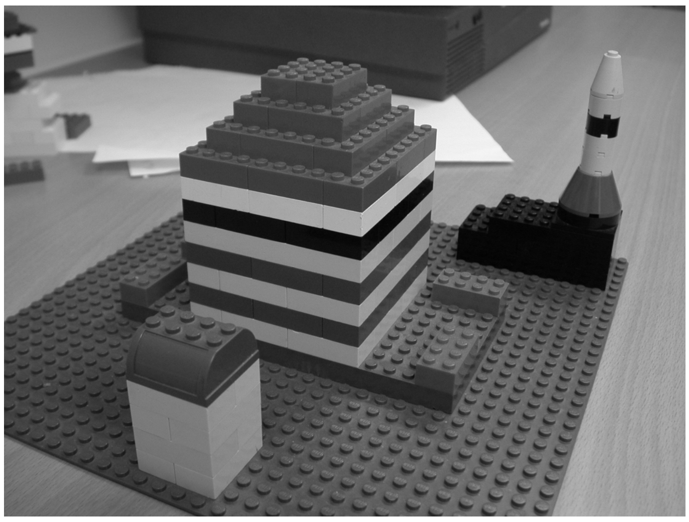


Figure 6.1: The exterior of the logo model as the camera will see it

6.2 Problems

Due to problems with the images from the endoscope a computer generated model has also been used that can be seen in figure 7.23. These problems are discussed in chapter 10.

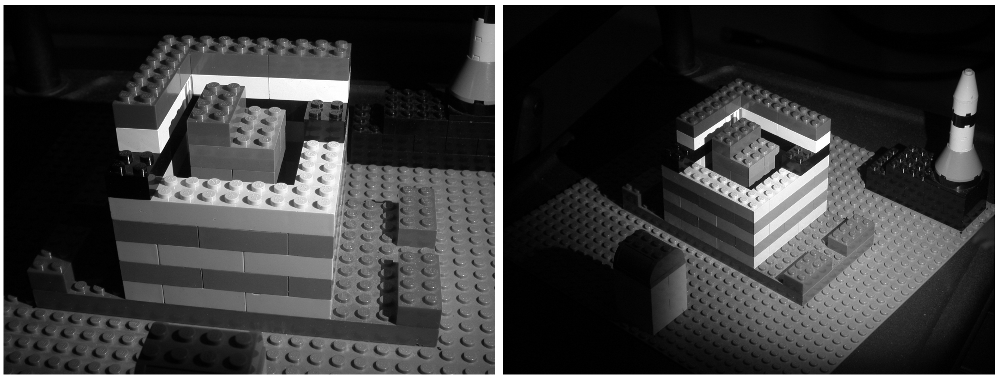


Figure 6.2: Two views of the interior of the model which contains the hidden object that should be superimposed onto the video stream.

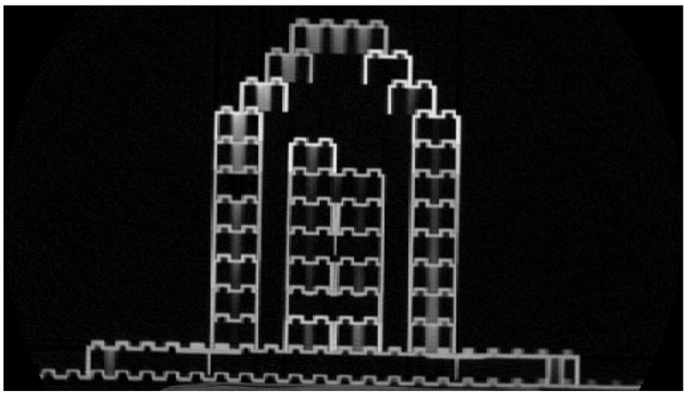
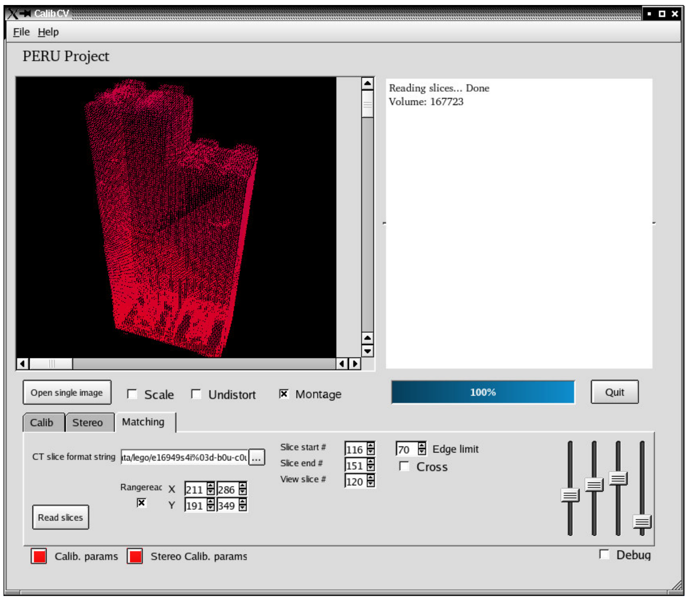


Figure 6.3: Slice of the lego model from a CT scan. The inner hidden object (the smaller house) is clearly visible.



Reading slices... Done
Volume: 167723

Scale [ ] Undistort [ ] Montage [✓]

CT slice format string: ta/lego/e16949s4i%03d-b0u-c0u

Rangereac [✓] X 211 286 Y 191 349

Slice start # 116
Slice end # 151
View slice # 120

70 Edge limit
Cross [ ]

🟥 Calib. params 🟥 Stereo Calib. params

Debug [ ]

Figure 6.4: Slices 116 to 151 in the range of the hidden house of the model. Data visualized as a pointcloud using a ray casting algorithm with disparity color visualization.

Chapter 7

Theory

This is the main part of this thesis where the different algorithms are discussed and tested. Section 7.4 discusses how the stereo capability of the camera can be exploited to find the endoscopes position and orientation relative the CT scanned object. Section 7.5 aims for the same goal, but here a tracking system is used to directly get the position and orientation of the endoscope. Finally section 7.6 discusses how these two techniques can be used together. Section 7.3 discusses the calibration of the camera that is used in all approaches to the problem. First section 7.1 starts with a simplification of the problem by making certain assumptions.

7.1 Assumptions

In this thesis we are going to assume that the segmentation has already been done with correct labeling. We also assume that the hidden part of the CT data is at rest relative to the rest of the CT dataset which means that no deformations of the dataset are expected. This means that correctly knowing the position and orientation of the surface of the exterior of the CT dataset makes a precise positioning of hidden parts possible.

7.2 Segmenting the CT-data

We need to make a segmentation of the 3D angio dataset to extract the LIMA and then project it onto the video stream. There are several methods to do this, these methods are not considered in this thesis as the anatomical dataset is assumed to be already segmented. Though a simple segmentation of the experimental model need to be done as described in section 7.2.1.

We also need to find which points in the dataset that are the points visible to the camera or in other words; the points that are on the external surface of the object. These points are called the exterior surface. These exterior surface points are going to be used in a surface registration phase while for the LIMA it will consist of the points that are needed for visualization. The last mentioned points aimed for visualization is the interior surface.

7.2.1 Simple segmentation of the model

Due to the geometry of the model and the hidden object it's an easy task so segment it into exterior and interior. The Peru application provides methods for reading user specified sub cubes, and with this functionality it's easy to make a segmented version of the model dataset. Just scroll through the slices until you find the correct starting slice and enter that in the *Slice start* box. Don't forget to enter the preferred *Edge limit* which is the thresholding value. The result of this thresholding is directly visualized in the image window. Then locate the slice that contains the largest part of the hidden object and by using the middle mouse button press in upper left corner hold and drag and then release at the bottom right corner. The coordinates appear in the *Rangeread* boxes. Next find the first slice that do not contain your hidden object and note that in the *Slice end* box. Now check the *Rangeread* box and apply *Read slices*. The surface points are found as desribed in section 7.2.2 and can now be directly 3D visualized to see if you are satisfied with the result. The visualization is done by dragging the sliders to the right in the interface. The dataset is saved to disk under the filename pointcloud.data.

7.2.2 Surface extraction

Even when reading the subcube which contains the hidden object that we want to visualize we don't need to use all the points in the visualization. An algorithm that finds the surface points and ignores the inner points was written. Using neighbor relationships, the connected zeros are found (empty space around object), then localization of object points connected with the marked space using the neighbor relationship is made. Thus only points that are neighbors to the empty spaced are selected. Currently two neighbor relationships are supported, the 26 cube neighbors and a 3D cross using 6 neighbors. This algorithm reduces the number of points in the dataset heavily. Though this method can yield more points than necessary due to the fact that it follows the outer connected space and if the model has a hole it will fill it. Thus a sphere with a tiny hole would preserve all it's inside surface points which of course are not very interesting from a camera point of view. This is not a problem that will be handled here because of the simple closed geometry of the model. But a quick thought is to use only those points where it's possible to draw a straight line to the outside of the object, along which no object points are intersected.

7.2.3 Tessellating a pointcloud using marching cubes

Visualizing a set of many points in 3D can be a good visualization if we have a few numbers of points; due to the fact that we can see the structure of the whole object at once. Movement of the point cloud is often needed to make us perceive the structure. But when the point set becomes very large it gets hard to make out all the structures and it gets computationally intensive, so instead we want to render the points using surfaces. The most basic surface made out from points is the triangle and is thus used extensively in computer graphics. The process of finding what triangles to use in a set of points is called tessellation and one algorithm that tessellates a set of points in 3D is the marching cubes algorithm which is described in this section and was introduced by Lorensen and Cline in 1987 [LORENSEN-87].

The marching cubes algorithm as commonly described takes a volumetric dataset and tessellates the iso surface at a specified iso value. Applied to a binary cube (point cloud) we can use 0.5 as iso value. Using a non binary dataset a simple thresholding is applied at the iso value and a binary dataset is obtained.

The algorithm handles so called voxels (a primary three dimensional cell, topological equivalent with the hexahedron where the six sides is perpendicular to one of the coordinate axes, [SHROEDER-98]) and looks at the eight corners. After thresholding the dataset it's voxels can be in several different states depending on the values of the corners. 8 corners with 2 possible states each yields $2^8 = 256$ cases. Of course several of these cases are equivalent for the tessellation, for example all 8 cases with just one point. The 256 cases can be shown to be reduced to 15 equivalence classes due to the symmetry. If you leave out the case where all or none of the corners are above the iso value which would not yield any triangles we are left with 14 equivalence classes of tessellation. One of the classes can be seen in figure 7.1.

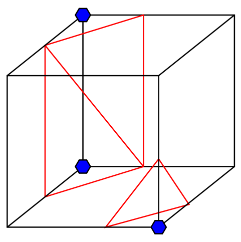


Figure 7.1: One of the cases in the marching cubes algorithm where the blue dots indicates pixels with values above the threshold and in red lines the tessellation being made in this configuration.

The algorithm looks at each voxel in the dataset and tessellates according to which of the 15 equivalence classes the current voxel belongs to. Unfortunately it's not so easy as just deciding which equivalence class the voxel belongs to because some of the classes shows ambiguity, in other words there are several ways to tessellate some of the classes. If this is done carelessly there can be holes in the surface. Some solutions to this problem exists and one is to use an alternate technique which is called marching tetrahedron. The marching tetrahedron does not show any ambiguity for the classes and thus leaves no holes, but marching tetrahedron gives more triangles and bumps can appear in the surface, [SHROEDER-98]. Another approach is the *asymptotic decider* by Nielson and Hamann [NIELSON-91]. In the other methods the values of the dataset are assumed to vary linearly along the edges, but in the asymptotic decider the data is assumed to vary bilinearly over the face. So to find the edge cut points a bilinear interpolation is carried out. It can be verified that the contour curves of the bilinear interpolation are hyperbola. Looking at one face of the cube we thus get two hyperbola. When both the hyperbola intersects the domain (the cube face) we have an ambiguity. The criteria then used to connect the edge cut points by Nielson and Hamann is based upon whether or not they are joined by a component of the hyperbolic arc. This selection can be determined by comparing the contour value with the value of the bilinear interpolant. See [NIELSON-91] for details.

A third approach is to use complementary cases, these are cases that fits with the neighboring cases and prevents holes. This is an efficient and simple solution to the problem.

The implementation of the algorithm can be done efficiently using lookup tables. First an 8 bit index looking up the configuration of the corners in a lookup table which returns a 12 bit number indicating which of the possible 12 edges that is cut by the iso surface. The intersection points is calculated using linear interpolation when the data is assumed to vary linearly along the cell edges. Looking up another table at the same 8 bit index tells us which triples of edge cut points that makes up the triangles (at most five).

The making of a surface from the points enhances the visualization significantly. But implementing the whole algorithm including all the possible combinations and lookup tables is time consuming and therefore one finished implementation were tested which clearly showed to improve the ability to percept the shape of the visualized object. Unfortunately this code was not open source and permission was not given by the author. Therefore all images will show the dataset as a pointcloud. But even though this has to do with estetics and not the fundamental algorithms of the thesis it needs to be held in mind that this improvement is of significant importance to make a good impression of where the hidden object is and what orientation it has. Complex shapes are very difficult to percept as point clouds.

7.3 Calibration

To be able to make a good 3D-reconstruction from camera images, camera calibration is essential. The epipolar constraint is one of the most fundamentally useful parts in the reconstruction phase, so a correction such that the images are epipolar are highly wanted.

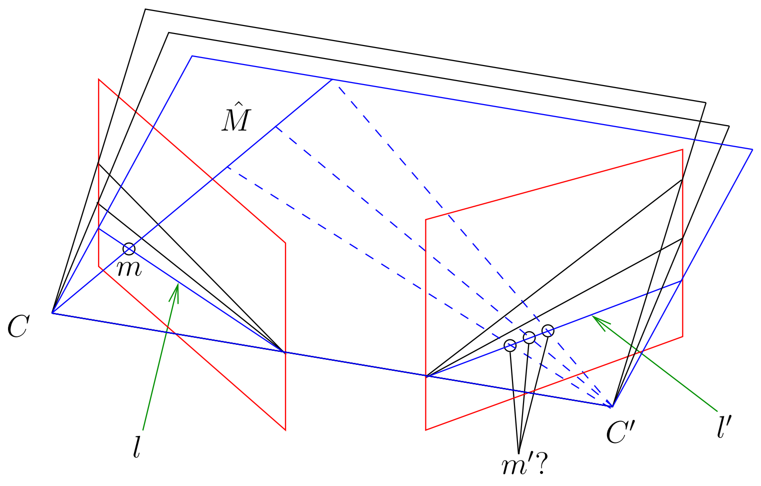


Figure 7.2: Epipolar geometry: We have two cameras C and C' capturing the left and right image planes, here in red color. The exact position of a point M in the scene is not known other than that it lies along the line $\hat{M}$ thus projected in point m on line l in the left image. The point M thus lies in the bottom plane visualized in blue. As the point lies along $\hat{M}$ it's bound to be projected at point m' somewhere along the line l' in the right image. l and l' are called epipolar lines. Points located at other locations in the scene yields the other epipolar lines indicated by the two other planes above the blue plane in the figure.

As seen in Figure 7.2 a given image point in one image restricts the position of the corresponding point in the other image. The two corresponding lines in the images are called epipolar lines and are said to be in epipolar correspondence. The plane that gives rise to the two epipolar lines can be tilted as the viewed point is moved and therefore defines a bundle of epipolar lines.

Of course there are always epipolar lines (curves) in a stereo image pair, but we want them to be horizontal (or vertical) lines such that the disparity search can be reduced to a 1D search along the rows (or columns) in the image. This process of image calibration is called rectification. The principle is to transform the images to parallel camera geometry, or in other words we want to transform the image planes such that the corresponding space planes are coinciding. The standard rectification technique is planar rectification and it selects a plane parallel to the camera baseline. Planar rectification requires however that the images are well calibrated in beforehand, even though there exists planar rectification algorithms for uncalibrated images. Another approach is polar rectification, and one easy algorithm is found in [POLLEFEYS-00]. This algorithm can deal with all possible camera geometries. Input needed is the oriented fundamental matrix.

For more details on the following section, refer to [POLLEFEYS-00]. The rectification aspect is not further examined in this thesis as the two cameras are mounted with parallel image planes in the endoscope. Though another aspect of calibration is that the coordinates in the image does not correspond with the physical coordinates in the retinal plane (image plane). For a CCD camera the relation depends on the size and shape of the pixels and the position of the CCD chip. For a standard camera the error is introduced in the digitization process. The image coordinates can be found using the following equations where x_R and y_R is the coordinates in the retinal plane...

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & (\tan(\alpha))\frac{f}{p_y} & c_x \\ & \frac{f}{p_y} & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ 1 \end{bmatrix} \tag{7.1}$$

The above equation is written simpler as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_R \\ y_R \\ 1 \end{bmatrix} \tag{7.2}$$

In equation 7.2 the upper triangular matrix is called the calibration matrix of the camera, where f_x and f_y are the focal length measured in width and height of the pixels, ans s is a skew factor. c_x and c_y are together the principal point[1]. The skew are for most cameras very close to zero. You have to remember that the calibration is made for fixed optics, a camera with changed focus or zoom makes the focal length change and thus needs other calibration parameters.

Looking at the output from the endoscope used in this thesis when it's aimed at a grid pattern reveals that camera calibration obviously is necessary. The distortion you directly notice with the eye is radial distortion where magnification is different in the edges than in the center of the image. This is illustrated in figure 7.3 and 7.4.

For radial distortion we can get the undistorted coordinates (x, y) from the observed image coordinates (x_0, y_0) by the following equations...

[1]The principal point is the projection of the point of sight upon the plane of projection

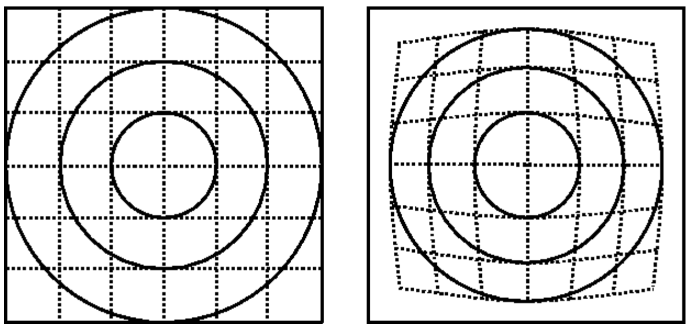


Figure 7.3: Illustrative example of radial distortion where the magnification is different in the edges than in the center of the image.

$$x = x_0 + (x_0 - c_x)(K_1 r^2 + K_2 r^4 + \ldots) \tag{7.3}$$
$$y = y_0 + (y_0 - c_y)(K_1 r^2 + K_2 r^4 + \ldots) \tag{7.4}$$

Here the $(K_1, K_2, \ldots)$ are the parameters of radial distortion where…

$$r = (x_0 - c_x)^2 + (y_0 - c_y)^2 \tag{7.5}$$

You should note that also this calibration is done for fixed optics. Should the optics change, the parameters will change. Usually only the first two parameters are used in the undistortion process as it's both expensive and numerically unstable to use too many.

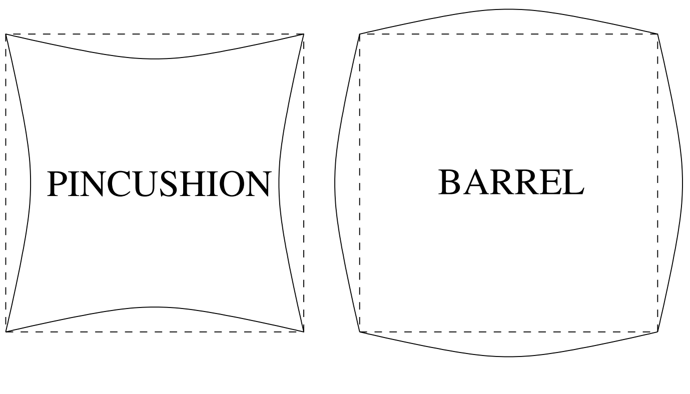


Figure 7.4: The two variants of radial distortion. To the left pincushion distortion and to the right barrel distortion. Pincushion distortion is mostly seen in tele lenses while barrel distortion is often shown in wide angle lenses. The endoscope suffers not surprisingly from the barrel variant.

All camera parameters can be divided into *intrinsic* or *extrinsic* parameters.

- ⋆ **Intrinsic parameters**
  The camera characteristics
  - **Focal length**
    The distance between the lens and the point where parallel rays through the lens diverge to a point
  - **Image center**
    Location of image center in pixel coordinates
  - **Effective pixel size**
    Describes the size of the pixel in both directions (s_x,s_y)
  - **Radial distortion coefficients of the lens**
    See equation 7.3 and 7.4.
- ⋆ **Extrinsic parameters**
  Spatial relationship between the camera and the world. If the we have calibrated for the intrinsic the following...
  - **Rotation matrix** Tells us about the rotation of the camera relative the world
  - **Translation vector** Translation of the camera relative the world

The calibration algorithms used in the Peru application are based on the work by Zhengyou Zhang [ZHANG-00].

In practice the implemented application together with the calibration works as follows. First a calibration pattern must be made. The calibration patterns should be something very much like a chessboard, but the algorithm seems to work better with patterns that do not have the same number of tiles in the x and y directions. The tiles could be rectangular and in the application there is an input parameter for the proportions of the tiles. The other input you must manually enter is the number of tiles in x and y direction. Using the camera that shall be calibrated you take pictures of the calibration pattern in several orientations. The more patterns the better result, but due to the fact that the algorithm is not completely stable too many patterns ($\approx 40+$) sometimes yields very strange results. Use of 15 ± 5 patterns should give useful results. Input to the algorithm using the Peru application is showed in figure 7.6.

The undistortion parameters are found by first locating the intersections/corners in the checker board patterns. Knowing that these corners should lie along straight lines in an undistorted image, the parameters (equations 7.3, 7.4) can be found.

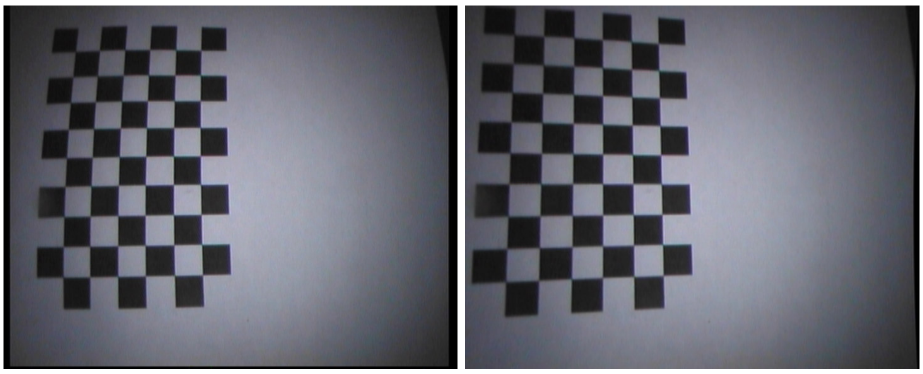
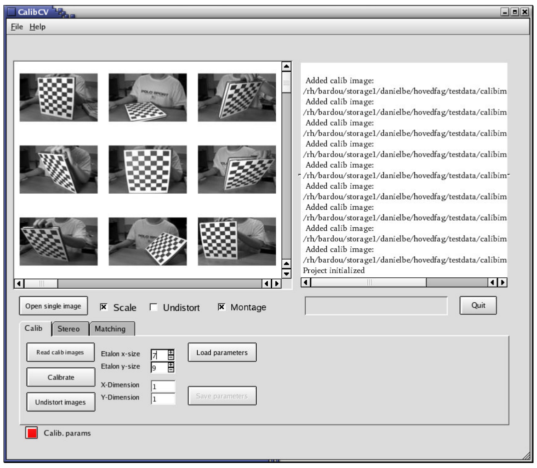


Figure 7.5: Example of image calibrated using the described algorithm. The left image is the original distorted one, and the right image is the undistorted result.

7.3.1 Preprocessing of calibration images

What images are suitable to use as calibration patterns ? Too extreme angles of the checker board pattern makes it hard for the algorithm to find the corners, so use moderate angles. Sufficient and even lighting in the calibration images is also preferable. If this is not possible some preprocessing can help the algorithm. The endoscopic camera for example uses a ring of light around the lens and thus illuminates the image in a circular round spot, see figure 7.7.

This yields images that is darker in the edges than in the middle, which gives the corner detection algorithm trouble. This can be corrected with morphology using a method called top hat transform.



```
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Added calib image:
/rh/bardou/storage1/danielbe/hovedfag/testdata/calibim
Project initialized
```

[✓ Scale] Undistort [✓ Montage]

Etalon x-size 7
Etalon y-size 9
X-Dimension 1
Y-Dimension 1

🟥 Calib. params

Figure 7.6: Screenshot from the Peru application showing the input to the calibration algorithm

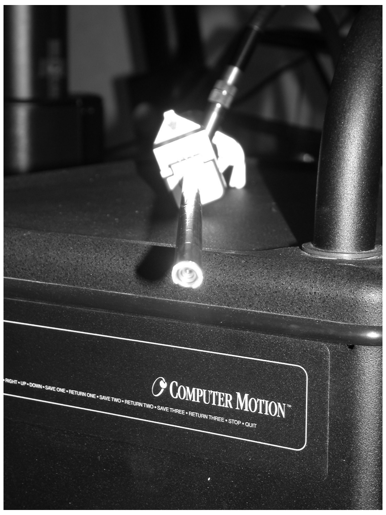


Figure 7.7: As can be seen as a glowing ring on the endoscope the light will be elliptically shaped in the images. Viewing an object at close range yields images where the light do not cover the entire image thus giving images with uneven lighting. This can clearly be seen in figure 7.5.

A top hat transformation extracts objects from images with variations in lighting.

The top hat transformation consists of one morphological operation called opening. To understand what the morphological opening is, one must be familiar with two other operations called morphological erosion and morphological dilation. Erosion and dilation is performed using a structuring element. Both erosion and dilation are here described using Minkowski's formalism, [SONKA-98]. Dilation $\oplus$ using structuring element B is defined as…

$$X \oplus B = \{p \in \varepsilon^2 : p = x + b, x \in X, b \in B\} \tag{7.6}$$

Thus dilation $X \oplus B$ is the point set of all possible vector additions of pairs of elements, one from each of the sets X and B. Examples of the morphological transformations can be seen in figure 7.10.

Erosion $\ominus$ using structuring element B is defined as…

$$X \ominus B = \{p \in \varepsilon^2 : p + b \in X \; \forall \, b \in B\} \tag{7.7}$$

Thus erosion $X \ominus B$ is given by all the points p for which all possible $p + b$ are in X.

Contour extraction can for example be made as $I - (I \ominus B)$ or $(I \oplus B) - I$ where I is an image and B a structure element.

The morphological opening of an image is defined as…

$$X \circ B = (X \ominus B) \oplus B \tag{7.8}$$

…and the closing as…

$$X \bullet B = (X \oplus B) \ominus B \tag{7.9}$$

Closing will connect objects that are close to each other while the opening of an image will split objects that are thinly connected. How close and how thin is decided by the structure element B.

The top hat transform is now defined as…

$$X \setminus (X \circ K) = X \setminus ((X \ominus K) \oplus K) \tag{7.10}$$

…or in words the original image minus it's opening. The opening gives us all the parts of the image that can not fit into the structuring element. Note that this includes the background with it's uneven light. By removing the opening from the original we are left with an image where the badly illuminated (in our point of view) background is removed and those parts of the image that we are interested in is kept because they are larger than the structuring element. An illustration of top hat performed on an image of a chess board with a radial light source is illustrated in figure 7.8, the morphological operations involved is illustrated in figure 7.10 and a 1D illustration in figure 7.9.

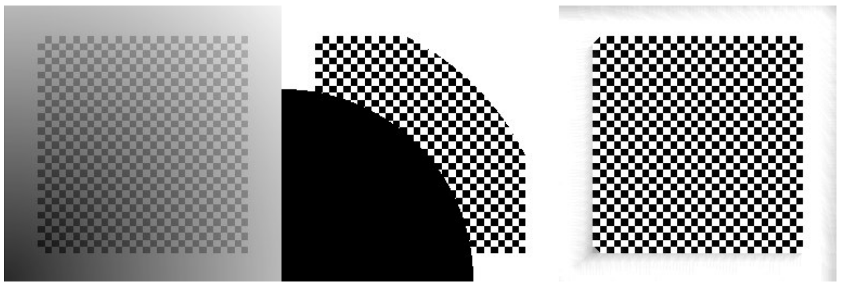


Figure 7.8: To the left the original image with a radial light source illuminating the pattern. The middle image shows a desperate try with simple thresholding yielding an unsatisfactory result. The right image shows what the top hat transform gives us after postprocessing with contrast stretching.

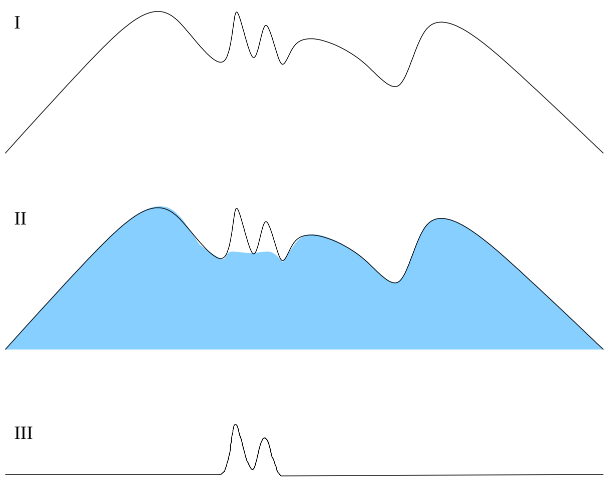


Figure 7.9: Top hat transformation in 1D. (I) is the original function before transformation. (II) Shows the opening of the function in blue. (III) Shows (I) - (II).

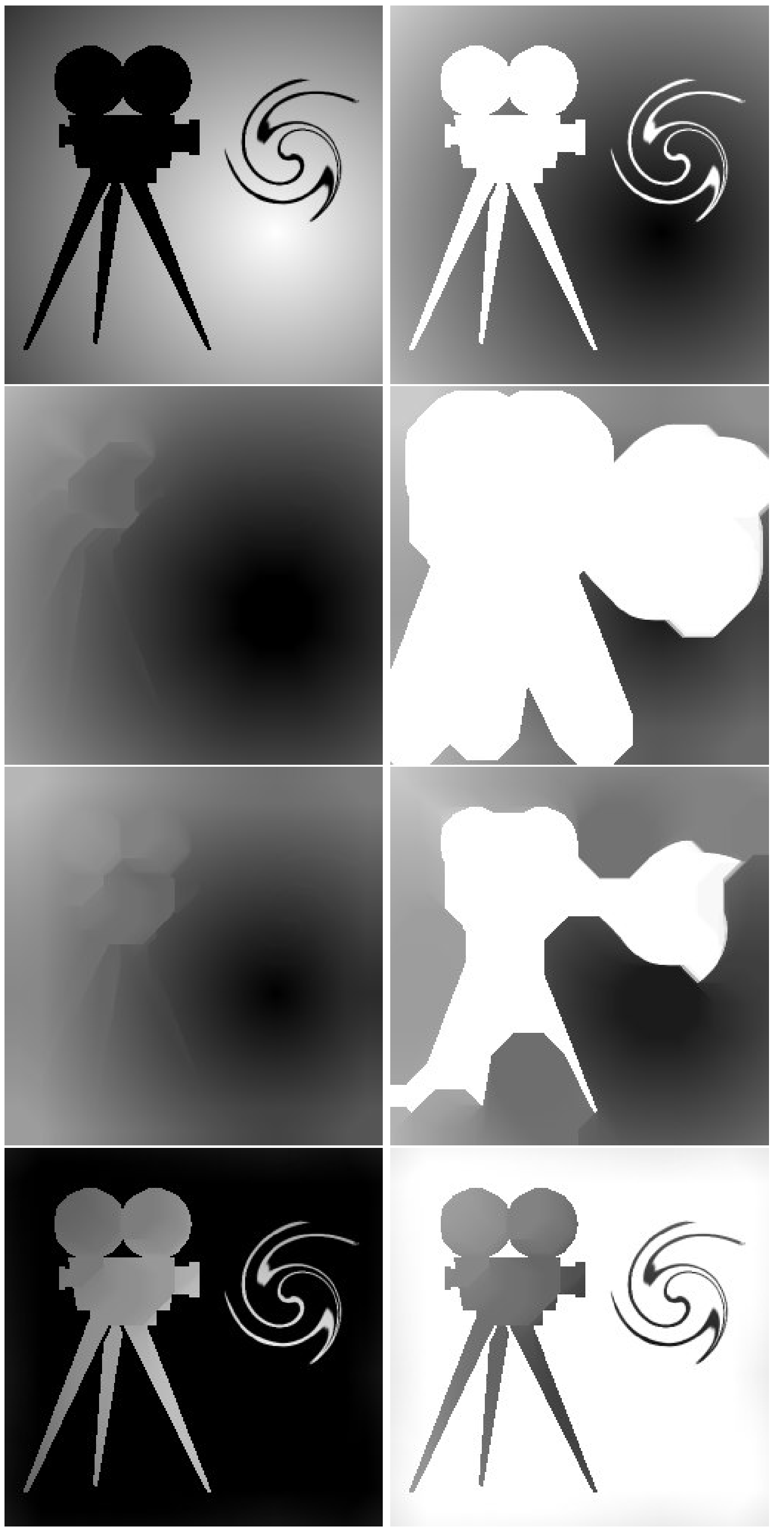


Figure 7.10: This figure illustrates the use of different morphological transformations. The top left image is the original image before any transfomations. Top right shows the complement of the original image, this is such that the structural element will work on the correct parts of the image. Second row left shows erosion performed on the complemented original image. Second row right shows the dilated version. Third row left is the opening, while third row right shows the closing. Bottom row left is the tophat transformation, and bottom row right is the complement of the tophat transformation such that we return to the original image. The structuring element used on these images was a disk with radius 25.

7.3.2 Rotation

The images from the stereo endoscope is not completely aligned such that we need to look at the problem of finding the rotation and translation of one image to fit the other which will be described in this subsection.

Problem description

Given two images of a suitable calibration pattern, find the relative rotation between them. The two images can originate from the same camera where the second image is taken with the camera rotated relative the first image, or the two images can originate from two different cameras with images of the same scene. The two images can also be slightly translated relative each other as for example is the case for a stereo image pair. An illustrative overview can be seen in figure 7.11.

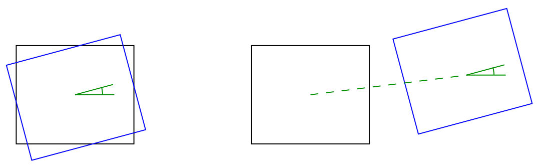


Figure 7.11: To the left only rotation and to the right rotation and translation

Choosing an appropriate pattern

Several patterns were thought of. The final decision was made according to a set of wanted features which were. . .

- ⋆ Should handle rotations from 0-360 degrees (-180,180)
- ⋆ Easy to make (drawn by hand)
- ⋆ Accurate
- ⋆ Easy to find relative translation

The idea of the final decision was to make a pattern consisting of circles which would provide an easy way of finding translation by deciding the relative movement of the center of the circles. To distinguish the circles from each other they are drawn in different sizes. This is defined as only different sizes, not specific sizes. The number of circles to use was set to 4 to be fairly moderate and provide accurate results. The results are satisfying even if the circles are drawn by hand as the algorithm detects ellipses with the circle as a special case. The position of the 4 different sized circles makes it easy to find rotations if 0 - 360 degrees while this would not have been possible with equal sized circles where we are not guaranteed to see any difference in a 90 degree rotation and a 0 degree rotation. Two images of such a pattern is shown in figure 9.1.

The algorithm

The OpenCV library [OPENCV] includes a function for ellipse finding in an image which is used. The first trial was a pattern with filled circles, but it was hard to get a single contour around the circles using that strategy. The contours must be extracted for the ellipse finding part, and are found using the canny edge detector. There were success with the filled circles, but the preprocessing was unnecessary complicated by using top hat transform and a specialized thresholding method that looked for the valley closest to the mean value in the histogram. Using non filled circles the problem of finding the significant contours directly was simplified such that any preprocessing with the canny edge detector and specialized thresholding algorithms become unnecessary.

The algorithm to find the correct relative rotation was set as follows…

I Make the image gray scale

II Apply canny edge detector and make a binary image with the found edges

III Extract the outer contours

IV Sort the found circles by size

V Calculate angles between circle centers

VI Redo the previous steps with the second image

VII Compare the found angles in both images and calculate the difference

VIII If wanted, correct one of the images such that both have same rotation

Trimming the corrected images

When we correct one of the images to get the same rotation as the other one we get areas in this image that is not represented in the other image. This is illustrated in figure 7.12.

As the stereo algorithms works on strictly rectangular images with no empty areas we want to select a rectangular area that is common to both images after the rotation. One way to do this is to find the common internal rectangle with largest possible area (the common image). Suppose the images share a common center which the other image is rotated around, figure 7.13. This assumes translation is already found where areas outside are easily cropped away.

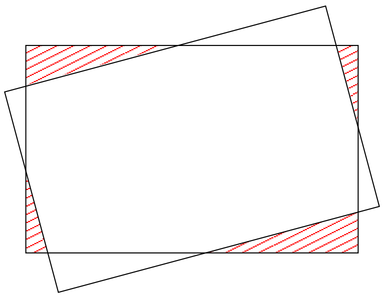


Figure 7.12: Areas not represented after a correction, here marked with slanted red pattern

As a first idea you could determine the largest possible area by placing the top of the internal rectangle Δy down from the not rotated image. This turned out to be a relatively messy equation, and the solution gives a common image asymmetrically placed in the common area. This is illustrated in figure 7.13. A better and more elegant solution is to set up the following equations. . .

$$y_t = \tan(\theta)x_t + \frac{h}{2\cos(\theta)} \tag{7.11}$$

$$y_b = \tan(\theta)x_b - \frac{h}{2\cos(\theta)} \tag{7.12}$$

Equations 7.11 and 7.12 is the top respective bottom line of the rotated image in a coordinate system centered at the center of the image that we see as not rotated

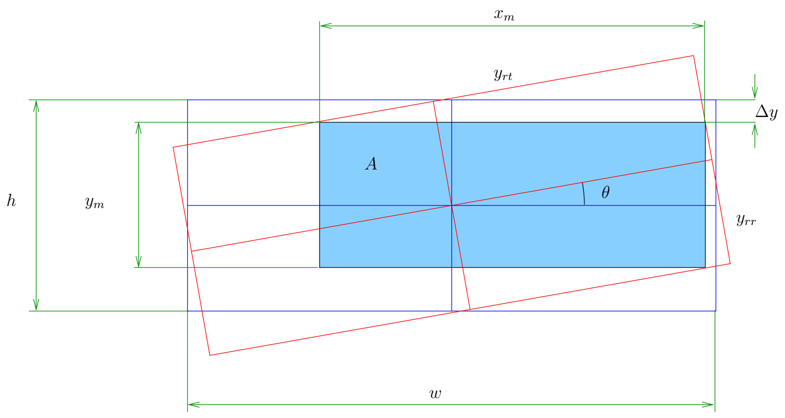


Figure 7.13: Finding the largest area A in a common rectangle for both images by setting the common rectangle Δy down from the top of the not rotated frame. Using the angle Θ and the distances shown in this figure a relatively messy equation arises for the maximum area of the common rectangle.

and the axes aligned with it. Now we want to find the largest possible area A in a common rectangle centered at the center, so we set x in the equations to $-\Delta x$ respective Δx which gives…

$$y_t = -\tan(\theta)\Delta x + \frac{h}{2\cos(\theta)} \tag{7.13}$$

$$y_b = \tan(\theta)\Delta x - \frac{h}{2\cos(\theta)} \tag{7.14}$$

This means that the height of the created internal rectangle becomes…

$$y_t - y_b = \left(-\tan(\theta)\Delta x + \frac{h}{2\cos(\theta)}\right) - \left(\tan(\theta)\Delta x - \frac{h}{2\cos(\theta)}\right) = \frac{h}{\cos(\theta)} - 2\tan(\theta)\Delta x \tag{7.15}$$

So the area A becomes…

$$A = 2\Delta x\left(\frac{h}{\cos(\theta)} - 2\tan(\theta)\Delta x\right) = 2\Delta x\left(\frac{h - 2 * \Delta x \sin(\theta)}{\cos(\theta)}\right) \tag{7.16}$$

But to not cut the rotated frame we add a restriction that the distance Δx must not be larger than where the rotated frame cuts the not rotated frame. Using equation for the right side of the rotated image 7.17 we get the intersection in equation 7.18.

$$y_r = -\frac{1}{\tan(\theta)}x + \frac{w}{2\sin(\theta)} \tag{7.17}$$

$$y_r = \frac{h}{2} \Rightarrow x = \frac{w - h\sin\theta}{2\cos\theta} \tag{7.18}$$

A plot representing the area in color with the angle and distance as axis is shown in figure 7.14.

As seen in figure 7.14 the area tends to be largest at the border against the restricted limit for angles less than approximately 40 degrees, so a good and simple choice would be to use the restricted limit as the with of the internal common rectangle. For applications where it's interesting to look at larger angles you could differentiate the area and find the maximum area below the restricted limit.

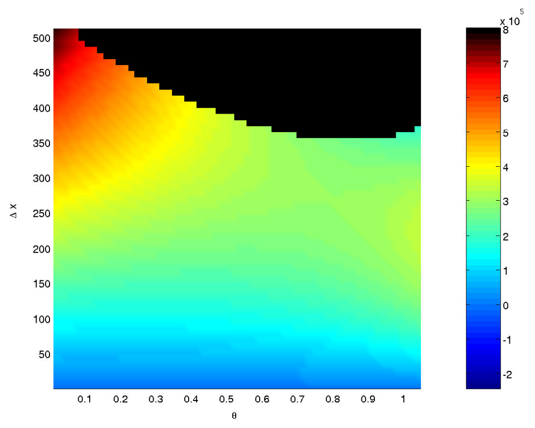


Figure 7.14: The area of the rectangle represented as color, with the axis θ for the rotated angle and Δx for the half width of the internal common rectangle. The black area is restricted due to cutting of the rotated frame.

Finding the translation

We can find the translation of the pattern by looking at the mean centre of all the circles in the not rotated image. The position we get when we rotate this center around the image center the found number of degrees is then compared to the mean circle center of the rotated image. Images from the output of the rotation algorithm can be seen in section 9.1.

7.4 Stereo approach

This chapter will discuss and examine how the stereo capability of the endoscope could be used to find the correspondence between the camera view and the CT data set.

7.4.1 Surface reconstruction based on stereo information

Based on the information of two separated images viewing the same scene, surface and depth information can be extracted. The key problem in stereo vision is to decide which points in the left and right image that corresponds. When a correspondence is set up between two points p_l and p_r, we say that the disparity for these two points is the relative movement along a scanline between the left and right frame where a scanline is an epipolar line as explained in section 7.3. For rectified images we get epipolar lines along the rows in the image and the disparity value is simply...

$$d = p_{lx} - p_{rx} \tag{7.19}$$

where p_{lx} means the x coordinate of pixel p_l.

The relationship between disparity and distance is shown in figure 7.15.

We see from figure 7.15...

$$d = s\frac{f}{z} \tag{7.20}$$

When you are to calculate the depth/disparity information there are two distinguished techniques, sparse depth maps, and dense depth maps. The sparse depth maps only calculates the depth for certain points or areas in the images such as edges found by an edge detector, while dense depth maps calculates the depth for all pixels in the image.

Computations for dense disparity maps are time consuming. But it's not always all information is needed, often the target of the stereo analysis is to calculate the distance to objects in the scene from the camera, and it can be sufficient with a sparse edge based disparity map. One method to calculate sparse disparity maps are described in [KLETTE-95] and is based on disparity histograms.

However we want to find a surface such that we will be able to find the corresponding match on the CT dataset and thus dense maps will give us most information.

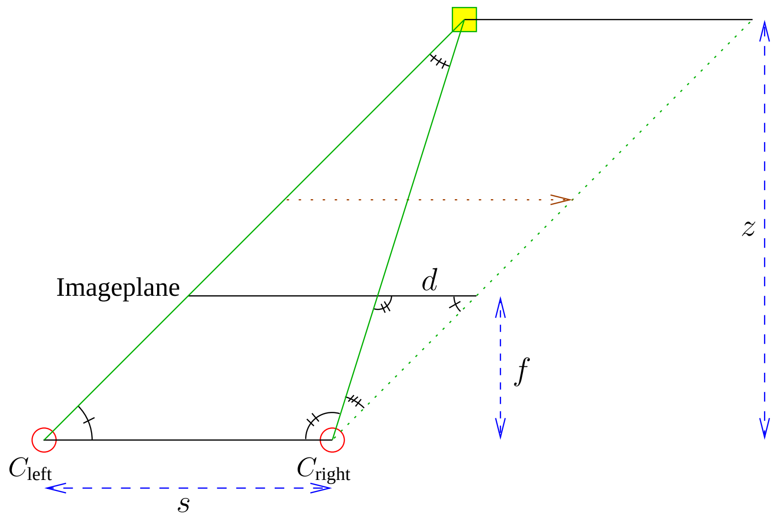


Figure 7.15: Illustration of disparity. The left and right cameras (C_{left} and C_{right}) are looking at the yellow square. z is the depth coordinate in camera coordinates. f is the focal length and s is the separation between the cameras. The distance d is the relative movement of the square in the imageplane, which is defined as the disparity.

There are some constraints to exploit in stereo imaging. First we have the...

- ⋆ **Epipolar constraint**
  See figure 7.2, with which the searchspace can be reduced to one dimension. The epipolar lines are often referred to as the scanlines

In [ROJAS-97] the following constraints that can additionally be used are listed...

- ⋆ **The continuity or smoothness constraint:**
  Disparity will vary slightly
- ⋆ **The ordering constraint:**
  The stereo projection almost always preserves the order of the primitives extracted from the two images along matching epipolar lines.
- ⋆ **The photometric constraint:**
  The disparity assignments should map points in left image to points in right image with similar photometric attributes.

To satisfy the photometric and the smoothness constraint the problem can be set up as a minimization of…

$$E = \sum_{i=1}^{N}(L(i) - R(i - d(i))^2 + \mu \sum_{i=2}^{N}(d(i) - d(i - a))^2 \tag{7.21}$$

where the first term penalizes the difference in image features between corresponding points (photometric constraint), and the second term penalizes the difference in disparity of neighbouring pixels (smoothness constraint) using a weight μ. $L(i)$ and $R(i)$ are photometric measures of the neighborhood of pixel i in one scanline.

7.4.2 Preprocessing

Some preprocessing of the left and right stereo image before applying the stereo algorithm can improve the results. It should be noted that preprocessing is more expensive than postprocessing as the preprocessing is applied to both left and right frame while the postprocessing is only applied to the generated disparity map.

7.4.2.1 Camera calibration

The first preprocessing step is to calibrate the camera and correct the images according to the theory described in section 7.3.

7.4.2.2 Noise reduction

One problem with finding corresponding points/parts of two images is noise. To deal with this problem a noise filter can help. There exists several types of noise filters.

The **Mean filter** looks at a pixel neighborhood and set the pixel value to the mean of that neighborhood. This filter reduces noise but is not good to remove so called salt and pepper noise where some pixels have incorrectly got the highest or lowest possible value. These pixels contribute too much to the mean value calculation. Also edges get blurred by this filter.

The **Median filter** is better at handling noise such as salt and pepper. Here the pixel we look at get the median value of it's neighborhood and thus is not as influenced by single extreme values as the mean filter is. But the median filter is more expensive as the values in the window used has to be sorted. And the filter does not handle gaussian noise very well.

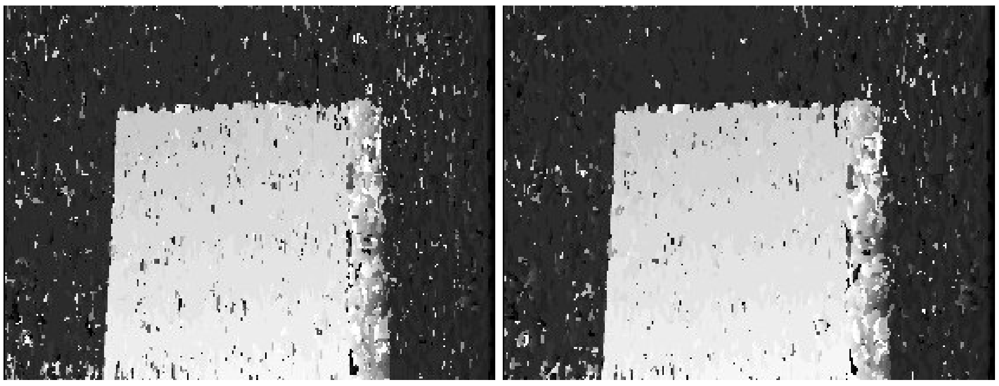


Figure 7.16: Example of noise removal pre filtering on a stereo pair of a slanted area. Left image is without pre noise removal and to the rigt the same algorithm on the same scene with pre noise removal. The gaussian filter has been used, and no post processing.

Then we have the **Gaussian filter** which is quite similar to the mean filter except that it uses different weights in the neighborhood giving the pixel we look at highest weight. The kernel is derived from an gaussian function, hence the name. This filter preserves the edges better than the mean filter but is also bad at handling salt and pepper noise, but handles gaussian noise better than the median filter.

In figure 7.16 we can see that noise preprocessing with noise removal can increase the results from the stereo algorithms. There will probably not be very much salt and pepper noise in the input left and right image such that median or gaussian pre filtering could be more advantageous here.

7.4.2.3 Mean correction

If the cameras light sensitivity is different the images will get different mean values. This can be compensated for by computing the mean value of all pixels for both images and add the difference to one of the images. This can help the algorithm where the difference is significant, but can also be negative if we look at an extreme example. Imagine that one stereo scene consists of a totally black object to the left and a totally white object to the right. Then both images from the cameras will have different mean value naturally as the right image would contain more white.

7.4.2.4 Histogram equalization

If the two cameras differ more than a small amount there can be beneficial to use histogram equalization on the images. Histogram equalization redistributes intensity distributions and can also be called contrast enhancement. Contrast for two regions is defined as...

$$c = \frac{F - B}{F + B} \tag{7.22}$$

where F and B is the mean gray levels of the regions for which the contrast is evaluated, [SONKA-98].

Histogram equalization increases contrast for tops in the histogram and decreases contrast for valleys in the histogram. A histogram equalization on two images that has histogram that differs slightly can be more similar after histogram equalization has been performed on both images. If you apply this filter on each channel separately you can get shifts in color that can visually appear a bit odd, this is not necessarily bad for the algorithms that will work on the image.

7.4.3 Stereo algorithms

This section contains an overview of different stereo algorithm techniques and those implemented and tested in this thesis.

7.4.3.1 Blockmatching

Block matching compares equally sized blocks in the left and right image, and the Mean Square Error (MSE) between the pixel values inside the respective blocks defines a measure of the similarity. For gray value image the MSE is defined as...

$$MSE(x, y, \delta) = \frac{1}{\Delta w^2} \sum_{i=-k}^{k} \sum_{j=-k}^{k} |E_R(x+i, y+j) - (E_L(x+i+\delta, y+j)|^2 \tag{7.23}$$

where δ is an offset describing the difference $(x_R - x_L)$ between the column positions in the left and right image. When using color images we must measure the distance between colors, and it shows that the method used for deciding the distance between colors have no significant value [KLETTE-95], so when the euclidean distance in the RGB color cube is used the MSE can be calculated as…

$$MSE_{COLOR}(x, y, \delta) = \frac{1}{\Delta w^2} \sum_{i=-k}^{k} \sum_{j=-k}^{k} dist_c(C_R(x+i, y+j), C_L(x+i+\delta, y+j)) \tag{7.24}$$

where for two colors $c_1 = (R_1, G_1, B_1)$ and $c_2 = (R_2, G_2, B_2)$ the distance $dist_c(c_1, c_2) = |R_1 - R_2|^2 + |G_1 - G_2|^2 + |B_1 - B_2|^2$

The disparity d between the blocks is the distance between positions of the blocks showing the minimum MSE or MSE_{COLOR}. The search can be limited by defining a maximum disparity D_{max}.

A technique based on block matching was found to be rather efficient in [KLETTE-95] for obtaining dense stereo correspondence. Different color models where investigated for this method (RGB, XYZ, $I_1I_2I_3$, HSI), where $I_1I_2I_3$ showed to provide the best information for stereo matching. $I_1I_2I_3$ is defined as…

$$I_1 = \frac{R+G+B}{3}, \quad I_2 = \frac{R-B}{2}, \quad \text{and} \quad I_3 = \frac{2G-R-B}{4}$$

Several block matching algorithms have been implemented. One fast version using a constant window size of 3, one with variable window size with more accurate results called standard blockmatching and one fast method that advances a full block at the time. Finally a hierarchical method has been tested which is described in the next section 7.4.3.2. The earlier mentioned different approaches will now be described in more detail.

Standard Blockmatching

The standard method takes three parameters as input in addition to the two frames, D_{max} which limits the maximum disparity between two blocks, blocksize Δw which is the width of the window used and finally what type of error measure that should be applied to measure the error between blocks. The error measures implemented is either the absolute difference or the squared difference. Δw is currently limited in the implementation to positive odd values to get a center pixel in the block. Algorithm follows…

I Use left and right images with extended borders such that we get values for every pixel in the image. The extended images have the size: $X + \Delta w - 1 + D_{\text{max}}, Y + \Delta w - 1$ where X and Y is the width and height of the original images.

II Looking at the window around each pixel in the right image we start scanning using blocks in the left image toward the right until D_{max} is reached. The block where minimum difference is reached is the matched block. Now the pixel in the right image is assigned the difference in x-values between itself and the center pixel in the matched block.

III Advance to next pixel in the right image

The algorithm is illustrated in figure 7.17.

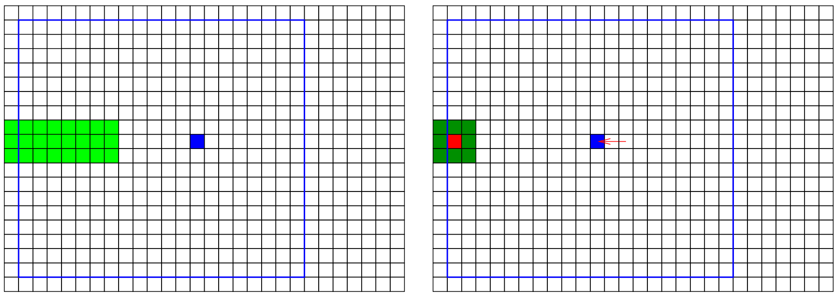


Figure 7.17: This is an illustration of the block matching algorithm. The block around the red pixel in the right frame is about to be matched against a block in the left frame. Tested blocks will slide through the left frame toward the right, here indicated by light green pixels. The blue frame indicates the original frame sizes, and the grid extends to the enlarged frames used in the calculations. In this example the max disparity D_{max} is set to 5 and the width of the block $\Delta w = 3$. The blue pixel that appears in both frames is shifted 2 pixel to the left in the right frame to indicate a pixel that would get disparity 2.

Fast Blockmatching

The fast method uses a constant window size of 3. The difference between the standard and the fast method that makes it possible to speed it up is that the fast method compares block sum to block sum where the standard method compares the difference between blocks by a pixel to pixel strategy. This difference makes it possible to precompute the block values such that one block sum does not have to be calculated more than once even though the testing block will slide over it several times. This speeds the algorithm up a great deal and it gives better result than comparing the images pixel to pixel (standard method with block size 1). Though the standard method with block size 3 gives better result than the fast method in terms of the error, as can be seen in section 9.2.

Blockadvancing Blockmatch

This method is similar to the Standard with the difference that as we get the disparity value for a pixel it is set for all pixels belonging to it's surrounding block. Then we advance block size number of pixels. This will of course give a less detailed disparity map, where the detail decreases with increasing block size. Though we get a faster algorithm that also gives less pixel noise in the resulting disparity map.

A in depth comparison between the methods is found in the Result chapter (Ch 9).

Due to the problems with large areas of about the same intensity, a algorithm that better handles large areas of same intensity is wanted. One approach is the hierarchical one that uses an image pyramid as described in section 7.4.3.2.

7.4.3.2 Pyramid blockmatching

Large areas of the same intensity is a problem for the ordinary blockmatching method as the blocks will get very similar values.. One method to address this problem is by using an hierarchical image pyramid during the processing as in [KOSCHAN-96]. The use of image pyramids was introduced in the 1970's by Tanimoto and Pavlidis in [TANIMOTO-75]. Examples of image pyramids can be seen in figure 7.18.

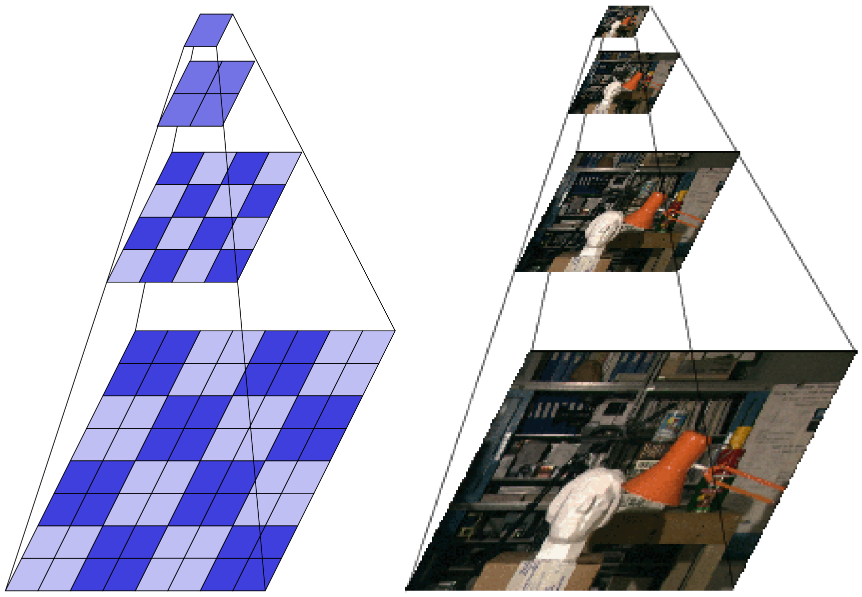


Figure 7.18: Hierarchical image pyramid or Gaussian image pyramid which is created by smoothing and subsampling the original to lower resolution. To the left an example pattern, and to the right using an image.

The idea is that when we reduce the resolution of the image we get one pixel that represents several or many depending on how large the reduction. For large untextured areas we soon get one pixel that represents the area. Blockmatching on this resolution reduced image assigns a disparity value to the pixel. Then looking at an image with increased resolution we can use the disparity value of the pixel p as a hint for the pixels that pixel p represents on a lower level. This restriction that comes from the images up in the pyramid prevents pixels in large untextured areas to slide away too much further down in the pyramid. This would of course yield a problem in scenes where the depth oscillates rapidly. For example a stereo image of a tree would probably have this problem with the pyramid blockmatch approach. You have to know what type of scene you are capturing before deciding to use a particular stereo algorithm.

We can think of the original image and the resolution reduced images as the image pyramid where the base is the original image and upward in the pyramid we have images with lower resolution. If we keep on going we will have the top of the pyramid as only one pixel which is the mean value of all the pixels in the original image. The pyramid blockmatching algorithm does not need to create a pyramid so large that we get only one pixel at the top, as block matching on a single pixel is meaningless. For a normal sized image, the use of 2 - 5 levels in the pyramid is reasonable.

The following rules are set up…

$$D_{\Delta}(s) = 2^{s-1} D_T \tag{7.25}$$

$$D_{\min}(s) = \begin{cases} D(0) - D_{\Delta}(s) & \text{for } s = 1 \\ D_{\min}(s-1) - D_{\Delta}(s-1) & \text{for } s > 1 \end{cases} \tag{7.26}$$

$$D_{\max}(s) = \begin{cases} D(0) + D_{\Delta}(s) & \text{for } s = 1 \\ D_{\max}(s-1) + D_{\Delta}(s-1) & \text{for } s > 1 \end{cases} \tag{7.27}$$

In the equations we have that $D(s+1)$ is the disparity at pyramid level $s+1$ can be derived from the disparities at level s. D_T is the tolerance of the disparities on the next level, which controls the smoothness as small D_T gives smaller width of the search space D_{Δ}.

Output disparity maps from the blockmatching algorithms can be seen in figure 9.6.

7.4.3.3 Birchfield

The birchfield algorithm matches individual pixels in scanline pairs while allowing occluded pixels being unmatched.

An occluded pixel is a pixel which contains the intensity originating from of a point on an object for which a match do not exist in the other image. This is because the viewline from the camera to this part of the object is blocked by an object in it's path. Easy experiment to understand this fact is the classical stereo finger exercise; when you look at your finger held up in front of your eyes with one eye closed, the finger is covering some part of your view. When you look with the other eye you cover more or less another part of the background. Thus some part of your view is just visible for one of the eyes and occluded for the other.

The algorithm handles large untextured areas better than some other algorithms. Large untextured areas is hard because it's not enough to look at a small neighborhood to state the correlating pixels in the other image.

For each scanline, which should be an epipolar line, in the image a matching sequence is found which tells which pixles in the left scanline that corresponds to which pixels in the corresponding scanline in the other image. A matching sequence is labeled M. Each M is associated with a cost which tells how unlikely it is that this sequence is the true correspondence. The definition of this cost sequence in equation 7.28 is a constant penalty for each occlusion and a reward for each match such that this cost function would give us match sequences with as few occlusions as possible and as many matches as possible.

$$\gamma(M) = N_{occ}\kappa_{occ} - N_m\kappa_r + \sum_{i=1}^{N_m} d(x_i, y_i) \tag{7.28}$$

The disadvantage with this cost function is that it prefers piecewise constant disparity maps. This means that a slow real change in disparity over the scanline results in a single constant disparity in the algorithm. The pros is that it detects depth discontinuities very accurate. Also the simplicity of the cost function makes it easy to implement in opposite to other algorithms that uses very complex cost functions.

In equation 7.28 we have that κ_{occ} is the constant occlusion penalty, κ_r is the constant reward, $d(x_i, y_i)$ is the dissimilarity between pixels x_i and y_i. N_{occ} and N_m are the number of occlusions and matches respectively in M.

To understand the terms in the cost function one can imagine κ_{occ} as an increasing evidence as we get more and more occlusions that we should change the disparity. κ_r can be thought of as the maximum amount of dissimilarity that can be expected for two matching pixels. In the original version of the birchfield algorithm in [BIRCHFIELD-98] $\kappa_{occ} = 25$ and $\kappa_r = 5$ are used.

Looking at how the dissimilarity $d(x_i, y_i)$ is calculated it is not only the difference between the pixels. Many algorithm do this for simplicity, but sampling of the same edge can be quite different in two images. This problem can be dealt with in several ways. One can use subpixel resolution and thus get a more accurate position of an edge, but this is a quite expensive operation. Using blocks of pixels as in the block matching algorithms (also called windows) is another alternative. Birchfield states that a third method of using *linearly interpolated intensity functions* surrounding two pixels can be proved insensitive to sampling.

The interpolation function linearly interpolates one pixel around the pixel being tested in the right image and uses the pixel itself, y_i and the two interpolated values I_R^- and I_R^+ in the comparison with the pixel x_i in the left image. Finding the minimum I_{min} and the maximum I_{max} of I_R^-, I_R^+ and y_i, the dissimilarity is defined in equation 7.29.

$$d(x_i, y_i) = \max\{0, x_i - I_{max}, I_{min} - x_i\} \tag{7.29}$$

There is one restriction here and it is that the vicinity of x_i and y_i on the sensor must be either concave or convex. This problem is illustrated by a simple stereopair constructed which contains two textured squares on a white background one shifted more than the other. Illustration of the problem can be seen in figure 7.19. The algorithm thus genereates better results if the lens has been slightly defocused or by use of gaussian smoothing before the computation.

In addition to the cost function some constraints are used. The first set of constraints deal with the problem of untextured regions. This constraints make the algorithm handle this regions better than many other stereo algorithms. The property of the images to be fulfilled for this to work is that the depth discontinuities has an intesity variation of at least 5 gray levels. Other constraints is maximum disparity and that no matched pixel has occluded pixels on both sides of itself.

Using the algorithm as this far described could give *strange* behaviour looking at the scanlines next to each other as each scanline is treated individually. Extending the costfunction to a two dimensional const function that also lookes at neigborhoods of scanlines could overcome this problem. However as this is a costly way of doing it, the birchfield algorithm instead uses a global postprocessing method that only increases the time used by 30% instead of the 2D extension which sometimes can cost as much as 800% in computation time. In the postprocessing method each pixel is assigned a reliability that it has this disparity when looking at it's column. After looking at the columns, reliability is checked along rows. The disparity map are now corrected by removing isolated disparities which is surrounded by pixels with high reliability. Finally a mode filtering is made to clean the disparity map. Mode filtering uses the most frequently occurring inten-

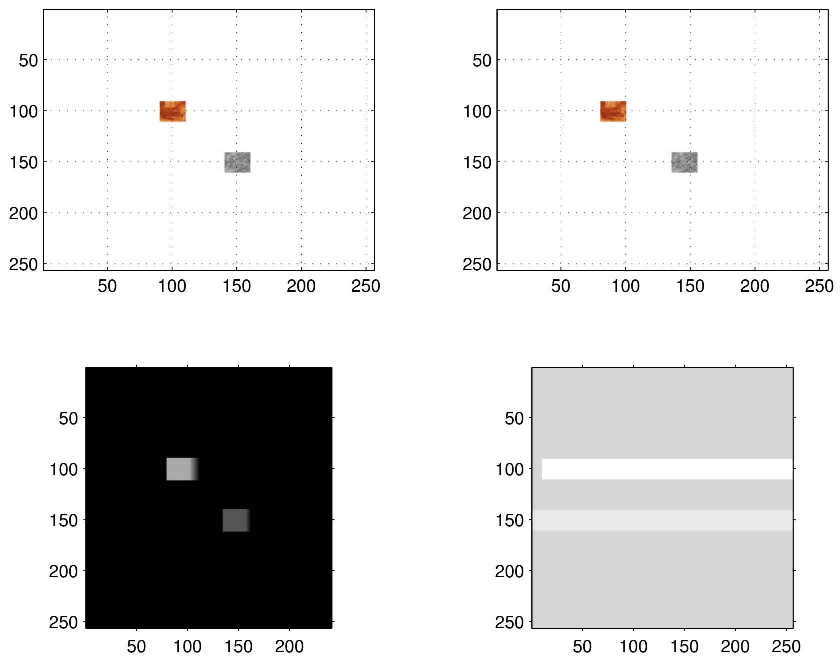


Figure 7.19: Top row contains left and right image of the constructed pair. Lower left shows blockmatching performed on the pair, while the lower right shows birchfield on the same pair.

sity in a kernel of specified size.

7.4.3.4 Sparse methods

Because we know the geometry of the object we are seeing in the stereoscope sparse methods can be at least something to have in mind.

[ROJAS-97] describes two sparse procedures using minimization and least-cost strategies that can be used either separately or in union. The first procedure is based on use of dynamic programming to find the shortest distance between two strings. Two strings $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ are given from the pixels in the left and right images respectively. A function $F : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, m, \Theta\}$ maps pixels from string A to string B. $F(i) = \Theta$ means that no correspondence is found, while $F(i) = k$ tells that the i'th pixel of A can be paired with the k'th pixel of B. A cost function C_F is associated with the mapping, where $F(i) = \Theta$ gives a constant cost of τ, and $F(i) = k$ gives a cost that are influenced by attributes of both pixels in A and B. The distance between the two strings are defined as a minimization of the cost function over all the pixels. The minimum total cost of matching the first i symbols in A with the j first symbols in B, can now be written using a recurrence relation, that allows use of dynamic programming techniques, equation 7.30…

$$CTM(i,j) = \min\{CTM(i-1,j-1)+cost(i,j), CTM(i,j-1)+\tau, CTM(i-1,j)+\tau\} \tag{7.30}$$

The procedure starts with finding edgepixels in left and right image using the Laplacian-of-Gaussian mask (LoG). The laplacian is an 2D isotropic measure of the 2nd spatial derivative of an image. The 2nd spatial derivate highlights regions of rapid intensity change, which makes it suitable for edge detection. The gaussian is used to smooth the image before the laplacian is applied to reduce the noise sensitivity. The two filters can be applied as one, as this convolution is assosiative. The LoG output contains zero-crossings in the image…

- ⋆ Zero at a long distance from the edge
- ⋆ Positive just to one side of the edge
- ⋆ Negative just to the other side of the edge
- ⋆ Zero at some point in between, on the edge itself

For each horizontal line the zero-crossings are found and put in string A and B for the left and right image respectively. The costfunction in equation 7.30 is defined as follows…

$$cost(i,j) = \sum_{k=1}^{p} w_k (L_k(i) - R_k(j))^2 \tag{7.31}$$

$L(i)$ and $R(j)$ represents image features and w_k a weight. Now equation 7.30 can be applied to find the optimal correspondence between the pixels. The described procedure only calculates the correspondence between edge pixels, and thus only the disparity for the edgepixels. In [ROJAS-97] a second procedure is described which takes two complete epipolar lines from the left and right images. Using procedure one we can make use of the fact that the pixels in an intervall l_k between two pixels in A must correspond to pixels in the intervall between pixels in B that matched the two pixels in A. Now we can minimize the following equation which is an extension of equation 7.21.

$$E = \sum_{i \in l_k} v(i)(L(i) - R(i - d(i))^2 + \sum_{i \in l_k} (1 - v(i))\tau + \mu \sum_{i \in l_k} v(i)(d(i) - d(i - a))^2 \tag{7.32}$$

where

$$v(i) = \begin{cases} 1 & \text{if pixel i of L is visible in the right image,} \\ 0 & \text{otherwise} \end{cases} \tag{7.33}$$

Now the CTM matrix is set up where each row corresponds to one pixel's ($\in l_k$) possible disparities. The minimization problem is now solved by finding the shortest path in a weighted directed graph, where the vertices are the possible disparities. In figure 7.20 the matrix elements are drawn as vertices, and a top node is inserted. The edges are weighted and the minimization is solved by finding the shortest path from the top-node to the bottom.

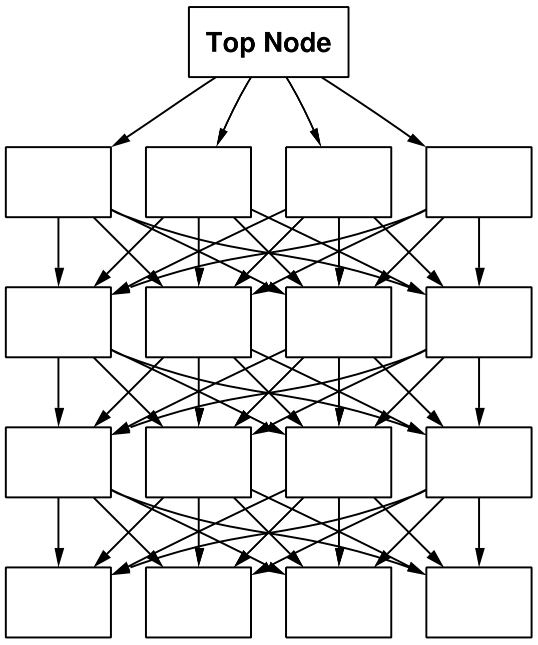


Figure 7.20: CTM-matrix as a weighted directed graph. To find the best string match we start at the top node and finds the shortest path to the bottom. Each edge is weighted and the columns are the different possible disparities for the pixel representing each row.

7.4.3.5 Other methods

Other methods to achieve depth information from images are *shading based shape recovery*, see [HORN-89], where you move lightsources around and by studying the shading (shadows), depth information can be extracted. It should also be mentioned that depth information can be found using focusing, see [KRISTENSEN-93].

7.4.4 Postprocessing

The error of the output of the stereo algorithms can be further reduced using filtering directly on the resulting disparity map.

7.4.4.1 Median filtering

Here the filter is applied on the disparity map. The median filter is explained in section 7.4.2.2 discussing the prefilters. The median filter is a costly operation, but applying it as a postfilter reduces the computational efforts since we don't have to apply it to both left and right images, only the resulting disparity map. This filter has shown to be very beneficial in terms of reduced error, see section 9.2 in the result chapter. The raw disparity output map is often relatively noisy where some pixels incorrectly has received completely incorrect disparity which also often differs significantly from the surrounding pixels, thus a median filter could be appropriate.

7.4.4.2 Gradient Removal

This filter is specially implemented to deal with a problem that arises in the block-matching algorithms. The problem occurs around abrupt changes in disparity where the background is uniform. Looking at a block to the right of an abrupt change in disparity in the right image, the searching block in the left image will find the best match when it passes the edge (the abrupt change). When moving the block in the right image the searching block in the left image will again find the best match as the first block pass the edge as the background is uniform. Note that which of the disparities (where several best matches occur) that is chosen is dependent on the implementation. Here the lowest disparity is assumed to be chosen. This makes the disparity after the edge drop by one for each step thus resulting in gradient behavior after edges. The gradient removal filter handles this and looks for gradients where the gradient drops by one for each pixel and removes these gradients. Such gradients is vary rarely occurring naturally in a disparity map. Gradient that only consists of one drop is spared. An example of this filter is shown in figure 7.21.

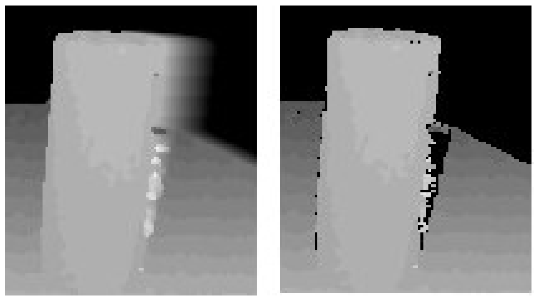


Figure 7.21: Gradient Removal where the left image has a gradient on the right hand side of the cylinder which has been removed in the right image.

7.4.5 Surface registration

To be able to correlate the calculated depth map generated by the stereo algorithms with the 3D model we turn to the subject of surface registration. In [JOHNSON-97] surface registration is described to be the process that aligns 3D data sets acquired from different view points or at different times. In this case we want to align the depth map/surface with the 3D model (in the experiments in this thesis the Lego model). When we find the correct alignment between the depth surface and the model we know the transformation that takes the model and orients it against the depth surface or vice versa. This transformation is used to find the camera position relative the model, or equivalent the models position relative the camera. By having this information we can superimpose the hidden and segmented objects of the model onto the video stream.

In general we can think of the stereo camera as a range sensor and the Lego house as our 3D model. Then we want to know the range sensors position relative the model by registering the surface generated by the range sensor with the 3D model.

Section 7.4.5.1 gives a brief overview over different methods of surface registration, while section 7.4.5.2 gives a more detailed description of the implemented method.

7.4.5.1 Surface registration overview

In the general case surface registration can be partitioned into three issues. . . [AUDETTE-00]

- ⋆ Choice of transformation
- ⋆ Elaboration of surface representation and similarity criterion
- ⋆ Matching and global optimization

Choice of transformation

The choice of transformation can be divided in two different cases. A *Rigid body transformation* or a *Nonrigid transformation*. A rigid body transformation is applied if the deformation between the two surfaces in negligible, while a nonrigid transformation is used if we have significant deformations of the surfaces. Noise should not be treated as a significant deformation. A rigid body transformation is as simple as. . .

$$\boldsymbol{x}_B = \boldsymbol{R}_{AB}\boldsymbol{x}_A + \boldsymbol{t}_{AB} \tag{7.34}$$

where a point $\boldsymbol{x}_B$ in the point set B is given by the point $\boldsymbol{x}_A$ in the point set A rotated $\boldsymbol{R}$ and translated $\boldsymbol{t}$. So a rigid body transformation preserves the shape and size, only position and orientation are affected. The function we want to minimize in the rigid body transformation is typically…

$$\min_{R,t} \sum_{i=1}^{N} \|\boldsymbol{x}_{B_i} - (\boldsymbol{R}\boldsymbol{x}_{A_i} + \boldsymbol{t})\|^2 \tag{7.35}$$

where i goes through all points and calculates the squared distance between corresponding points. Thus a minimization of the squared distance between the point sets.

Looking at the nonrigid transformation we must deal with a more general case of transformations called *affine transformations*. The affine transformation is written as…

$$\boldsymbol{x}_B = \boldsymbol{A}_{3\times3}\boldsymbol{x}_A + \boldsymbol{b}_{3\times1} \tag{7.36}$$

An affine transformation preserves collinearity, which means that all points lying on a line before the transform will still lie on a line after the transform, as well as ratios of distances. For nonrigid transformations there are also several other approaches such as using global polynomial functions or piecewise polynomials such as splines for local nonrigid transformations, [AUDETTE-00].

In the rigid body transformation we have a orthogonality constraint on the elements of R which is a 3×3 matrix which we do not have in the nonrigid case.

Surface representation and similarity criterion

Four approaches to represent the surfaces is **feature**, **point**, **model based** and **global similarity**. What is chosen here can depend on a variety of factors such as if the surfaces are nearly aligned from the start or if we expect an arbitrarily large transformation, also of interest is the smoothness of the surfaces.

The **Feature based** methods uses surface morphology to find features in a preprocessing step. This makes the description compact in comparison to the point and model based methods which uses all or a large subset of all points. These methods are commonly used on rigid transformations.

The **Point based** methods uses the surface points as primitives and tries to minimize distances between point pairs. Another name for surface registration based on point sets is **free-form** surface matching [BESL-92]. The free form surface matching generally assumes that the two point sets are close to aligned. The minimization function can be to minimize the squared distances between mutually closest points. There exists many free form surface matching methods where the most common difference is in the choice of distance metrics in the minimization function. One common distance metric is…

$$d(\boldsymbol{x}_B, \boldsymbol{R}_k \boldsymbol{x}_{A,\min} + \boldsymbol{t}_k) = \min_{\boldsymbol{x}_A \in X_A} d(\boldsymbol{x}_B, \boldsymbol{R}_k \boldsymbol{x}_A + \boldsymbol{t}_k) \tag{7.37}$$

where the point $\boldsymbol{x}_B \in X_B$ and $\boldsymbol{x}_A \in X_A$ where X_A is the transformed point set. To minimize the distance between closest point pairs there is a widely used method called *Iterative Closest Point* (ICP) which was introduced in [BESL-92].

The **Model based** methods often uses an implicit criterion such as an external force. A brief overview is found in [AUDETTE-00].

The **Global similarity** methods are often used on relatively featureless surfaces where we can expect to find arbitrarily large transformations. The use of 2D *footprints* and examples of global similarity methods and can be further studied in [YAMANY-99] and [JOHNSON-97].

Matching and global optimization

This is the stage of the process that finds corresponding points or feature pairs of the two surfaces. If feature pairs are used this can be a comparison of discrete candidates, or for point and model based methods this is usually a minimization of some objective function.

When corresponding points or features are found we search for the transformation that aligns these as good as possible. The iterative methods then finds new closest points and calculates a new transformation toward convergence.

7.4.5.2 Iterative Closest Point algorithm using the Fast Marching method and Singular Value Decomposition

In this thesis the problem consists of aligning two 3D point sets, the easiest and most used method doing this is the ICP algorithm mentioned earlier which shortly can be written as… [STEIN-02]

I Find corresponding points between two sets of data, $\hat{P} = [p_1 p_2 \ldots p_m]$ and $\hat{Q} = [q_1 q_2 \ldots q_m]$, such that p_i corresponds to q_i

II Transform one set of points such that some error metric between the two sets are minimized

III Restart and continue until convergence

The first problem of finding the corresponding points is the major difficulty in the ICP algorithm. If the point sets are fairly aligned from the beginning a method based on the nearest point in the other point set is advantageous. The nearest points are found according to their euclidean distance.

Taking one point $p \in P$ and searching for the closest point in the other pointset Q is very exhausting if it's done for every point in P and for each iterative step, IMN if we have M points in the point set P and N points in the point set Q and I iterative steps. One solution is to pre-compute a distance map for Q and using that as a lookup table for each step. This distance map would only have to be computed once defining a distance-cube in space where each cell would contain a pointer to it's nearest point in the point set Q and the distance to this point. The distance is not strictly necessary except during the computation, it's the pointer that's interesting. Another method to speed up the computation is the use of K-D search trees which can be studied in [FRIEDMAN-77], but as we assume that the model point set Q is static we use a distance map which then can act as a very quick lookup table for every position of the point set P. The downside is that this distance map can be quite memory intensive for large datasets.

So if we have the point set P in a certain orientation totally enclosed in the distance cube computed for Q we can directly look up the nearest points and thus the lists $\hat{P}$ and $\hat{Q}$ of point correspondences are given.

This precomputation of the distance map can of course also be computationally expensive for large datasets. We don't want to wait unnecessary long time for it to finish. There is a smart method to efficiently compute the distance map called Fast Marching Method described in [SETHIAN-99]. This method uses a growing surface concept to compute the distances. A brief description of the algorithm follows…

I Initialize all points of the distance map to ∞ distance

II Initialize those points that belongs to the point set with 0 distance and itself as pointer to closest point and add them to a priority queue sorted on distance.

III Lift the point p with shortest distance from the priority queue.

IV For all p's 26 neighbors n, check if the distance d is ∞. If $d = \infty$ update it's distance to the euclidean distance to p's nearest point and set n's nearest point to p's nearest point. Add n to the priority queue.

If n's $d \neq \infty$, check if the distance to p's nearest point is closer than n's current closest distance. If so update with new distance and nearest point. Do not add n to the priority queue.

V Restart from III until priority queue is empty.

Using the Fast Marching Method on Q to obtain the distance map to be used in registration of P on Q, we get the lists $\hat{P}$ and $\hat{Q}$ of correspondences where p_i corresponds to q_i. Visualization of the distance map can be seen in figure 7.22.

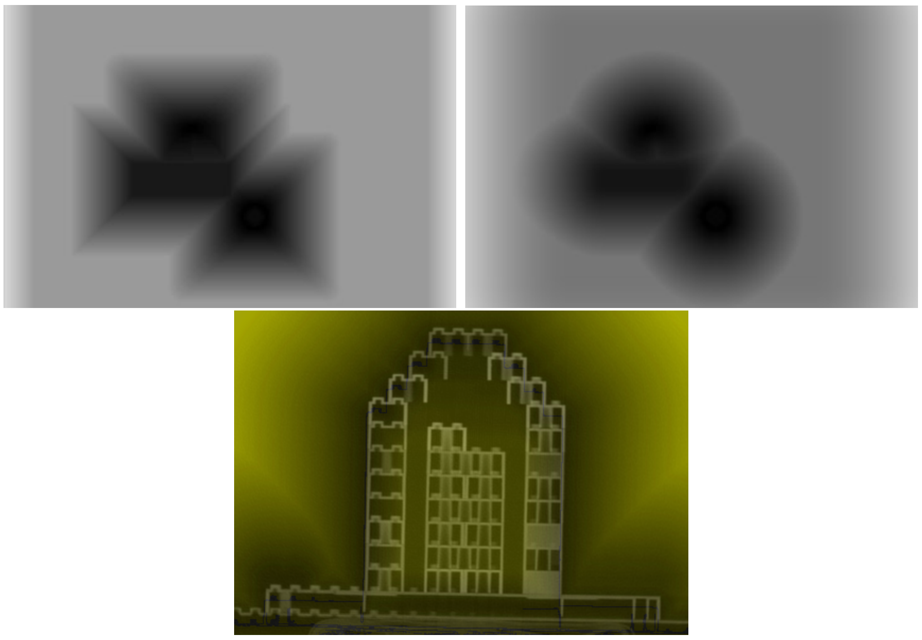


Figure 7.22: First row visualizes one of the slices in the distance map cube. Darker color mean a closer distance to the surface. Two methods are supported by the Peru application. One implements the distance as steps in number of pixels (top left) which is faster than exact euclidean distances which is showed in the top right figure. But as this is only done once the exact implementation is preferable. The example comes from a 3D-Studio scene that can be seen in figure 7.23 where the right image has nearly the same viewpoint as these distance maps. Bottom image shows a similar visualization for the lego model where the CT slice has been superimposed. The distance map also contains pointers to the nearest points on the surface, this is not easily visualized and is therefore not included in these visualizations.

When the correspondence lists has been set up we want to find the transformation that minimizes the distance between the point sets. Typically used methods doing this are quaternions or Singular Value Decomposition (SVD). The quaternions approach is used in [BESL-92], and SVD in [ARUN-87]. SVD is used to find the best transformation according to minimization of the distance between the corresponding points. The full mathematical description of the Singular Value Decomposition can be found in books about numerical analysis as for example in [CHENEY-96]. The reason that SVD was chosen is that several papers has found it to be the most stable one compared to the quaternion approach, see for example [LORUSSO-95].

Our problem can be set up using the matrices $\hat{P}$ and $\hat{Q}$ which is the lists of corresponding points of the two point sets as earlier described and R which is the transformation matrix. The equation where we want to find R such that we find the minimal solution is...

$$\hat{P}R = \hat{Q} \tag{7.38}$$

First we use SVD on $\hat{P}$...

$$\hat{P}_{m\times 3} = U_{m\times m} D_{m\times 3} V_{3\times 3} \tag{7.39}$$

where...

$$UU^* = I \text{ and } VV^* = I \tag{7.40}$$

where $U^* \equiv \bar{U}^T$ is the adjoint matrix, which is as noted the transposed conjugate matrix. In this application we do not deal with complex values but to be correct this notation is used. The matrix D is a diagonal matrix which will contain the singular values $\sigma_1 \dots \sigma_n$ of $\hat{P}$ on the diagonal. They are the nonnegative square roots of the eigenvalues of $\hat{P}^*\hat{P}$. Depending on the implementation (or arbitrary choices in the performance of the SVD) there can be differences in the ordering of $\sigma_1 \dots \sigma_n$ hence a matrix can have several different singular value decompositions. In our case we have $n = 3$.

We state what we want to minimize...

$$\begin{aligned} \rho &= \inf_R \|\hat{P}R - \hat{Q}\|_2 \\ &\underset{\text{by eq. 7.39}}{=} \inf_R \|UDVR - \hat{Q}\|_2 \\ &= \inf_R \|U^*(UDVR - \hat{Q})\|_2 \\ &= \inf_R \|DVR - U^*Q\|_2 \end{aligned} \tag{7.41}$$

If we now let...

$$c = U^*\hat{Q} \text{ and } y = VR \tag{7.42}$$

we see that...

$$\rho \underset{\text{by eq. (7.41)}}{=} \inf_y \|Dy - c\|_2 \tag{7.43}$$

so…

$$\|Dy - c\|_2^2 = \sum_{i=1}^{r} (\sigma_i y_i - c_i)^2 + \sum_{i=r+1}^{m} c_i^2 \tag{7.44}$$

is minimized by setting $y_i = \frac{c_i}{\sigma_i}$ for $1 \leq i \leq r$ and $y_{r+1}, \ldots, y_n$ can be chosen arbitrary. Thus we have…

$$\rho = \sqrt{\sum_{i=r+1}^{m} c_i^2} \tag{7.45}$$

Of all the y vectors that gives ρ we can chose the one with $y_{r+1} = \cdots = y_n = 0$ which is given by…

$$y = D^+ c \tag{7.46}$$

where D^+ is the pseudoinverse of D which here can be written as…

$$D_{m\times n} = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix} \longrightarrow D^+_{n\times m} = \begin{bmatrix} \sigma_1^{-1} & & & & & & \\ & \sigma_2^{-1} & & & & & \\ & & \ddots & & & & \\ & & & \sigma_r^{-1} & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix} \tag{7.47}$$

Now we can set up the final equation to solve for R…

$$R \underset{\text{by eq. 7.42}}{=} V^* y \underset{\text{by eq. 7.46}}{=} V^* D^+ c \underset{\text{by eq. 7.42}}{=} V^* D^+ U^* \hat{Q} \tag{7.48}$$

This takes one point set to the other, but as can be observed this does not imply that R is orthogonal and thus is not a rigid motion. As we want a rigid motion we have to look at a slightly different approach. We restate our problem with the extra constraint…

Find the matrix $R_{\in \mathbb{R}^{m\times n}}$ with orthogonal columns such that the matrix $\hat{P}_{\in \mathbb{R}^{m\times n}} R_{\in \mathbb{R}^{n\times n}}$ approximates $\hat{Q}_{\in \mathbb{R}^{m\times n}}$ as closely as possible.

This problem is called *Procrustes matrix problem* after Procrustes in the greek mythology. Procrustes offered hospitality to passing strangers, who was invited on a meal and a nights rest in Procrustes very special iron bed that would fit anyone who tried it. What the visitors did not knew, was that Procrustes made the visitors fit the bed, not the bed fit the visitors. He stretched them if they were too short or cut of their limbs if they were too long.

By using the Frobenius norm, Procrustes problem is set up as minimizing the following equation [EVERSON-98]...

$$\|\hat{Q} - \hat{P}R\|_F^2 \quad \text{with } R^T R = I \tag{7.49}$$

where R^T means the transpose of R as Procrustes problem is defined in $\mathbb{R}$, and the Frobenius norm of a matrix A is defined as...

$$\|A\|_F = \left(\sum_{i,j} a_{i,j}^2 \right)^{1/2} \tag{7.50}$$

Since R has orthogonal columns we can write...

$$\begin{aligned} \|\hat{Q} - \hat{P}R\|_F^2 &= \mathrm{Tr}\hat{Q}^T\hat{Q} + \mathrm{Tr}(\hat{P}R)(\hat{P}R)^T - 2\mathrm{Tr}\hat{Q}R^T\hat{P}^T \\ &= \mathrm{Tr}\hat{Q}^T\hat{Q} + \mathrm{Tr}\hat{P}^T\hat{P} - 2\mathrm{Tr}\hat{Q}R^T\hat{P}^T \\ &= \mathrm{Tr}\hat{Q}^T\hat{Q} + \mathrm{Tr}\hat{P}^T\hat{P} - 2\mathrm{Tr}\hat{P}^T\hat{Q}R^T \end{aligned} \tag{7.51}$$

where we can see that minimizing $\|\hat{Q} - \hat{P}R\|_F^2$ is equivalent to maximizing $\mathrm{Tr}\hat{P}^T\hat{Q}R^T$. Tr is the trace. In a technique similar to the one earlier described we can also here find R by using SVD. We decompose $\hat{P}^T\hat{Q}$ using SVD...

$$\hat{P}^T_{n\times m}\hat{Q}_{m\times n} = U_{n\times n} D_{n\times n} V_{n\times n} \tag{7.52}$$

where $UU^T = I$ and $VV^T = I$ and and D contains the singular values $\sigma_1 \ldots \sigma_r$ on the diagonal. We now have...

$$\mathrm{Tr}\hat{P}^T\hat{Q}R^T = \mathrm{Tr}UDV\hat{R}^T = \mathrm{Tr}U^T RV^T D = \mathrm{Tr}TD = \sum_{i=1}^{n} t_{ii}\sigma_i \leq \sum_{i=1}^{n} \sigma_i \tag{7.53}$$

where $T = U^T R V^T$ which easily can be seen to be orthogonal $T^T T = I$. As T is orthogonal, the trace in equation 7.53 is maximized when $T = I_{n \times n}$, which is true for…

$$R = UV \tag{7.54}$$

There is an alignment issue to be taken care of before the evaluation depending on which plane the model has been sliced in. This is demonstrated on a simple 3D model made in 3D-Studio which can be seen in figure 7.23. The point set Q generated using the peru application is visualized in figure 7.24.

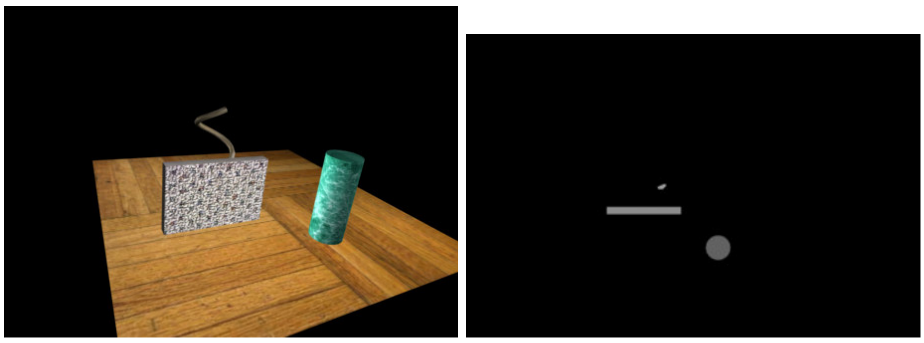


Figure 7.23: To the left a render of a 3D scene made in 3D-Studio. The scene consists of some simple textured objects. One plane (floor), one rectangular box (wall), one cylinder and one spiral standing behind the wall. The render is part of an movie made using two virtual cameras to obtain a stereo flyby of the scene. To the right one slice from the same scene. The scene is sliced top-down. The CT-slicing emulation was constructed by using an AND-operation between a plane that was sliding down the scene and the scene objects. A virtual orthographic camera was placed on top of the scene pointing down to capture the slices.

As can be seen this scene is sliced in another plane than the depth maps will be in. For the initial guess we need to transform the coordinates of the depth maps correctly to align them correctly with the point set Q. This is a very simple operation computationally but something to be aware of when starting the iteration of ICP as it assumes we are relatively near the correct orientation. The implementation places the model inside a cube and thus we take as a parameter to the program which face of the cube/model we should start iterating from. We can choose between north, south, east and west. We assume that the start position is one of these sides. We also assume that the depth maps is not upside down, i.etc. that the cameras upvector is fairly aligned with the models up orientation.

The implemented algorithm is as follows…

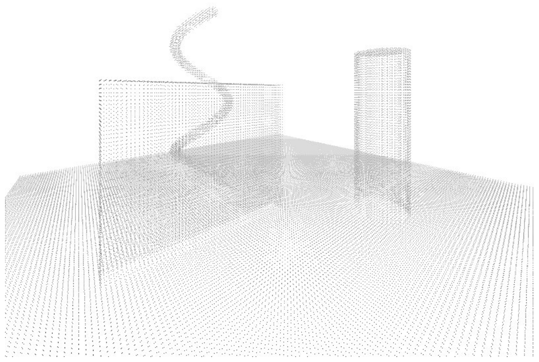


Figure 7.24: This is a render from the peru application showing the point set from a virtual camera.

I From the model dataset Q the distance map is calculated as described earlier

II Check which points in the surface P to be registered that are inside the distancemap cube and set up the correspondences $\hat{P}$ and $\hat{Q}$

III Find center of mass (CM) for $\hat{P}$ and $\hat{Q}$ and translate P toward Q the vector $\hat{Q}_{CM} - \hat{P}_{CM}$

IV Find new correspondences $\hat{P}$ and $\hat{Q}$ from this positioning

V Use procrustes algorithm to find the rotationmatrix that minimize the frobenius norm between the two pointsets and apply this rotation on P

VI Restart from II until convergence

7.4.5.3 Difficulties

One problem with the generated depth maps that make the surface registration more difficult is the discreteness. One surface with a linear change in distance will result in a stair shaped disparity map as can be visualized in figure 7.25.

Another problem regards perspective. A stereo image of a square shaped object will result in s disparity map where the square shaped object will not appear as square shaped (if we are not viewing it from right above) due to the perspective. A simple algorithm to deal with this has been tested which stretches areas of the

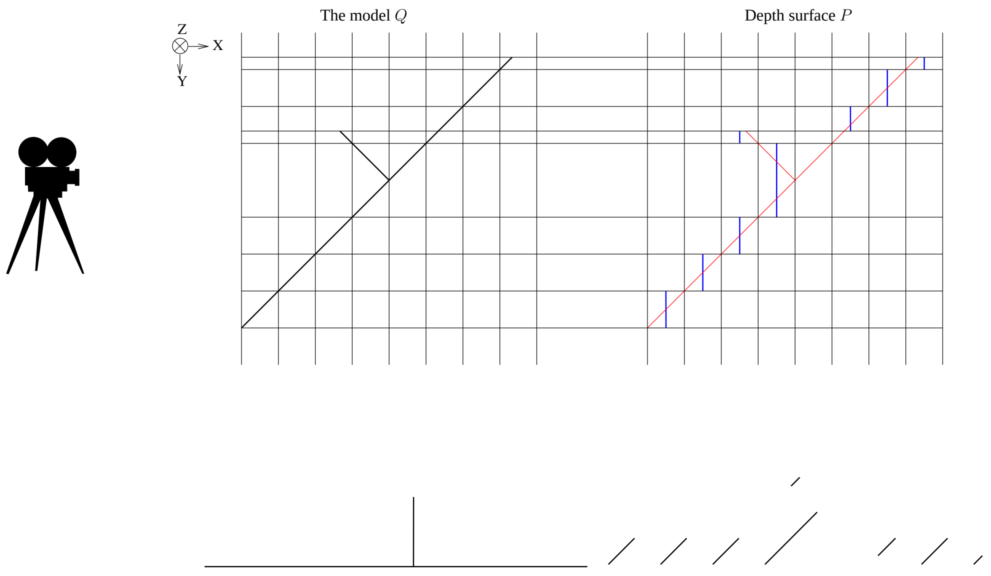


Figure 7.25: This shows one of the difficulties in trying to match the depth surface to the model. We have a stereocamera to the left observing the black tilted plane with a smaller perpendicular plane on it's upper side, this is the model noted as Q. Imagine that Q is a surface that continues into the picture along the z-axis. Depending on the resolution of the images we are able to separate between a finite number of discrete levels of distance. This levels are here indicated by vertical lines. Thus the upper right image is the generated depth map P in blue of the scene with the model Q marked in red. Thus the problem is to match a discrete surface like P to the right to the model Q. Below is Q and P rotated such that Q's lower plane is aligned along the x-axis. This shows that there are many ambiguous orientations that could arise where the depth surface finds a match along it's diagonal lines instead of through them, which would be the correct match in this case.

disparity map depending on the disparity. Small disparities will be stretches less than large. This algorithm is visualized in figure 7.26. This algorithm is very simple and does not work in all cases, for example it is very noise sensitive as dark noise will expand.

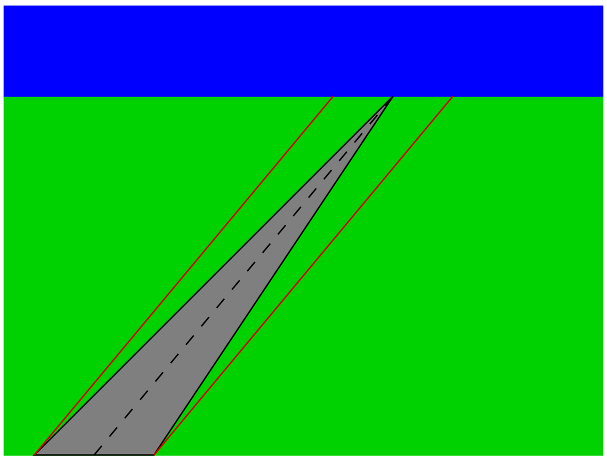


Figure 7.26: The perspective correction would in this example of a road stretch areas of same distance from the camera with a factor x depending on the distance. Areas far away from the camera would be stretched the most while areas closest to the camera would be stretched very little. In this figure the original road would be extended to the red borders.

7.5 Optical tracker approach

Another simpler approach to find the position and orientation of the camera relative the surface extracted CT dataset is the use of a tracking system that constantly measures the orientation and position of the camera. This approach assumes that the object recorded is not moved relative the tracker coordinate system (tcs). If it moves a correction has to be made, see section 7.6.

7.5.1 The lego model

The Peru application reads the CT slices and threshold them at an user specified value. The dataset is now represented as a binary cube. Visualizing all the points in this binary cube is not very interesting. We want to segment the dataset such that we can visualize only the hidden part. For the segmentation we could use specialized algorithms such as those described in section 7.2, but for the simple model the Peru application supports a rangeread of the slices, see section 7.2.1, such that we can read a subcube from all slices.

The points found in the surface extraction algorithm of the segmentation are then directly placed in an Open Inventor/Coin environment or are tessellated using for example the marching cubes algorithm.

In the first phase of the project we are using a model which depicts the medical problem. The model consists of layers, one outer layer and one hidden inner layer, see Chapter 6. The purpose is to overlay a video of the complete model with the not visible inner layer of the model.

The system is set up as follows: The optical tracking system Flash Point (Image Guided Technologies Inc., Colorado, USA) (figure 7.27) consists of three optical sensors (IR cameras) that track three infrared emitters positioned on a probe. The probe is mounted on the camera thus giving us the position and orientation of the camera. The information that is received is the position plus one normal and one transversal vector.

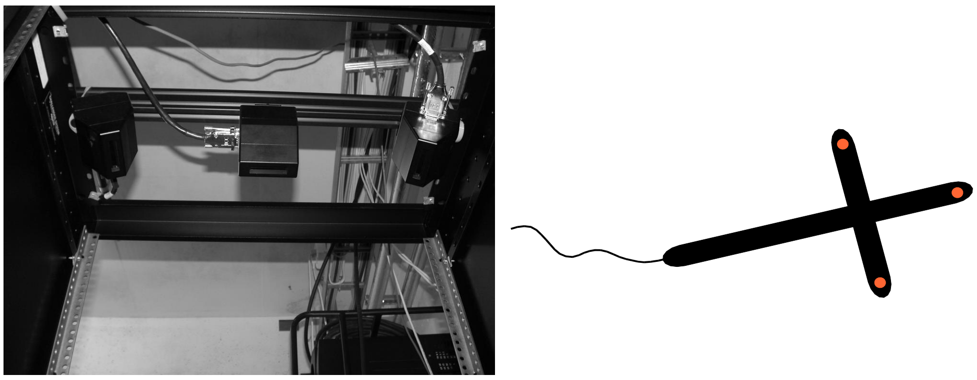


Figure 7.27: Left: The flashpoint rig with it's three infrared cameras. Right: The T-shaped LED probe that the cameras monitors.

Thus three vectors of interest can be obtained, the direction in which the camera is pointing, the up vector (the orientation in which the top of the camera is pointing), and the location of the camera. One fourth vector of interest is the x-direction of the image plane and is easily found by the crossproduct of the camera direction and the up vector.

The developed application receives the coordinates of the camera and places a virtual camera in this position and orientation in an Open Inventor/Coin environment.

The video from the camera that will show the actual model is textured using direct OpenGL on a far plane in the scene. The distance to this plane should be as far as is necessary to not intersect with the objects in the scene. The position of the corners of the plane are found by using the camera coordinate system to place it perpendicular to the camera view. The size is determined by simple geometry such that the plane fills the camera view.

$$\begin{aligned} \text{Width} = &\quad 2D\tan\left(\frac{\phi A}{2}\right) \\ \text{Height} = &\quad 2D\tan\left(\frac{\phi}{2}\right) \end{aligned} \tag{7.55}$$

In equation 7.55 D is the distance from the camera to the plane, ϕ is the height angle of the camera view and A is the aspect ratio.

To place the plane aligned with the camera orientation we need the camera up vector U and the direction vector of the camera view V. OpenInventor do not give these directly for the virtual camera and thus it must be calculated from the displacement from an original position using the following equations which is a simplified version of the rotationmatrix round an arbitrarty vector…

$$\begin{aligned} \boldsymbol{V} = \; & (1 - \cos(\theta)) * \text{axis}[0] * \text{axis}[2] + \sin(\theta) * \text{axis}[1], \\ & (1 - \cos(\theta)) * \text{axis}[1] * \text{axis}[2] - \sin(\theta) * \text{axis}[0], \\ & \cos(\theta) * (1 - \cos(\theta)) * \text{axis}[2] * \text{axis}[2] \end{aligned} \tag{7.56}$$

$$\begin{aligned} \boldsymbol{U} = \; & (1 - \cos(\theta)) * \text{axis}[0] * \text{axis}[1] - \sin(\theta) * \text{axis}[2] \\ & \cos(\theta) + (1 - \cos(\theta)) * \text{axis}[1] * \text{axis}[1] \\ & (1 - \cos(\theta)) * \text{axis}[2] * \text{axis}[1] + \sin(\theta) * \text{axis}[0] \end{aligned} \tag{7.57}$$

Here the camera is rotated Θ around the coordinate axis axis from an original position. $*$ is used as multiplication sign to improve readability.

Shaking and jiggling of the image plane prevented direct use of the received camera vectors.

The scene is now using the tracker coordinate system, and thus we need all our coordinates in this coordinate system.

If the interior of the model (the hidden part of the dataset that we want visualize) is now placed inside the scene, the far video plane is overlaid with the interior at the correct position as pictured in figure 7.28. The setup with the model can be seen in figure 7.29.

7.5.2 Position correlation

To be able to place the hidden object into the scene we need to transform the object coordinates (ocs) to the tracker coordinate system (tcs) which is received from the flashpoint system. The object coordinates is the coordinates we get when we place the origo in the upper left hand corner of the first CT slice with the x-axis extending to the right and the y-axis extending downwards and the z value as the slice number. This transformation is found using the following procedure…

Four markers are placed on the object in ocs coordinates, and this is achieved using the Peru application, see figure 7.30. These markers are saved to file together with the datastructure that holds the model. Localization of these markers in the

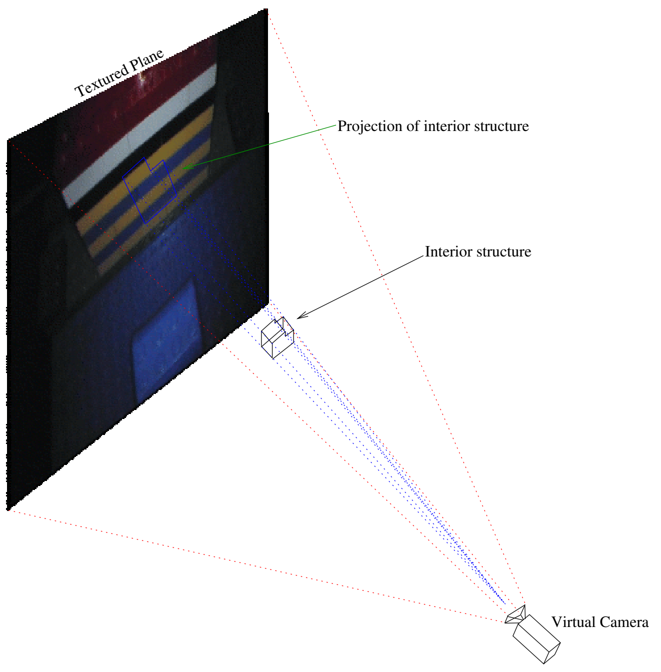


Figure 7.28: The Open Inventor scene with the textured plane in the background, and the part of the CT data that we want to visualiza as a scene object. A virtual camera is continuously updated with the orientation and position of the real camera with the help of the tracking system. The output from the virtual camera is then used as output to a TV screen or monitor.

tracker coordinate system is done using a tracked pointer that is pointed at the corresponding markers on the real model one at a time. It's important to point out the markers on the model in correct order.

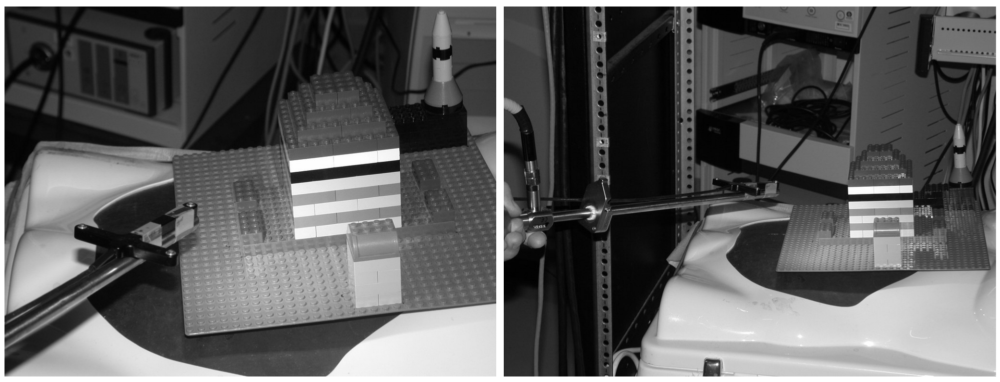
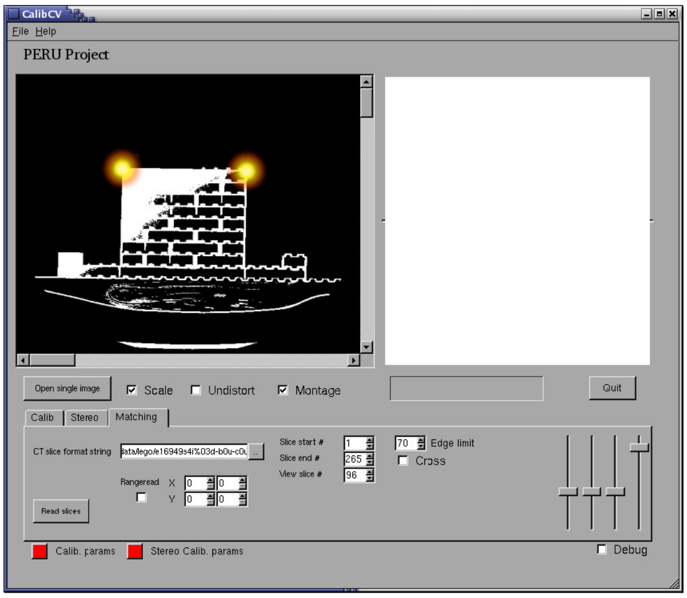


Figure 7.29: The setup for the model, the camera/endoscope is in this stage hand held and the T-formed black object in the front is the probe that the flashpoint system tracks. The tracking is made on the small bright dots on the probe which is infrared emitters.



[✓] Scale [ ] Undistort [✓] Montage

CT slice format string: data/lego/e16949s4i%03d-b0u-c0u

Rangeread [ ] X 0 0 Y 0 0

Slice start # 1

Slice end # 265

View slice # 96

70 Edge limit

[ ] Cross

[ ] Debug

Figure 7.30: Markers are placed on the model, here exaggerated in yellow to enhance visibility in the figure.

Four markers in both ocs and tcs coordinates makes it possible to find the coordinate transformation matrix from ocs to tcs by solving three equation systems. The following must hold…

$$\boldsymbol{V_{ocs}} * T = \boldsymbol{V_{tcs}} \tag{7.58}$$

Expanding equation 7.58 for one of the four markers using homogenous coordinates we get…

$$\begin{bmatrix} \boldsymbol{M1_{ocs_x}} & \boldsymbol{M1_{ocs_y}} & \boldsymbol{M1_{ocs_z}} & 1 \end{bmatrix} * \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{M1_{tcs_x}} \\ \boldsymbol{M1_{tcs_y}} \\ \boldsymbol{M1_{tcs_z}} \\ 1 \end{bmatrix} \tag{7.59}$$

Using all the markers we get the following systems…

$$\begin{bmatrix} \boldsymbol{M1_{ocs_x}} & \boldsymbol{M1_{ocs_y}} & \boldsymbol{M1_{ocs_z}} & 1 \\ \boldsymbol{M2_{ocs_x}} & \boldsymbol{M2_{ocs_y}} & \boldsymbol{M2_{ocs_z}} & 1 \\ \boldsymbol{M3_{ocs_x}} & \boldsymbol{M3_{ocs_y}} & \boldsymbol{M3_{ocs_z}} & 1 \\ \boldsymbol{M4_{ocs_x}} & \boldsymbol{M4_{ocs_y}} & \boldsymbol{M4_{ocs_z}} & 1 \end{bmatrix} * \begin{bmatrix} a \\ d \\ g \\ t_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{M1_{tcs_x}} \\ \boldsymbol{M2_{tcs_x}} \\ \boldsymbol{M3_{tcs_x}} \\ \boldsymbol{M4_{tcs_x}} \end{bmatrix} \tag{7.60}$$

$$\begin{bmatrix} \boldsymbol{M1_{ocs_x}} & \boldsymbol{M1_{ocs_y}} & \boldsymbol{M1_{ocs_z}} & 1 \\ \boldsymbol{M2_{ocs_x}} & \boldsymbol{M2_{ocs_y}} & \boldsymbol{M2_{ocs_z}} & 1 \\ \boldsymbol{M3_{ocs_x}} & \boldsymbol{M3_{ocs_y}} & \boldsymbol{M3_{ocs_z}} & 1 \\ \boldsymbol{M4_{ocs_x}} & \boldsymbol{M4_{ocs_y}} & \boldsymbol{M4_{ocs_z}} & 1 \end{bmatrix} * \begin{bmatrix} b \\ e \\ h \\ t_y \end{bmatrix} = \begin{bmatrix} \boldsymbol{M1_{tcs_y}} \\ \boldsymbol{M2_{tcs_y}} \\ \boldsymbol{M3_{tcs_y}} \\ \boldsymbol{M4_{tcs_y}} \end{bmatrix} \tag{7.61}$$

$$\begin{bmatrix} \boldsymbol{M1_{ocs_x}} & \boldsymbol{M1_{ocs_y}} & \boldsymbol{M1_{ocs_z}} & 1 \\ \boldsymbol{M2_{ocs_x}} & \boldsymbol{M2_{ocs_y}} & \boldsymbol{M2_{ocs_z}} & 1 \\ \boldsymbol{M3_{ocs_x}} & \boldsymbol{M3_{ocs_y}} & \boldsymbol{M3_{ocs_z}} & 1 \\ \boldsymbol{M4_{ocs_x}} & \boldsymbol{M4_{ocs_y}} & \boldsymbol{M4_{ocs_z}} & 1 \end{bmatrix} * \begin{bmatrix} c \\ f \\ i \\ t_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{M1_{tcs_z}} \\ \boldsymbol{M2_{tcs_z}} \\ \boldsymbol{M3_{tcs_z}} \\ \boldsymbol{M4_{tcs_z}} \end{bmatrix} \tag{7.62}$$

Solving these three systems we get the coordinate transformation matrix T from ocs to tcs.

Transforming the points in the dataset from ocs to tcs we can then place it inside the Open Inventor/Coin environment where the camera is already placed in the same coordinate system.

Results and experiments with the optical tracker approach are presented in section 9.4.

7.6 Combined stereo and tracker approach

Both the optical tracker approach, section 7.5 and the stereo approach, section 7.4 tries to find the position and orientation of the camera relative the CT dataset. Both approaches suffers from problems.

One problem with the stereo approach is to get a good initial guess to the surface registration algorithm as the algorithm assumes that the surfaces are relatively close to each other.

One problem with the optical tracker approach is that it assumes that the objects in the scene are not moving relative to the tracker coordinate system (tcs).

A combination of the two techniques could help both of this problems. The initial guess of the surface to be registered could be taken from the tracking system. And if the objects has moved a small distance relative tcs the stereo algorithm should be able to handle that.

Chapter 8

The implemented application and its GUI

An application that integrates the earlier discussed parts was developed. This chapter is devoted to describe it's various parts.

Nearly all of the earlier described algorithms has been implemented in this application with the exception of some coding (mostly the OpenGL/Coin/Open inventor code) that had to be done on a SGI (Silicon Graphics, Inc) computer which received the videostream from the endoscope.

Intels OpenCV library [OPENCV] has been chosen as the main library used in the implementation of the application which is named Peru. The name Peru has it's origin from the motivation of the thesis which is to find LIMA. And where better look for LIMA than in Peru ?

Peru consists of four main classes; **CCOCV**, **CalibCV**, **Stereo** and **Matcher**. **CalibCV** is handling all the gui using trolltechs Qt [Qt]. **CCOCV** (abbr CameraCalibrationOpenCV) contains all functionality for finding camera calibration parameters, and undistort images using the found parameters. **Stereo** is the baseclass for all implemented stereo algorithms while **Matcher** contains the surface registration algorithms.

The class **Renderer** is a special implemented class for visualization of point clouds using a Raycasting algorithm. This class visualized for example all output of the surface registration algorithms (section 9.3) and the subcube segmentation that was made in section 7.2.1. Class **ImageWidget** handles all output of images to the screen which includes saving of shown image, change of colormap and scaling. Class **Pixel** is the baseclass for all Pixels/Points with three coordinates x,y,z. **FPixel** uses floating points values and **SPixel** includes information of direction

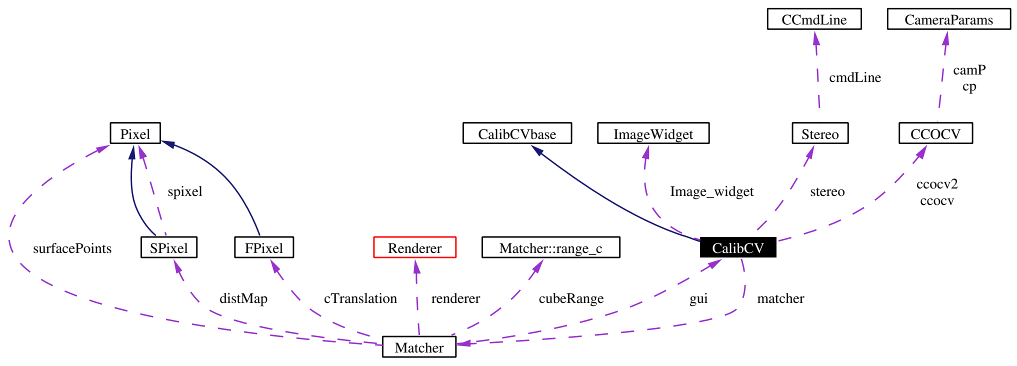


Figure 8.1: Class overview of the application with main focus on the four main classes.

used in the distance maps in the surface registration phase. There is also a class **Filter** not shown in figure 8.1 that is the baseclass for all Post and Prefilters.

8.1 Programming language and libraries

The programming language of choice was C++. The reasons for this choice was several. But what languages are there to chose between when making a scientific computer program? Let's make a short lists with brief descriptions.

- ⋆ **Fortran**
  Fortran is a shortening of FORmula TRANslation and has been around for a long while. Before Fortran most programming was made in assembler. The first version was worked on by IBM in the 1950's and many versions have come since then. As a convention all version has been named by the two last digits of the year when it's standard was proposes. Thus we have...

  – Fortran 66

  – Fortran 77

  – Fortran 90

  Fortran was constructed as a language for scientific computing, mainly because at that time the only people who used computers was physicists and mathematicians. The benefits with Fortran is that it is relatively straightforward and have good performance. The considerations is that it is somewhat antique today. Specially Fortran 77 which is the most widely used cause there are lot's of compilers for it, and there exists free compilers. Fortran 90 has corrected some of the considerations with Fortran 77, but there exists no free compilers what I am aware of for Fortran 90.

⋆ **C**
C is also a language that has been around for a while, it was created in 1971-1972. It was originally developed by computer scientists to write operating systems, for example all UNIX and Linux systems are programmed in C. But because C was not originally developed with the aim of scientific computing some features are missed. For example the arrays are quite primitive in C. But on the other hand C is a low level language which gives the programmer control to make the code extremely efficient.

⋆ **C++**
Just by the name of this language you understand that it's tightly bound to C. C++ is an extension of C that takes advantage of object oriented programming. C is a fairly simple language in the way that everything always will boil down to macros, pointers, structs, arrays and functions. C++ is much richer and includes private and protected members, function overloading, default parameters, constructors, destructors, user-defined operators, inline functions, references, friends, templates, exceptions, namespaces, and more [MEYERS-98]. This makes the language very flexible, you can always do the same task in many ways. This does also make it demanding of the programmer to know all the opportunities that the language provides. But C++ is a language that is used by millions of programmers, and resources of libraries and source code is large.

⋆ **Java**
Java is another object oriented programming language that has gain much popularity. Java is developed by Sun Microsystems and the first version saw the light of this world in 1995 which makes it a fairly new language to the earlier discussed ones. Javas portability is one of it's great advantages. The code can be run on almost every platform using a virtual machine. This portability comes at a cost however and makes Java less efficient than the other languages mentioned here. It should be mentioned though that the use of just in time compilers java can come pretty close to C++ and sometimes in certain affects even beat it. Also java is often more simple to debug, while bugs in C++ can be very tricky.

As I wanted to take advantage of the object oriented programming benefits the choice stood between Java and C++. Java I already knew, while I had little experience in C++. Java is easier to program in with it's ease of development, reusability, portability and memory management. But for computational purposes and scientific computation C++ is still the language that is mostly used due to the higher performance, and that the existing software almost extensively are programmed in C++. I also had a desire to learn something new, so my choice was C++ and this thesis also become a great course in the language.

Looking at the libraries used the most fundamental and most used is the earlier mentioned OpenCV library. I investigated some other libraries such as...

- ⋆ Gandalf `http://gandalf-library.sourceforge.net/`
- ⋆ Tina `http://www.tina-vision.net/index.php/`
- ⋆ ImageMagick `http://www.imagemagick.org/`

Both Gandalf and Tina are written in C. There seemed to be much more life and development on the OpenCV library at the time such as a nice news group and frequent updates. Otherwise Tina is an interesting library which contains a lot of useful methods, but unfortunately a bit hard to get an overview and many parts are quite old. Tina has been under development since 1986 and consists of over 150000 lines of code. I also used ImageMagick for some parts of the application at first, this though being phased out as the OpenCV library had or during the development got all of the operations I did with ImageMagick anyway. ImageMagick is more aimed at image manipulation than scientific calculations. One more important fact to chose OpenCV is that it also exist for Windows. While development is being done on a linux machine it's very nice to able to port it some day if that should be desirable.

Finally there was the question of a gui to make it easier to operate. OpenCv includes a small library for gui but it's more aimed for simple tasks such as displaying an image on the screen. My choice fell soon on the Qt library `http://www.trolltech.com` which is rapidly gaining in popularity and is a gui that is portable to many different platforms.

8.2 Layout

The layout of the Peru application is modulated in tabs such that it's easy to separate the different parts and thus disable or enable different functionality easy if that should be desirable. Currently there are three tabs, calibration, stereo and matching shown in figure 8.2.

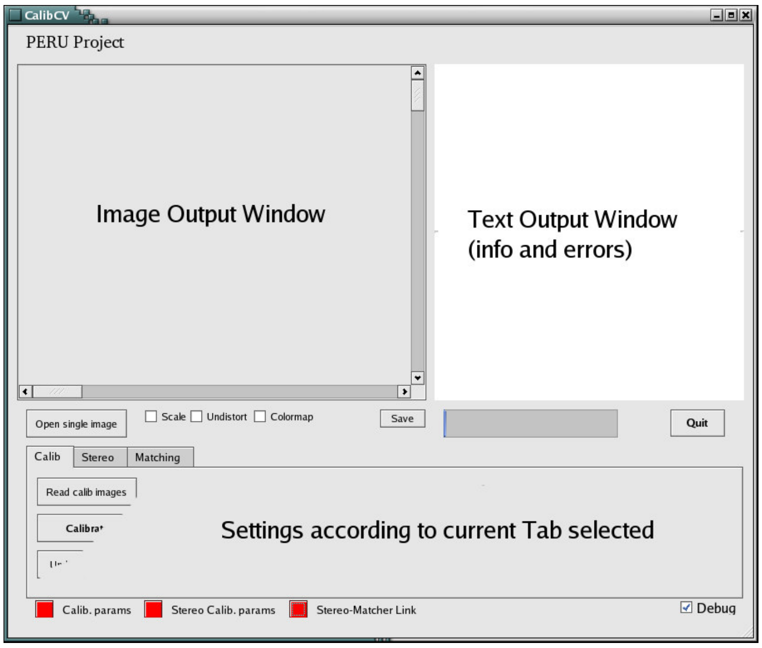


Figure 8.2: Overview of the graphical user interface for the implemented tool Peru.

Except for the tabs there are two main windows, one for text output in form of info and errors, the other one shows image output. There are also a progressbar that indicates the progress of most operations. Under the image output windows there are some controls for it. *Undistort* undistorts the current displayed image with the current undistort parameters. *Scale* scales the image to fit in window.

There are also some status leds at the bottom to indicate the state of different parts of the program. As of now there are three leds indicating if parameters for the undistort method has been specified and if a so called Stereo-Matcher link has been established. This link tells that the output from the stereo tab should be used as input in the matcher tab, in other words the output disparity maps from the stereo algorithms is directly sent to the input of the surface registration algorithm.

Help in short format is available for all of the features by hovering the mouse cursor over it.

8.3 Maintainability

The application has been written such that additions and separations will be as painless as possible.

Separability:
The three tabs contains functionality implemented as three different classes and is thus easy to separate if desired. The gui is one class that talks to the other classes. The other classes talks minimally to the gui such that they easily can be connected to some other gui. The only communications toward the gui are info messages to the text output window.

Additions:
Additions of new stereo algorithms are easy, you only have to make a subclass of **Stereo** and implement calculateDisparity() that operates on the images named left and right. The output should go into the image dispI and that's it. Of course you also have to add the algorithm to the gui such that it may be chosen. One detail to have in mind is that the Stereo super class takes two parameters to it's constructor which is *int argc* and *char** argv*. This is because the Stereo super class easily can be used as a commandline driven program if separated from the gui. If you are just adding a new stereo algorithm this is nothing to think about, but the interface that uses the **Stereo** super class must know of this.

Prefilters and postfilters are also easy to add as it's built up in the following way: One abstract class **Filter** is used as super class for each specific filter. The stereo class then has two Filter vectors that can be filled with wanted filters. One vector for pre filters and one for post filters. Adding a new filter is as simple as to subclass **Filter** and implement the pure virtual method *apply(IplImage* image)* with your filter. Then add functionality to the gui class to put that filter in the appropriate filter vector.

Chapter 9

Results

This chapter presents the results from the different parts of the thesis in tables and figures. A discussion of the results is found in Chapter 10. Some results are presented in other forms earlier in the thesis and are being referenced to.

9.1 Calibration

The computational performance of the camera calibration step is not of high importance due to the fact that it only has to be done once for each camera. But the stereo images should be undistorted continously. The grabbed images from the endoscope has a resolution of 720x288 pixels. The performance of the undistort implementation was measured to be about 16 fps on a 2.4GHz machine, this was measured when first the left and then the right frame was undistorted. The undistort procedure is of course very simple to parallelize such that the left and right channel is undistorted simultaneously by two CPU's. But more importantly the process as here measured uses a function to undistort the images, this function defines a mapping which a lookup table could take advantage over. The lookup table approach would probably give the undistort process a major speedup.

Another aspect of the calibration is the rotation correction. Examples of the output from the implemented code follows. . .

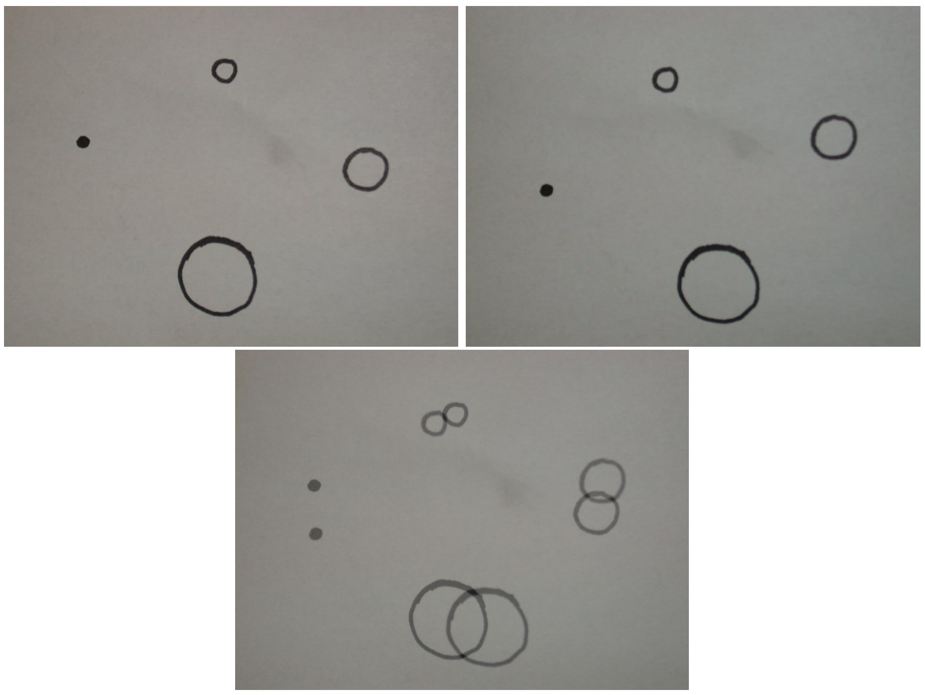


Figure 9.1: First Row:Two images taken by a hend held Nikon 995 digital camera of a hand drawn test pattern. Second Row: The images overlaid to visualize the difference between them.

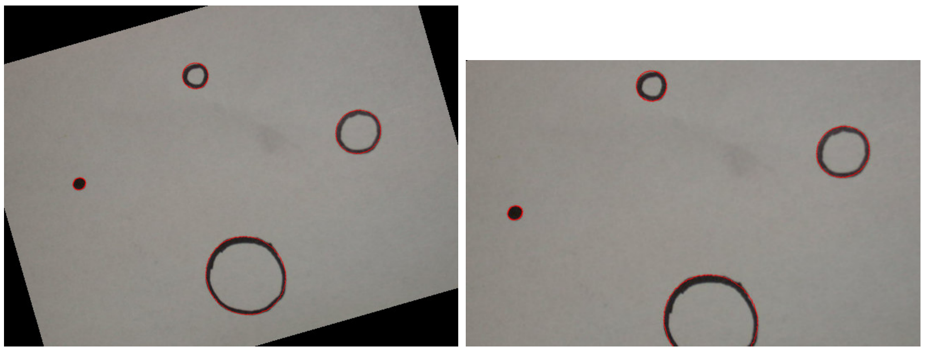


Figure 9.2: The left image shows one of the images rotated by the implemented algorithm to match the other image with the resulting black areas. The right image shows the same image after cropping it according to section 7.3.2. The found ellipses are marked with red color by the application.

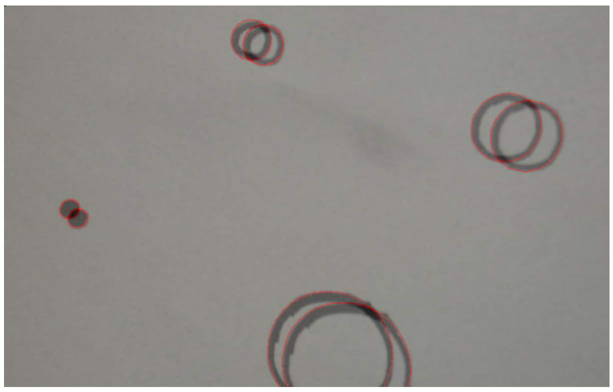


Figure 9.3: Both the cropped and corrected images overlaid. This is without translation correction.

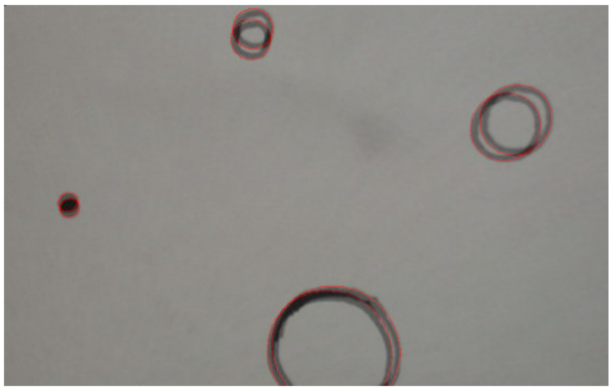


Figure 9.4: Both the cropped and corrected images overlaid. This time with translation correction. As the camera was handheld we seem to have some perspective or scale difference and thus it's not possible to align all circles perfectly.

9.2 Stereo Algorithms

This section will present the results of the stereo algorithms. All tests are performed on the Tsukuba test scene that can be seen in figure 9.6. The scene has been used since ground truth is available, thus making error measurement simple. The tsukuba scene also includes both textured, untextured and smooth areas.

The images from the endoscope turned out to be very hard to work with as the quality was poor. It should be mentioned that the signal was analog and that both left and right channel was squeezed into one stream using each other image line as left and right channel respectively thus reducing the resolution of the images in y-direction by half. Examples from endoscopic images of the lego model can be seen in figure 9.5.

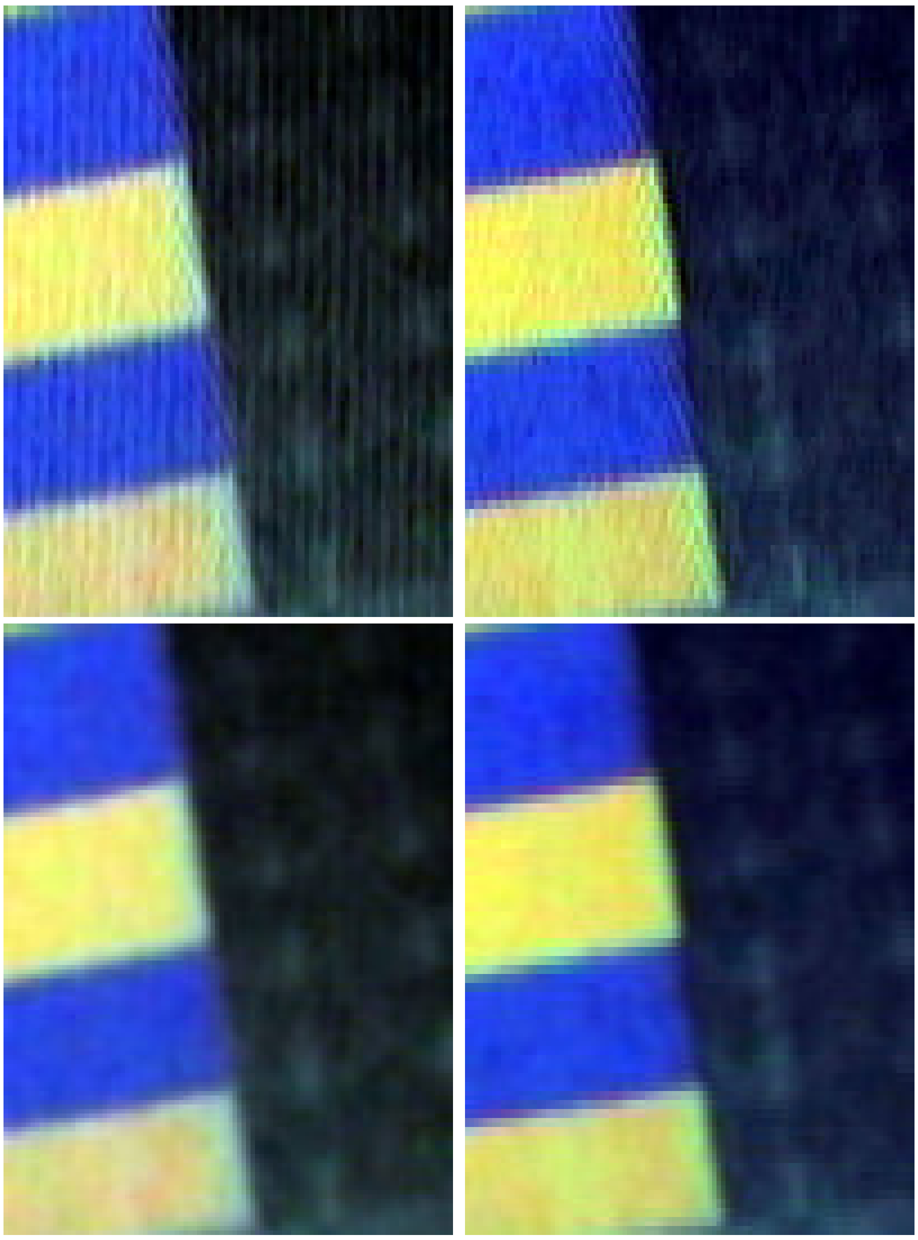


Figure 9.5: Top left is channel 0 from the endoscope where vertical stripes can be seen. Top right is same part of the image for channel 1. The images has been histogram equalized but some of the colors are still not very similar. Also some block artifacts can be seen that almost looks like jpeg compression artifacts even though the images are in RAW format. Second row shows the images after a smoothing filter has been applied that smoothes orthogonal to the stripe pattern, thus removing most of it. This though removes some of the details in the image. There is a problem with areas of same intensity which gets blob artifacts in them as can clearly be seen in both channels. Also to be seen here are some floating of the colors in channel 1, this is most easily seen in the bottom right image where the yellow bands extends further to the right than the blue. Another aspect that can be mentioned is the high demand for light that the endoscope had. The available light was very strong, actually I wore sunglasses when I worked with surface registration to not hurt my eyes. This light was sufficient to get good illumination on surfaces no more than 7-10 cm away from the endoscope. This can be seen in this image as the dark background which is the floor of the legomodel is too far away to get sufficient light from the endoscopes light. One 1000 W lamp was tried to give good illumination of the background but this gave off so much IR radiation that it interfered with the optical tracking system.

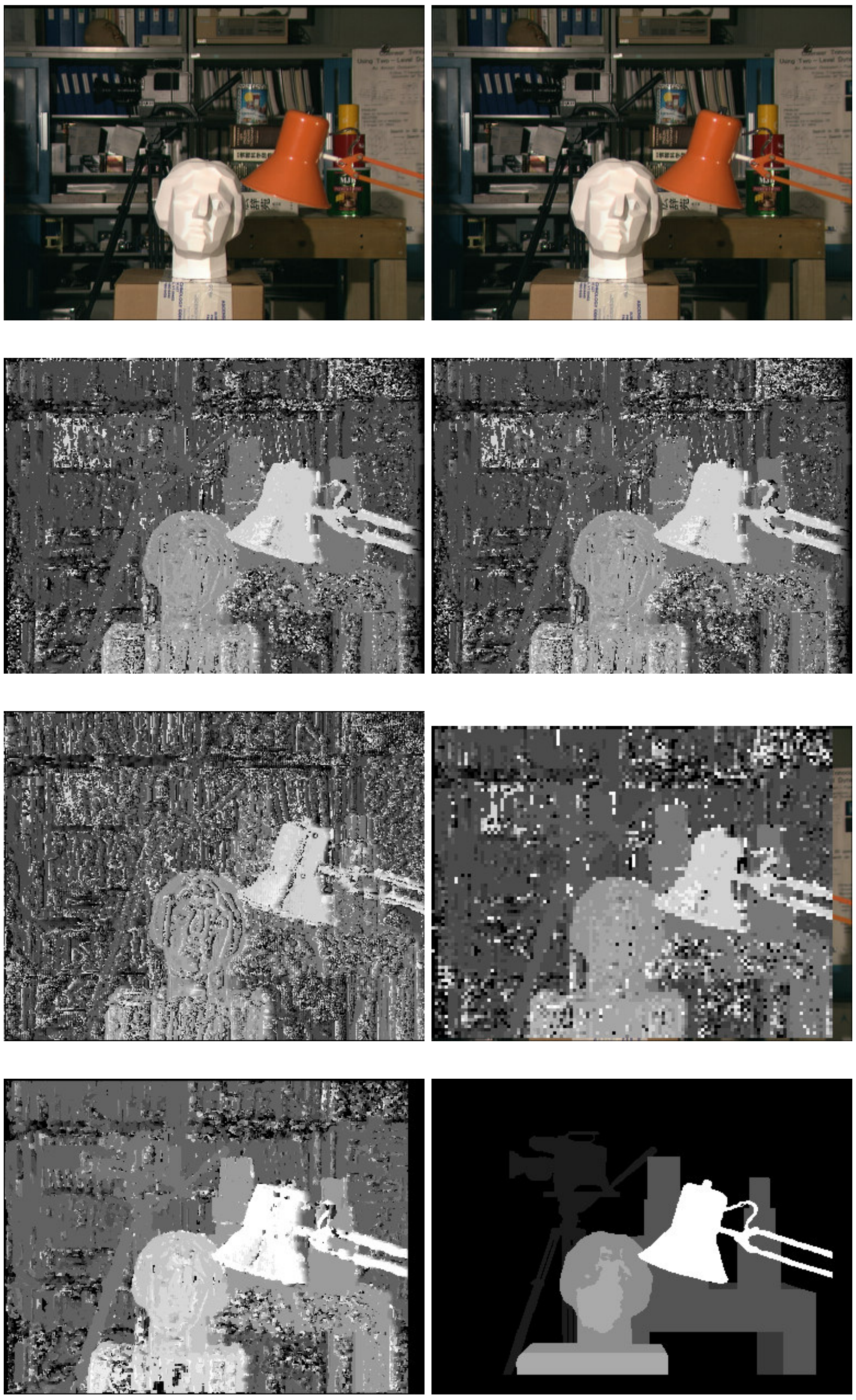


Figure 9.6: This is the output of the different blockmatching approaches used. The images (except the first row) are the disparity maps where larger disparity is shown in as brighter color implying objects closer to the camera. Note that this is without any post or preprocessing at all, and the block size $\triangle w$ is 3 for all methods to make them comparable. Top row shows left and right image of original scene. Second row to the left shows Standard blockmatching using absolute difference while the right image shows Standard blockmatching using squared difference. Third row to the left shows Fast blockmatching and to the right the Blockadvancing blockmatching method. Bottom row left shows Pyramid Blockmatching using two levels and a tolerance of 7 per level. Bottom row right shows the exact disparity map. The colors does not completely correspond as the images has been contrast stretch to lie in the interval 0-255 instead of 0-14 for visual enhancement. As can be seen in comparison to the other algorithms the pyramid method addresses much of the noise without any form of filtering.

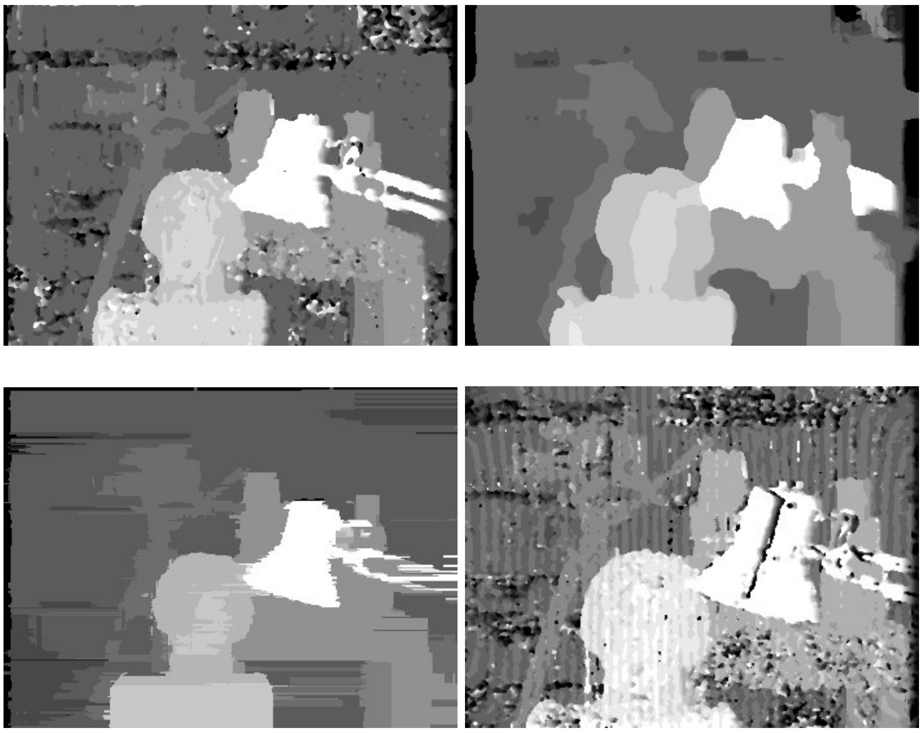


Figure 9.7:
Top left: Improved results of standard blockmatching using $\Delta w = 5$, Pre Gaussian filtering and Post Median filtering. An example of only pre gaussian filtering can be seen in figure 7.16.
Top right: Standard blockmatching using a very large window ($\Delta w = 21$) shows that noise is almost totally removed but the position of the edges are not correct. And this disparity map took 16 seconds to generate.
Bottom left: Output of the birchfield algorithm without any pre or postfiltering.
Bottom right: An example of the output an uncalibrated image pair of the tsukuba scene with barrel distortion. A striped effect appears as the scanlines bend differently at the same scene position in the frames. Same parameters as the top left image.

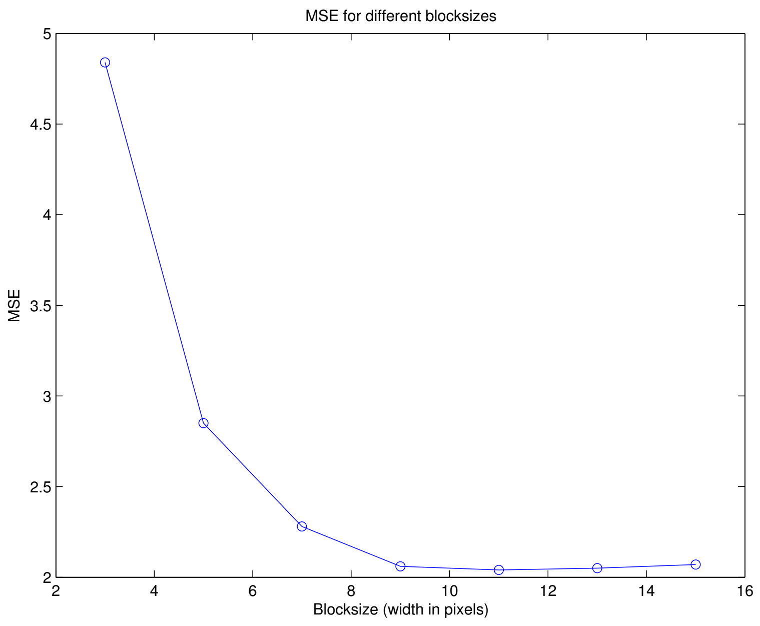


Figure 9.8: This figure shows how the mean square error changes as a function of the blocksize. The test is performed on the tsukuba scene where ground truth is available. $D_{MAX} = 14$ has been used which is the true max disparity of the scene. Standard blockmatching is used without any filtering.

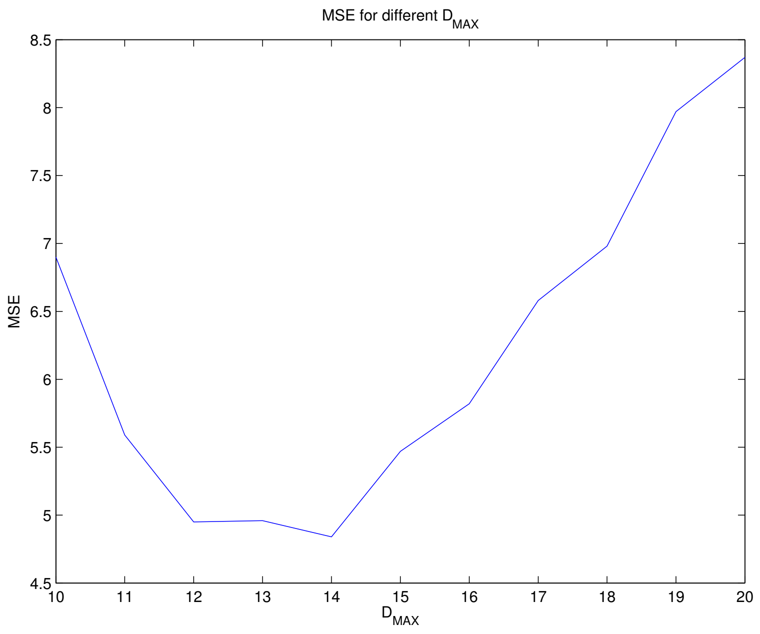


Figure 9.9: This figure shows how the mean square error changes as a function of D_{MAX}. The test is performed on the tsukuba scene where ground truth is available and the known max disparity is 14. Standard blockmatching is used without any filtering.

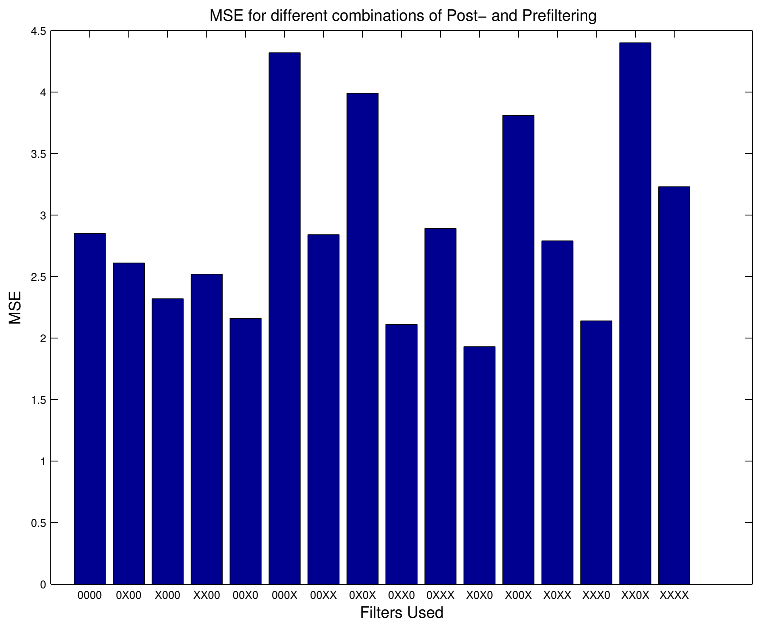


Figure 9.10: This figure shows the mean square error resulting from different combination of pre-filters and post-filters. The code under the bars should be read in the following way: There are four markers where an X stands for *filter applied* **and an O stands for** *filter not applied***. The order of the filters are: PreGaussian smoothing, Pre Medianfilter, Post Median Filter and finally Post Gradient Removal. Standard blockmatching was used on the tsukuba scene with a D_{MAX} of 14 and a blocksize Δw of 5.**

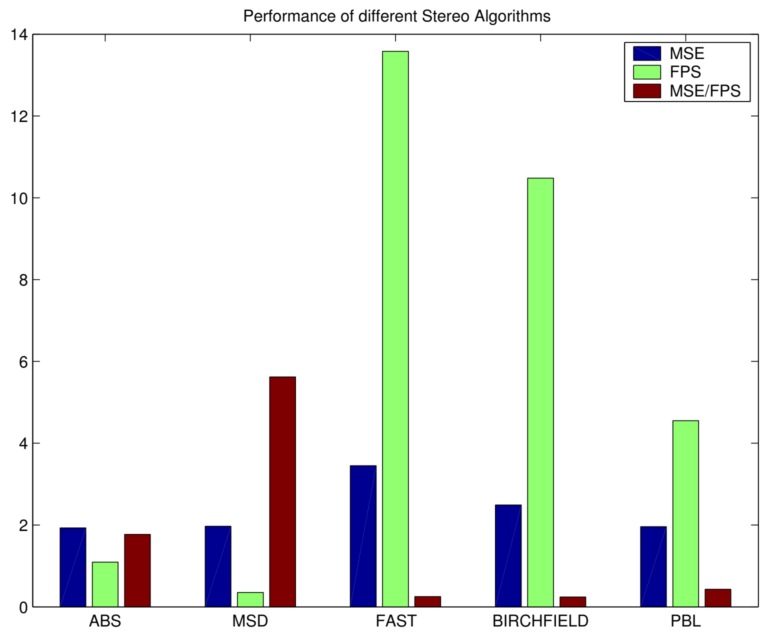


Figure 9.11: This figure demonstrates the performance of the different implemented stereo algorithm on the tsukuba scene. The performance is measured in both the terms of the mean square error and the speed of the algorithms in FPS(Frames per second). There is also included the ratio between the MSE and FPS which tells us something about the achieved quality per time unit. The abbreviations is ABS = Standard Blockmatching with minimum absolute difference between the blocks, MSD = Standard Blockmatching with min squared difference between the blocks, FAST = Fast blockmatching and PBL is Pyramid Blockmatching. Here Pre Gaussian filtering is used in combination with Post Median filtering. The algorithms are described in section 7.4.3.

9.3 Surface Registration

This section presents the results from the experiments with the surface registration algorithm.

Model	Dimensions	Total Points	Reduced Points	Iterations to convergence	Seconds per iteration	Noise	MSE per pixel
Legohouse	202,278,85	1428287	183677	44	0.14	0	0.09
Legohouse	202,278,85	1428287	183677	38	0.15	1	0.97
Legohouse	202,278,85	1428287	183677	40	0.17	3	2.89
Legohouse	202,278,85	1428287	183677	40	0.17	5	4.88
Inner Legohouse	79,175,35	201152	41404	21	0.031	0	0.68
Inner Legohouse	79,175,35	201152	41404	25	0.034	1	0.97
Inner Legohouse	79,175,35	201152	41404	22	0.036	3	2.88
Inner Legohouse	79,175,35	201152	41404	38	0.035	5	4.83
3DS model	360,240,100	122726	91765	60	0.065	0	1.27
3DS model	360,240,100	122726	91765	100[1]	0.075	1	1.12
3DS model	360,240,100	122726	91765	80	0.072	3	3.05
3DS model	360,240,100	122726	91765	40	0.055	5	5.51

Table 9.1: Registration performance. This table demonstrates both the time efficiency performance and the quality of the registration made in terms of the error. A Pentium 4 2.4GHz CPU was used on a computer with 1024MB RAM using RedHat Linux 8 with kernel version 2.4 as operating system. All the models used here are described in more detail in Chapter 6 and Section 7.4.5.2. All displacements for testing the algorithm is here 5,5,-5 degrees around x,y and z axis respectively. The test is performed such that a transformed copy of all surface points are entered to the ICP algorithm (Section 7.4.5.2) and a registration is performed. The Legohouse dataset was thresholded at threshold level 60 before the surface extracting algorithm. The surface extraction algorithm here always uses all 26 closest neighbors as the neighborhood. A range $[146, 88, 91] \rightarrow [348, 366, 176]$ was extracted from the CT scan to give us a cube containing the large legohouse. The Inner legohouse dataset was also thresholded at level 60, but with a range $[210, 185, 116] \rightarrow [289, 360, 151]$ such that only the small inner legohouse was extracted. The 3DS model is not thresholded before the surface extraction algorithm as the CT slices are binarized in the CT-emulation described in Figure 7.23. The MSE error is the mean squared distance from a point in the transformed and possibly noise distorted pointset to the correct corresponding point in the original dataset. The error measure is the mean per point.

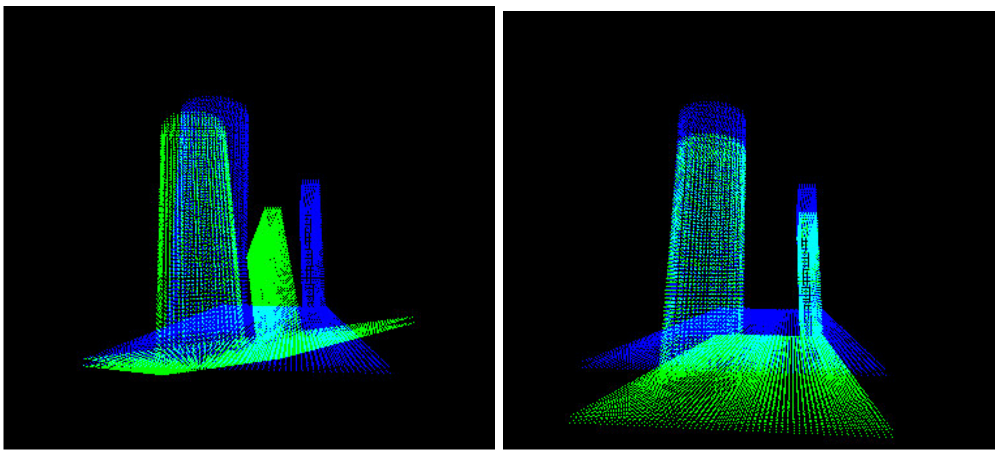


Figure 9.12: These images shows the result of 70 iterations with the ICP algorithm on a part of the 3D-Studio model. The left image shows the configuration of the datasets P and Q before registration and the left image shows the configuration of the datasets after 70 iterations of ICP. P is green and Q is blue. The cyan color comes from overlapping pixels. As can be seen the rotation retrieval is very good while the translation contains an error in the Z-axis. Because of the planar nature of these point sets almost every point has found a very close neighbor except for the points in the bottom floor. The bottom floor of P is outside the distance cube and thus not included in the calculations which explains the small MSE converged to at the position in the right image after 70 iterations. A plot of the mean square error per pixel is shown in figure 9.13. Here P is rotated 5,15 and -15 degrees around the x,y and z axis respectively. Translation is 5,10 and 10 distance units in x,y and z from correct alignment.

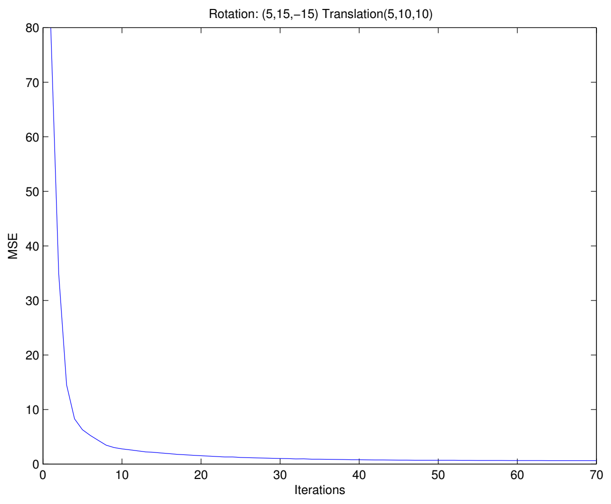


Figure 9.13: A plot over how the mean square error per pixel drops for each iteration. As can be seen the largest steps is done in the first 10 iterations and then slowly dropping until convergence at 70 iterations. See figure 9.12 for a visualization of the process.

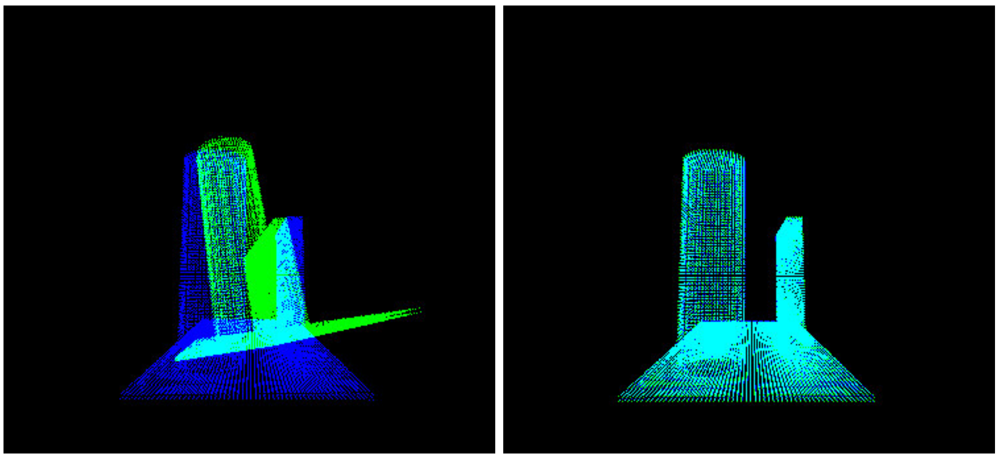


Figure 9.14: These images shows the result of 70 iterations with the ICP algorithm on a part of the 3D-Studio model. This time there is no translation difference between the two sets. Convergence is here reached in about 28 iterations, but 70 is used to be consistent with the last trial. The left image shows the configuration of the datasets P and Q before registration and the left image shows the configuration of the datasets after 70 iterations of ICP. P is green and Q is blue. The cyan color comes from overlapping pixels. A plot of the mean square error per pixel is shown in figure 9.15. As in the last trial in figure 9.12 P is rotated 5,15 and -15 degrees around the x,y and z axis respectively. But no translation is applied. The resulting alignment is here perfect.

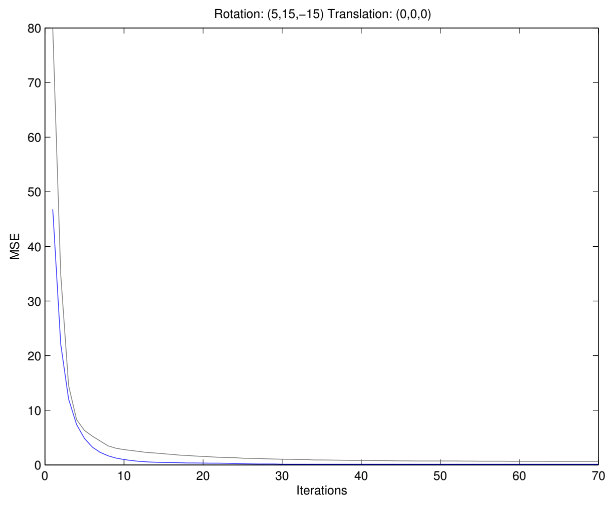


Figure 9.15: A plot over how the mean square error per pixel drops for each iteration. Convergence is here reached after 28 iterations. The plot from figure 9.13 is included here in gray as a comparison. See figure 9.14 for a visualization of the process.

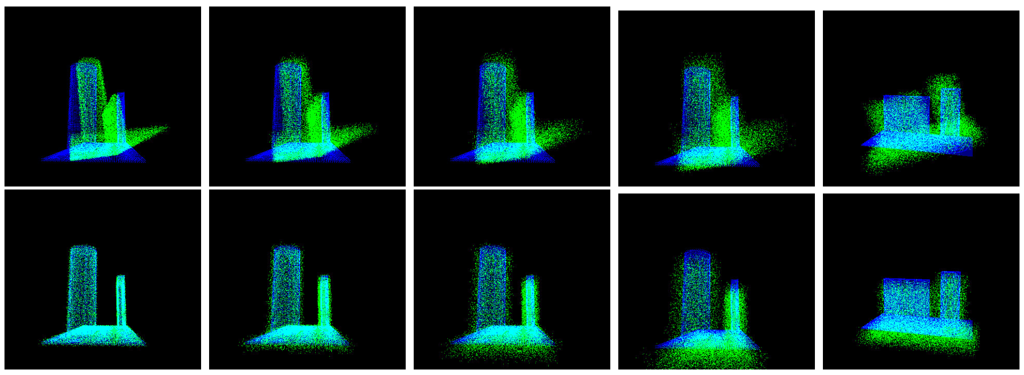


Figure 9.16: This shows a noise test on the ICP algorithm. Top rows contain the model point set Q (blue) and noise distorted and rotated copy of Q used as P (green) before ICP algorithm is applied. Bottom row shows how P has been rotated and translated by the parameters found in the algorithm. From left to right we have an addition of $\pm$ 1,3,5,8 and 10 distance units of random uniform noise. The dataset has dimensions $120 \times 56 \times 100$. Note the misplacement of P in the last two columns.

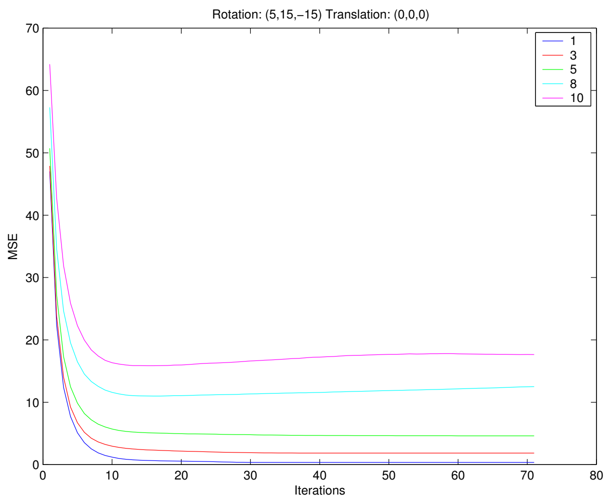


Figure 9.17: This plot shows the robustness of the ICP algorithm against noise in the surface to be registered. There is an addition of $\pm$ 1,3,5,8 and 10 distance units of random noise to each point in Q. The plot shows the mean square error per point to the point which is found as the corresponding point. Here we can see the steady convergence of the ICP algorithm, although we also note that for noise levels of 8 and 10 we get a slight increase in error from iteration 15 and higher.

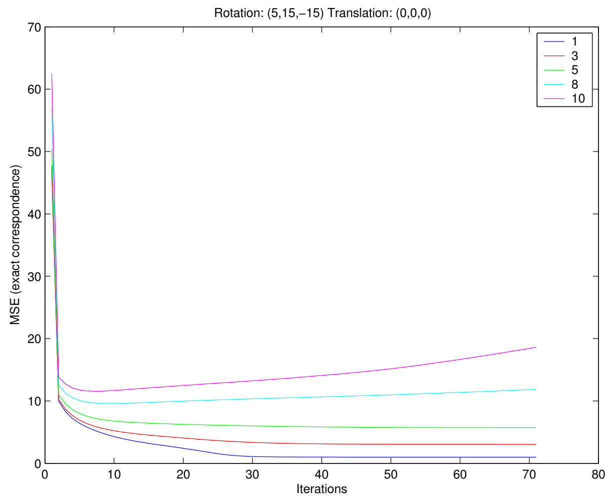


Figure 9.18: This plot shows a similar plot for the noise cases in figure 9.17, but here the error is recorded as the mean square error per point to the point known to be the correct corresponding point. This plot shows more clear that noise levels 8 and 10 has trouble converging to the correct position, they are instead pushing the surface outside as can be seen in the last two columns of figure 9.16.

9.4 Optical tracker

This section will show some example images where the interior of the legomodel is superimposed onto the videostream from the endoscope. The positioning was quite good but the system had a small delay when moving the endoscope fast by hand. Using the robot to control the camera this was not an issue as the speed of the robot is slower. Some drifting of the superimposed structure could also be noted when it was located far out in the edges of the image.

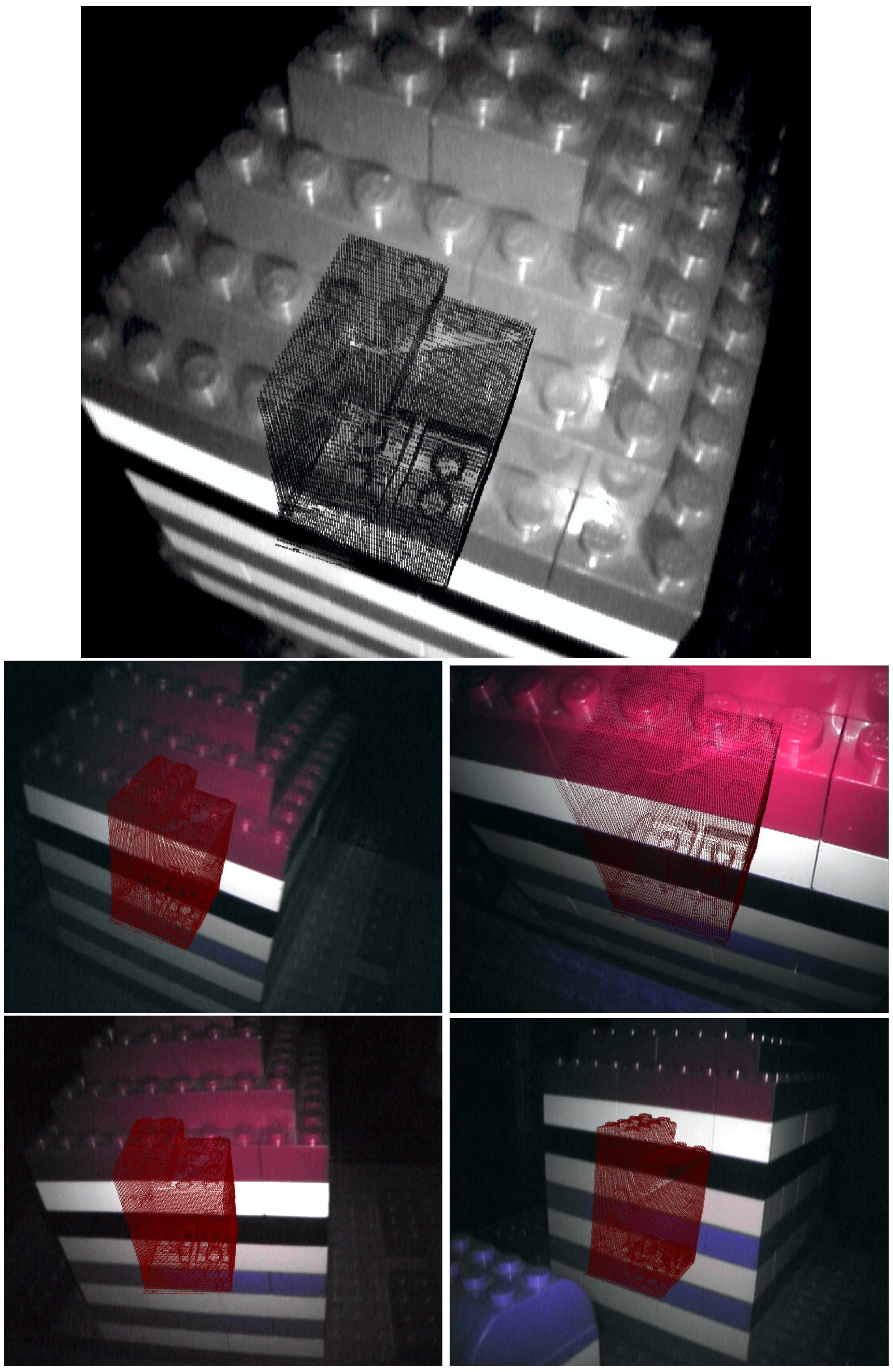


Figure 9.19: The interior of the legomodel is superimposed onto the videostream. All images has been level adjusted or contrast stretched to compensate for the poor output quality.

Chapter 10

Discussion

The output of the stereo algorithms should both be low in error and of fast performance as the application should run in realtime. The experiments presented in the Result chapter 9 has information concerning both these aspects and of special interest is figure 9.11. But first we look at the different parameters and filters that can be applied before and after the algorithm.

Figure 9.8 shows the error relative the width of the block used for the blockmatching types of algorithms. We clearly see that the error reduces as we increase the size of the blocks. But using too large blocks in fact increases the error. This increase for very large blocks come from blurring of edges. Here the problem is that as the size of the block increases we get less precision in the spatial position of the disparity, but a small block size will get good spatial precision but less good block similarity precision. It should also be noted that increasing the size of the block increases the computational effort of the algorithm significantly (quadratic dependent on the width of the block). Using a block size of five instead of three yields a much improved result, larger block size gives very bad performance in terms of time used so I suggest as a compromise to use 3, 5 or at the very most 7 as the size of the block. Further tests of the blockmatching algorithms uses a block size of 5.

Figure 9.9 looks at the choice of the max disparity to search for. The scene tested has a known max disparity of 14, and we note that using the same value of the max disparity as the true max disparity yields the best result. This is maybe not surprising as larger values makes possibility for pixels to get a disparity that is higher than possible, and a value of the max disparity that is too low of course introduces errors for all pixels that has a true disparity that is higher than the max searched for. To know the scene you are observing and thus using a max disparity that is as close to the true as possible should be the goal. Though you should make sure that you do not chose a value that is too low to avoid that some pixels are not even possible to be assigned their true disparity.

Figure 9.10 examines the use of different post and prefilters. The first thing to notice is that the eight worst combinations of filters all include the Post Gradient Removal. This added error comes from that when removing the gradients the algorithm currently replaces it with a value of 0. This is not always a good choice as the correct disparity after the edge may be far from 0. A better way to implement this filter would probably be to look at values above the current scanline and replace the removed gradient with values based on this above values. Anyway this added error that now results may not be a problem for the surface registration algorithm as pixels with a disparity value of zero is assumed to belong to some distant background and are thus not included in the calculations. In this case the Post Gradient Removal is in fact positive even though the mean square error for the total disparity map has increased. We secondly note that of the eight worst combinations, four of them are not as bad as the four worst. The difference here is that none of the four worst filters include the Post Median filter while worst filters number 5 - 8 all includes Post Median Filter. In fact every filter combination which do not include the Post Median Filter benefits from including it. We can also note that Pre Median Filter does not nearly give the same improvement as the Post version. Also the Pre Median Filter is much more expensive as it is applied to both left and right image. Pre Gaussian Filtering seems to do a better job to make the blocks match at the correct disparities. The best combination reached in the experiment is using Pre Gaussian Filter in combination with Post Median Filter. There are two filters not considered in this experiment which is the Histogram equalization and the Mean Value Correction. These filters has shown to be beneficial if the images from the two cameras differ more than slightly. The mean correction filter is very fast and can give good improvements from cameras that do not have the same sensitivity.

It should also be mentioned that no difference between the RGB space and the $I_1 I_2 I_3$ space as [KLETTE-95] had found to be a better choice was found. Though Klettes statement that using the information in all three channels improved the result by 20% to 25% was confirmed and therefore all the results in the last chapter are using all RGB channels.

Figure 9.11 finally say something about both speed and the quality according to small errors in the algorithms. The first thing to notice is that using squared difference to measure the difference between blocks in the Standard Blockmatch algorithm is not very meaningful, the performance drops a great deal while in fact the error is slightly larger which makes MSD the worst of the algorithms according to error per time (the brown bar). We also note that the fastest algorithm has the largest error. Fast blockmatching, Birchfield and Pyramid blockmatching shows a trend where faster algorithms yields larger error. Pyramid blockmatching which is not nearly as optimized as fast blockmatching could probably be done faster and thus be a good candidate for the winner among the implemented algorithms when considering both speed and low error. It's current performance of about 5 FPS on a 2.4GHz machine shows potential for realtime performance. Reaching higher speed can also be accomplished using several CPU's where a current available CPU would pick up the next stereo pair from the video stream yielding a small delay rather than a slowdown.

The surface registration part shows that most of the error seems to be reduced in about the first 10 iterations independently of the dataset used, this can be seen in figures 9.13, 9.15, 9.17 and 9.19. Looking at table 9.1 we can see that the time used per iteration can sometimes be quite large. Even though the dataset has been reduced by only using the surface points there should be potential to improvements here as it's not necessarily obvious that we need to use all the points on the surface. One idea would be to pick out feature points to use in this registration which could reduce the time used in this part significantly. The surface registration algorithm has shown to be very noise robust and has handled uniform random noise up to ± 5 pixels in the 3DS scene where the diameter of the cylinder is 20 pixels. Note also that during operation the position of the surface in the last time frame can be used as guide for initial position for the current time frame. Thus the surface is nearly aligned and the number of iterations needed is reduced.

Chapter 11

Conclusions

The output from the implemented algorithms with the endoscopic input images is not usable by it's own for good disparity maps. Chapter 12 discusses what can be done about this.

Some conclusions can be drawn from the stereo experiments, the algorithms has shown to give good quality disparity maps for calibrated images of higher quality than the stereoscopic endoscope delivers.

Good matching against the CT data relies on a disparity map that is as similar as possible to the dataset, this requires smoothing of the discrete disparity map which in turns destroys information about abrupt true changes in the disparity map. Therefore a scene consisting of a continuous smooth surface is easier to work with than the discontinuous surface like the lego model yields where you can't smooth the disparity map without destroying significant information.

The performance of the stereo algorithms has potential to be used in realtime. Also many of these has simple solutions for parallelization, for example the block-matching methods could be calculated in several threads which each operated on different parts of the image. Extensive time has not been used to top optimize the algorithms, such that it should be potential to improve them to some extent.

As earlier mentioned the surface registration phase has been shown to be noise robust. Robustness to deformations has not been tested. It seem to have more trouble with translations than rotations.

The optical tracking part has shown to be the most successful part of this thesis where a realtime application has been implemented and tested with good results. This part should be useful on it's own for applications where the observed object is not moving relative the tracker coordinate system.

Chapter 12

Future work

The stereo approach needs much more work before it can be a usable strategy for finding the correspondence between the camera and the dataset. The problem is complex and involves many steps as can be seen in this thesis. Better image quality from the cameras is needed. Use of fiducial markers or additional sensors could help in the areas of the images where the stereo algorithms are unable to find good correspondence points. Much like the principle of hierarchical block-matching (Pyramid Blockmatch), such markers or sensors could restrict disparity search in certain areas. Also the perspective problem needs to be dealt with. As the geometry of the observed object is known in advance this info could be used. Finally a smoothing method for the disparity map should be investigated, where this is easiest if the observed object is known to have a continuous surface. In the not continuous surface case this is harder as we do not want to smooth the sudden large changes in disparity, but also here the fact that we know the geometry of the observed object could be exploited.

Appendix A

Article submitted to CARS 2003

Augmented reality for safer coronary artery bypass

Lars Aurdal, Daniel Bengtsson, Ole Jakob Elle and Eigil Samset [*]

The Interventional Center, Rikshospitalet, 0027 Oslo, Norway

Abstract

The standard procedure for left anterior descending (LAD) coronary artery bypass requires a sternum split. The internal mammary artery (IMA), most typically the left one (LIMA), is dissected free from behind the sternum and is then anastomosed onto the heart below the occlusion of the LAD.

Totally endoscopic coronary artery bypass (TECAB) is a less invasive alternative to the standard procedure. This procedure results in heavily reduced invasiveness, but also leads to loss of precision, reduced force feedback and loss of overview. The purpose of this work is to alleviate the problems related to loss of overview by generating an augmented reality for the surgeons in which they are given the impression of 'seeing through' the tissues surrounding the LIMA thus making localization of the LIMA a simple matter. Using preoperative CT or MR data, the LIMA is located with respect to the sternum and the tissues surrounding it. Intra-operatively, the sternum is located and the tissues surrounding the LIMA are tracked using a stereo videoscope held by a robotic arm. Knowing the position of the videoscope relative to the sternum now makes it possible to calculate where in the videoscopic images the LIMA is located. This information is then used to generate an augmented reality showing the tissues surrounding the LIMA as transparent, thus revealing the position of the LIMA within them.

Key words: augmented reality, coronary artery bypass

[*] Corresponding author: Lars.Aurdal@labmed.uio.no,Tel.: int+47 23070100, Fax.: int+47 23070110

1 Introduction

During the standard procedure for left anterior descending (LAD) artery bypassing, access to the heart is gained through a sternum split. The left internal mammary artery (LIMA) is dissected free from where it is located behind the sternum and then anastomosed onto the heart below the occlusion of the LAD. Figure 1 shows an image of the standard procedure making it possible to appreciate its invasiveness.

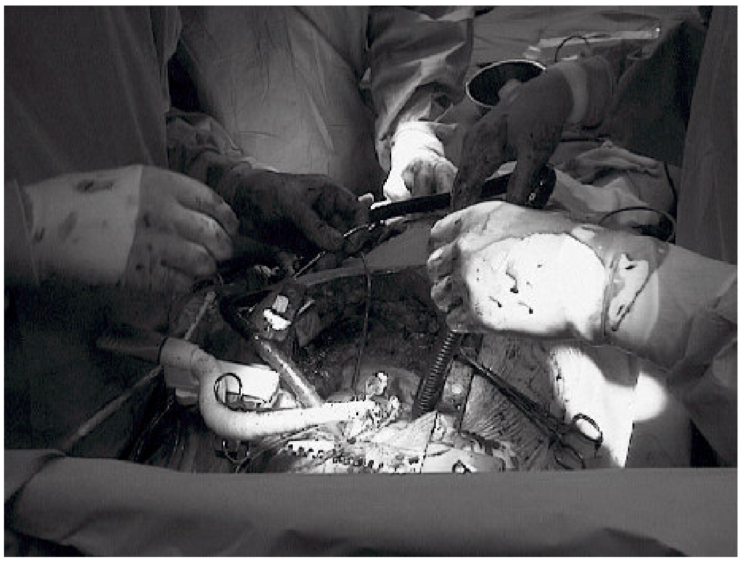


Fig. 1. Offpump coronary bypass surgery through a split sternum.

The invasiveness of this procedure has inspired much research on alternatives. Totally endoscopic coronary artery bypass (TECAB) is one such alternative where access to the thoracic cavity is gained through small incisions in the thoracic wall [1], [2], [3], [4], [7], [8]. The main advantage of this technique is the heavily reduced invasiveness resulting in reduced recovery time and less scarring. The main disadvantages are loss of precision, reduced force feedback and loss of overview. The loss of precision during TECAB was one of the driving forces behind the development of surgical robots capable of reducing tremor in the instruments as well as downscaling movements. The introduction of such robots will unfortunately completely remove all force feedback. As a consequence, palpating the tissues surrounding the LIMA, and thereby locating the LIMA through pulsations is no longer possible. Dissection and mobilization of the LIMA now become very difficult procedures.

The purpose of this work is to alleviate this problem by generating an augmented reality for the surgeons in which they are given the impression of 'seeing through' the tissues surrounding the LIMA thus making localization of the LIMA a simple matter.

2 Methods

Prior to surgery the patients thoracic cavity is imaged using CT or MR to allow for localization of the LIMA with respect to its surrounding tissue and with respect to the sternum. The LIMA and its surrounding tissues are segmented from these images by standard image segmentation methods. In addition, the coordinate system of the robots are located with respect to the patients sternum. During surgery, a stereoscopic videoscope and two instruments, all held by robotic arms and operated via an external console, are introduced into the thoracic cavity. Once introduced into the thoracic cavity, the videoscope can visualize the tissue surrounding the LIMA. Since the position of the videoscope is known and since the position of the videoscope relative to the LIMA is known it now becomes possible to superimpose a 3D image of the LIMA on the 3D image of the tissues behind the sternum. This effectively gives the operating surgeons the impression of seeing through the tissues surrounding the LIMA.

In order to allow for precise and safe dissection of the LIMA, it is obvious that the virtual image of the LIMA must depict the position of the LIMA as precisely as possible. The allowed tolerance is expected to be less than +/- 2mm in any spatial direction.

The position of the LIMA relative to the sternum is well known (from the segmented images), however, the position of the robot relative to the sternum (and thus relative to the LIMA) is much harder to establish. Primarily, this is so because the sternum moves with the respiratory movements of the patient. Secondly, the exact position of the sternum is difficult to asses even in the absence of these movements. The precision required for this procedure can therefore not be achieved only by positioning the robotic system relative to the patients sternum.

To solve this problem, a two-step method is used. The robot held camera is first positioned as precisely as possible relative to the sternum. The remaining errors in position are then removed by continuously tracking the movements of the tissues behind the sternum. The shape of this tissue is known from the CT/MR imaging step. Observing this tissue using the stereo videoscope provides two image streams that can be used to define a depth map of the observed scene. This is done by using standard image processing algorithms for depth recovery from stereo video data [6]. Once the depth map is known, the position of the camera relative to the tissues behind the sternum is determined by correlating the depth map to the shape of the same tissues as determined from the CT/MR images. This procedure is facilitated by the fact that a good estimate of the camera position relative to the sternum can be obtained by proper alignment of the robot relative to the patients sternum.

3 Results

The problem of segmenting the LIMA from the pre-operative CT or MR images is a relatively simple one to solve, primarily because of the fact that this processing does not need to take place in real time, it is done before the patient enters the operating room. The segmentation can thus be performed with manual intervention. We will therefore not consider this problem any further in this paper.

Superimposing images of the LIMA onto the videoscopic images of the tissues behind the sternum must be done in real time. To test our procedure we have made a solid model of complex shape that captures the essential challenges of the given problem. The model is shown in figure 2 and is constructed using LEGO^{TM}pieces. As the figure clearly shows that the model is two-layered, consisting of an outer shell enclosing an inner structure thus simulating the LIMA embedded in its surrounding tissue. As the last part of this figure shows, we have also imaged the model in a CT scanner. In our experimental setup the stereo video camera is held by a robotic arm (Aesop 3000, Computer Motion), the position of the robotic arm is known relative to the outer shell of the solid object. The camera generates stereo images of the solid object. Knowing the position of the camera relative to the outer shell of the model, and knowing the inner structure of the model from segmented CT images of it, it now becomes possible to superimpose an image of the *inner structure* of the model onto an image showing its surface. This is illustrated in figure 3

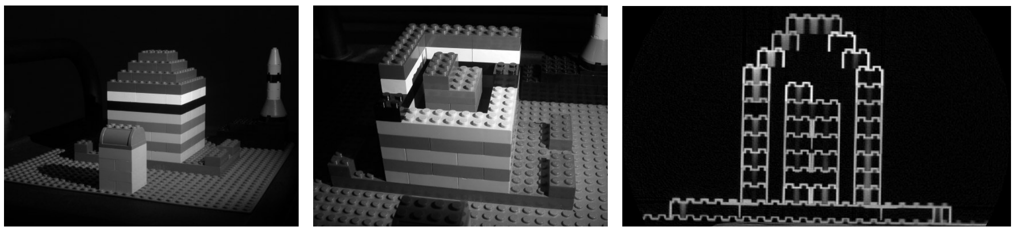


Fig. 2. Lego model, overview, opened to reveal interior structure and image from CT scan.

The problem of generating the depth maps based on the stereo video data must also take place in real time. We have developed a tool to calibrate the stereo video camera, and done extensive work on realtime stereo map generation algorithms [5]. We have also started work on the last problem, that of correlating the depth maps generated from the video camera with the surface of the 3D model in order to increase the precision of the placement of the videoscope relative to the model. Since the approximate position of the video camera relative to the patient or model is provided by the robotic system, the search space in which to look for the best correlation is heavily reduced and makes it possible to do the correlation by finding the affine transformation that will reduce the mean square distance between the model and the depth map to a minimum.

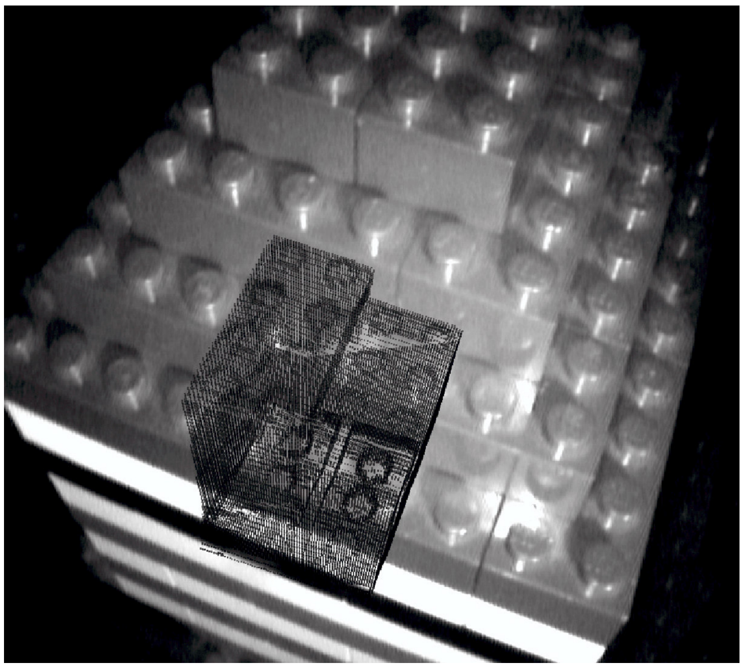


Fig. 3. Lego model with superimposed inner structure.

4 Conclusion

We have developed a prototype of a system for generating an augmented reality for surgeons performing totally endoscopic coronary artery bypass (TECAB). Our main aim is to facilitate the mobilization of the left internal mammary artery (LIMA) by making it (artificially) visible inside the tissue surrounding it. This will partially alleviate the problems experienced by surgeons performing TECAB such as loss of visual and haptic feedback. We have solved the main problems related to this task and have obtained good results on artificial models. We will in the near future perform the same tests on data collected on human subjects. As a future extension of this work, we hope to allow for handheld cameras, this requires that the positioning of the camera can be made exclusively based on the images.

References

[1] D. Boehm, H. Reichenspurner, H. Gulbins, C. Detter, B. Meiser, P. Brenner, H. Habazettl and B. Reichart, *Early experience with robotic technology for coronary artery surgery*, Annals of Thoracic Surgery, vol. 68, pages 1542-1546, 1999.

[2] V. Falk, J. McLoughlin, G. Guthart, J. Salisbury, T Walther, J. Gummert, F. Mohr, *Dexterity enhancement in endoscopic surgery by a computer- controlled mechanical wrist*, Minimally Invasive Therapy and Allied Technologies vol. 8, pages 235-42, 1999.

[3] V. Falk, A. Diegeler, T. Walther, S. Jacobs, J. Raumans, F. Mohr, *Total endoscopic off-pump coronary artery bypass grafting*, Heart Surgery Forum vol. 3, pages 29-31, 2000.

[4] H. Reichenspurner, D. Boehm, H. Gulbins, C. Detter, R. Damiano, M. Mack and B. Reichart, *Robotically assisted endoscopic coronary artery bypass procedures without cardiopulmonary bypass*, Journal of Thoracic Cardiovascular Surgery, vol. 118, pages 960-961, 1999.

[5] Z. Zhang, *A Flexible New Technique for Camera Calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 22. pages 1330-1334, 2000.

[6] A. Koschan, V. Rodehorst, K. Spiller, *Color stereo vision using hierarchical block matching and active color illumination*, Proc. 13th Int. Conf. on Pattern Recognition ICPR96, Vienna, Austria, Vol. I, pp. 835-839, 1996.

[7] F. Devernay, F. Mourgues, E. Coste-Manire, *Towards endoscopic augmented reality for robotically assisted minimally invasive cardiac surgery*, Proceedings of Medical Imaging and Augmented Reality, 2001.

[8] P. Nataf, L. Lima, M. Regan, S. Benarim, R. Ramadan, A. Pavie, I. Gandjbakhch, *Thoracoscopic Internal Mammary Artery Harvesting: Technical considerations*, Ann Thorac Surg, vol. 63, 1997.

Bibliography

[ARUN-87] K. Arun, T. Huang, and S. Blostein, ***Least-squares fitting of two 3-d data point sets***, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5) (1987). 61

[AUDETTE-00] M. Audette, F. Ferrie, and T. Peters, ***An algorithmic overview of surface registration techniques for medical imaging***, Medical Image Analysis, 4(3) (2000), http://citeseer.nj.nec.com/audette99algorithmic.html, pp. 201–217. 15, 57, 58, 59

[BAREQUET-97] G. Barequet and M. Sharir, ***Partial surface and volume matching in three dimensions***, IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997), http://citeseer.nj.nec.com/article/barequet94partial.html. 13, 15

[BEASLEY-02] R. A. Beasley and R. D. Howe, ***Tactile tracking of arteries in robotic surgery***, IEEE Intl. Conf. Robotics & Automation Washington DC May 2002, (2002), http://www.biorobotics.harvard.edu/pubs/Beasley_Howe_ICRA2002.pdf. 9, 16

[BENGTSSON-03] L. Aurdal, D. Bengtsson, O. J. Elle, and E. Samset, ***Augmented reality for safer coronary artery bypass***, Computer Assisted Radiology and Surgery (CARS), June 2003. 7

[BESL-92] P. J. Besl and N. D. McKay, ***A method for registration of 3d shapes***, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2) (1992), pp. 239–256. 15, 58, 59, 61

[BIRCHFIELD-98] S. Birchfield and C. Tomasi, ***Depth discontinuities by pixel-to-pixel stereo***, 1998, `http://robotics.stanford.edu/~birch/publications/p2p_iccv1998.pdf`. 15, 50

[CHENEY-96] K. C. Cheney, ***Numerical Analysis***, Brooks/Cole, 1996. 61

[CLARKE-98] T. A. Clarke and J. G. Fryer, ***The development of camera calibration methods and models***, Photogrammetric Record, 16(91) (1998), `http://www.vision.caltech.edu/bouguetj/calib_doc/papers/Clarke98_calib_history.pdf`, pp. 51–66. 15

[DIEGELER-99] A. Diegeler, T. Walther, S. Metz, V. Falk, R. Krakor, R. Autschbach, and F. W. Mohr, ***Comparison of midcab versus conventional cabg surgery regarding pain and quality of life***, The Heart Surgery Forum, 2(4) (1999), `http://www.hsforum.com/vol2/issue4/1999-32610.html`, pp. 290–296. 7

[EVERSON-98] R. M. Everson, ***Orthogonal, but not orthonormal, procrustes problems***, Advances in Computational Mathematics, (1998), `http://www.dcs.ex.ac.uk/academics/reverson/pubs/procrustes.pdf`. 64

[FITZPATRICK-00] J. M. Fitzpatrick and M. Sonka, ***Handbook of Medical Imaging, Volume 2: Medical Image Processing and Analysis***, Society of Photo-optical Instrumentation Engineers, 2000. 15

[FRIEDMAN-77] J. H. Friedman, J. H. Bentley, and R. A. Finkel, ***An algorithm for finding best matches in logarithmic expected time***, ACM Transactions on Mathematical Software, 3(3) (1977), pp. 209–226. 60

[GRAY-00] H. Gray, ***Anatomy of the Human Body***, New York: BARTLEBY.COM, 2000, `http://www.bartleby.com/107/`. 6

[HOLMES-00] ***Mayo clinic proceedings***, vol. 75(11), November 2000, `http://www.mayo.edu/proceedings/2000/nov/7511e1.pdf`. 6

[HORN-89] B. Horn, ***Height and gradient from shading***, International Journal of Computer Vision, Vol. 5, No. 1, pp. 37–75, August 1990, (1989), http://www.ai.mit.edu/people/bkph/papers/newsfs.pdf. 55

[JOHNSON-97] A. E. Johnson and M. Hebert, ***Surface registration by matched oriented points***, Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling, May 1997, pp. 121–128. 15, 57, 59

[KAPPERT-01] U. Kapperta, R. Cichona, J. Schneidera, V. Gulielmosa, T. Ahmadzadea, J. Nicolaib, S. Tugtekina, and S. Schueler, ***Technique of closed chest coronary artery surgery on the beating heart***, European Journal of Cardio-thoracic Surgery, 20 (2001), pp. 765–769. 8

[KIAII-00] B. Kiaii, W. D. Boyd, R. Rayman, W. B. Dobkowski, S. Ganapathy, G. Jablonsky, and R. J. Novick, ***Robot-assisted computer enhanced closed-chest coronary surgery: Preliminary experience using a harmonic scalpel® and zeus***$^{\mathrm{TM}}$, Heart Surgery Forum, 3 (2000). 8, 9

[KLETTE-95] R. Klette, A. Koschan, K. Schlüns, and V. Rodehorst, ***Surface reconstruction based on visual information***, tech. rep. , Department of computer science, The University of Western Australia, 1995, http://iristown.engr.utk.edu/˜koschan/paper/TR95.pdf. 41, 45, 46, 99

[KOSCHAN-96] A. Koschan, V. Rodehorst, and K. Spiller, ***Color stereo vision using hierarchical block matching and active color illumination***, 1996, http://citeseer.ist.psu.edu/koschan96color.html. 15, 48

[KRISTENSEN-93] S. Kristensen and H. I. Christensen, ***Continuous reconstruction of scene objects***, SPIE, September 1993, http://www.cvmt.dk/˜sk/. 55

[LORENSEN-87] W. Lorensen and H. Cline, ***Marching cubes: A high resolution 3d surface construction algorithm***, in Proceedings of the 14th annual conference on Computer graphics and interactive techniques, ACM Press, 1987, pp. 163–169, http://doi.acm.org/10.1145/37401.37422. 24

[LORUSSO-95] A. Lorusson, D. Eggert, and R. B. Fisher, ***A comparison of four algorithms for estimating 3-d rigid transformations***, Proc. British Machine Vision Conference, BMVC95, Birmingham, (1995), http://www.dai.ed.ac.uk/papers/documents/rp765.html. 61

[MEYERS-98] S. Meyers, ***Effective C++, Second Edition***, Addison-Wesley, 1998. 77

[MORGAN-00] J. Morgan, ***Robotics revolutionizing heart bypass***, USA Today, Feb 29 (2000), http://images.usatoday.com/life/health/doctor/lhdoc101.htm. 7

[MULLANY-03] C. J. Mullany, ***Coronary artery bypass surgery***, (2003), http://circ.ahajournals.org/cgi/reprint/107/3/e21.pdf. 6, 7

[NATAF-96] P. Nataf, L. Lima, M. Regan, S. Benarim, R. Ramadan, A. Pavie, C. Cabrol, and I. Gandjbakch, ***Minimally invasive coronary surgery with thoracoscopic internal mammary artery dissection: Surgical technique***, Journal of Cardiac Surgery, 11 (1996), pp. 288–292. 9

[NATAF-97] P. Nataf, L. Lima, M. Regan, S. Benarim, R. Ramadan, A. Pavie, and I. Gandjbakhch, ***Thoracoscopic internal mammary artery harvesting: Technical considerations***, Ann Thorac Surg, 63 (1997). 7

[NIELSON-91] G. M. Nielson and B. Hamann, ***The asymptotic decider: Resolving the ambiguity in marching cubes***, in Proc. of Visualization'91, San Diego, CA, 1991, pp. 83–91. 25

[OPENCV] ***Intel's OpenCV***, http://www.intel.com/research/mrl/research/opencv/. 37, 75

[POLLEFEYS-00] M. Pollefeys, ***Tutorial on 3d modeling from images***, tech. rep. , Center for Processing of Speech and Images (ESAT-PSI) of the K.U.Leuven (Belgium)., 2000. 28

[Qt] ***Trolltech's Qt***, http://www.trolltech.com. 75

[ROJAS-97] A. Rojas, A. Calvo, and J. Muñoz, ***A dense disparity map of stereo images***, Pattern recognition letters, (1997), pp. 385–393. 15, 42, 52, 53

[SCHARSTEIN-02] D. Sharstein and R. Szeliski, ***A taxonomy and evaluation of dense two-frame stereo correspondence algorithms***, (2002). 15

[SETHIAN-99] J. A. Sethian, ***Fast marching methods***, SIAM Review, 41(2) (1999), http://carlos.lbl.gov/RESEARCH/OtherRefs/Sethian99.pdf, pp. 199–325. 60

[SHROEDER-98] W. Shroeder, K. Martin, and B. Lorensen, ***The Visualization Toolkit***, Prentice Hall, 1998. 24, 25

[SOMMERER-99] C. Sommerer, L. Migninneau, and R. Lopez-Gulliver, ***Time lapse immersive interaction with historic 3-d stere images***, Virtual Systems and Multimedia, September 1999, http://www.mis.atr.co.jp/˜gulliver/papers/vsmm99.pdf. 14

[SONKA-98] M. Sonka, V. Hlavac, and R. Boyle, ***Image Processing, Analysis, and Machine Vision***, PWS Publishing, 1998. 32, 45

[SSB] S. sentralbyrå, http://www.ssb.no/. 5

[STEIN-02] A. N. Stein, ***Modeling real-world objects from noisy data***, http://www.andrewstein.net/research/ICP/ICPmodeling.pdf. 59

[TANIMOTO-75] S. Tanimoto and T. Pavlidis, ***A hierarchical data structure for picture processing***, Computer graphics and image processing, 4 (1975), pp. 104–119. 48

[YAMANY-99] S. M. Yamany and A. A. Farag, ***Free-form surface registration using surface signatures***, vol. 2, IEEE International Conference on Computer Vision (ICCV'99), IEEE, September 1999, pp. 1098–1104, http://www.cvip.uofl.edu/wwwcvip/research/publications/Pub_Pdf/1999/7-ICCV99.pdf. 59

[ZHANG-00] Z. Zhang, ***A flexible new technique for camera calibration***, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), http://dx.doi.org/10.1109/34.888718, pp. 1330–1334. 15, 30

[ZHANG-92] ——, ***Iterative point matching for registration of free-form curves and surfaces***, Tech. Rep. RR 1658, Institute National de Recherche en Informatique et en Automatique, Mars 1992. 15