Structure-from-motion for enclosed environments

A THESIS
SUBMITTED TO THE DEPARTMENT OF APPLIED MATHEMATICS
OF THE UNIVERSITY OF STELLENBOSCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Henri Hakl
December 2007

Supervised by:
Prof Ben Herbst
Dr Karin Hunter

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Signature: ..... Date: .....

Abstract

A structure-from-motion implementation for enclosed environments is presented. The various aspects covered include a discussion on optimised luminance computations—a technique to compute an optimally weighted luminance that maintains a greatest amount of data fidelity. Furthermore a visual engine is created that forms the basis of data input for reconstruction purposes; such an inexpensive solution is found to offer realistic environments along with precise control of scene and camera elements. A motion estimation system provides tracking information of scene elements and an unscented Kalman filter is used as depth estimator. The elements are combined into an accurate reconstructor for enclosed environments.

Abstrak

'n Struktuur-deur-beweging implementasie vir geslote omgewings word beskryf. Optimisieerde luminasie word beskryf, dit is 'n metode om optimale gewigte vir luminasie berekeninge te vind sodat so veel as moontlik inligting behoue bly. 'n Virtuele omgewings simulator word voorgestel wat die basis vorm vir rekonstruksie data. Die metode is goedkoop en kan omgewings realisties en met 'n groot mate van kontrolle bereken. 'n Unscented Kalman filter word gebruik om diepte te bereken.

Acknowledgements

An endeavor of the magnitude of a PhD study cannot be accomplished by a single person. The years that accumulated into these pages were filled with a wide range of people that were instrumental in their creation. I wish to express my gratitude to all who have aided me in this time.

Specifically I wish to thank my promoter, Ben Herbst, for his friendly yet consequent guidance in my research. His insight and experience were an exceptional sounding board for the methods proposed in this study.

I had the good fortune to receive significant financial assistance from the National Research Foundation of South Africa. The grant provided to me over the course of three years was a valuable aid for which I am grateful; my research would not have been possible without it.

A great variety of people, too numerous to enumerate, supported me throughout the PhD work: colleagues and friends who proved both useful distractions and meticulous advisers.

Finally, I can feel no greater sense of debt and gratitude than that which I feel for my mom. Her diligent work, unswerving trust and unquestioning love have supported me and my efforts throughout my life. It is to her that I dedicate this work.

Contents

Chapter 1

Introduction

> *To suppose that the eye with all its inimitable contrivances for adjusting the focus to different distances, for admitting different amounts of light, and for the correction of spherical and chromatic aberration, could have been formed by natural selection, seems, I confess, absurd in the highest degree.*
>
> Charles Robert Darwin

The release of "*On the origin of species by natural selection*" by Charles Darwin in 1859 could be described as the intellectually most profound landmark of the industrial revolution. The theory forwarded by Darwin was not unique[1], nor novel, but it was presented at a time when the metaphoric scientific soil was fertile and the idea was well received. The theory is of great significance in the field of biology—but, more profoundly, it represents the culmination of scientific awakening in that it achieved the separation of church and science.

The theory of evolution, though, is a theory, and as such is subject to constant scrutiny by those who oppose the principles held forth by it. This text does not attempt to involve itself in the debate on the merits of evolutionary theory. Instead it recognizes that a point of contention between supporters and detractors of the theory is the biological miracle of the eye. Evolutionary theorists hail the eye as one of the greatest achievements of natural selection—yet those that attempt to demean the theory extol the eye as the perfect exhibit to demonstrate the implausibility of reinforced chance creating such a finely tuned and highly complex organ.

Credit, though, should not only fall to the eye but also the brain, which in conjunction with the eye allows the collected data from light rays to be processed into identifiable entities and

[1] Equal credit on the theory of evolution should be given to both Charles Darwin (1809—1882) [26] and Alfred Wallace (1823—1913) [122] who both independently developed what is known as the theory of evolution—now commonly referred to as the Darwin-Wallace Theory of Evolution. Additionally, the work by Jean Baptiste Lamarck (1744—1829) [65] contributed significantly to the development of evolutionary theories. Though Lamarck's work was discredited during his lifetime his plausible theories anticipate, and in places even supercede, the work done by Darwin and Wallace.

allows depth to be perceived. The faculty to infer spatial relation and recognition of features is deeply profound to the existence of highly complex organisms such as mammals, and is likely to be one of the early contributors to increasing complexity and size of the cerebral cortex.

Mathematically the problem of emulating the eye for purposes of creating an artificial system of spatial and feature recognition is formidable, yet tractable. The consequences of formulating such a system successfully would be varied and far reaching, finding applications in many forms and fields—including, though not limited to intelligent security systems, safety assessment equipment, motion-capture systems, medical topographic systems, and autonomous robotics such as humanoid androids and robots used to explore hostile environments.

Different attempts have been made to formulate a solution to the problem of depth perception. A biologically inspired approach is that of stereoscopic systems—which uses two cameras to model two eyes [9]. The small discrepancies in perception due to the slight displacements of the cameras may be used to infer the depth of objects. Structure-from-motion solutions [8, 126] only use one camera; however, they make use of time-variant data in image sequences to reconstruct depth. A third approach is known as shape-from-shading [127]: in it single images are used to deduce depth parameters; this is accomplished through analysing feature relations and shading of objects.

Two more approaches are known as shape-from-focus [80] and shape-from-silhouette [20]. The former, shape-from-focus, primarily finds application in microscopic reconstruction of depth information. At very large magnification the depth of focus of optical microscopes is not sufficient to convey a sharp image of non-planar samples. Usually the solution proposes to acquire a series of images with a constant change in focus—the final image is computed by creating a collage of the sharp regions in each image.

Shape-from-silhouette is a conceptually simple approach to generate three-dimensional models from images. Each image is taken from a different viewpoint and the object's silhouette on an image represents a conic volume in object space. By intersecting each of the conic volumes a three-dimensional model of the object can be generated.

> *The purpose of this study is to advance understanding and insight into using structure-from-motion techniques to successfully reconstruct depth-information from image sequences for architectural environments.*

To state the intention less succinctly: the structure-from-motion problem possesses many facets that are related to each other, yet still differ significantly. For example, reconstructing an object from an image sequence is similar to reconstructing an environment from an image sequence. However, the former benefits from a larger set of simplifying assumptions that allow the problem to be solved using simpler means. In the later case the assumptions need to be more universal, requiring ever more complex and robust solutions to yield strong results.

The ideal solution to the structure-from-motion problem would emulate the biological eye-brain duo and be able to recognize and correlate depth data in unrestricted environments in the presence of partially transparent and deformable entities, as well as arbitrary dynamic motion and lighting. This feat is well beyond the scope of this text. Instead, this text presents an approach to gain structure-from-motion data for static, fully enclosed environments—such as, for example, the inside of a building.

The difficulty herein is twofold. Firstly, only a part of the environment is exposed to the reconstruction process at any given moment in time. As the source of perception moves new parts of the environment become visible and others become occluded or leave the field of view.

Secondly, the data density is increased tremendously compared to the problem of object reconstruction. To yield a structurally sound and meaningful representation of a reconstructed environment it may not be sufficient to identify key features and analyse their depth components. The approach adopted in this text makes use of a dense optical flow that yields a good structural indication of the environment that is reconstructed.

1.1 Previous work

The problem of reconstructing environments is difficult but has a large number of real-world applications. Therefore it is not surprising to find that extensive work has been done on the problem. A related problem to structure-from-motion reconstruction is known as SLAM (simultaneous localisation and mapping)—the problem of an autonomous robot navigating an unknown environment requires the robot to be able to perform environmental mapping and localisation of itself within that environment.

An early solution is offered by Kristensen and Christensen [64] who describe an autonomous robot navigation system for indoor environments that makes use of two reconstruction systems: stereoscopic based reconstruction to provide a broad overview of the environment and aided by a depth-from-focus system that resolves depth queries in areas where potential occlusion are detected.

Bohn and Thornton [15] make use of a Laser Range Finder, in conjunction with geometric transforms and artificial neural networks to reconstruct enclosed environments. The data is subsequently laser sculpted using a spatial data structure called an octree.

Nedevschi *et al* [81] present a method to reconstruct three-dimensional environment data using a trinocular system—in other words they extend normal stereoscopic reconstruction to include a third camera. Their implementation makes use of optimised code to process data in parallel using SIMD[2] instructions such as the MMX and SSE2 instruction sets.

[2]Single Instruction, Multiple Data—CPU instructions that allow processing of several data inputs and producing multiple outputs in a single instruction

Clark *et al* [23] make use of a single video camera along with positional equipment to perform environment reconstruction in both sparsely furnished room interiors as well as urban environments. The interior variant makes use of an inertial measurement unit to determine translational and rotational properties, whereas the exterior version makes use of GPS information. Both motion systems are supplemented by an extended Kalman filter to estimate corrections to the information provided by the motion tracking systems. The environment is reconstructed sparsely using a MAPSAC (Maximum A Posteriori SAmple Consensus) framework.

A comprehensive solution was presented by Dissanayake *et al* [28, 29] that made use of several existing strategies that addressed the indivual problems in reconstruction and combined them into a single complete system. Their contributions emphasize efficient construction, manipulation, management and storage of scene maps.

Abuhadrous *et al* [3] forward a solution that consists of a robot equipped with a laser scanner to map the environment as well as GPS, INS and odometer to determine the location and orientation of the robot within its environment. An internal estimator derives an estimate of the current robot state using the various sensors it is equipped with.

Weingarten and Siegwart [123] describe an improvement on an extended Kalman filter based SLAM solution. Their system makes use of a laser scanner to generate a point cloud from which planar features are extracted using stochastic means; subsequently, as the robot moves, the map and localisation is updated incrementally by estimating its motion and pose using an extended Kalman filter.

Recently Panzieri *et al* [84] suggested a simplified form of applying the interlaced extended Kalman filter to the SLAM problem. The interlaced Kalman filter is actually a set of filters that each estimate a different aspect of the system: camera modeling and structural modeling. Their suggestion achieves a better balance between computational load and precision.

Considerable effort has been made with structure-from-motion solutions as well. These will be presented in Chapter 6 *Structure from motion*.

1.2 Contributions

Let us take a moment to recap. The general goal of this thesis is to develop a system to reconstruct 3D objects from 2D images. More precisely, to densely reconstruct interior environments such as the interior of a room. As we have seen in the previous section a number of options exist, some of them available commercially. The available options include, but are not limited to, laser scans, stereo vision utilising structured lighting, shape-from-silhouette, photoconsistency, and so forth. All of these find important applications but also suffer from certain limitations. Laser scans are expensive, so is stereo vision utilising structured lighting. In addition stereo vision requires calibrated cameras, or some automated calibration procedure.

Shape-from-silhouette only reconstructs the convex hull and is not suitable for the reconstruction of interior environments. Photoconsistency also needs some kind of calibration procedure and again is less suitable for the reconstruction of interior scenes [49].

That brings us to the approach investigated in this thesis, namely structure-from-motion. Of course this is not a new idea, more detail about that follows in a moment. Let us first note some of the advantages of using structure-from-motion for dense reconstruction of specifically interior environments. Imagine a scenario where a robot, equipped with a video camera, enters an unknown environment inaccessible to humans. The idea is to build a full 3D environment from the video stream recorded by the camera. Note the modest requirements if the structure-from-motion route is followed—a single camera, attached to a robot. And the latter, or an equivalent devise, is required for all 3D reconstructions. The fact that the camera does not need to be calibrated increases the robustness of the procedure. Although the same results may be obtained if the robot is equipped with two calibrated cameras, since the robot is moving through the environment, it means that the cameras need to be calibrated and synchronised. A loss of either the calibration or synchronisation can be catastrophic for the reconstruction. This is not the case for structure-from-motion. Therefore, even in situations where stereo vision is possible, even preferable, it is still a good idea to confirm the stereo reconstruction using structure-from-motion. In fact, this is exactly how structure-from-motion is employed at iThemba Laboratories where patients undergoing fixed-beam proton therapy are positioned using stereo vision. Structure-from-motion is used as a safeguard against accidental loss of calibration, see [118].

Perhaps rather surprisingly, but to the best of our knowledge structure-from-motion has not been extensively investigated as a means for densely reconstructing interior environments. In this thesis the limits and limitations of structure-from-motion are investigated.

One limitation is immediately apparent—for indoor environments structured lighting cannot be used, not even for stereo reconstruction. This is because reference points are lost the moment the camera (and structured light source) changes position. This means that recognisable features in the environment are required—for a dense reconstruction, dense features are required. Even if sufficient texture is available, the limitation on accuracy is unclear, as is the computational complexity and need to be investigated.

In order to explore the accuracy of the reconstruction, a ground-truth is required. For this purpose a virtual environment is developed. This is then used to thoroughly test the limitations of the approach. For example, the reconstruction is based on the unscented Kalman filter, to the best of our knowledge, first used by Venter in the structure-from-motion context [119]. It has been observed that the unscented Kalman filter takes about 100 frames to converge, but the effect of the frame-rate has not been investigated. The virtual environment provides a convenient setting for changing the frame-rate, and allows a detailed investigation of this issue.

In the course of developing the system, numerous detailed problems needed to be addressed and solutions provided. These are detailed in the text. Let one example suffice for now. The full

reconstruction is done in a piecewise fashion and the whole then reassembled from the different pieces. Commercial applications using structured lighting typically use a manually provided reference markers. We do not have this luxury. Instead the relation of the reconstructed pieces is inferred from the camera motion which in general be monitored and controlled for robot-mounted cameras.

Let us finally list some of the specific contributions that, to the best of our knowledge, are new.

1. *Optimal luminance space.* Since structure-from-motion is performed on gray scale images, a conversion from colour space is required. The conversion is not unique and there is a danger that information can be lost in the transformation process. The conversion parameters that are generally used, work well for general scenes. In Chapter 2 the idea of an optimal luminance space, optimised for a specific image, is introduced. It is fast to compute and is designed to preserve more information as compared to normal luminance conversion.

2. *Using structure-from-motion for the dense reconstruction of interior scenes.* This needs to be qualified. Structure-from-motion has been used for 3D reconstructions in some form or another, for a long time. However, we are not aware of any detailed study exploring its potential for the dense reconstruction of interior scenes. Stereo reconstruction and laser scans are typically used instead. Shape-from-silhouette is popular for the reconstruction of 3D objects, and is often used in conjunction with stereo reconstruction. Since it relies on views of the silhouettes of objects, it is not suitable for the reconstruction of interior scenes. Even the term *dense* reconstruction needs to be qualified. Since we rely on the availability of features inherent in the scene, the reconstruction can only be as dense as the number of available features. However, we explore in depth the potential of the approach in those cases where dense features are available as it is important to understand how the problem scales with an increase in the number of features.

3. *Performing structure-from-motion within the context of a fully controlled virtual environment.* Again it is necessary to qualify the statement. Virtual environments or artificial data is commonly used, even for structure-from-motion. In the earliest study of structure-from-motion at this institution, for example, Venter [117] tested his scheme on virtual objects. In this thesis however, we are more ambitious. A full interior virtual environment was developed that provides full control over the camera motion and environment, including texture, camera motion, structural complexities, etc. Since it provides a ground-truth, it becomes a useful, if not an indispensable, tool for a detailed study of the limitation of structure-from-motion.

4. *Reconstruction of the environment from smaller pieces using information inferred from camera motion.* It is common practice to divide large computational tasks into smaller pieces, simply because computational complexity scales non-linearly with size. In our application, the environment is not visible from a single viewpoint and it is therefore quite

natural, and computationally more efficient, to do a piecewise reconstruction and then assemble the different pieces into a coherent reconstruction of the total scene. Usually such methods make use of reference points that relate the different pieces to each other. These reference points may be derived automatically from the scene itself, and are therefore prone to misalignment (requiring further processing such as the RANSAC algorithm to identify miss-alignments), or the reference points are manually super-imposed onto the scene (some applications require little stickers to be pasted on the objects to be reconstructed). In this thesis a complementary approach is developed where the relationships between the different pieces are inferred from controlled camera motion. As is always the case in Computer Vision, a combination of independent, complementary approaches lead to increased robustness.

The structure of the thesis is as follows. In the next chapter optimal luminance space is discussed. The 3D virtual environment developed for this thesis is described in Chapter 3. Motion estimation, including the dense and sparse optical flow procedures used in this thesis are described in Chapter 4. Chapter 5 discusses the unscented Kalman filter, followed by a chapter where the integration and implementation of the different parts are discussed. Chapter 7 gives the experimental results, and is followed by a chapter where we draw various conclusions.

In whatever manner God created the world, it would always have been regular and in a certain general order. God, however, has chosen the most perfect, that is to say, the one which is at the same time the simplest in hypothesis and the richest in phenomena.

Gottfried Wilhelm von Leibniz

Chapter 2

Color space and data retention

Der Fortschritt ist eine gute Sache—sofern man sich über die Richtung einig ist.

Albert Schweitzer

To phrase Albert Schweitzer's words in English: "Progress is a good thing—provided one is agreed on the direction." In any venture, however, it is not uncommon when occasional aberrations bend the intended path into wholly unexpected directions.

This chapter presents such an aberration from the intended path of structure-from-motion research: the concept of optimised luminance space. Luminance space is a color space that describes the brightness of image elements; in simplest terms it describes a gray-scaled version of an image—which may be interpreted as a weighted average of its color components. In this thesis the notion of an optimised luminance space refers to adjusting the weights of the weighted average in such a manner as to retain the largest possible amount of information.

Like many chance discoveries, the concept of optimised luminance space is a by-product of research of an entirely different nature. Digital image processing techniques often rely on color space conversions and color sub-spaces to create particularly efficient algorithms. For example, many motion estimation approaches make use of luminance space for increased computational efficiency.

Luminance space contains all the essential data relevant for motion estimation techniques. Typically only eight data bits per pixel are used in luminance space, compared to twenty-four bits in raw RGB color space. This does not directly affect the theoretical computational complexity, but in terms of practical implementation the decrease in per-pixel data size allows applications to process data considerably faster. Such advances in speed are especially relevant in real-time or resource intensive systems such as, for example, motion estimation and image/video compression. Structure-from-motion solutions benefit from optimised luminace through the increase in accuracy that can be achieved in motion tracking with the use of

optimised luminance.

As the remainder of this chapter will demonstrate, a good estimate to optimised luminance space can be computed more efficiently than normal luminance space; furthermore, a significant increase in information is observed in optimised luminance space compared to normal luminance.

The following sections will introduce optimised luminance space and its background, as well as various implementations which attempt to make use of the concept. To illustrate the significance the research results are demonstrated and concluded with a discussion on relevance to current developments in the field of computing.

Definitions

Intensity, or brightness, of a color is a measure of the flow of power that is radiated by that color over some interval. In the case of a light wave it can be understood as the amplitude of that wave. On digital systems the intensity of a color simply refers to its digital value, typically 0 indicates complete absence of intensity and 255 refers to highest intensity. Brightness perception is a complex phenomenon and is, in the context of this thesis, simplified to *luminance*: the weighted average of color components.

Hue, or chromaticity, describes color; in other words red, green, brown, purple, and so on. In terms of a light wave it can be understood as the dominant wavelength of that wave. On digital systems the hue is usually described using three color components: RGB.

Saturation refers to the relative bandwidth of the visible output from a light source. As more wavelengths of color are present the apparent color becomes more faded and approaches white. A color possesses a high saturation if that color is pure, meaning it shares little or no bandwidth with other wavelengths of light.

2.1 Background

Human perception of colors is remarkably complex. Two primary types of sensory nerves exist in the human eye: rods and cones. Rods are able to respond to minute intensities of light—thus facilitating night-vision—and cones are able to respond to color stimuli.

According to Hecht [42] experiments indicate that three types of cones exist in the human eye that are responsible for sensing colors: cones that react to red, green and blue color hues. The overall palette of colors perceived by the human mind is then a fusion of the stimuli received over these nerves.

This insight led to the realization that mathematically a triplet of values could be used

to uniquely describe a color. Three parameters, in turn, are then a sufficient and minimal description to completely describe color sensation—the concept of color space.

One of the earliest mathematically defined color spaces is the CIE XYZ color space, sometimes referred to as CIE 1931. It was created by the International Commission on Illumination (CIE) in 1931.

The CIE 1931 color space was originally derived from work by David Wright [125] and John Guild [40]. Their work was at first merged into the CIE RGB color space, from which CIE XYZ was developed shortly afterwards.

Numerous other color spaces have since then been defined. Their properties vary, some are suitable to certain applications—others are suited to different applications. For example, computers commonly make use of RGB color space as a conceptually simple and efficient model to control visual hardware components. Color spaces such as CIE L*a*b* [74], on the other hand, attempt to offer a complete and device-independent description of all the colors visible to the human eye.

The concept of device dependence or independence is relevant in color spaces. Independent models such as L*a*b* describe a color absolutely, whereas a simple RGB color space does not guarantee the same appearance of a particular set of values across different devices.

This may lead to some confusion when attempting to convert an image described in a device-dependent color space to an independent one, or vice versa. For example, to convert an image from generic RGB space to L*a*b* space requires the converter to know or assume an absolute reference[1]—as there is no direct relation between absolute and non-absolute color spaces.

CIE L*a*b* is conceptually simple to understand. The components **a** and **b** refer to the hue—that is to say the chromatic component—of the color. The first component, **L**, is the luminance and refers to the brightness of a given color, shades of gray ranging from complete black to bright white.

The brightness of image colors is generally refered to as the luminance of the image. The color range described by the **L** component of the CIE L*a*b* standard may therefore be interpreted as a color subspace, commonly referred to as luminance space. Opposed to luminance space is chroma space consisting of the set of the remaining two components **a** and **b**—chroma describes the hue of a particular color. A simple conceptual illustration of these different spaces is represented in Figure 1, on page 11:

Luminance and chroma spaces are not merely of mathematical interest, they possess distinct properties with regard to human perception. To illustrate: human perception is optimised to react to changes in light intensity, thus the human visual system is more sensitive to changes in luminance than to chromatic changes.

[1] Among others, Hewlett-Packard and Microsoft Corporation have proposed such an absolute color space for the RGB color model. It is known as standard RGB or simply sRGB.

(a) Luminance space (b) Chroma space

(c) Normal image: Union of luminance and chrome components

Figure 1: Illustration of luminance, chroma and normal color space

This knowledge is applied in various fields such as, for example, video compression: video sequences depicting intense and rapid action scenes are best compressed with a bias towards luminance data, whereas slow and still sequences are biased towards chromatic data. Though the overall data density remains unchanged, human perception rates the quality of the video sequences as higher using such an approach compared to using no bias.

As described previously many algorithms, such as motion tracking [108], make use of luminance space. It is well suited for processing purposes due to its compact nature and it offers a good estimate of the image in spite of the lower data density.

Technically, the computation of luminance from RGB data on modern computing systems is a simple process—luminance (Y) is the weighted sum of red (R), green (G) and blue (B) color components [87]:

$$Y = 0.2125R + 0.7154G + 0.0721B \ .$$

The weights are chosen in a manner to optimally stimulate the perception of brightness in the

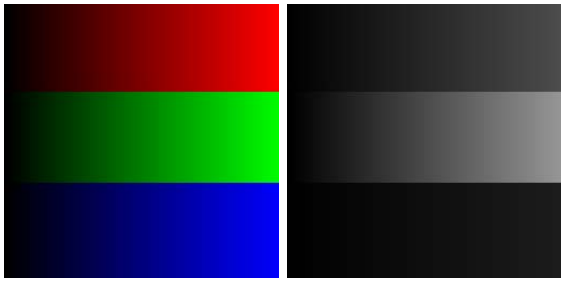
human visual system. Older systems make use of different coefficients, corresponding to the phosphors typical in monitors during the inception of NTSC television in 1953:

$$Y = 0.299R + 0.587G + 0.114B \ .$$

The apparently odd choice of coefficients reflects differences in perceived brightness in human vision—if three light sources (red, green and blue respectively) were to radiate equally brightly, then human perception would make them appear to have different brightnesses. Green light would be considered brightest, followed by red, and blue would be rated darkest of all.

A consequence of the small factor blue plays in computing brightness is that images with extensive blue areas and images with primarily blue hues will possess visible contours due to low data discrimination, as demonstrated in Figure 2: Each color bar—red, green and blue—is shown with uniformly increasing intensity from least to most prominent in 2a. However, in the corresponding luminance image 2b the blue bar appears to be nearly uniformly black due to the low discrimination of blue data in luminance space.

The following section addresses this observation and introduces optimised luminance space. This concept is intended to optimise the data retention in luminance space and thereby improve computational accuracy in motion estimation, video compression and other applications making use of luminance space as a compact representation of image data.



(a) Original image (b) Luminance image

Figure 2: Loss of data fidelity due to luminance coefficients

2.2 Optimised luminance

The diminished data retention in images when converting from normal color space to luminance space is not just a problem in pathological cases. Images, or sub-sections of images, with predominantly blue, red and purplish hues can suffer a considerable loss of detail.

For the purposes of this text, data fidelity in an image is loosely defined as the richness of information within the image—this can be likened to how well such an image compresses without loss of information. Correspondingly, a monochromatic image possesses a low data

fidelity and an image of white noise possesses a high data fidelity. Figure 3 illustrates this phenomenon.

(a) Full color image

(b) Blue hue image

(c) Luminance of full color image

(d) Luminance of blue hue image

Figure 3: Histogram data illustrating loss of data fidelity

Each image has its corresponding color distribution histogram superimposed. The vertical axis represents the frequency of occurence and the horizontal axis represents the brightness of colors from least to most intense.

The images demonstrate how an image with predominantly blue hues suffers a considerable loss of detail when transformed into luminance space: the histogram in 3c displays a far greater diversity of intensities across the image than the histogram in 3d. The diversity of intensities in 3c, when compared to 3d, indicates that the image possesses a greater degree of information and can thus be analysed more accurately with algorithms such as motion estimation.

Conversely, in images with predominantly blue hues the probability of errors increases when making use of corresponding luminance data. For example, motion estimation algorithms may fail to find correct motion vectors due to the lack of sufficiently distinct features in the data.

The problem, however, is not purely confined to the reduction of significant bits in image

data. In practise many problems suffer from a certain amount of noise, be it from measurement error or background noise innate to the process in question.

Such contamination of image data can often be tolerable in color space. However, if the loss of data fidelity is prominent when converting to luminance space (such as in areas of the image with predominantly blue color tones), then the presence of generic noise may cause algorithms to become unreliable as the signal to noise ratio may decrease.[2]

It follows that if luminance space is used purely due to the resultant compact data size, then in general a more optimal choice of compact space could be used. Such an optimally chosen compact space will be referred to as optimised luminance space in this text. The term may be misleading as it creates the impression that some form of luminance is chosen that optimally stimulates human perception—whereas the intention is actually to choose a luminance that can be optimally perceived by artificial systems, such as computers.

The idea of computer-based perception is apt: mathematically there is no loss in information (apart from chromatic information) when computing luminance space out of color space—all that changes is data emphasis. The loss of information is a result of storing the computed luminance data in integer form—truncation and rounding errors are responsible for the loss of information. Thus distinct color intensities may be mapped onto the same luminance intensity: a many-to-few mapping that results in a loss of information.

In principle the computed luminance data could be stored in 32-bit floating-point number format. This would ensure a high data precision and would practically eliminate loss of information as a source of computational inconsistencies. However, computationally the cost of processing 32-bit floating-point numbers is several times higher than the processing of 8-bit integers. For each 32-bit floating-point number, four 8-bit values could be computed in parallel. This computational efficiency of 8-bit data outweighs the relatively small loss in data fidelity when computing the weighted RGB averages.

Mathematically any choice of coefficients α_r, α_g, and α_b (corresponding to the coefficients for red, green and blue hues respectively) is suitable to compute the average weighted luminance such that

$$Y = \alpha_r R + \alpha_g G + \alpha_b B$$

provided that

$$\alpha_r + \alpha_g + \alpha_b = 1 \tag{1}$$

$$0 \leq \alpha_r \leq 1$$

$$0 \leq \alpha_g \leq 1$$

$$0 \leq \alpha_b \leq 1$$

holds.

[2]It should be noted that noise is not always a problem as occasionally it may actually aid perceived accuracy. For example, a fine noisy grain applied to poor digital video footage can improve the perceived quality of the media [2].

At this point the question is raised: which values of α_r, α_g, and α_b are such that the information of the luminance data is maximised? To make meaningful statements in this regard a mathematically rigorous metric is required against which different solutions for the coefficients can be compared and declared more or less optimal. Depending on which metric is chosen, different solutions and computational strategies present themselves.

For the purposes of this text the choice of metric is based on histogram equilibrium. The histogram equilibrium is sometimes referred to as histogram variance, or simply variance in this thesis. A given solution for luminance coefficients is rated more optimal if it produces a histogram with a more even distribution of colors. Mathematically histogram equilibrium is computed by the summation of absolute differences between incidence of colors and the expected incidence of colors given a uniformly random distribution of pixels in the image:

$$h = \sum_{i=0}^{n-1} |c_i - c_e|$$

where h is the computed histogram variance, n indicates the number of colors that could be used in the image, c_i represents the frequency of the ith color and c_e is the number of expected occurences of a color in the image. Given additionally that p is the total number of pixels in the image then the expected incidence of colors is simply:

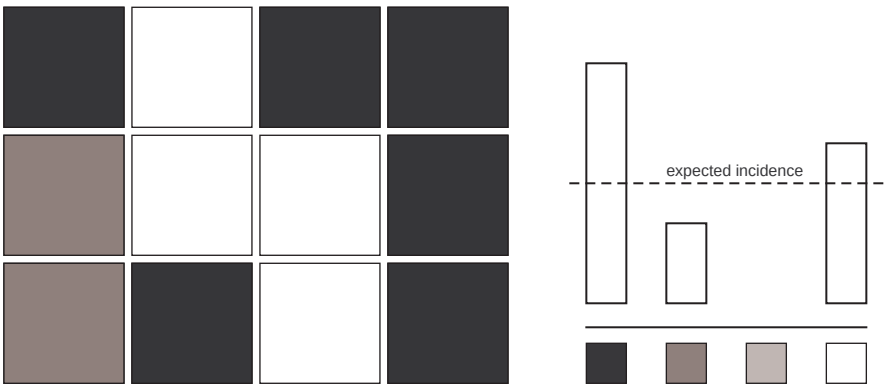
$$c_e = \frac{p}{n} \ .$$

A lower histogram equilibrium h means that the image possesses a more even distribution of colors—and therefore more information.

This is illustrated with a simple example: the histogram equilibrium of Figure 4 is computed. The image is four pixels wide and three pixels high. Furthermore it possesses a 2-bit color depth, corresponding to four different colors that could be used in the image (the example image makes use of only three of these four colors).

The expected incidence of colors in the image is:

$$\begin{aligned} c_e &= \frac{p}{n} \\ &= \frac{wh}{n} \\ &= \frac{4 \times 3}{4} \\ &= 3 \ . \end{aligned}$$



(a) Magnified image (b) Histogram of image

Figure 4: Example image to compute histogram variance

The histogram variance in the image is:

$$\begin{aligned} h &= \sum_{i=0}^{n-1} |c_i - c_e| \\ &= \sum_{i=0}^{3} |c_i - 3| \\ &= |6-3| + |2-3| + |0-3| + |4-3| \\ &= 8 \end{aligned}$$

The result of "8" does not convey much information in itself; however, it is a valuable measure when compared to the variance of a different image. Lower values of histogram variance suggest a more even distribution of colors and hence are considered more optimal in this study.

The next section introduces the approaches that were considered and implemented to compute optimised luminance. Demonstrative results are included that illustrate the gain in histogram variance.

2.3 Optimised luminance implementations

To achieve insight into the nature of optimised luminance space a simple naive algorithm was implemented to compute the optimal coefficients using brute force. The algorithm is listed below in Program 1.

The program is simple to understand: *bestH* indicates the current best rating for tested R, G and B values, and *resolution* specifies the granularity of tested values. Two conditional loops are used to iterate through all possible unique combinations of R, G and B that satisfy the

Program 1 Naive Optimal Coefficients

```
bestH = +infinity
resolution = 0.01
R = 0
while R <= 1.0 do begin
  G = 1 - R
  while G >= 0.0 do begin
    B = 1 - R - G
    curH = GetH( GetWeightedLuminance(image, R, G, B) )
    if curH < bestH then begin
      bestR = R
      bestG = G
      bestB = B
      bestH = curH
    end
    G = G - resolution
  end
  R = R + resolution
end
```

constraints, (1), shown on page 14. These coefficients are then used to compute the weighted luminance data and the histogram variance is computed. This is compared to the current best rating and, if necessary, the best R, G, B and $bestH$ are updated.

Such a naive approach is tremendously expensive computationally, and will only find the best solution afforded by the resolution—or granularity—of the search. This implies that in arbitrary images it is wholly plausible, and for that matter likely, that a slightly better solution can be found by using a smaller resolution parameter.

The computational complexity of "Naive Optimal Coefficients" can easily be demonstrated to be $O(n^2)$, where n specifies the resolution of the search: the algorithm iterates through a loop that in turn passes through another loop. The iteration count for each of these loops is determined by the *resolution* parameter—in other words it possesses a computational expense of order *resolution* $\times$ *resolution*.

Generally the most time-consuming part of the algorithm on each loop iteration is the computation of the functions GetH and GetWeightedLuminance. However, for the purposes of evaluating the computational complexity of "Naive Optimal Coefficients" they are considered to resolve in constant time.

The images in Figure 5 illustrate the results of the algorithm. Figure 5a is the original image, for comparison Figure 5b illustrates normal luminance space for the image, Figure 5c represents the image in optimal luminance space, and finally Figure 5d demonstrates the coefficient solution space.

Figure 5d is sometimes referred to as a tri-color diagram in this text and is a representation of the coefficient solution space. Darker areas denote lower values and brighter areas denote higher values of histogram equilibrium. A lower histogram variance is considered more optimal, so darker areas indicate a more optimal choice of coefficient than lighter areas. The figure illustrates clearly that in this particular case green is a poor choice to predominate the coefficients of the luminance computation. Instead, blue or red make good coefficients to give a lot of weight to, but they should not receive equal weight.

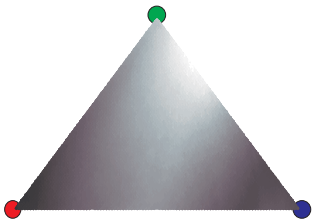
The best coefficients, according to the algorithm, are given by $R = 1.0$, $G = 0.0$ and $B = 0.0$. These correspond to a histogram variance of 25368; the histogram variance of the normal luminance image is computed as 27901—which falls short of the optimal luminance balance by a margin of approximately 10%.

It is tempting to compare optimal luminance to histogram equalization, which is a technique to adjust image contrast by spreading out histogram data. However, it is important to distinguish between them. Optimal luminance attempts to compute a luminance image that retains as much information as possible from the original image; histogram equalization, on the other hand, artificially enhances images by changing information. Histogram equalization aids human perception, but it does so through falsifying information in an image. In some cases such falsification is not detrimental and can aid applications, such as feature detectors. In such cases it is still possible to first use optimal luminance followed by histogram equalization to achieve a greater degree of accurate information compared to normal luminance computations.

(a) Original image

(b) Normal luminance

(c) Optimal luminance



(d) Histogram variance distribution (tri-color diagram)

Figure 5: Example of results of Naive Optimal Coefficients algorithm

When comparing the images for normal luminance (Figure 5b) and optimal luminance (Figure 5c) even casual observation suffices to verify that the optimal luminance image emphasizes differences in data more than the normal luminance image. The histogram equilibrium for normal and optimal luminance images confirm the increase in data fidelity—this implies that processes such as motion estimation can be computed more accurately and more efficiently on the optimal luminance image than on the normal luminance image.

The increase in accuracy is afforded by the greater amount of information which allows a more accurate detection of features and patterns. The increase in computational efficiency is indirect: due to the increase in accuracy it is possible to use smaller sub-images for feature and pattern matching, and this in turn allows computations to be resolved quicker.

The primary drawback of this approach to computing optimal luminance is the high computational cost of calculating the optimal luminance coefficients. A modern desktop can be expected to require several seconds to compute the optimal weights. For example, the image in Figure 5a ($256 \times 256 \times 24$ at a resolution of 0.01) requires approximately 5 seconds of processing time on a 2Ghz desktop system to resolve the implementation of "Naive Optimal Coefficients" used throughout this text. Although there is room for improvement in the implementation, it cannot be expected to meet the expectations of high performance or real-time applications.

At this point some results presented later in this chapter are preempted to allow the introduction of a faster algorithm to compute optimal variance. The results displayed in Section 2.4 illustrate three significant features of the optimal luminance solution space:

> *Optimal coefficients are nearly always found along the edge of the tri-color diagram. Histogram equilibrium varies continuously throughout the tri-color diagram. A tri-color diagram may possess multiple local optima.*

Intuitively, and according to the Lambertian model [83], it makes sense that optimal coefficients would tend to be found along the edges of the solution space. The reason for this is that the majority of image data is duplicated to some extent throughout the color channels—by giving equal weight to each color the amount of information that is stored per color is reduced, which in turn decreases the amount of information.

The continuous variation of histogram variance indicates that a greedy strategy could be used in an algorithm to solve for optimal luminance weights. However, as Figure 12b on page 25 demonstrates, images may possess a highly varied and complex optimal luminance solution space. The presence of local optima implies that a greedy algorithm is not guaranteed to find the most optimal solution.

Program 2, below, stipulates a heuristic algorithm that can be used to compute optimal luminance weights. The heuristic used is twofold. Firstly, only solutions along the edges of the tri-color diagram are considered. Secondly, a greedy approach is used, in spite of the inability to guarantee the best possible solution.

Program 2 Heuristic Optimal Coefficients

function GetHeuristicWeights

Takes input image and search resolution and returns near-optimal weights and variance along red-green, green-blue and red-blue borders

input: image, res
output: (α_r, α_g, α_b), h

```
begin
  sr = GetH( GetWeightedLuminance(image, 1, 0, 0) )
  sg = GetH( GetWeightedLuminance(image, 0, 1, 0) )
  sb = GetH( GetWeightedLuminance(image, 0, 0, 1) )

  rg = GetRecursiveHeuristic(image, res, 1, 0, 0, sr, 0, 1, 0, sg)
  gb = GetRecursiveHeuristic(image, res, 0, 1, 0, sg, 0, 0, 1, sb)
  rb = GetRecursiveHeuristic(image, res, 1, 0, 0, sr, 0, 0, 1, sb)

  return best solution based on rg, gb, rb
end
```

function GetRecursiveHeuristic

Takes input image, search resolution, starting point, starting variance, end point, and end variance—returns near optimal weights and variance along start and end points

input: image, res, (wr, wg, wb), wh, (vr, vg, vb), vh
output: (α_r, α_g, α_b), h

```
begin
  tr = 0.5 * (wr + vr)
  tg = 0.5 * (wg + vg)
  tb = 0.5 * (wb + vb)
  th = GetH( GetWeightedLuminance(image, tr, tg, tb) )

  if max( abs(wr - tr), abs(wg - tg) ) < resolution then begin
    return best solution based on wh, th, vh
  end else begin
    if wh < vh then begin
      GetRecursiveHeuristic(image, res, wr, wg, wb, wh, tr, tg, tb, th)
    end else begin
      GetRecursiveHeuristic(image, res, tr, tg, tb, th, vr, vg, vb, vh)
    end
  end
end
```

The second heuristic requires a slightly more elaborate justification. The heuristic considers that an optimal solution can produce the best possible histogram variance; however, finding such an optimal solution may require extensive processing time. A sub-optimal solution falls short of the best solution; however, the heuristic assumes that the sub-optimal solution is near the optimal solution and still offers a considerable improvement in histogram balance. To summarise, the heuristic trades accuracy of solution in favor of higher execution speed.

The "Heuristic Optimal Coefficients" algorithm makes use of recursion to compute the result. The *GetHeuristicWeights* function makes use of the recursive function *GetRecursiveHeuristic* to compute the best solution along each of the three edges of the tri-color diagram, and then returns the best of the three solutions.

The *GetRecursiveHeuristic* function receives two coordinates (start- and end-point) on the tri-color diagram, along with an evaluation of the variance at those points. It then computes the mid-point between the starting and ending coordinates and evaluates the variance at that location. If the distance between start-point and end-point is less than the search resolution the recursion is halted and the best current solution is returned, otherwise the recursion proceeds between start-point and mid-point or mid-point and end-point—depending on whether the starting or ending coordinates possess a more optimal variance.

The computational complexity of "Heuristic Optimal Coefficients" is of the order $O(\ln n)$, where n specifies the resolution of the search. To see that it has a logarithmic complexity consider that it computes three sets of points through recursion, where each recursive call halves the search space until the resolution condition is met.

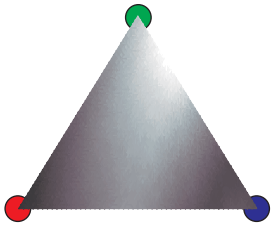
The following section illustrates the results of optimal luminance experiments. Additionally comparative data on the "Naive Optimal Coefficients" and "Heuristic Optimal Coefficients" is provided.

2.4 Optimal luminance results

A set of different images with varying color properties was used to test the efficiency of "Naive Optimal Coefficients" and "Heuristic Optimal Coefficients" in maximising data fidelity. Each image is accompanied by a histogram variance rating for each of these algorithms, as well as a rating based on normal luminance computations.

The format will present the original image and a tri-color diagram for luminance weights, as well as the normal and the optimal luminance images. The histogram data of both the normal and optimal luminance are presented along with a summarizing table. Note that it is possible for the heuristic algorithm to find solutions that are more optimal than the naive algorithm as they do not consider the same set of points. They merely share a common resolution parameter that indicates to what degree a solution is searched for.

(a) Original image



(b) Tri-color diagram

(c) Normal luminance

(d) Optimal luminance

Figure 6: Image - Cape

Weights	Red	Green	Blue	Variance
Naive	1.000	0.000	0.000	25368
Heuristic	1.000	0.000	0.000	25368
Luminance	0.299	0.587	0.114	27901
Gain from Naive algorithm				9.08%
Gain from Heuristic algorithm				9.08%

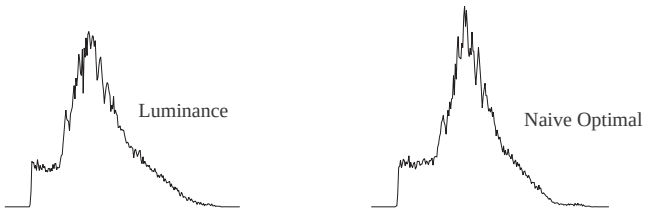
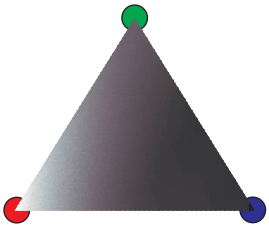


Figure 7: Histographs - Cape

(a) Original image



(b) Tri-color diagram

(c) Normal luminance

(d) Optimal luminance

Figure 8: Image - Hikaru

Weights	Red	Green	Blue	Variance
Naive	0.000	0.000	1.000	11798
Heuristic	0.000	0.030	0.970	11663
Luminance	0.299	0.587	0.114	19486
Gain from Naive algorithm				39.45%
Gain from Heuristic algorithm				40.15%

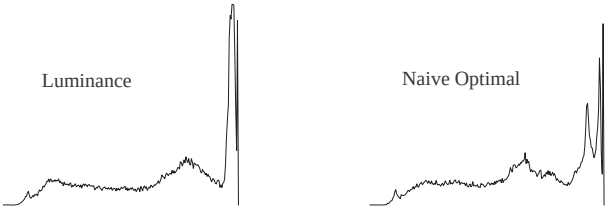
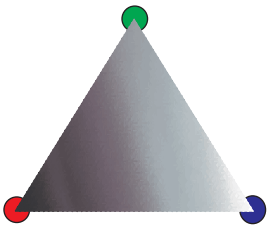


Figure 9: Histographs - Hikaru

(a) Original image



(b) Tri-color diagram

(c) Normal luminance

(d) Optimal luminance

Figure 10: Image - Girls

Weights	Red	Green	Blue	Variance
Naive	0.980	0.020	0.000	14319
Heuristic	1.000	0.000	0.000	14462
Luminance	0.299	0.587	0.114	19509
Gain from Naive algorithm				26.60%
Gain from Heuristic algorithm				25.87%

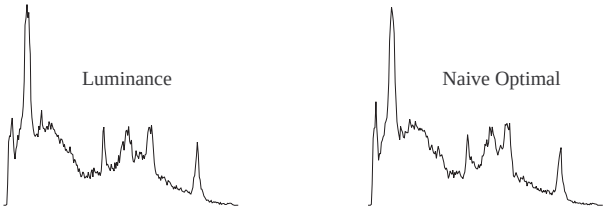
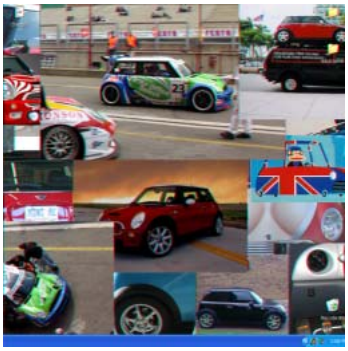
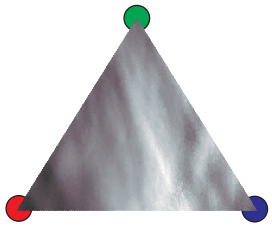


Figure 11: Histographs - Girls



(a) Original image



(b) Tri-color diagram

(c) Normal luminance

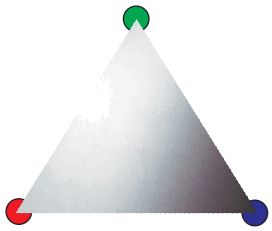
(d) Optimal luminance

Figure 12: Image - Cars

Weights	Red	Green	Blue	Variance
Naive	0.970	0.000	0.030	11555
Heuristic	0.970	0.000	0.030	11555
Luminance	0.299	0.587	0.114	12885
Gain from Naive algorithm				10.32%
Gain from Heuristic algorithm				10.32%

Figure 13: Histographs - Car

(a) Original image



(b) Tri-color diagram

(c) Normal luminance

(d) Optimal luminance

Figure 14: Image - Hut

Weights	Red	Green	Blue	Variance
Naive	0.000	0.000	1.000	26440
Heuristic	0.000	0.000	1.000	26440
Luminance	0.299	0.587	0.114	31139
Gain from Naive algorithm				15.09%
Gain from Heuristic algorithm				15.09%

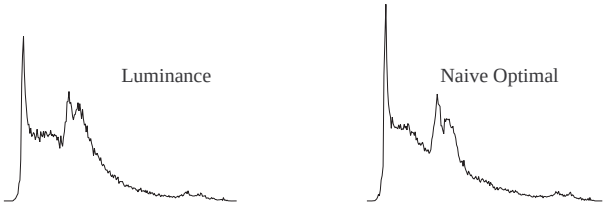
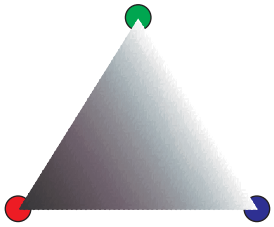


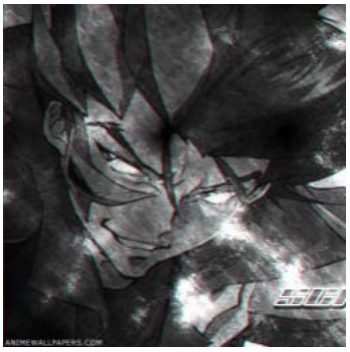
Figure 15: Histographs - Hut

(a) Original image



(b) Tri-color diagram

(c) Normal luminance



(d) Optimal luminance

Figure 16: Image - Scryed

Weights	Red	Green	Blue	Variance
Naive	0.990	0.010	0.000	26170
Heuristic	0.990	0.010	0.000	26170
Luminance	0.299	0.587	0.114	40532
Gain from Naive algorithm				23.17%
Gain from Heuristic algorithm				23.17%

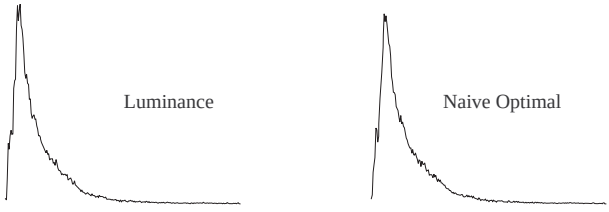
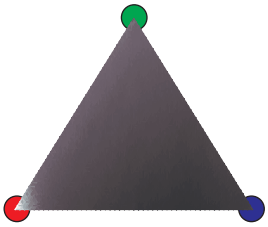


Figure 17: Histographs - Scryed

(a) Original image



(b) Tri-color diagram

(c) Normal luminance

(d) Optimal luminance

Figure 18: Image - Sung

Weights	Red	Green	Blue	Variance
Naive	0.330	0.000	0.670	33874
Heuristic	0.250	0.000	0.750	33862
Luminance	0.299	0.587	0.114	34444
Gain from Naive algorithm				1.65%
Gain from Heuristic algorithm				1.69%

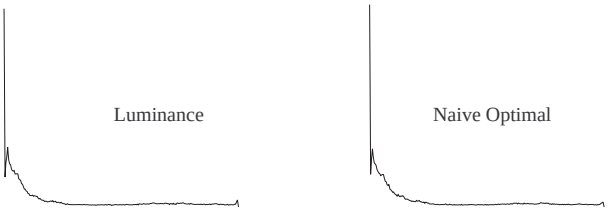
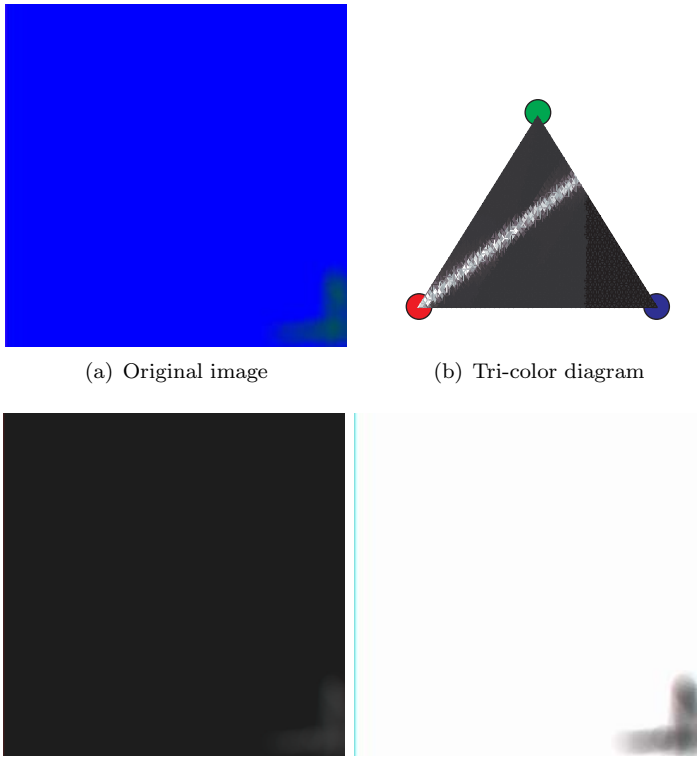


Figure 19: Histographs - Sung



(a) Original image (b) Tri-color diagram

(c) Normal luminance (d) Optimal luminance

Figure 20: Image - Test

Weights	Red	Green	Blue	Variance
Naive	0.000	0.580	0.420	63078
Heuristic	0.010	0.000	0.990	63078
Luminance	0.299	0.587	0.114	63139
Gain from Naive algorithm				0.10%
Gain from Heuristic algorithm				0.10%

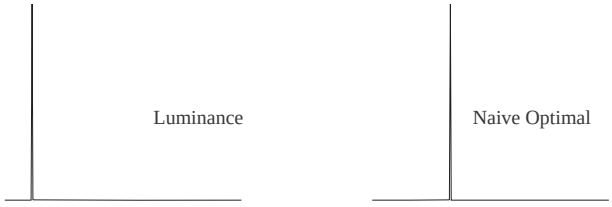
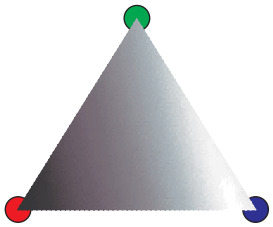


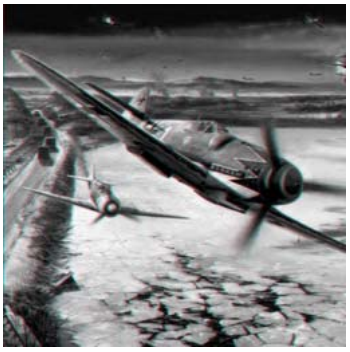
Figure 21: Histographs - Test

(a) Original image



(b) Tri-color diagram

(c) Normal luminance



(d) Optimal luminance

Figure 22: Image - War

Weights	Red	Green	Blue	Variance
Naive	0.960	0.020	0.020	9538
Heuristic	0.940	0.060	0.000	9548
Luminance	0.299	0.587	0.114	17804
Gain from Naive algorithm				46.42%
Gain from Heuristic algorithm				46.37%

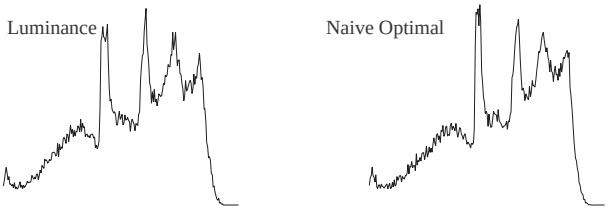


Figure 23: Histographs - War

The results on the previous pages demonstrate that in nearly all cases optimal coefficients are found along the edges of tri-color diagrams. The only exception is found in Figure 23 on page 30, and even in that case the optimal solution is near the edges of the tri-color diagram. The examples are certainly not conclusive, but they do allow the formulation of a heuristic which produces results comparable to the "Naive Optimal Coefficients" approach.

Some may note that in some cases, such as Figure 8, the heuristic outperforms the naive solution. This is due to the fact that the two algorithms do not consider the exact same sample points—the naive algorithm linearly considers all samples within a resolution parameter, while the heuristic approach uses a divide-and-conquer recursion. These differences cause the two algorithms to consider different samples and it is thus possible for the heuristic to slightly improve on the naive approach.

The heuristic, however, can be simplified further by noting that optimal results are not only found along the edges of tri-color diagrams, but more specifically near their vertices. This simplification allows applications to only consider the existing red, green and blue color channels to determine a compact image representation with high data fidelity.

The simplified heuristic allows implementations to save the processing time of computing the luminance of an image, in favor of a quicker analysis of the histogram balance of the red, green and blue color channels. Such an analysis requires less computing time than a luminance computation, and the resulting optimal luminance generally possesses a higher data fidelity than the corresponding standard luminance.

Table 1 summarizes the results of this chapter and illustrates the near optimal nature of color-channel-only optimal luminance.

Image	Standard Luminance	Naive Optimal Coefficients	Optimal Channel	Naive Optimal Gain %	Channel Gain %
Cape	27901	25368	25368	9.08	9.08
Hikaru	19486	11798	11798	39.45	39.45
Girls	19509	14319	14462	26.60	25.87
Cars	12885	11555	11724	10.32	9.01
Hut	31139	26440	26440	15.09	15.09
Scryed	40532	26170	26170	35.43	35.43
Sung	34444	33874	33913	1.65	1.54
Test	63139	63078	63078	0.10	0.10
War	17804	9538	9613	46.42	46.01

Table 1: Summary of optimal luminance results

The last section of this chapter presents closing thoughts on optimal luminance. Suggestions for additional work, as well as an alternative approach to achieve high data fidelity are presented.

2.5 Closing thoughts and suggestions for further work

This chapter presented findings on optimal luminance computations, a method for achieving greater data fidelity in compact space. The solution presented in this chapter utilizes a metric that considers histogram balance in an image as an indicator to data fidelity.

It is possible to define alternative metrics that emphasize other image properties to maintain a high data fidelity. One such alternative is to keep the existing histogram variance metric, but refine it to sub-images of the graphic data. For example, typically motion estimation at a particular point only considers a small subset of the full image data when performing estimations—it is therefore possible to use the histogram variance metric localized to the current region to ensure high data fidelity for that particular subset.

On the other hand, it is possible to approach the problem of computing a compact representation of the image data that maintains a high threshold of information from a different angle. For example, it is possible to use the method of histogram equalization on a normal luminance image to ensure that data subsets of the image can be easily differentiated. Such a method is suitable for some computational problems, but not all, as the histogram equalization process typically introduces new data to the image. This can lead to falsification of results.

The following chapter presents the visual environment used during the research process. It describes the strengths and weaknesses of the representation and offers a short account on its development.

> *Basic research is like shooting an arrow into the air and, where it lands, painting a target.*
>
> Homer Burton Adkins

Chapter 3

Visual Environment

> *A rock pile ceases to be a rock pile the moment a single man contemplates it, bearing within him the image of a cathedral.*
>
> Antoine De Saint-Exupery

The words of Antoine De Saint-Exupery remind us that perception is greater than the sum of individual visual stimuli—the questions of intent and potential are part of the considerations on perception. Two questions that present themselves in the field of structure-from-motion are *what is to be reconstructed* and *how is the data to be reconstructed obtained*.

The reconstruction process under consideration for this thesis is that of fully enclosed, static environments. Thus not individual objects are processed; but instead immersive, continuous environments such as the inside of a building are taken into account. To draw the comparison to the preceding quote: the reconstruction should not reproduce a model of a cathedral as seen from the outside, but instead produce a model of the depths perceived while traversing the rooms and halls inside of a cathedral.

More precisely, the reconstruction process considers input data that is within the boundaries defined by the following assumptions:

- *The visual data represents a fully enclosed environment.* This implies that all of the visual data at each point in time represent only the environment to be reconstructed. An example of an environment that is not fully enclosed is that of a room with an open door that permits sight onto the horizon.
- *The environment is static.* This implies that the environment is rigid and undergoes no changes. A door opening and closing is an example of dynamic data in an enclosed environment.

- *The environment consists entirely of opaque elements that possess no light reflective or refractive properties.* Correspondingly, no partially transparent, reflective or refractive entities, such as a pool of water, are present in the environment.
- *The environment is perceived through a simple perspective camera model.* This ensures that the visual data mimics the data perceived by the human eye. However, unlike the human eye, the perspective camera model does not distinguish objects through depth focus. All entities perceived are clearly in focus.
- *The input data is free of all forms of distortions.* This implies that the perceived data is not geometrically transformed through lens refraction, chromatically transformed due to lighting conditions, nor is the data contaminated with visual artifacts such as lens glare or environmental hueing. The human eye, in contrast, changes its perceptive qualities depending on the brightness and ambient-hue of the environment, as well as the distance of objects relative to the eye.

It is not strictly necessary to perfectly adhere to these restrictions. Slight deviation from the restrictions above do not cause the reconstruction process to fail—it merely tarnishes the quality of the results slightly. For example, distortion in the input data may produce a distortion in the resulting three-dimensional model; the degree of the distortion in the input determines whether the final results are acceptable.

The visual environment that serves as input to the reconstruction process is, for the purposes of this thesis, generated in real time using a virtual three-dimensional environment. Some experiments are performed using real samples, but the majority are computed from virtual models.

The advantages of a virtual, simulated environment are numerous. It is a simple matter to ensure that all the assumptions regarding input data are met. Precise control over camera motion and camera properties can be exerted. Furthermore the true structure and the reconstructed structure data can be easily correlated and compared. A virtual environment allows the easy reproduction of input data, as well as affecting slight variation in input data for experimental purposes. Allowing the virtual environment to run in real-time allows interactive control over the visual input. Finally, the use of a virtual environment is exceptionally cost-effective—expenses for quality camera equipment as well as precise positioning machinery and software are eliminated.

The following section will present the background and current development of interactive virtual environments. Section 3.2 will describe and illustrate the virtual environment developed for this study.

3.1 Background and current development

Virtual environments have evolved at a staggering pace in the last two decades. The range of implementations include both military, industrial, educational and research applications—as well as the gaming products of the electronic entertainment industry.

The entertainment industry is the dominant driving force in the development of new algorithms and hardware in the quest to create ever more immersive and complete virtual environments. This should come as no surprise, as the market for electronic entertainment has increased rapidly in the last decade. The growth in recent years has allowed the industry to exceed even Hollywood in annual earnings [89].

Early efforts were primarily made as proof of concept demonstrations by dedicated groups of enthusiasts in a digital-based subculture known as the Demoscene [105]. The Demoscene evolved from small graphical and musical demonstrations that were used by crackers as signatures in the early 1980's.

Typically a gaming company attempts to produce products that are stable and possess the desired features necessary to the game. In contrast, demosceners are concerned foremost with the mastery of advanced special effects and excellence of artistic representation. Traditionally a demo programmer would go to great lengths to find ways to exploit undocumented hardware features to allow his effects to excel.

Arguably the most influential demo group in the history of the Demoscene is the Future Crew. They have a surprisingly small selection of releases, but each was a technical masterpiece. Their demo "Second Reality", released in 1993, could be described as pivotal in that it shaped content and production standards of demo releases for years to come.[1]

One of the earliest commercially successful fully immersive virtual environments was encapsulated by the Freescape Solid 3D engine developed by Incentive Software[2]. The engine served as the basis for a series of games such as "Driller" (1987), "Total Eclipse" (1988) and "Castle Master" (1990).

The breakthrough for virtual reality gaming began with "Wolfenstein 3D" in 1992. It was developed by id Software and served as their first three-dimensional gaming milestone. However, it was the release of "Doom" in 1993 that completely opened the floodgates to first-person perspective gaming. Finally "Quake" (1996) initiated a trend into completely three-dimensional immersive environments.

id Software went on to develop both the Doom and Quake franchises. Though each id Software product is a financial success, the company is primarily known for producing outstanding

[1] Quite a substantial number of demosceners have made their passions a career. For example, some of the various members of Future Crew have parted ways and formed companies such as FutureMark (known for its 3DMark series—programs to benchmark graphics hardware) and Remedy Entertainment (known for popular gaming titles such as Max Payne).

[2] The company changed its name to Superscape in 1991.

gaming engines that are licensed out to other companies. Typically after an engine reaches five years of age id Software releases it under the GNU GPL, essentially placing the engine into the public domain.

The progress of virtual reality systems has in recent years reached an exceptionally high standard of immersion. Environments are, visually, near photo-realistic, complex physics systems allow natural interaction with objects in the world, and environmental, positional audio cues complete the immersive illusion.

A side-effect of this ever more complete and realistic representation of the real world in a virtual environment is the development of non-photorealistic rendering methods. Modern graphics hardware supports the use of shaders that can control how vertices and pixels are processed by the rendering pipeline. Such shaders can be used to create both realistic effects—such as modelling light glare—as well as non-natural, artistic effects—such as cel-shading[3]; as demonstrated in Figure 24.

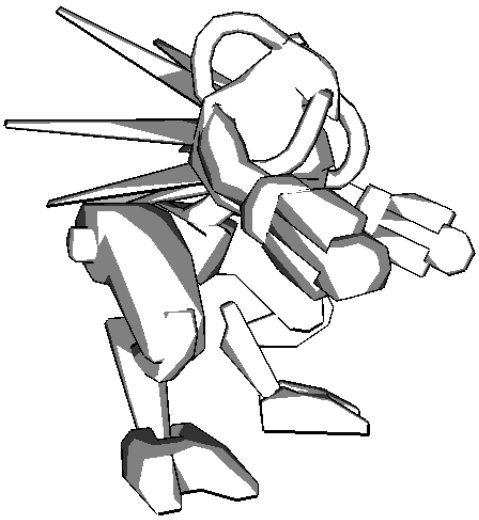


Figure 24: Example of cel-shading

The following section describes the virtual environment adopted for the thesis. Additionally it describes its development as well as the mechanisms involved.

[3]Cel-shading is a rendering technique that creates cartoon-like shading of objects.

3.2 Virtual environment

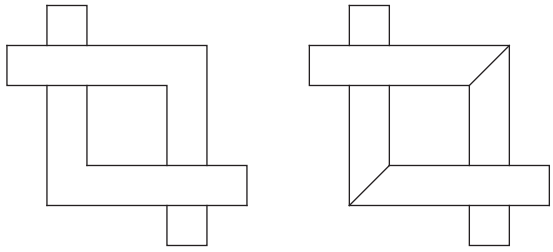
The Quake 3 engine has been a popular foundation for several entertainment titles and has proven itself as an exceptional rendering tool for indoor and enclosed environments. Additionally several free fan-made applications exist to create custom environments and maps for the engine. These features make the Quake 3 environment an effective tool for virtual reality application—correspondingly the virtual reality environment used in context of this thesis is an in-house graphics application capable of loading and interactively rendering Quake 3 map files.

The Quake 3 engine and its source was placed under GNU GPL in August 2005 [45], allowing free public domain access to use and learn from the sources. Numerous websites on the internet offer tutorials and explanations on the various functions, features, formats and file specifications used by the Quake 3 engine. Given the extensive online documentation of the engine it is a relatively simple task to implement a Quake 3 map file loader and renderer.

Algorithmically, the most significant aspect of an enclosed environment renderer is the occlusion and visibility determination. In the case of Quake 3 these are achieved primarily through binary space partitioning trees (BSP trees) and potentially visible sets (PVSs)—the remainder of this section introduces the background and theory of these abstract data structures.

3.2.1 Binary space partitions

Binary space partitions were first formulated by Fuchs [32] in 1980. They were initially intended as a solution to the problem of depth-sorting polygons and resolving special cases: as illustrated in Figure 25a it is impossible to determine which polygon is in front of the other as each polygon is partially in front of and partially behind the other. A BSP solves this problem by dividing the polygons, so that the smaller polygons can then each be accurately classified as in front of or behind the others. However, since their inception, the applications of binary space partitions have increased substantially.



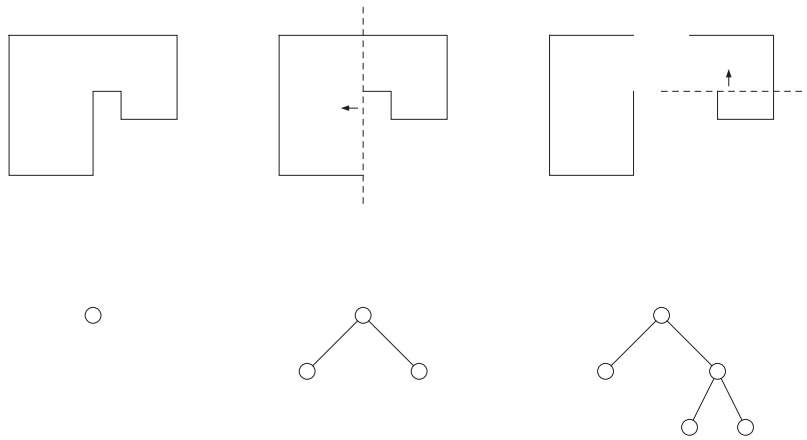
(a) Which polygon is deeper? (b) BSP solution

Figure 25: Depth conflict and resolution

Binary space partitions are used to accelerate a wide variety of occlusion related problems, for example: Chin and Feiner [21] who presented their approach to using BSP trees to efficiently compute shadow volumes in 1989; as well as Bittner *et al* [14] who use BSP trees as a scene representation to accelerate tests on an occlusion tree—a hierarchy of shadow frusta. The interested reader is referred to Wade [120] who maintains an excellent online FAQ on binary space partitions.

A binary space partition can be likened to a binary tree, applied not to numbers but to geometrical space—it sorts polygons into a logical tree. Similar to a binary tree the height—meaning the length of the path from root to its furthest leaf—of a balanced BSP tree is of the order of $\log n$, where n is the number of polygons in the tree.

A given node represents a logical plane that passes through the geometric space and divides all entities in the space into two subsets: that part of space that is in front of the dividing plane and that part of space that is behind the dividing plane. The child nodes of their respective parent nodes in turn divide the remaining subset of geometric space into an in front and a behind space. Figure 26 below demonstrates the process—the example chooses to create a binary space partition where dividing planes are chosen based on existing polygons, though this is not necessary for BSP trees in general. Furthermore, a leaf is not further refined if the set of polygons in it describe a concave space, instead the leaf contains a pointer to the set of polygons—as illustrated in Figure 27.



(a) Initial world space (b) First division (c) Final division

Figure 26: Binary space partition creation

The choice of dividing plane impacts the quality of the binary space partition. At first glance the best choice of the dividing plane is one which separates the space into two equally sized subsets. However, if a dividing plane passes through a polygon, that polygon needs to be replaced with two polygons that describe the original polygon as well as the cut made by the

dividing plane.

This process implies that the number of polygons to be sorted into a BSP tree can increase considerably during its creation, depending on the choice of dividing plane. Therefore it is paramount to choose a plane that divides the world into relatively equal subsets, while possessing a bias towards producing as few as possible additional polygons.

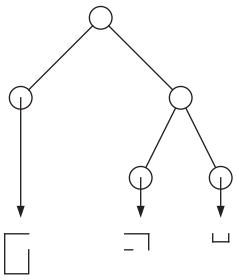


Figure 27: Binary space partition result

The requirements to choose a dividing plane that both produces a balanced tree, as well as the fewest possible polygon splits, are mutually exclusive. The problem of creating an optimal BSP tree is NP-complete [120]; in practice heuristics are used that are biased towards the intended use of the binary space partition: if the rendering context is emphasized then polygon splitting is minimised as the rendering performance is a function of the contents of the entire BSP tree. In contrast, in applications where the BSP trees are primarily used to apply algorithms on the spatial divisions of the tree, a balanced tree is favored.

The advantages of applying algorithms to binary space partitions is that in many cases the order of computational complexity can be reduced considerably [120]. Problems that would require operations in the order $O(n)$ when applied to the full geometric set of a world are reduced to $O(\log n)$, and problems that are solved in the order of $O(n^2)$ are reduced to $O(n \log n)$.

The reason for these improvements is that when a particular node can be excluded from consideration, its children also fall away from consideration. Examples of problems that can be accelerated using BSP trees are: collision detection, light mapping, shadow computation, physics computation, and frustum culling.

Although binary space partitions are effective structures for spatial operations, they are not commonly used in their original function as an ordering mechanism to solve the problem of visible surface determination[4]. Particularly advances in graphics hardware have solved many rendering difficulties that required extensive coding as little as two decades ago. In the case of BSP trees the key advance that makes binary space partitions obsolete as a rendering vehicle is the use of hardware Z-buffering.

[4]The problem of determining which polygons, and parts thereof, are visible. Practically equivalent to the problem of hidden surface removal—the problem of determining which polygons are not visible.

A Z-buffer is conceptually simple to understand and implement—especially on graphics hardware. The algorithm solves the problem of visible surface determination by storing the depth value of computed pixels and drawing a pixel only when it passes the Z-buffer test, in other words, it renders a pixel only if its corresponding depth value is nearer than the relevant Z-buffer entry. The development of the Z-buffer is generally credited to Edwin Catmull [19][5], though the first to publish the idea of the Z-buffer was Wolfgang Straßer [102].

Modern advances in graphics hardware make binary space partitioning trees unnecessary in terms of their rendering properties, however, Z-buffers and other hardware accelerations alone are not sufficient to meet the requirements of high-end interactive virtual reality environments. The next subsection briefly examines potentially visible sets (PVSs) as an additional algorithm for visible surface determination.

3.2.2 Potentially visible sets

Potentially visible sets are precomputed data sets that specify which parts of a graphics scene are visible from other parts of the scene. For example, in BSP trees, typically a PVS considers BSP leafs and stores a list of all other leafs that are visible from them.

When rendering a scene from a binary space partition, the rendering algorithm parses into the BSP tree to locate the leaf in which the camera is currently positioned. Then it references the potentially visible set of that leaf and uses the list of potentially visible leafs to render the scene. Typically some additional culling of potentially visible leafs is performed, for example, the bounding boxes of potentially visible leafs is compared with the view frustum of the camera—any leafs rejected from this culling step are not rendered.

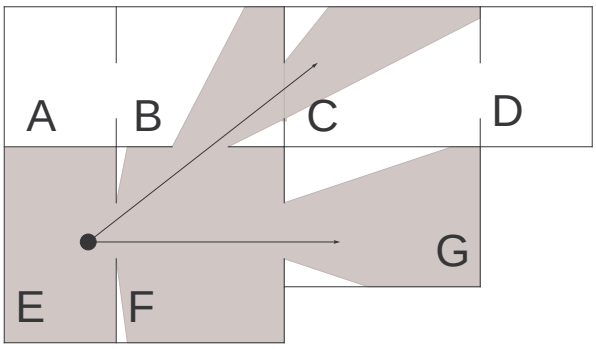


Figure 28: Example of potentially visible set

[5]Edwin Catmull is possibly the most celebrated academic figure in the field of computer graphics. His research output includes not only the Z-buffer, but also texture mapping and bi-cubic patches. Dr Catmull is the president, and co-founder, of Pixar Animation Studios—he was the lead developer of Renderman, rendering software which is responsible for a wide range of computer animated clips and movies such as *Toy Story* and *Finding Nemo*.

Figure 28 illustrates the idea of potentially visible sets. Room E is used as source leaf for the demonstration and the potentially visible rooms are B, C, F and G. Thus rooms A and D are not considered for rendering. Figure 29 below depicts the corresponding stab graph.

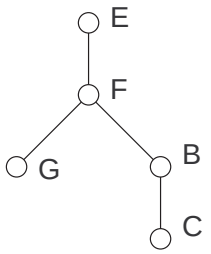


Figure 29: Stab graph of potential visibility from room E

Stab graphs can be used in conjunction with view frustum culling. If a particular node can be eliminated, then all its children fall away as well. To illustrate, consider Figure 30: in the stab graph above, if room B is determined to be outside the view frustum, then it and all its children (room C) are culled as well, even though room C may be within the view frustum.

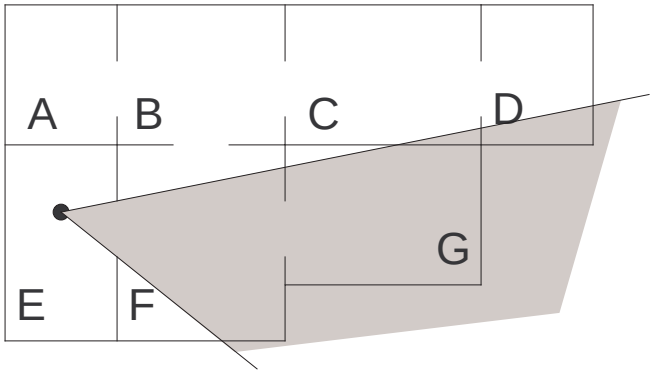


Figure 30: View frustum culling

Potentially visible sets are a common preprocessing step in many algorithms. For example, Airey [4] as well as Teller and Séquin [107] detail the use of such sets to perform occlusion culling.

Potentially visible sets are also used in other environments, for example Stewart [101] proposes the use of potentially visible sets for terrain rendering. However, typically a PVS is used in urban and indoor environments which are populated with many large occluders that allow efficient culling of graphics data.

A prominent problem in the use of potentially visible sets is that of PVS data storage.

Complex scenes, by their nature, require a great number of convex sections, sometimes known as cells, which form the basis of potentially visible tests. This implies that for n cells an order of n^2 potentially visible pairings need to be considered.

Different strategies exist to resolve the problem of PVS data size. For example, Gotsman *et al* [39] make use of a hierarchical visibility classification scheme to reduce data volume. Koltun *et al* [62] achieve a similar effect through the definition of a *virtual occluder*—a view-dependent object that is occluded from any point within the current cell; also by the same authors is a hardware accelerated attempt at computing relevant PVS information in real time [63]. The idea of real-time PVS computation is not entirely new. It is reminiscent of an earlier approach by Luebke and Georges [70] which makes use of cells and connecting portals to perform an on-demand PVS computation.

For this thesis the method adopted, by default, is the one used by the Quake 3 engine. It is elegant in its simplicity: each cell is uniquely identified by a cell identification number, the potentially visible data is a sequence of booleans such that the first boolean indicates whether cell number one is visible, the second boolean refers to the second cell, and so forth. The actual boolean information for particular cell visibility can be stored in a single bit, and thus the PVS data is represented as a continuous bitstream.

The model that is described is usually an indoor or urban environment, meaning large parts of the scene are typically occluded from cells. Thus the resulting potentially visible data consists of predominantly 0's—indicating that corresponding cells are not visible—as well as occasional clusters of PVS data with higher complexity. The Quake 3 engine efficiently stores this data using zero run-length encoding, which reduces the required storage space by approximately two orders of magnitude.

The following subsection briefly discusses alternative virtual environments that could be considered as visualisation tools. Additionally, in closing, it presents images from the visual environment used for this study.

3.2.3 Alternatives

Instead of a custom-made Quake 3 map renderer, it is possible to consider a host of alternate options to serve as a visualisation tool. These can be roughly divided into external libraries and custom made libraries.

A custom visualisation engine could be created from first principles, and can thus, in theory, reach any desired complexity. However, in practice creating a suitably advanced virtual environment requires a considerable investment in time and extensive expertise in the field of graphics. Furthermore the basic rendering engine needs to be supplemented with content creation tools that allow importing and exporting of various formats, as well as creation and editing of maps and materials.

Often, when creating a custom visualisation tool, it is prudent to pick an existing rendering package and implement its specifications. This normally limits the features available to those implemented by the relevant source engine. However, it reduces research and programming time and offers considerable pre-existing infrastructure in the form of meta-applications to design and edit content, as well as online forums, mailing lists and support groups dedicated to the engine.

Most popular entertainment titles have online documentation of the relevant formats used in maps and models, as well as third party tools that allow customization of content. Popular examples include the series of Quake engines, as well as Battlefield [7], Half-Life and Half-Life Source [112]. Often it is not necessary to code a custom engine based on an existing one, but to make use of its custom content creation tools—such as the custom modification Source SDK that is based on the Half-Life Source engine [113]. Such solutions can make use of tools such as Fraps[6] to export the output from the renderer to video or image sequences [12]. In principle Fraps allows the use of any existing visual environment tool to function as the basis for video sequences.

Instead of creating a custom engine it is possible to use existing rendering engines. Depending on the requirements it is possible to licence commercial libraries, such as the cutting edge Doom 3 engine, which are sometimes made available at a sizable discount when used for academic purposes. Alternatively a host of free real-time rendering libraries is available, including high profile candidates such as Irrlicht [35], OGRE [103], and Genisis3D [31].

Figure 31: Visual environment

[6] A real-time video capturing tool.

Additional alternatives include the use of VRML (Virtual Reality Modeling Language) [25], CAD (Computer Aided Design) programs, and naturally the use of a video camera. VRML is readily accessible to computer users as it requires little programming and graphics knowledge. CAD programs, such as 3D Studio Max [1], require no programming skills—though creating complex 3D sequences may require considerable artistic effort. Finally, making use of a video camera is perhaps the easiest method of creating realistic video sequences to serve as reconstruction input. However, it is difficult to ensure accurate perspective and motion parameters and it is hard to produce controlled experimental variation for real world video.

In closing, Figure 31 above illustrates the capabilities of the graphics engine utilized in the thesis. The following chapter describes the motion estimation techniques used in conjunction with the renderer.

> *Because of the nature of Moore's law, anything that an extremely clever graphics programmer can do at one point can be replicated by a merely competent programmer some number of years later.*
>
> John Carmack

Chapter 4

Motion estimation

Colors answer feeling in man; shapes answer thought; and motion answers will.

John Sterling

Though taken well out of context, the words by John Sterling echo the basic stratagems employed in motion estimation techniques: motion estimation is, at heart, performed by tracking the flow of colors in image sequences, or—in more resource intensive approaches—tracing the motion of shapes.

The task of accurate and fast motion estimation is a crucial cornerstone for a great number of applications including, though not limited to, surveillance and navigation tasks as well as image and video compression. Mitiche and Bouthemy [77] provide a useful overview of the problems inherent in motion estimation, and Beauchemin and Barron [11] offer a survey and classification of motion estimation techniques.

This chapter introduces the motion estimation solution employed in this thesis. The following sections outline the existing work on motion estimation, as well as the solution and optimization processes employed for the purposes of this study.

4.1 Background

The taxonomy of motion estimation is hard to discuss without making reference to motion compensation. These two fields are traditionally closely linked—often techniques employed in solving either motion estimation or motion compensation are interchangeably utilized in the other as well. Nonetheless, the two fields of motion estimation and motion compensation attempt to achieve different goals and therefore require the underlying paradigms to shift emphasis.

Motion estimation is concerned with the detection of changes in an image sequence that is due to the motion of image objects relative to the camera. Motion compensation, on the other hand, attempts to predict pixel values in a particular image frame from the pixel intensities in other frames; the purpose of motion compensation is to eliminate inter frame (or temporal) redundancy in image sequences.

Motion estimation primarily focuses on the accuracy of results, whereas motion compensation typically emphasizes how well the data in image sequences can be compressed. It is relatively easy to compare different motion compensation techniques based on the compression gain that is afforded by the techniques, however, motion estimation algorithms cannot easily be evaluated as they require comparison with known real world motion data to make meaningful statements on respective accuracy and efficiency. As a consequence, motion estimation techniques often have to rely on qualitative evaluation or evaluation based on artificially generated data[1].

Most solutions to both motion estimation and compensation can be divided into two categories: *Feature correspondence* and *differential* methods, also referred to as *correlation* and *pixel-recursive* methods. The following subsections will detail how features may be chosen, feature correspondence methods, differential methods, and the Kanade-Lucas-Tomasi tracking algorithm.

4.1.1 Feature selection

Some, though not all, motion tracking methods rely on identifying strong features that can be easily discerned. This begs the question how such suitable features can be found.

A popular method [99] relies on image gradients: given an image that is described through its pixel intensities $I(x, y)$ and a window W that defines the feature under consideration, compute a gradient $\mathbf{g}$ such that

$$\mathbf{g} = \nabla I_W = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

where g_x is the derivative of intensity at an image point in the horizontal direction and g_y is the derivative in the vertical direction. Using $\mathbf{g}$, define the product

$$\mathbf{g}\mathbf{g}^{\mathrm{T}} = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix},$$

and then the integral Z is given by integrating over area W such that

$$\mathbf{Z} = \iint_W \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \omega \, dx \, dy \tag{2}$$

where ω is a weighting function. Choosing $\omega(x, y) = 1$ places equal emphasis on all parts of the feature window, though other choices are common as well. For example, some applications

[1] Well-known examples of artificial sequences are the Translating Tree [68] and Diverging Tree [10].

choose a Gaussian weighting function to emphasize feature elements in the centre of the window. A centre-biased weighting function is particularly relevant in applications where scenes may undergo complex and unpredictable transformations.

The 2×2 matrix Z contains purely textural data that can be analysed to detect feature edges, corners and intersections within W. Typically the eigenvalues of Z are considered. The eigenvalues are small for uniform intensity patterns within W. One small and one large eigenvalue indicate the presence of a unidirectional change in intensity, and two large eigenvalues correspond to a bidirectional change in intensity—strong changes in two directions—in other words a readily trackable feature within W.

A common problem is defining when eigenvalues are sufficiently large to be classified as a viable feature for tracking purposes. The magnitude of the eigenvalues of the most prominent trackable features in one image may differ significantly from the magnitude of the eigenvalues in another image.

The problem is typically solved by accepting a minimum threshold that eigenvalues need to achieve before being classified as trackable features. Alternatively, some approaches compute the feature viability of a complete image and choose the most prominent set of features from that analysis.

A popular measure is the Harris *cornerness* function, originally presented by Harris and Stephens [41]. Large values of this function indicate the presence of a corner: the larger the value, the stronger the corner. The function is defined as

$$R = \det \mathbf{Z} - k(\text{trace } \mathbf{Z})^2$$

with

$$\begin{aligned} \det \mathbf{Z} &= \lambda_1 \lambda_2 &&= Z_{11} Z_{22} - 2 Z_{12} \\ \text{trace } \mathbf{Z} &= \lambda_1 + \lambda_2 &&= Z_{11} + Z_{22} \end{aligned}$$

where λ_1 and λ_2 are the eigenvalues of $\mathbf{Z}$, and k is a constant. Usually k is chosen to be equal to 0.04, which yields optimal results according to Harris and Stephens [41].

The Harris cornerness function makes intuitive sense if the two terms are considered independently: the determinant of $\mathbf{Z}$ is simply the product of its eigenvalues and, as indicated previously, larger eigenvalues in $\mathbf{Z}$ indicate more reliably tracked features.

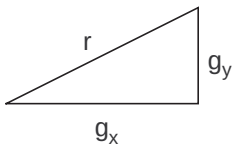


Figure 32: Significance of the trace term

To appreciate the significance of the second term, consider the case where only a single pixel is considered, in other words no integration step is required

$$\mathbf{Z} = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}.$$

It follows that the trace in the second term is computed as trace $\mathbf{Z} = g_x^2 + g_y^2$. In other words the trace of $\mathbf{Z}$ can be interpreted as the square of the hypotenuse, r, as shown in Figure 32.

Note that if $g_x + g_y$ is constant, then the values for r in the equation $r^2 = g_x^2 + g_y^2$ possess a global minimum when $g_x = g_y$. The magnitude of the hypotenuse is smaller the more alike the values of g_x and g_y are. Since the second term is subtracted from the first term it follows that a smaller r results in a larger R. Although the above description uses a special case to illustrate the significance of the second term, the conclusion applies to all $\mathbf{Z}$—in other words the second term creates a bias towards equivalent gradients.

This concludes the description of feature identification techniques. The following section presents the class of motion estimation techniques known as feature correspondence methods.

4.1.2 Feature correspondence

Feature correspondence, or correlation, methods attempt to match patches of a particular frame with the most similar patch in other frames. The underlying assumption of these approaches is that, locally, pixel motion is uniform—therefore pixels may be grouped into small patches and their motion may be considered collectively.

Similarity measures

It is possible to define some form of similarity metric for such pixel patches and search for the best matching patches in subsequent or prior image frames. Most commonly such similarity metrics are defined as the per-pixel sum of a comparison function:

$$D = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f\left(I(x, y, t), I(x + u, y + v, t + \Delta t)\right)$$

where D is the measure of similarity, n is the length of the square pixel patch considered, f is the comparison function, and $I(x, y, t)$ is the pixel intensity at position (x, y) in frame t. The displacement vector (u, v) is used to search for the pixel patch in the next frame $t + \Delta t$ that yields the greatest similarity to the reference patch.

Good initial definitions for the comparison function are the absolute difference, $f(a, b) = |a - b|$, the squared difference, $f(a, b) = (a - b)^2$, or the correlation coefficient, $f(a, b) = ab$. These definitions for f are often used in motion compensation problems, though for the purposes of motion estimation these functions have subtle flaws.

Both the absolute and squared difference functions are sensitive to uniform intensity variation in patches—such as changes in ambient lighting—and the correlation coefficient is sensitive to changes in intensity magnitude—as may be caused by localised lighting. In the problem of motion compensation these sensitivities can benefit the analysis. The motion estimation problem, however, prefers to emphasize the texture of pixel patches—that is to say, it favors sensitivity to perceived shapes.

The generally preferred solution for motion estimation is the normalized correlation function. It makes use of the basic correlation coefficient function with adjustments to eliminate the effects of pixel intensities. For a given square pixel patch of size n define:

$$h(\mathbf{x}, t) = I(\mathbf{x}, t) - \bar{I}_t \ ,$$

where $I(\mathbf{x}, t)$ is the intensity of the pixel at position $\mathbf{x} = (x, y)$ in frame t, and $\bar{I}_t$ denotes the mean intensity of the pixel patch in frame t. Furthermore define the search location $\mathbf{y}$ as the sum of the position vector $\mathbf{x}$ and the displacement vector $\mathbf{u}$, and the search frame s as the sum of the reference frame t and a frame delta Δt:

$$\mathbf{y} = \mathbf{x} + \mathbf{u}$$

$$s = t + \Delta t \ .$$

Based on these definitions the normalized correlation coefficient function is given as:

$$f_{NCC} = \frac{\sum h(\mathbf{x}, t) h(\mathbf{y}, s)}{\left[\sum h^2(\mathbf{x}, t) \sum h^2(\mathbf{y}, s)\right]^{\frac{1}{2}}} \ ,$$

where the summation is performed over $\mathbf{x}$, the pixels inside the pixel patch.

It is possible to define other forms of correlation functions, for example: instead of performing a similarity measure based on direct pixel intensities the class of ordinal metrics ranks pixel patches according to individual pixel intensities and performs similarity computations based on these ranks. The rank data is then compared in the same manner that intensity data is usually compared. Figure 33 demonstrates such a ranking.

12	44	32
27	11	14

(a) Intensities

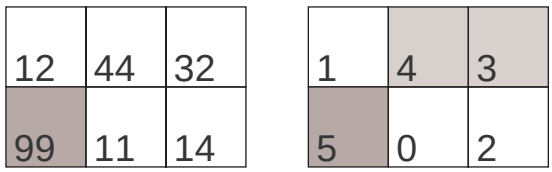
1	5	4
3	0	2

(b) Rank matrix

Figure 33: Ordinal ranking

Bhat and Nayar [13] demonstrated the application of such ordinal methods in the problem of stereo matching. Based on their work it is known that ordinal methods are robust against

single pixel variations (see Figure 34)—as the associated rank matrices with and without the single-pixel variation differ insubstantially. However, it is sensitive to Gaussian noise, especially in image regions with subtle variation in intensity. In contrast, direct intensity based approaches, such as the normalized correlation coefficient function, are prone to errors in single pixel variation and robust to Gaussian noise.



(a) Intensities (b) Rank matrix

Figure 34: Single pixel variation

Pixel patch size

The methods described up to now in this section assumed a fixed, square, pixel patch, or zone, that serves as reference to perform motion estimation. Typically 8x8 or 16x16 pixel blocks are used in practice. However, patch-matching techniques are not limited to square, or even fixed-size pixel regions—any shape, constant or dynamically changing could serve as the basis for matching algorithms.

It is well known that the size of pixel patches affects the sensitivity to motion [33, 59, 73]. Smaller pixel regions are sensitive to noise, while bigger regions are insensitive to fine motion. Several approaches attempt to compensate by adapting pixel patch sizes to underlying requirements, adaptively increasing size when the variation in local pixel intensity is low and decreasing in size when it is high. For example, according to Seferidis and Ghanbari [97] quad-tree segmentation is a typical solution to variable-sized pixel zones, as it offers an efficient balance between quad-tree refinement and representational complexity.

Several criteria exist for selecting suitable segmentation sizes. Kanade and Okutomi [59] present a method that determines patch size according to local pixel intensities and disparities. Seferidis and Ghanbari [97] make use of the *absolute temporal difference*; defined as the sum of absolute differences between matched blocks in subsequent frames.

Malo *et al* [73] classify such approaches as either based on prediction error, or based on a measure of the entropy of the resulting encoding. They note that refinement criteria do not always take into account the encoding method, for example, MPEG-1 compression transforms blocks into the frequency domain, and consequently basic prediction error metrics give a poor indication of compression performance. In such cases, according to Malo *et al*, it is prudent to

adapt a suitable metric to determine pixel patch refinement. In the case of MPEG-1 encoding they suggest a metric based on the spectral entropy of frame differences.

Apart from pixel patch size, the shape of the pixel region may also vary over time. This allows a more accurate tracking of specific motion events—at the cost of increased complexity since optimal transformation and translocation parameters need to be calculated. Examples of such work are found in Seferidis and Ghanbari [96, 97] who allow dynamically changing quadrilaterals of any convex shape.

Search methods

The final point of consideration in feature correspondence is that of search methods. Recall the use of a displacement vector (u, v) in similarity measures,

$$D = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f\left(I(x, y, t), I(x + u, y + v, t + \Delta t)\right) \ .$$

The displacement vector is used to search a region of an image frame for the best matching pixel patch for a given reference block. This begs the question how the set of search vectors should be determined.

The simplest approach is to exhaustively search all potential locations in the target frame; though typically this approach is made more efficient by assuming that there is a maximum velocity that needs to be considered—as features usually move relatively slowly. This velocity refers to motion in image space, rather than real space, thus allowing motion tracking to only consider displacements up to a particular radius. This naive approach is known as the full search algorithm, Figure 35 below illustrates the process.

Such a search strategy requires a computational complexity of $O(n^2)$, where n is the search radius—to show that the complexity if $O(n^2)$ consider that the matching block is matched to every pixel patch of the same size in the search region. It is not surprising that such an approach possesses relatively large computational cost, and is therefore unsuited for real-time applications. Several algorithms have been suggested that improve on the basic full search algorithm. Such alternatives either attempt to eliminate unlikely search vectors, or reduce the complexity of the search space by only considering subsections.

As an example of the latter, assume that the similarity increases monotonically the closer the search vector is to the optimal match [48]. Such an assumption implies that a greedy approach to block matching can effectively reduce the search complexity to the order of $O(\log n)$. Unfortunately in most cases the assumption does not hold, implying that local optima exist that distort the search for the global optimum.

Nonetheless many search algorithms are willing to accept a loss of accuracy for a gain in execution speed. The most well-known of these greedy approaches was presented by Koga

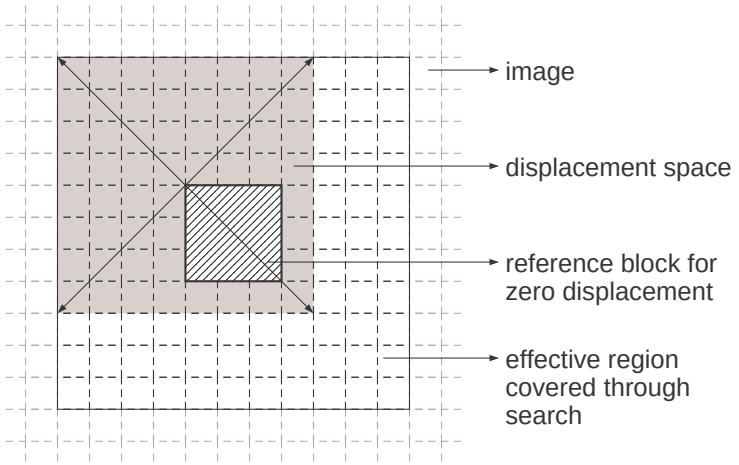


Figure 35: Example search of radius 4 for a 3×3 block

et al [61] and is known as the three-step-search, which makes use of a search pattern that recursively refines the motion estimate. Several improvements have been made on the basic three-step-search. Li *et al* [67], for example, note that typically motion vectors are biased towards no motion and thus present a centre-biased version of the algorithm known as the new-three-step-search. The four-step-search [86], discussed by Po and Ma, nearly reaches the accuracy of the former approaches, but achieves this result at a lower computational cost.

In some applications, though, the assumption that similarity decreases monotonically with increasing distance from the best match does not hold, causing local minima to falsify the results from greedy searches. Decroos *et al* [27] demonstrate that it is possible to maintain the order of execution of greedy search methods in such applications by introducing a search region that exceeds the immediate neighbourhood.

4.1.3 Differential approaches

Differential, or pixel recursive, algorithms for motion estimation underlie the basic assumption that perceived pixel motion over moving areas is based on translation, or stated in terms of equations:

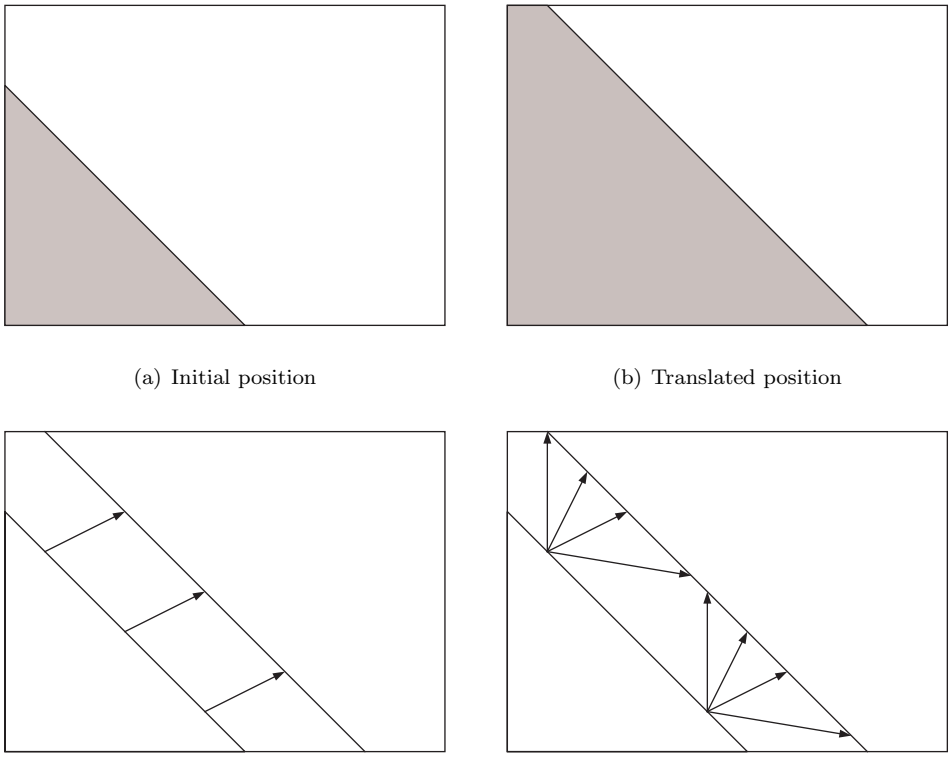
$$\begin{aligned} 0 &= I(x,y,t) - I(x-\Delta x, y-\Delta y, t-\Delta t) \\ &= \frac{\partial I}{\partial t} \end{aligned}$$

where $I(x,y,t)$ is the intensity of the pixel at (x,y) and time t, also $\mathbf{u} = (u,v) = (\Delta x, \Delta y)$ is the translation of the pixel, and Δt is the frame time difference. Define:

$$E_M = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}$$

where E_M is known as the *motion gradient constraint*, sometimes also referred to as the *optical flow constraint*.

The motion gradient constraint actually describes a line in velocity space. This implies that solutions to the constraint lie on a line, rather than at a particular point—this limitation is known as the *aperture problem*. The physical interpretation of this limitation, shown in Figure 36, is that if an edge can only be observed locally, then only its motion components normal to that edge can be determined.



(a) Initial position (b) Translated position

(c) Actual motion vector (d) Possible perceived motion vectors

Figure 36: Aperture problem

Additional constraints are required to allow the motion estimation to specify a single point, rather than a line. Different pixel recursive techniques primarily differ in their choice of additional constraints. The simplest constraint is to pick the shortest possible translation—though it is a poor choice in practice, as this choice makes no provision for the actual underlying motion.

Horn and Schunck [43] proposed the *smoothness constraint*, E_S, that assumes that motion varies slowly over the image. It is defined as,

$$E_S = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 .$$

It allows the determination of the motion field by minimizing the error function

$$E = E_M^2 + \lambda E_S$$

with λ a parameter to control the relative weighting of the motion and smoothness constraints.

Some differential algorithms, such as the one forwarded by Uras *et al* [111], solve the aperture problem by assuming that the second order derivatives of the motion constraint are also constant. Symbolically this is stated as follows:

$$\nabla E_M = 0$$

$$\begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \frac{\partial^2 I}{\partial x \partial t} \\ \frac{\partial^2 I}{\partial y \partial t} \end{bmatrix}$$

where the matrix on the left-hand side is the Hessian **H** of I. Provided **H** can be inverted, there exists a unique solution to u and v such that:

$$\begin{aligned} u &= \left(\frac{\partial^2 I}{\partial x^2} \frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial x \partial y} \right)^2 \right)^{-1} \left(\frac{\partial^2 I}{\partial y^2} \frac{\partial^2 I}{\partial x \partial t} - \frac{\partial^2 I}{\partial x \partial y} \frac{\partial^2 I}{\partial y \partial t} \right) \\ v &= \left(\frac{\partial^2 I}{\partial x^2} \frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial x \partial y} \right)^2 \right)^{-1} \left(- \frac{\partial^2 I}{\partial x \partial y} \frac{\partial^2 I}{\partial x \partial t} + \frac{\partial^2 I}{\partial x^2} \right) . \end{aligned}$$

Uras *et al* make use of numerical techniques to approximate the partial derivatives to determine **H**, then solve for u and v by finding its inverse. Though the algorithm is susceptible to noise, the errors can be controlled by smoothing the image intensities and making use of pruning techniques that identify and discard erroneous values.

4.1.4 Kanade-Lucas-Tomasi tracking

The Kanade-Lucas-Tomasi tracking algorithm is widely used for feature tracking. It is based on early work by Lucas and Kanade [69] in 1981, fully developed a decade later by Tomasi and Kanade [108]. Shi and Tomasi presented a clear and complete paper [99] in 1994 that serves as the primary reference for the algorithm.

The algorithm itself straddles the line between feature correlation and differential-based approaches. It makes use of feature identification based on examining the eigenvalues of **Z**, as defined by Equation (2) in Section 4.1.1 on page 46. Features are tracked between two search windows by minimizing the error between them.

Similar to differential methods, the basic premise of the Kanade-Lucas-Tomasi tracking algorithm is that

$$I(x, y, t) = I(x - \Delta x, y - \Delta y, t + \Delta t) .$$

In other words the intensity of a point in frame t is the same as the intensity in frame $t + \Delta t$, though the point is translated by $(\Delta x, \Delta y)$. This displacement is defined as $\mathbf{d} = (\Delta x, \Delta y)$ and the purpose of the tracking algorithm is to find the best choice for $\mathbf{d}$.

To simplify the notation the time term is dropped such that the image frames are defined as

$$\begin{aligned} \mathbf{x} &= (x, y) \\ B(\mathbf{x}) &= I(x, y, t) \\ A(\mathbf{x} - \mathbf{d}) &= I(\mathbf{x} - \mathbf{d}, t + \Delta t) \\ &= I(x - \Delta x, y - \Delta y, t + \Delta t) \end{aligned}$$

so that the relationship between the images is given as

$$B(\mathbf{x}) = A(\mathbf{x} - \mathbf{d}) + n(\mathbf{x})$$

where $n(\mathbf{x})$ is a noise term to compensate for errors due to observation inaccuracies and intensity distortions due to light scattering and other ambient factors.

Given a feature window area W an error function can be defined. The error function, also referred to as the residual function, is the surface integral of the weighted squared differences between A and B,

$$\epsilon(\mathbf{d}) = \iint_W \left[A(\mathbf{x} - \mathbf{d}) - B(\mathbf{x})\right]^2 \omega \, d\mathbf{x}$$

where ω is a weighting function. Typically the bias is chosen as $\omega \equiv 1$, though occasionally a center-biased or application specific weighting function is used.

The task at hand is to find a displacement vector $\mathbf{d}$ such that $\epsilon(\mathbf{d})$ is minimised. Assuming that the displacement $\mathbf{d}$ is sufficiently small, a first-order Taylor expansion may be applied to yield

$$A(\mathbf{x} - \mathbf{d}) = A(\mathbf{x}) - \mathbf{g}^{\mathrm{T}}\mathbf{d}$$

where $\mathbf{g}$ refers to the gradient vector defined in Section 4.1.1 on page 46. Thus the error function becomes

$$\epsilon(\mathbf{d}) = \iint_W \left[A(\mathbf{x}) - B(\mathbf{x}) - \mathbf{g}^{\mathrm{T}}\mathbf{d}\right]^2 \omega \, d\mathbf{x} \ .$$

To minimize the error function the derivative of ϵ with respect to $\mathbf{d}$ is set to zero, yielding

$$\nabla\epsilon = \iint_W \left[A(\mathbf{x}) - B(\mathbf{x}) - \mathbf{g}^{\mathrm{T}}\mathbf{d}\right] \mathbf{g}\omega \, d\mathbf{x} \ .$$

Furthermore, by noting the identity $(\mathbf{g}^{\mathrm{T}}\mathbf{d})\mathbf{g} = (\mathbf{g}\mathbf{g}^{\mathrm{T}})\mathbf{d}$ it is possible to rearrange the equation above to produce

$$\left(\iint_W \mathbf{g}\mathbf{g}^{\mathrm{T}}\omega \, d\mathbf{x}\right)\mathbf{d} = \iint_W \left[A(\mathbf{x}) - B(\mathbf{x})\right]\mathbf{g}\omega \, d\mathbf{x} \ . \tag{3}$$

Equation (3) is known as the Kanade-Lucas-Tomasi tracking equation. It can be written as

$$\mathbf{Z}\mathbf{d} = \mathbf{e}$$

where

$$\mathbf{Z} = \iint_W \mathbf{g}\mathbf{g}^{\mathrm{T}} \omega \, d\mathbf{x} \quad \text{and} \quad \mathbf{e} = \iint_W [A(\mathbf{x}) - B(\mathbf{x})] \, \mathbf{g}\omega \, d\mathbf{x} \; .$$

It is obvious that to solve for $\mathbf{d}$ it is essential that $\mathbf{Z}$ be invertible. For $\mathbf{Z}$ to be invertible the region at $A(\mathbf{x} - \mathbf{d})$ must contain gradient information in both the x and y direction. Provided that features are chosen appropriately—by making use of the feature acquisition techniques discussed in Section 4.1.1—it is ensured that $\mathbf{Z}$ is invertible.

4.1.5 Higher order motion estimation

Although the motion estimation techniques described above are efficient and accurate in the context of reasonably tractable problems, some complex motion analysis problems require more sophisticated algorithms to produce acceptable results. Problems that require the use of higher order solutions may, for example, require tracking of a particular entity among a set of candidates in the presence of dynamically varying partial or complete obfuscation of objects. A suitable real-world analogy is the tracking of a particular individual in a crowded area.

An implementation of Kalman filters for motion tracking is a popular choice for higher order motion estimation. The intrinsic process and measurement models used by the Kalman filters serve to describe the behavior of tracked objects and the limitations of the observations. This allows Kalman filters to integrate Newtonian motion models, contour transitions, or other processes to improve the estimate of the motion. Examples of such work include Gennery [37] as well as Rehg and Kanade [91].

The primary limitation of Kalman filters is that they function best in linear problem spaces, though the various improvements of the Kalman filter (such as the extended Kalman filter and the unscented Kalman filter) offer good solutions for weakly non-linear problems. However, for highly irregular, non-linear problems even the most advanced Kalman filters fail to offer consistently accurate results.

A solution that is robust in even the most non-linear cases was forwarded by Isard and Blake [47] in 1998. The algorithm, dubbed Condensation, is a variant of the class of particle filter solutions known as Sampling Importance Resampling filters and extends the earlier work performed by the same authors in 1996 [46]. Condensation is a fusion of the terms "**con**ditional **dens**ity propag**ation**".

The algorithm relies on probabilistic weights to randomly select pixels. The weights describe the density distribution of the tracked objects—meaning the odds that the pixel is part of the tracked object. Subsequent observations adjust the weights of the sampled pixels, which in turn describes the sampling space from which new random pixels are selected.

In practice only a relatively small set of sample pixels is required for Condensation to yield good results. As few as 100 samples per frame have been shown to offer good performance

and accuracy. Typically, though, between 500 and 1000 samples are used in highly cluttered problems.

Such intricate solutions to motion estimation well exceed the requirements of this thesis; but nonetheless it proves useful to be aware of these algorithms. Both Kalman filters and particle filters will be discussed in greater detail in Chapter 5.

4.2 Implementation

Structure-from-motion methodologies rely on a source of relative motion data to perform reconstruction. Motion tracking is a common source of such motion data and is utilised in this thesis. This section describes the motion tracking solution used.

The requirements of a given task mould the tools that may be used to accomplish that task. In the case of structure-from-motion problems the intended result dictates the nature of the motion data utilised. Some applications, such as the reconstruction research by Rautenbach [90], rely on a sparse solution making use of feature tracking to accurately depict depth details in reconstructed objects.

The intention of this thesis, however, is to reconstruct a complete and immersive closed environment. This shifts the emphasis away from detail-based reconstruction, suggesting that a dense optical flow solution is better suited to the task at hand.

Classically a dense optical flow solution is ill-suited for reconstruction purposes, as dense flow solutions do not follow a particular feature point. Instead the optical flow is determined at pre-set locations. This severely limits the accuracy of reconstruction attempts of iterative techniques, such as the Kalman filter that is used within this thesis.

It follows that a compromise between dense and sparse flow solutions may serve as a suitable vehicle to accomplish the motion tracking needed for reconstruction purposes. This thesis utilises a solution that simulates a dense optical flow by densely covering the sample space with features that should be tracked. This dense feature set is divided into smaller groups that are monitored across image sequences in a manner similar to the tracking employed in sparse optical flow methods. This yields a large field of accurate motion data over a long period of time, sufficient to allow the Kalman filter to converge to a solution.

The motion estimation utilised in this thesis makes use of feature correspondence; in other words it is a block matching solution to motion tracking. The scene is densely covered in tracked features—these features are not selected using a feature selector but instead are designated statically: the scene is evenly divided into 360 vertical sections and each of these sections is divided into an even distribution of 32 features. These values were chosen as they provided a suitably dense reconstruction of the environment.

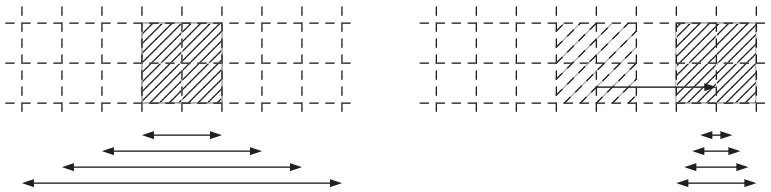
This ensures a dense set of features that allows an accurate reconstruction of the environment. The feature set has the advantage that the fixed rules describing the feature locations allow an easy correspondence of the relation between features: each point is connected to its neighbouring points.

However, since features are not chosen using a measure of feature prominence, as described in Section 4.1.1, it is harder to ensure that features can be accurately tracked. It is hoped that by making use of a block matching algorithm that the large blocks, rather than the specific feature points, possess sufficient textural detail to be readily trackable.

The implementation of motion estimation is designed to offer a high degree of accuracy whilst maintaining an acceptable rate of computation. The implementation chooses to forego heuristic searching algorithms in favor of a more accurate full search algorithm. The matching process is accelerated by the assumption that camera motion is confined to horizontal translation; which reduces the search space from $O(n^2)$ to $O(n)$ without loss of accuracy.

The choice to limit camera motion to the horizontal plane is precipitated by design decisions detailed in Section 6.2.1. By imposing a structure on dense feature selection and camera motion it is possible to ensure that features are not lost prematurely and that depth cues are maximised. The motion estimation subproblem benefits from these design choices by allowing optimisations and simplifications to be used in the implementations.

It is important to note that the choice to limit camera motion and motion tracking to a single axis is purely intended to expedite computation. The algorithm for depth reconstruction via the unscented Kalman filter used in this study places no limitations on the camera motion and makes no assumptions regarding the nature of the motion present in input data.



(a) Step 1—pixel accurate estimate

(b) Step 2—refine estimate using sub-pixel testing

Figure 37: Motion estimation procedure

The estimation process is performed in two steps—a coarse search and a fine search. The first step makes use of highly optimised block matching routines that are hand-assembled in SIMD[2] code to perform rapid pixel-accurate block matching. The estimated point results from this first step are used in a localised matching search that only considers the immediate sub-pixel

[2]Single Instruction, Multiple Data—CPU instructions that allow processing of several data inputs and producing multiple outputs in a single instruction

neighbourhood around that point to find the optimal match, as illustrated in Figure 37.

It is common to choose a pixel patch size of 8×8 or 16×16 pixels in general purpose block estimators. The implementation used within this thesis uses a patch size of 16×3 pixels for coarse block matching, and 16×1 for sub-pixel matching. These sizes prove to be sufficient for the purposes of this thesis and offer an additional improvement to the execution speed of the motion estimation system.

An unexpected problem that materialised in this study is the accurate tracking of selected features. The dense flow methodology assumes no inherently characteristic features in the samples chosen for motion estimation. This implies that the accurate location of a particular sample cannot be verified against a particular reference block—the technique is limited to determining the best matching pixel patch to the current reference.

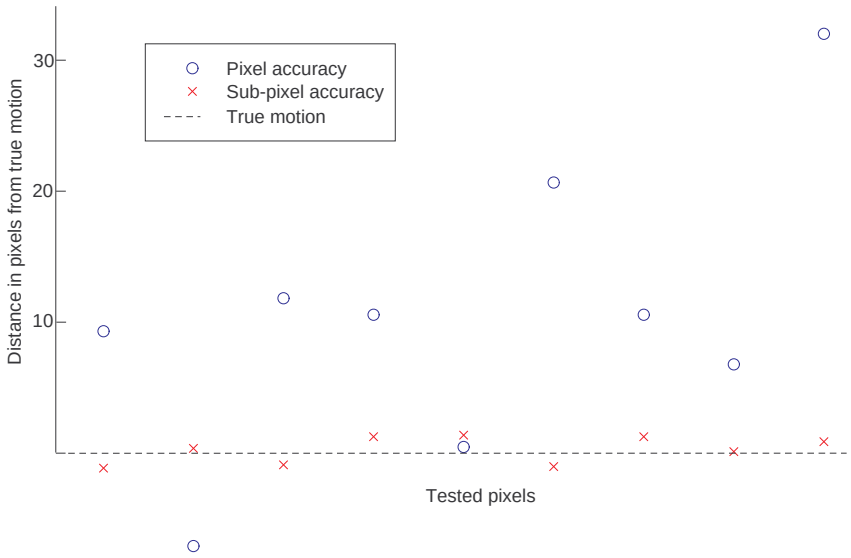


Figure 38: Pixel accuracy versus sub-pixel accuracy

At first glance, this appears to be sufficient to trace the motion of a pixel block; however, the problem becomes prominent over successive frames due to aliasing of data. The error in motion estimation across several frames can accumulate—progressively over- or under-estimating the position of the desired pixel region. This, in turn, changes the reference block that is being tracked to such an extent that over time it shares no common real-world features with the original reference block.

It is possible to make reference to the initial pixel block throughout the motion estimation process. Such a solution, however, severely restricts the range of applications of the motion estimation, as scene-transforms and camera motion over time may change the features of a reference block.

Instead, the solution adopted within the thesis is that of sub-pixel interpolation. Instead of matching the reference block to the pixels in the search patch a sub-pixel patch is computed: the colors of neighbouring pixels are linearly interpolated to emulate the effect of aliasing. These sub-pixel patches are then matched to the reference block. It was found that for the purposes of this study a sub-pixel interval of 0.1 is sufficient to track pixel patch motion. Figure 38 on page 59 compares the relative accuracy of the two approaches during experimentation. It is clear from the graph that the pixel-based tracker is considerably less accuate and sub-pixel tracker.

4.3 Discussion

Up to this point the basic elements in motion estimation, and the implementation in this study, were discussed. Now follow several additional topics that are relevant to motion tracking and that will be briefly discussed in this section, namely

- dense vs sparse optical flow,
- feature tracking and validation,
- hierarchical approaches,
- multi-solution approaches,
- accuracy-efficiency trade-off, and
- sub-pixel resolution of motion.

Dense vs sparse optical flow

Tracking of individual features does not allow a detailed analysis of full scene motion. This is known as sparse optical flow and is commonly used in applications where specific scene elements need to be identified and accurately tracked, such as surveillance cameras that automatically follow moving entities and object tracking systems used in sporting events.

On the other hand, some applications—such as the encoding of video data—require full scene motion data. Instead of tracking specific features, dense flow solutions attempt to estimate the motion of regions [5]—thus the motion of a specific element is approximated as the motion of the region that it is part of. In other words, dense optical flow models are less accurate, but provide a motion estimate for every single scene element.

Feature tracking and validation

Feature tracking through sparse flow methods can take advantage of increased processing time compared to dense flow methods, as they generally require small portions of the full image scene to be searched. However, the increased processing time that is available is offset by the increased accuracy required by feature trackers and the need to verify tracked features.

Feature verification, or validation, attempts to ensure that features tracked with motion estimation techniques are in fact the features desired. Verification methods typically employ the assumption that tracked objects possess some set of characteristic properties that can be evaluated and tested. Objects may be assumed to go through a limited set of possible transformations, and correspondingly the feature state perceived in scenes is continuously monitored to validate that tracked objects transform in a manner appropriate to their expected behavior. For example, a soccer ball will always be expected to appear roughly spherical; when a triangle is found where an oval was expected then the feature validator has successfully falsified a tracked feature estimate.

Hierarchical approaches

Motion in scenes tends to occur on a variety of scales. It follows then that it is possible to track motion on coarse scales, and use the information gained to seed search estimates at finer scales or, alternatively, use results of fine-grained search to determine coarse motion parameters.

Such hierarchical considerations can be applied to both correlation and differential motion estimation methods. Examples include Glazer *et al* [38] who present a hierarchical implementation of Horn and Schunck's [43] differential approach that makes use of low resolution computations of the image scene to determine coarse motion vectors. Similarly Anandan [6] proposes a framework for a top-down approach to motion estimation that refines coarser results into fine results. Anandan makes use of correlation based methods, but notes that the approach is easily applied to differential techniques as well. The Kanade-Lucas-Tomasi algorithm may be adapted for hierarchical implementation as well; see for example Wagener [121].

Hierarchical approaches, sometimes also known as pyramidal implementations, find use in a variety of ways; for example the DivX [2] technology video codec[3] has parameters to make use of *global motion compensation* for encoding purposes. Global motion compensation allows the codec to more optimally encode video sequences that have extensive global motion—such as camera panning, common in nature documentaries.

[3] codec—en**co**der/**dec**oder

Multi-solution approaches

The field of motion estimation has brought forth a wide range of motion estimation techniques, each effective for a particular class of image scenes—but none is known to outperform all others under all conditions. Given such a scenario it is not surprising that multi-solution strategies emerge.

Peacock [85], for example, makes use of information fusion strategies that combines the results of several algorithms to achieve a more accurate overall result. Specifically Peacock makes use of the methods presented by Camus [18], Horn and Schunck [43], and Uras *et al* [111]. Fusion strategies can intelligently distinguish between high speed and high accuracy requirements, allowing any desired balance (within the confines of the algorithms involved) to be used.

Accuracy-efficiency trade-off

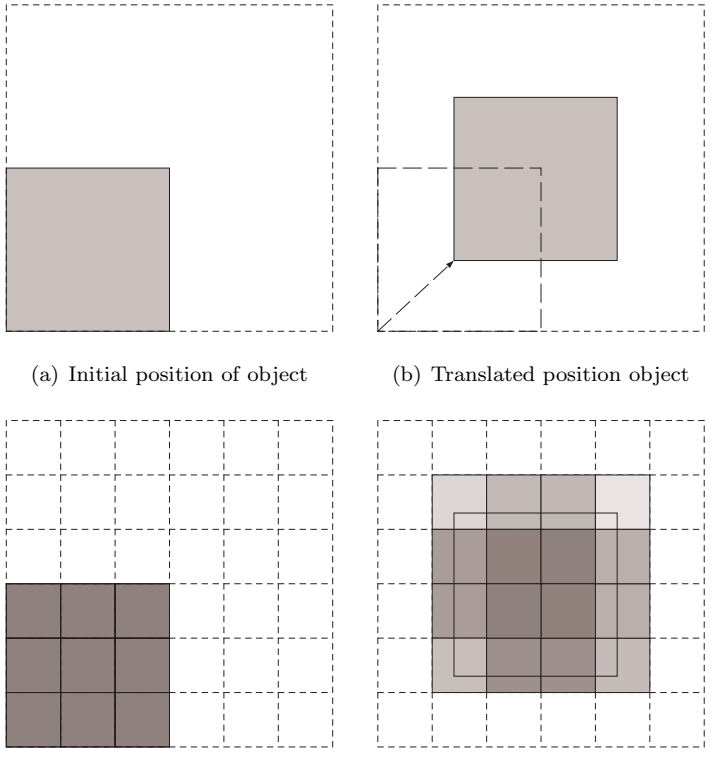
The question of accuracy versus efficiency is prominent in motion estimation. As a general rule it is possible to achieve greater accuracy at the cost of slower execution speed, or to increase the computational efficiency at the cost of loss of accuracy.

Liu *et al* [68] characterise the trade-off using an accuracy-efficiency curve—indicating a measure of the estimation error on the X-axis against the run-time required on the Y-axis. The performance with respect to accuracy and efficiency of algorithms can be compared on such a graph, though this requires the algorithms to run on comparable computational resources and on image sequences with accurately known real-world motion vectors.

Sub-pixel resolution of motion

A final point of interest is that of sub-pixel accuracy in motion estimation. Since only a limited number of positions are searched for matching purposes it follows that correlation techniques are typically discrete. The expected order of accuracy is usually given as 1 pixel per frame considered. Methods that compute motion estimates using several frames, such as the algorithm forwarded by Camus [18], may increase the precision to $1/n$ pixels per frame, where n is the number of frames evaluated.

It is possible to increase the accuracy of motion estimates to sub-pixel level on single-frame evaluations by making use of interpolation techniques to compute pixel values at points between pixels. This is viable as it mimics the aliasing effect common in digital media, as portrayed in Figure 39: since a moving feature within an image does not always perfectly align with the pixels of that image it is common that a pixel is not fully covered by that feature. In such cases the final color of that pixel is a blend of all the colors that share that pixel. As was previously discussed in Section 4.2 on page 59 of this chapter, the motion tracking algorithm employed in this study utilizes linear interpolations to achieve the necessary accuracy of motion estimates.



(a) Initial position of object

(b) Translated position object

(c) Digital perception initially

(d) Digital perception after translation

Figure 39: Anti-aliasing

This concludes the chapter on motion estimation. The following chapter details the Kalman filter, which serves as the computational tool for the structure-from-motion process.

All motion is cyclic. It circulates to the limits of its possibilities and then returns to its starting point.

Robert Collier

Chapter 5

Kalman filter

Kalman filters are about as sweet a method as you can find in applied mathematics.

Dana Mackenzie

Dana Mackenzie's words, published in a SIAM article on meteorological modelling [71], might not be the most eloquent words uttered in favor of Kalman filters—but they certainly echo the sentiments of many that have applied the filter to their problem. This chapter is dedicated to the primary tool utilised in the structure-from-motion problem within this thesis.

Rudolph Emil Kalman is the inventor of the Kalman filter—an efficient, recursive filter that can estimate the state of a dynamic system in the presence of noise and incomplete measurements. Surprisingly, his seminal paper [57], published in 1960, was initially met with scepticism and only several years later found the attention it deserved when Stanley Schmidt at the NASA Ames Research Center made use of the filter to solve the problem of trajectory estimation, which was subsequently utilised in the Apollo program.

Since the pioneering work in the 1960's the Kalman filter has evolved into various guises and has seen use in a wide variety of applications. These applications include, but are not limited to, navigational systems, guidance and tracking systems, seismic processing, meteorological modelling, nuclear reactor instrumentation, motion capture, audio reconstruction and, more recently, structure-from-motion modelling.

Credit for the development of the Kalman filter may be jointly attributed to Peter Swerling [104], Rudolph Kalman [57], and Richard Bucy [58]. The original Kalman filter, also refered to as the Simple or Linear Kalman filter, has produced a great number of variants designed to apply to different types of problems, such as the extended Kalman filter, by Stanley Schmidt [95], and the unscented Kalman filter, by Simon Julier and Jeffrey Uhlmann [52, 54].

The remainder of this chapter describes three useful implementations of the Kalman filter: the Linear Kalman filter, the extended Kalman filter and the unscented Kalman filter. The

emphasis is placed on the unscented Kalman filter, as it serves as the non-linear estimator for solving the structure-from-motion problem within this thesis. Additionally a section is dedicated to the description of particle filters—a class of filter that is useful in highly non-linear problems.

5.1 Linear Kalman filter

The original Kalman filter is now often referred to as the Linear or Simple Kalman filter and, as the name suggests, applies to linear problems. The Linear Kalman filter is described first as it offers a relatively simple introduction into the Kalman process. The ideas presented for the Linear Kalman filter generally map to the more complex non-linear counterparts with little effort.

Generally speaking all Kalman filters are minimum mean-square error estimators. They have the distinct advantage that they achieve this result iteratively, taking into account new measurements and observations. This has the advantage that the filter does not need to refer to all prior measurements to maintain the accuracy of estimates.

5.1.1 Problem definition

The Kalman filter estimates the state $\mathbf{x}$ at time step i of a dynamic system in the presence of observation $\mathbf{y}$. More fully the state of a system is given as

$$\mathbf{x}_i = \mathbf{F}_{i-1}\mathbf{x}_{i-1} + \mathbf{B}_{i-1}\mathbf{u}_{i-1} + \boldsymbol{\mu}_{i-1} \tag{4}$$

and the observation is given as

$$\mathbf{y}_i = \mathbf{H}_i\mathbf{x}_i + \boldsymbol{\nu}_i \tag{5}$$

where $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{H}$ are matrices describing the underlying models for the system, control input and measurements, $\mathbf{u}$ is an external control vector, and $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ represent state and measurement noise respectively.

The problems considered in this study do not need the control term; it is omitted in subsequent discussion.

5.1.2 Assumptions and notation

The underlying assumption of a typical Kalman filter is the adherence to a first order Markov process—implying that the current state only depends on the previous state, so that

$$p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \ldots, \mathbf{x}_0) = p(\mathbf{x}_i|\mathbf{x}_{i-1}) \ .$$

Additionally the current observation only depends on the current state:

$$p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{x}_{i-1}, \ldots, \mathbf{x}_0) = p(\mathbf{y}_i|\mathbf{x}_i) \ .$$

Kalman filters make use of Gaussian probability distributions, specifically:

- normal distribution,
- additive,
- white—thus $E(\boldsymbol{\mu}_i \boldsymbol{\mu}_j^T) = 0$ and $E(\boldsymbol{\nu}_i \boldsymbol{\nu}_j^T) = 0$ provided $i \neq j$,
- zero-mean—thus $E(\boldsymbol{\mu}) = 0$ and $E(\boldsymbol{\nu})$, and
- uncorrelated—thus $E(\boldsymbol{\mu}\boldsymbol{\nu}^T) = 0$.

Since the problems solved by the linear Kalman filter are linear, the Gaussian distributions map onto Gaussian distributions, thus allowing the filter to iterate.

The following notation applies:

$$\overline{\mathbf{x}}_{i|i-1} = E(\mathbf{x}_i|\mathbf{y}_1, \ldots, \mathbf{y}_{i-1})$$

where $\overline{\mathbf{x}}_{i|i-1}$ is the predicted state mean which is the expected value of $\mathbf{x}_i$ given observations $\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}$. On the same basis

$$\overline{\mathbf{y}}_{i|i-1} = E(\mathbf{y}_i|\mathbf{y}_1, \ldots, \mathbf{y}_{i-1})$$

indicates that $\overline{\mathbf{y}}_{i|i-1}$ is the expected value of $\mathbf{y}_i$ given the observations $\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}$. Additionally, note that $\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ signify that $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are drawn from a zero-mean Gaussian probability density function $\mathcal{N}$ with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ respectively.

The notation utilised in algorithmic representation largely ignores subscripts in favor of a presentation that mimics the flow of program execution. Although the algorithmic presentation requires a slight increase in interpretation of details, it is considerably easier on the eye and contributes to the understanding of the conceptual flow of the computational process: Following the nature of variable assignments in computer programs, unless otherwise indicated the most recent value of a variable is used. Additionally, assignment variables are always shown in italics. For example the symbolic representation

$$\mathbf{x}_i = f(\mathbf{x}_{i-1})$$

$$\mathbf{x}_{i+1} = g(\mathbf{x}_i)$$

is equivalent to the algorithmic representation

$$\boldsymbol{x} \longleftarrow f(\mathbf{x})$$

$$\boldsymbol{x} \longleftarrow g(\mathbf{x})$$

5.1.3 Kalman process

The basic Kalman process that underlies the majority of Kalman filters, including the extended and unscented Kalman filters, is shown in Figure 40 and can be described as follows:

The state of the system is predicted given the best prior estimate, which allows the observation to be predicted. These form the basis to compute the Kalman gain (defined below), which in turn allows the correction of the predicted state using the difference between observed and predicted measurements.

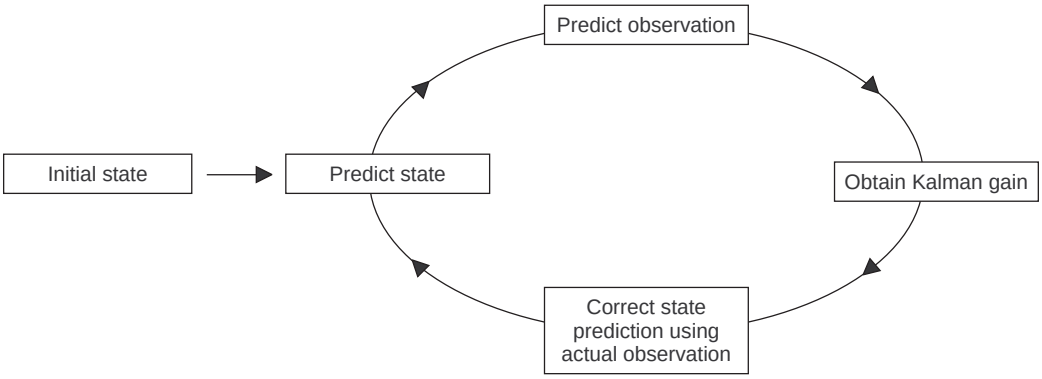


Figure 40: Kalman process

The basic state transition and observation transforms are respectively given as

$$\mathbf{x}_i = \mathbf{F}_{i-1}\mathbf{x}_{i-1} + \boldsymbol{\mu}_{i-1}$$
$$\mathbf{y}_i = \mathbf{H}_i\mathbf{x}_i + \boldsymbol{\nu}_i \ .$$

These form the basis to predict the current state of the system

$$\overline{\mathbf{x}}_{i|i-1} = \mathbf{F}_i\overline{\mathbf{x}}_{i-1|i-1}$$
$$\mathbf{P}_{\mathbf{x}_{i|i-1}} = \mathbf{F}_i\mathbf{P}_{\mathbf{x}_{i-1|i-1}}\mathbf{F}_i^{\mathrm{T}} + \mathbf{Q}_i$$

as well as predict the observation

$$\overline{\mathbf{y}}_{i|i-1} = \mathbf{H}_i\overline{\mathbf{x}}_{i|i-1}$$
$$\mathbf{P}_{\mathbf{y}_{i|i-1}} = \mathbf{H}_i\mathbf{P}_{\mathbf{x}_{i|i-1}}\mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i \ .$$

The predicted state and observation allow the computation of the Kalman gain, given as

$$\mathbf{K}_i = \mathbf{P}_{\mathbf{x}_{i|i-1}}\mathbf{H}_i^{\mathrm{T}}\mathbf{P}^{-1}_{\mathbf{y}_{i|i-1}}$$

which in turn allow the evaluation of the corrected state for the current time step:

$$\overline{\mathbf{x}}_{i|i} = \overline{\mathbf{x}}_{i|i-1} + \mathbf{K}_i(\widetilde{\mathbf{y}}_i - \overline{\mathbf{y}}_{i|i-1})$$
$$\mathbf{P}_{\mathbf{x}_{i|i}} = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{P}_{\mathbf{x}_{i|i-1}}$$

where $\widetilde{\mathbf{y}}_i$ represents the actual observation at time i. Program 3 summarises the process in algorithmic notation.

Program 3 Linear Kalman algorithm

State transition

$$\boldsymbol{x} \longleftarrow \mathbf{F}\mathbf{x} + \boldsymbol{\mu}$$

Observation transform

$$\boldsymbol{y} \longleftarrow \mathbf{H}\mathbf{x} + \boldsymbol{\nu}$$

Predict state

$$\overline{\boldsymbol{x}} \longleftarrow \mathbf{F}\overline{\mathbf{x}}$$
$$\boldsymbol{P}_x \longleftarrow \mathbf{F}\mathbf{P}_x\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$$

Predict observation

$$\overline{\boldsymbol{y}} \longleftarrow \mathbf{H}\overline{\mathbf{x}}$$
$$\boldsymbol{P}_y \longleftarrow \mathbf{H}\mathbf{P}_x\mathbf{H}^{\mathrm{T}} + \mathbf{R}$$

Kalman gain

$$\boldsymbol{K} \longleftarrow \mathbf{P}_x\mathbf{H}^{\mathrm{T}}\mathbf{P}_y^{-1}$$

Corrected state (where $\widetilde{\mathbf{y}}$ are actual observations)

$$\overline{\boldsymbol{x}} \longleftarrow \overline{\mathbf{x}} + \mathbf{K}(\widetilde{\mathbf{y}} - \overline{\mathbf{y}})$$
$$\boldsymbol{P}_x \longleftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_x$$

5.2 Extended Kalman filter

The extended Kalman filter is the obvious extension of the linear Kalman filter to apply to non-linear problems. As the problems solved by extended Kalman filters are non-linear it follows that the Gaussian distributions used to describe the system do not necessarily map onto Gaussian distributions. The extended Kalman filter solves this by linearizing the problem using a Taylor series expansion to approximate the system and observation models.

In practice the filter is more appropriately called the first order extended Kalman filter, as it makes use of a first order Taylor series expansion. However, although higher order expansions are certainly possible they are rarely used—thus the first order variant has become known as the extended Kalman filter. Symbolically the extended Kalman filter is described using the state transition and observation transforms:

$$\mathbf{x}_i = f(\mathbf{x}_{i-1}) + \boldsymbol{\mu}_{i-1}$$
$$\mathbf{y}_i = f(\mathbf{x}_i) + \boldsymbol{\nu}_i \ .$$

These form the basis to predict the current state of the system

$$\overline{\mathbf{x}}_{i|i-1} = f(\overline{\mathbf{x}}_{i-1|i-1})$$
$$\mathbf{P}_{\mathbf{x}_{i|i-1}} = \mathbf{F}_i\mathbf{P}_{\mathbf{x}_{i-1|i-1}}\mathbf{F}_i^{\mathrm{T}} + \mathbf{Q}_i$$

as well as predict the observation

$$\begin{aligned}\overline{\mathbf{y}}_{i|i-1} &= h(\overline{\mathbf{x}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{y}_{i|i-1}} &= \mathbf{H}_i \mathbf{P}_{\mathbf{x}_{i|i-1}} \mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i \ ,\end{aligned}$$

where $\mathbf{F}_i$ and $\mathbf{H}_i$ are the Jacobians

$$\begin{aligned}\mathbf{F}_i &= \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}|_{\overline{\mathbf{x}}=\overline{\mathbf{x}}_{i|i-1}} \\ \mathbf{H}_i &= \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}|_{\overline{\mathbf{x}}=\overline{\mathbf{x}}_{i|i-1}} \ .\end{aligned}$$

The predicted state and observation allow the computation of the Kalman gain, given as

$$\mathbf{K}_i = \mathbf{P}_{\mathbf{x}_{i|i-1}} \mathbf{H}_i^{\mathrm{T}} \mathbf{P}_{\mathbf{y}_{i|i-1}}^{-1}$$

which in turn allow the evaluation of the corrected state for the current time step:

$$\begin{aligned}\overline{\mathbf{x}}_{i|i} &= \overline{\mathbf{x}}_{i|i-1} + \mathbf{K}_i(\widetilde{\mathbf{y}}_i - \overline{\mathbf{y}}_{i|i-1}) \\ \mathbf{P}_{\mathbf{x}_{i|i}} &= (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i)\mathbf{P}_{\mathbf{x}_{i|i-1}}\end{aligned}$$

where $\widetilde{\mathbf{y}}_i$ represents the actual observation at time i. Program 4 displays the algorithmic form.

Program 4 Extended Kalman algorithm

State transition

$$\boldsymbol{x} \longleftarrow f(\mathbf{x}) + \boldsymbol{\mu}$$

Observation transform

$$\boldsymbol{y} \longleftarrow h(\mathbf{x}) + \boldsymbol{\nu}$$

Predict state

$$\begin{aligned}\overline{\boldsymbol{x}} &\longleftarrow f(\overline{\mathbf{x}}) \\ \boldsymbol{F} &\longleftarrow \frac{\partial f(\overline{\mathbf{x}})}{\partial \overline{\mathbf{x}}} \\ \boldsymbol{P}_x &\longleftarrow \mathbf{F}\mathbf{P}_x\mathbf{F}^{\mathrm{T}} + \mathbf{Q}\end{aligned}$$

Predict observation

$$\begin{aligned}\overline{\boldsymbol{y}} &\longleftarrow h(\overline{\mathbf{x}}) \\ \boldsymbol{H} &\longleftarrow \frac{\partial h(\overline{\mathbf{x}})}{\partial \overline{\mathbf{x}}} \\ \boldsymbol{P}_y &\longleftarrow \mathbf{H}\mathbf{P}_x\mathbf{H}^{\mathrm{T}} + \mathbf{R}\end{aligned}$$

Kalman gain

$$\boldsymbol{K} \longleftarrow \mathbf{P}_x\mathbf{H}^{\mathrm{T}}\mathbf{P}_y^{-1}$$

Corrected state (where $\widetilde{\mathbf{y}}$ are actual observations)

$$\begin{aligned}\overline{\boldsymbol{x}} &\longleftarrow \overline{\mathbf{x}} + \mathbf{K}(\widetilde{\mathbf{y}} - \overline{\mathbf{y}}) \\ \boldsymbol{P}_x &\longleftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_x\end{aligned}$$

5.3 Unscented Kalman filter

Although the extended Kalman filter is a reasonable tool to accommodate nonlinear models, its linearization method only achieves good predictions in weakly nonlinear problems. The unscented Kalman filter was introduced by Simon Julier *et al* in 1995 and 1996 [52, 56] to address the inadequacies of the extended Kalman filter and has since been refined in a series of publications led by Julier [50, 51, 53, 54, 55]. The unscented Kalman filter is able to make good estimates even in moderately nonlinear problems. However, it still fails in highly nonlinear problems.

To appreciate the need for a better alternative to the extended Kalman filter, consider Figure 41: the extended Kalman filter fails to account for the non-linearity of the system and predicts the state along the trajectory's tangent. Subsequently the mean and covariance need to be corrected to accommodate the true covariance, however, this compensation requires the covariance to be adjusted to an unreasonable degree. The unscented Kalman filter, in contrast, better estimates the behavior of the system.

Surprisingly, according to Rudolph van der Merwe and Eric Wan [116, 117], the extended and unscented Kalman filters possess comparable computational complexities of $O(n^3)$ to perform a state estimate. Additionally, the unscented Kalman filter implementations may be improved to execute in the order of $O(n^2)$ if only parameter estimation is required—as is the case in certain applications, such as artificial neural network training.

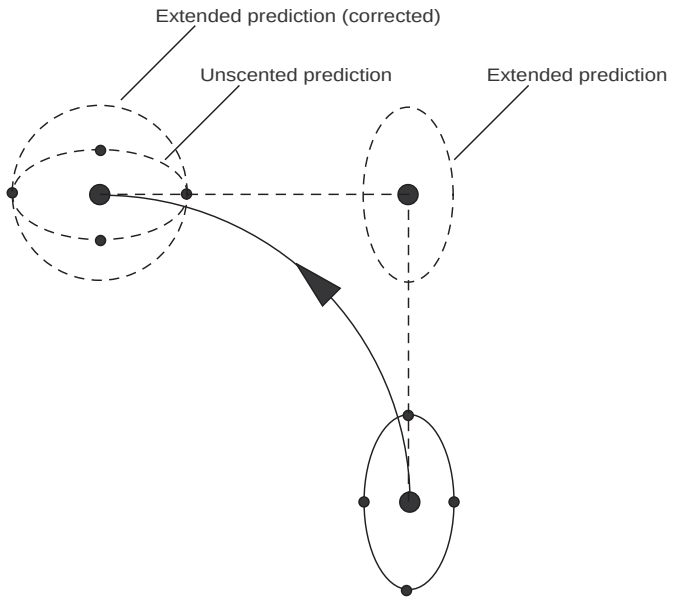


Figure 41: Predictions of extended and unscented Kalman filters

As would be expected from a first order estimator, the extended Kalman filter can model the first order moments of a Gaussian probability distribution function. In contrast, the unscented Kalman filter can model up to third order moments. In essence, the unscented Kalman filter has made the extended Kalman filter obsolete.

5.3.1 Unscented transform

Rather than linearizing the problem as with the extended Kalman filter, the unscented Kalman filter ensures that Gaussian distributions are maintained by making use of a set of points known as sigma points[1]. To understand the significance of the sigma points consider Figure 42.

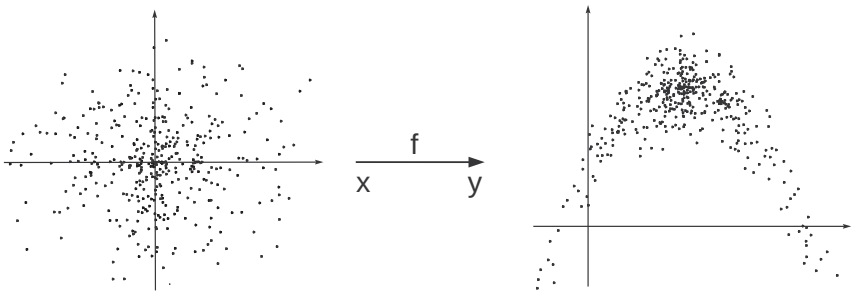


Figure 42: Many samples describing the nonlinear function $\mathbf{y} = \mathbf{f}(\mathbf{x})$

The graphs show a distribution of samples, $\mathbf{x}$, that are mapped to $\mathbf{y}$ using the nonlinear function $\mathbf{f} = \mathbf{x}^2 - \mathbf{x} + 1$; in general any arbitrary $\mathbf{f}$ may be used. Given that the mean $\overline{\mathbf{x}}$ and covariance $\mathbf{P}_x$ are known, what are $\overline{\mathbf{y}}$ and $\mathbf{P}_y$?

Monte Carlo methods, such as the particle filters discussed in Section 5.4, make use of a large cloud of n randomly sampled points $\{\mathbf{x}_i\}$. These samples are transformed using the nonlinear model itself to yield $\{\mathbf{y}_i\} = \mathbf{f}(\mathbf{x}_i)$ and subsequently the mean and covariance are calculated as $\overline{\mathbf{y}} = \frac{1}{n}\sum_i \mathbf{y}_i$ and $\mathbf{P}_y = \frac{1}{n}\sum_i (\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^{\mathrm{T}}$.

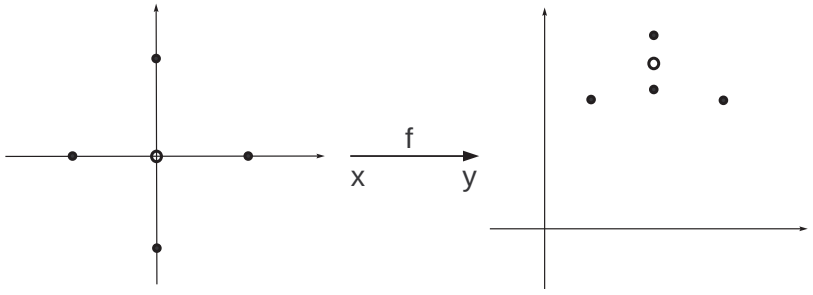


Figure 43: Sigma points describing the nonlinear function $\mathbf{y} = \mathbf{f}(\mathbf{x})$

[1]For this discussion we rely on notes that B. Sherlock has generously made available [98].

The Monte Carlo approach to the problem is effective, but it is not efficient. Often thousands of samples need to be evaluated before a good estimate is found. The unscented transform attempts to offer an efficient solution: instead of making use of thousands of random samples it makes use of a small set of carefully selected points, known as sigma points, as shown in Figure 43.

The sigma points $\{\mathbf{x}_i\}$ are chosen such that the sample mean of $\{\mathbf{x}_i\}$ is $\overline{\mathbf{x}}$ and the sample covariance of $\{\mathbf{x}_i\}$ is $\mathbf{P}_x$. Similar to the Monte Carlo method the sigma points are transformed using the nonlinear model such that $\{\mathbf{y}_i\} = \{\mathbf{f}(\mathbf{x}_i)\}$. The posterior mean and covariance are then given as $\overline{\mathbf{y}} = \frac{1}{n}\sum_i \mathbf{y}_i$ and $\mathbf{P}_i = \frac{1}{n}\sum_i (\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^{\mathrm{T}}$.

Typically about $2n+1$ sigma points are used for n-dimensional problems. To determine the sigma points it is necessary to compute $\mathbf{S}$, the matrix square root of $\mathbf{P}_x$, such that $\mathbf{P}_x = \mathbf{S}\mathbf{S}^{\mathrm{T}}$. In this study Choleski decomposition is used to compute $\mathbf{S}$, but any matrix square root works. For $1 \leq i \leq n$ the sigma points are then given as

$$\begin{aligned}
\mathbf{x}_0 &= \overline{\mathbf{x}} \\
\mathbf{x}_i &= \overline{\mathbf{x}} + \sqrt{n+\kappa}\mathbf{S}^{(i)} \\
\mathbf{x}_{i+n} &= \overline{\mathbf{x}} - \sqrt{n+\kappa}\mathbf{S}^{(i)}
\end{aligned}$$

where $\mathbf{S}^{(i)}$ denotes column number i of matrix $\mathbf{S}$ and κ is used to scale sigma points towards or away from the mean. Furthermore the sigma points are weighted when computing the posterior mean and covariance as follows

$$\begin{aligned}
w_0 &= \frac{\kappa}{n+\kappa} \\
w_i &= \frac{1}{2(n+\kappa)}, \quad i \neq 0 \\
\overline{\mathbf{y}} &= \sum_{i=0}^{2n} w_i \mathbf{y}_i \\
\mathbf{P}_y &= \sum_{i=0}^{2n} w_i (\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^{\mathrm{T}} \ .
\end{aligned}$$

According to Sherlock [98] the greatest accuracy for Gaussian distributions is achieved at $\kappa = 3 - n$. However, for $n > 3$ this implies that $\kappa < 0$ and thus that w_0 is negative. A negative w_0, in turn, implies that $\mathbf{P}_x$ might not be positive definite. This is a significant problem as the matrix square root only exists for matrices that are positive definite, and also, covariances are supposed to be positive definite. To overcome this problem the scaled unscented transform was developed.

5.3.2 Scaled unscented transform

The scaled unscented Kalman filter, presented by Julier [55], defines three parameters that control the scaling of sigma points: α, β and κ. The first, α, is constrained within $0 \leq \alpha \leq 1$

and exerts influence on the spread of sigma points around the mean. It it defaulted to $\alpha = 1$ and only changed when the filter needs to cope with extreme non-linearity.

The second parameter, β, affects higher order moments by adding or reducing the weight of the mean. β satisfies $\beta \geq 0$ and both Julier [52] and Van der Merwe *et al* [115] recommend $\beta = 2$ for Gaussian distributions. Finally, κ is used to ensure positive semi-definiteness. It satisfies $\kappa \geq 0$ and Sherlock [98] recommends $\kappa = 0$, which is the value used in this study.

The scaling parameters control two sets of weights that act on computed sigma points. The weights are defined through vectors v and w as

$$\begin{aligned} v_0 &= \frac{\lambda}{n+\lambda} \\ w_0 &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ v_j = w_j &= \frac{1}{2(n+\lambda)} \ , \quad j = 1, 2, \ldots, 2n \\ \lambda &= \alpha^2(n+\kappa) - n \end{aligned}$$

where n is the dimension of the problem. The scaled sigma points are now given as

$$\begin{aligned} \mathbf{x}_0 &= \overline{\mathbf{x}} \\ \mathbf{x}_i &= \overline{\mathbf{x}} + \alpha\sqrt{n+\kappa}\mathbf{S}^{(i)} \\ &= \overline{\mathbf{x}} + \sqrt{\alpha^2(n+\kappa)}\mathbf{S}^{(i)} \\ &= \overline{\mathbf{x}} + \sqrt{\lambda+n}\mathbf{S}^{(i)} \\ \mathbf{x}_{i+n} &= \overline{\mathbf{x}} - \alpha\sqrt{n+\kappa}\mathbf{S}^{(i)} \\ &= \overline{\mathbf{x}} - \sqrt{\alpha^2(n+\kappa)}\mathbf{S}^{(i)} \\ &= \overline{\mathbf{x}} - \sqrt{\lambda+n}\mathbf{S}^{(i)} \end{aligned}$$

for $i = 1, 2, \ldots, n$. The resulting sample mean and covariance are computed with the scaling weights such that

$$\begin{aligned} \overline{\mathbf{y}} &= \sum_{i=0}^{2n} v_i \mathbf{y}_i \\ \mathbf{P}_y &= \sum_{i=0}^{2n} w_i (\mathbf{y}_i - \overline{\mathbf{y}})(\mathbf{y}_i - \overline{\mathbf{y}})^{\mathrm{T}} \ . \end{aligned}$$

5.3.3 Unscented Kalman filter algorithm

Although it is not necessary, it is common to define input to the filter through super states of the estimate and covariance matrices by concatenating the state and covariances respectively into $\overline{\mathbf{a}}$ and $\boldsymbol{\mathcal{A}}$, defined below. This condenses the notation and hence is used in the description below. Furthermore, note that the computation of the sigma points yields matrix $\boldsymbol{\mathcal{A}}$ which can

be viewed as the augmentation of three matrices

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{\mathcal{X}} \\ \boldsymbol{\mathcal{Q}} \\ \boldsymbol{\mathcal{R}} \end{bmatrix}$$

where $\boldsymbol{\mathcal{X}}$ is the sigma point state matrix, $\boldsymbol{\mathcal{V}}$ is the sigma point state uncertainty matrix, and $\boldsymbol{\mathcal{N}}$ is the sigma point observation uncertainty matrix. Given these conventions the scaled unscented Kalman filter is described symbolically as:

- Augmented structures

$$\overline{\mathbf{a}} = [\overline{\mathbf{x}} \quad \mathbf{0} \quad \mathbf{0}]^{\mathrm{T}}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}$$

- Calculation of sigma points

$$\begin{aligned} \boldsymbol{\mathcal{A}}^{(0)}_{i-1|i-1} &= \overline{\mathbf{a}}_{i-1|i-1} \\ \boldsymbol{\mathcal{A}}^{(k)}_{i-1|i-1} &= \overline{\mathbf{a}}_{i-1|i-1} + \sqrt{(n+\lambda)\mathbf{A}_{i-1|i-1}}^{(k)} && k = 1, 2, \ldots, n \\ \boldsymbol{\mathcal{A}}^{(k)}_{i-1|i-1} &= \overline{\mathbf{a}}_{i-1|i-1} + \sqrt{(n+\lambda)\mathbf{A}_{i-1|i-1}}^{(k-n)} && k = n+1, n+2, \ldots, 2n \end{aligned}$$

  where $\mathbf{A}^{(k)}$ denotes column number $(k+1)$ of matrix $\mathbf{A}$. This is followed by the prediction of the state

$$\begin{aligned} \boldsymbol{\mathcal{X}}_{i|i-1} &= \mathbf{f}(\boldsymbol{\mathcal{X}}_{i-1|i-1}, \boldsymbol{\mathcal{Q}}_{i-1}) \\ \overline{\mathbf{x}}_{i|i-1} &= \sum_{j=0}^{2n} w_j \boldsymbol{\mathcal{X}}^{(j)}_{i|i-1} \\ \mathbf{P}_{\mathbf{x}_{i|i-1}} &= \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{X}}^{(j)}_{i|i-1} - \overline{\mathbf{x}}_{i|i-1}\right] \left[\boldsymbol{\mathcal{X}}^{(j)}_{i|i-1} - \overline{\mathbf{x}}_{i|i-1}\right]^{\mathrm{T}} \end{aligned}$$

- the observation prediction

$$\begin{aligned} \boldsymbol{\mathcal{Y}}_{i|i-1} &= \mathbf{f}(\boldsymbol{\mathcal{X}}_{i-1|i-1}, \boldsymbol{\mathcal{R}}_{i-1}) \\ \overline{\mathbf{y}}_{i|i-1} &= \sum_{j=0}^{2n} v_j \boldsymbol{\mathcal{Y}}^{(j)}_{i|i-1} \\ \mathbf{P}_{\mathbf{y}_{i|i-1}} &= \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{Y}}^{(j)}_{i|i-1} - \overline{\mathbf{y}}_{i|i-1}\right] \left[\boldsymbol{\mathcal{Y}}^{(j)}_{i|i-1} - \overline{\mathbf{y}}_{i|i-1}\right]^{\mathrm{T}} \end{aligned}$$

- computation of the Kalman gain

$$\begin{aligned} \mathbf{P}_{xy_{i|i-1}} &= \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{X}}^{(j)}_{i|i-1} - \overline{\mathbf{x}}_{i|i-1}\right] \left[\boldsymbol{\mathcal{Y}}^{(j)}_{i|i-1} - \overline{\mathbf{y}}_{i|i-1}\right]^{\mathrm{T}} \\ \mathbf{K}_i &= \mathbf{P}_{xy_{i|i-1}} \mathbf{P}^{-1}_{y_{i|i-1}} \end{aligned}$$

- and finally the corrected states

$$\overline{\mathbf{x}}_{i|i} = \overline{\mathbf{x}}_{i|i-1} + \mathbf{K}_{i|i-1}(\widetilde{\mathbf{y}}_i - \overline{\mathbf{y}}_{i|i-1})$$
$$\mathbf{P}_{x_{i|i}} = \mathbf{P}_{x_{i|i-1}} - \mathbf{K}_{i|i-1}\mathbf{P}_{y_{i|i-1}}\mathbf{K}_i^T$$

where $\widetilde{\mathbf{y}}_i$ is the actual observation at time i. The algorithmic representation of the unscented Kalman filter is given below in Program 5.

Program 5 Unscented Kalman algorithm

Structure augmentation

$$\overline{\boldsymbol{a}} \longleftarrow [\overline{\mathbf{x}} \quad \mathbf{0} \quad \mathbf{0}]^{\mathrm{T}}$$
$$\boldsymbol{A} \longleftarrow \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix}$$

Sigma point computation

$$\boldsymbol{\mathcal{A}} \longleftarrow \left[\overline{\mathbf{a}},\ \overline{\mathbf{a}} \pm \sqrt{(n+\lambda)\mathbf{A}}\right]$$

State transition

$$\boldsymbol{\mathcal{X}} \longleftarrow f(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{V}})$$
$$\overline{\boldsymbol{x}} \longleftarrow \sum_{j=0}^{2n} v_j \boldsymbol{\mathcal{X}}^{(j)}$$
$$\boldsymbol{P}_x \longleftarrow \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{X}}^{(j)} - \overline{\mathbf{x}}\right]\left[\boldsymbol{\mathcal{X}}^{(j)} - \overline{\mathbf{x}}\right]^{\mathrm{T}}$$

Observation transform

$$\boldsymbol{\mathcal{Y}} \longleftarrow \mathbf{f}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{R}})$$
$$\overline{\boldsymbol{y}} \longleftarrow \sum_{j=0}^{2n} v_j \boldsymbol{\mathcal{Y}}^{(j)}$$
$$\boldsymbol{P}_y \longleftarrow \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{Y}}^{(j)} - \overline{\mathbf{y}}\right]\left[\boldsymbol{\mathcal{Y}}^{(j)} - \overline{\mathbf{y}}\right]^{\mathrm{T}}$$

Kalman gain

$$\boldsymbol{P}_{xy} \longleftarrow \sum_{j=0}^{2n} w_j \left[\boldsymbol{\mathcal{X}}^{(j)} - \overline{\mathbf{x}}\right]\left[\boldsymbol{\mathcal{Y}}^{(j)} - \overline{\mathbf{y}}\right]^{\mathrm{T}}$$
$$\boldsymbol{K} \longleftarrow \mathbf{P}_{xy}\mathbf{P}_y^{-1}$$

Corrected state (where $\widetilde{\mathbf{y}}$ are actual observations)

$$\overline{\boldsymbol{x}} \longleftarrow \overline{\mathbf{x}} + \mathbf{K}(\widetilde{\mathbf{y}} - \overline{\mathbf{y}})$$
$$\boldsymbol{P}_x \longleftarrow \mathbf{P}_x - \mathbf{K}\mathbf{P}_y\mathbf{K}^T$$

Before closing this section, a brief note regarding the computation of the posterior error covariance: the results shown in the linear, extended and unscented Kalman filters only apply if the optimal Kalman gain is used. The more numerically stable posterior error covariance is given as

$$\mathbf{P}_{\mathbf{x}_{i|i}} = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{P}_{\mathbf{x}_{i|i-1}}(\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)^{\mathrm{T}} + \mathbf{K}_i\mathbf{P}_{\mathbf{y}_{i|i-1}}\mathbf{K}_i^{\mathrm{T}} \ . \tag{6}$$

Given that the optimal Kalman gain is used the above simplifies to the expressions used in this chapter. If the application is suffering from poor numeric stability, or a non-optimal Kalman gain is used, then the simplified equations do not hold and the above formula (6) should be used [36].

5.4 Beyond the Kalman filter

The Kalman filter successfully estimates the solutions to a wide variety of problems. Unfortunately, though, they are unable to cope with severe nonlinearities; thus there are some problems which cannot be solved effectively using these filters.

In some cases a sufficient degree of linearity can be restored to the problem by allowing the Kalman filter to act using smaller intervals of time. In such cases Kalman filters can offer an adequate solution to highly non-linear problems. Such a technique is, however, not universally applicable as practical considerations of many problems make it unfeasible to perform observations in sufficiently small steps of time.

Previously, in Section 4.1.5, an example of such a non-linear problem was mentioned: tracking of entities in a complex scene. The section proceeded to describe the Condensation algorithm which offers a robust solution to the problem. Although developed independently the Condensation algorithm and particle filters are considered to be the same. Particle filters themselves are also referred to as Sequential Monte Carlo methods—indicating that they are a subset of the more general Monte Carlo methods. This section will expand on the theory and application of such particle filters.

Particle filters may be likened to a form of mathematical simulation: they solve problems not deterministically, but through stochastic means—that is to say they make use of random samples and estimate the solution to a problem by observing the behavior of those samples using the underlying probabilistic model that describes the problem.

One of the earliest successes of this form of solution estimation was performed by Enrico Fermi in the 1930's. He used a random sampling method to calculate the properties of the newly discovered neutron as well as perform the simulations necessary to develop the Manhattan Project [66].

The nature of Monte Carlo methods, however, makes them arduous to use in manual calculations: the many repetitive computations required are laborious to perform by hand and

thus are particularly suited for electronic evaluation. The advent of computers has elevated Monte Carlo methods, and thus particle filters, to a popular tool for a wide variety of problems. Examples of applications of particle filters include cooperative localization of multiple robotic entities [92], multiple object tracking [44], and abnormality detection [79].

A thorough discussion on particle filters is given by Ristic, Arulampalam and Gordon in the book "Beyond the Kalman filter: particle filters for tracking applications" [93], or alternatively refer to Doucet *et al*'s treatise "On Sequential Monte Carlo sampling methods for Bayesian filtering" [30]. This text, in contrast, will only outline the basic methodology of particle filters.

Particle filters, similar to Kalman filters, assume the problem can be described using two models—a process model and an observation model:

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, \nu_k) \\ \mathbf{y}_k &= g(\mathbf{x}_k, \omega_k) \end{aligned}$$

where $\mathbf{x}_k$ describes the estimated state at time k, f is a known process modeling function, and ν is the noise inherent in the system; furthermore $\mathbf{y}_k$ is the observation at time k, g is a known observation modeling function, and ω represents measurement noise.

Particle filters then use samples that are propagated using the process and observation models to implicitly describe the probability density function of the solution. An example of such an approach is the variant known as the Sampling Importance Resampling filter, which will be described in greater detail at this point as a representative for particle filtering methods in general.

Sampling Importance Resampling filters require knowledge of the process and observation models, f and g, a means to draw samples from both the prior state estimates and the process noise distributions, as well as a likelihood function $p(\mathbf{y}_k|\mathbf{x}_k)$ (required to be accurate up to proportionality) which denotes the probability of observation $\mathbf{y}_k$ given state estimate $\mathbf{x}_k$.

Assume that a set of N samples from the posterior distribution $p(\mathbf{x}_{k-1}|\mathbf{y}_1, \ldots, \mathbf{y}_{k-1})$ with $k > 0$ is available. Denote that set as $\{\overline{\mathbf{x}}^i_{k-1} : i = 1, \ldots, N\}$. Then a set of predicted prior samples can be computed as follows:

$$\underline{\mathbf{x}}^i_k = f(\overline{\mathbf{x}}^i_{k-1}, \nu^i_{k-1})$$

where ν^i_{k-1} is an independent instance of system noise. These prior samples form a representation of the prior probability density function $p(\mathbf{x}_k|\mathbf{y}_1, \ldots, \mathbf{y}_{k-1})$.

The prediction step is followed by an update step that integrates the most recent observations into the solution estimate. The first part of the update step is the computation of a set of weights corresponding to samples in the predicted prior set, yielding

$$\hat{w}^i_k \propto p(\mathbf{y}_k|\mathbf{x}^i_k) \ .$$

These weights are then normalised for all $i = 1, \ldots, N$ to sum to unity

$$w_k^i = \frac{\hat{w}_k^i}{\sum_{n=1}^{N} \hat{w}_k^n} \ .$$

Finally a set of posterior samples, $\overline{\mathbf{x}}_k^i$, is created from the set of prior samples, $\underline{\mathbf{x}}_k^i$. Each element of the posterior set is selected from the prior set randomly. The probability of picking a particular sample is equal to its normalised weight. The same sample may be selected multiple times—the probability of picking a sample is not changed. The process of picking samples is repeated N times, thus forming a posterior set that contains as many elements as the prior set.

In other words the prior samples $\underline{\mathbf{x}}_k^i$ are resampled, with replacement, according to the normalised weights to produce a new set of samples $\{\overline{\mathbf{x}}_k^i : i = 1, \ldots, N\}$ such that

$$\Pr(\overline{\mathbf{x}}_k^i = \underline{\mathbf{x}}_k^j) = w_k^j \qquad \forall i, j = 1, \ldots, N \ .$$

The computed posterior samples describe the probability density function $p(\mathbf{x}_k | \mathbf{y}_1, \ldots, \mathbf{y}_k)$. The posterior samples may serve as input to compute the system at the next time step—thus a cycle of the algorithm is complete.

Numerous variations of particle filters exist which differ in the selection of samples, weight computation, choice of resampling methods, and other details. Problems that are addressed to varying degrees in different particle filtering methods include degeneration of sample space, sensitivity to noise, and efficiency.

An important aspect of particle filters is a good choice of initial sample set. It is not uncommon to make use of extended or unscented Kalman filters to compute a viable initial set of samples which can be used to initialise a particle filtering method.

Some approaches create an even more prominent interaction between Kalman and particle filters. For example, Van der Merwe *et al* [114] proposed the *unscented particle filter* which replaces the simple prediction step of a standard particle filter with a full unscented Kalman evaluation. The resulting filter exceeds both pure Kalman filters and pure particle filters in accuracy, robustness and insensitivity to extremes.

This concludes the discussion on Kalman filters. The following chapter details the structure-from-motion solution adopted in this thesis.

It is said that the present is pregnant with the future.

Voltaire

Chapter 6

Structure from Motion

Nothing that I can do will change the structure of the universe.

Albert Einstein

Though the universe may be immutable—it is certainly possible to discern some of its structure. This chapter presents a brief overview of the structure-from-motion problem, as well as an explanation of the methodologies employed in the solution adopted in the context of this thesis.

The structure-from-motion problem is ill-posed and sensitive to noise. Consequently a wide range of solutions exists that attempt to address a particular subset of the problem. However, as Oliensis described in his critique of structure-from-motion algorithms: "[...] any one SfM algorithm is unlikely to perform well in all situations" [82]. The following section outlines some of the approaches that have been followed.

6.1 Background

Informally, solutions to the structure-from-motion problem can be divided into two primary approaches: sequential and batch processing. Sequential—or fusion—solutions attempt to infer depth through an iterative procedure. When new data is obtained it is used to update existing estimates. Sequential solutions are not as robust or accurate as batch solutions, but they have the distinct advantage that they are suited to real-time applications.

Batch solutions, on the other hand, compute structural data once all measurements are available. Such batch solutions are, by their nature, computationally expensive—but they are more resistant to noise than sequential algorithms and generally lead to better results due to relying on a more robust large data set. Chiuso *et al* [22] offer a good example of a batched

structure-from-motion algorithm.

Oliensis [82] details the relative advantages and drawbacks of both batch and sequential methods. Although somewhat dated, his report captures and describes the majority of structure-from-motion algorithms and is recommended as a thorough overview into the background of, and issues concerning, the structure-from-motion problem.

Sequential solutions come in a variety of flavors, depending on the emphasis their respective authors placed on the methodologies and solution space. Many real-time, or near real-time, approaches make use of variants of the Kalman filter, for example Azarbayejani and Pentland [8] as well as Yao and Calway [126], Rautenbach [90], Malan [72] and Venter [119] who made use of the unscented Kalman filter that also forms the basis of structure estimation within this thesis.

Examples of non-Kalman based solutions include the work by Morita and Kanade [78] as well as Tomasi and Kanade [109]. Both of these make use of factorization methods in conjunction with orthographic projection to estimate shape parameters. The orthographic camera model is a special case of the more general perspective projection model that has the property that the structure-from-motion problem becomes linear, allowing it to be solved with the use of matrix factorization.

While most methods presuppose static environments or rigid objects, some algorithms are designed to cope with more general scenes such as multi-body or segmented-body reconstructors. Segmented structures, sometimes also referred to as articulated structures, consist of a single entity that consists of multiple bodies that can move independently—but are constrained by rules that describe the relation between bodies.

Consider, for example, how the human body consists of several parts that can move independently of each other, but are confined within particular relative motion parameters. Scheffler *et al* [94] and Taycher *et al* [106] are examples of solutions that can recover articulated structural data. Such solutions, that make use of some parametric model to describe the laws that govern the underlying motion, are also known as model-based methods.

The n-view problem is a subset of the structure-from-motion problem that considers scenes with more than one independently moving body. Solving such scenes is a formidable task and solutions require robust probabilistic analysis to determine which scene elements are associated with independent objects. These objects do not have to restrict their motion to some rule. Instead, they may move in any arbitrary manner. The solution methodologies for structural estimates is identical to single-object methods; however, an additional layer of complexity is needed to determine which parts of projected images correspond to which object. Costeira and Kanade [24], as well as Kanatani [60], present methods which can cope with such problems.

Some solutions to the structure-from-motion problem relax the requirement to consider only rigid objects, making use of elastic objects which may deform freely instead. The only constraint placed on such objects is that their deformation must exhibit some form of continuity.

Ullman [110] described a method for recovering depth data from "rubbery motion" as early as 1984, and Meyer *et al* [75] is a more recent addition to research performed in this subfield.

Model-based methods, as mentioned previously, make use of a parametric model that describes what range of motion is possible for a set of objects. Such solutions are particularly suitable for certain tasks such as human motion capture, but are less common outside scenarios in which particular motion characteristics may be presupposed.

Bregler and Malik [17], for example, assume the human skeletal structure as the underlying kinetic model for the purposes of human motion capture and reconstruction. On the other hand, Gavrila and Davis [34] initialize their solution by manually allocating limb and joint parameters, then reconstruct subsequent motions using twists and exponential maps. [1]

However, model-based paradigms may be applied to general solutions as well, as exemplified by Qian *et al* [88]. Qian *et al* make use of traditional structure-from-motion parameters of translation and rotation, and fuse them with a model of inertial kinematics. The resulting solution is particularly apt at reconstructing scenes containing natural motion.

The various efforts in the field of structure-from-motion have in common that they may cope well with a particular set of problems. However, they do not perform universally well over all scenarios. It might be supposed that information fusion strategies would excel in this situation, but unfortunately that is not yet the case: there are no consistently reliable structure-from-motion solutions for particularly difficult scenarios, let alone most real-world scenarios. Examples of such difficult scenarios include infering the structure of a body that undergoes unknown transformations and infering the structure of an object that is partially transparent or reflective.

The following section details the methods employed within this study for reconstructing dense enclosed environments.

6.2 Implementation

The previous chapters of this study outlined various issues related to the solution implemented in this study. These issues form the basis on which reconstruction can be performed. This section elaborates on the implementation of the solution. The following subsections will describe the reconstructive process, input state definition, as well as the adopted observation- and the process-models.

[1] Exponential maps are a compact representation of a motion matrix, much like quaternions describe a rotation matrix.

6.2.1 Reconstruction process

The overall reconstructive process can take many forms. In relatively straightforward situations the object can be chosen and its features tracked over time—these serve as input to a Kalman filter which estimates the depth parameters. In more complex situations features may be gained and lost over time, requiring dynamic adaptation during the reconstruction process.

In the structure-from-motion problem implemented in this thesis, two important issues influence the reconstruction process. Firstly, only a small subset of the enclosed environment is visible at any one moment in time. Secondly, a very large feature set must be evaluated to produce an accurate scene reconstruction.

Both these issues are addressed simultaneously: the scene is divided into small subsections. These pieces are reconstructed independently and recombined into a complete scene construction afterwards. In other words a piecewise reconstruction is performed.

Camera motion and each piece are defined in such a manner as to allow a complete reconstruction of the piece within the confines of the observational limits. Thus each piece is reconstructed without losing track of features—in other words this approach also eliminates the need to explicitly cater for gain and loss of feature points. Additionally a piecewise reconstruction greatly reduces the number of features that need to be considered at one point in time. This is an important consideration, as the order of execution of the unscented Kalman filter is of the same complexity as matrix multiplication. Computing several smaller batches can be performed significantly faster and with a smaller memory overhead, compared to a single large batch.[2]

The drawback of such an approach is that different piecewise reconstructions need to be recombined into a single overall reconstruction. Since only a scale, rather than absolute, reconstruction is possible there is the issue of compatibility between different parts of the reconstruction—as it is possible for different pieces to be reconstructed to different scales or with varying degree of distortions. Figure 51 on page 91 illustrates this issue: each vertical line in the wireframes represents a reconstructed piece, note the disjunct reconstruction in the typical model in comparison to the clear and well-defined reconstruction in the enhanced model.

Initial experiments using a typical observation model suggested that reconstructed pieces suffered from considerable incompatabilities. An improved observation model has been devised (detailed in Section 6.2.3) that produces experimental results that exhibit a high degree of compatibility—provided the Kalman filter acts on the piecewise inputs for an equivalent amount of time (meaning each input is processed until a fixed number of iterations is reached, as opposed

[2]A $n \times n$ matrix requires at least n^2 accesses to read each entry, hence the theoretically fastest matrix multiplication can be performed in $O(n^2)$ time. Typically the trivial $O(n^3)$ algorithm is used for matrices with relatively small n, and may be replaced by the Strassen algorithm, $O(n^{2.807})$, for greater n (no consensus has been reached on what values of n are sufficient to warrant a transition to Strassen's algorithm, with values cited in the literature ranging from 45 to 128). The current best solution is the Coppersmith-Winograd algorithm, $O(n^{2.376})$, however, it it is impractical for implementation and largely of theoretical interest, providing time-bounds on other algorithms.

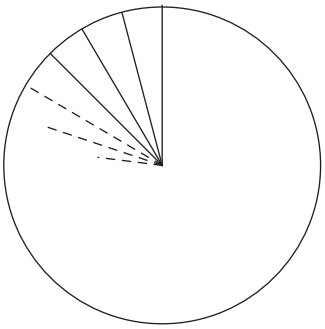
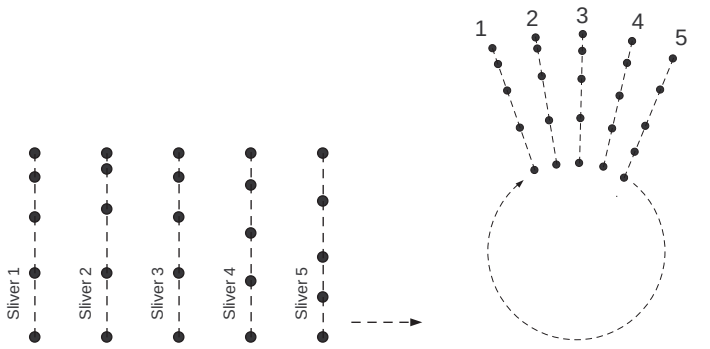


Figure 44: Piecewise reconstruction through slivers.

to processing until some degree of confidence is reached).

The distinction between processing time and confidence is important in this study. The reason for this is that a confidence value is specific to the features that are being reconstructed; neighbouring feature sets converge at approximately the same rate, but the confidence in the convergence depends on the complexity of the underlying structure.

Both the typical and improved observation models are discussed in greater detail in Section 6.2.3.



(a) Individual slivers

(b) Combination of slivers through appropriate rotation

Figure 45: Combining slivers

Figure 44 demonstrates the choice of piecewise reconstruction: typically a complete environment requires reconstruction through 360 degrees. The reconstruction process utilised in this thesis divides the environment into slivers, much like a cake, and independently estimates the depth cues of each sliver. The estimates are reassembled into a complete reconstruction once all slivers are computed, as shown in Figure 45. The figures show a series of pieces, or slivers,

that contain points representing structural estimates—the pieces are then recombined into a single reconstruction by aligning them correctly into a coherent scene.

Within this thesis each sliver spans an arc of one degree. When reconstruction is initiated a sliver is placed along the left-most edge of the field-of-view and features are uniformly distributed along the sliver, as shown in Figure 46—the dots in the figure represent features. Once the depth estimate of the sliver is completed the field-of-view is restored to the initial position and orientation, and then rotated to achieve the initial orientation for the subsequent sliver.

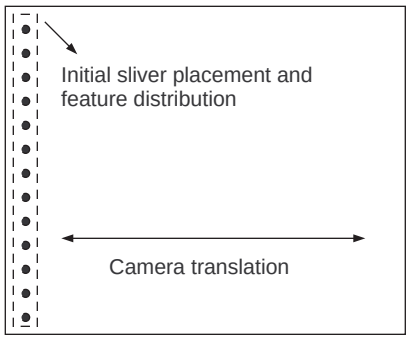


Figure 46: Inital sliver placement, feature choice, and camera translation.

Reconstruction of the sliver is performed through horizontal translation of the camera. The choice of feature points and the limitation of camera motion to the horizontal plane is done deliberately:

To enable the tracking of a dense set of features over a prolonged period of time it is prudent to impose a structure on features chosen for tracking as well as imposing limits on the possible camera motion. These two factors ensure that features are not lost prematurely and that significant information can be gained from their motion.

In structure-from-motion problems, translations of feature points due to camera motion produces depth cues that are interpreted to gain an estimate of the underlying structure. Smaller camera motion produces less pronounced depth cues and greater distortion due to noise—conversely greater camera motion offers more accurate depth cues which are less prone to distortion through noise.

By choosing a dense feature set that is uniformly distributed along the left-most edge of the field-of-view, and confining the motion of the field-of-view to the horizontal plane, the complete dense feature set can be tracked over the largest possible displacement from its initial location. This process produces depth cues that satisfy the demands on reconstruction for both a dense feature set and high accuracy without the need to rely on computationally expensive tracking methods or on error-prone algorithms that cater for feature loss and reacquisition.

A final note on fusion of reconstructions: each reconstruction creates an accurate three-dimensional reconstruction of the enclosed environment based on observations from a particular central point. Such a reconstruction, though useful in its own right, is not likely to fully reconstruct a complex enclosed environment. For example, separate rooms are usually not, or only partially, reconstructed as the initially chosen point of view may have no, or only limited, visual access to it.

Fortunately this is not generally a problem. The technique employed in this thesis allows the independent reconstruction from various locations. Given that the relative location from one center of observation to another is known, it is readily possible to combine the separate reconstructions into a single model using constructive solid geometry.[3]

This concludes the description of the reconstruction process. The following subsection defines the input state to the unscented Kalman filter used in this thesis.

6.2.2 Input state definition

The input state to the unscented Kalman filter used in this thesis consists of the following:

- *Quaternion*—describes the current camera orientation[4], 4 components
- *Vector*—describes the current angular velocity of the camera, 3 components
- *Vector*—describes the current translation of the camera, 3 components
- *Vector*—describes the current translational velocity of the camera, 3 components
- *List*—a list of current depth estimates, 1 component per feature

It is clear that the input to the Kalman filter is divided into two sections. The first part describes the dynamic part of the scene camera, catering for orientation and position, as well as change in orientation and change in position. The second part is a list of values representing the current depth estimates for the scene.

The initial camera orientation is initialized as $(1, 0, 0, 0)$—where $(1, 0, 0)$ represents the vector part and the final 0 represents the scalar part of the quaternion. The angular velocity, translation and translational velocity are all initialized as $(0, 0, 0)$—the camera is thus initially estimated to be stationary. The initial depth estimates are given as $(1, 1, 1, ..., 1)$ in other words the scene is initially estimated as a flat surface. Note that only the depth parameter of the structural estimate is given, since the other parameters are implicit in the reconstruction.

[3]Constructive solid geometry is a popular technique in computer science for the binary manipulation of geometric objects. Typically constructive solid geometry defines three operations (union, intersection and difference) which can be applied to sets of geometric data.

[4]Quaternions are a useful 4-component representation of rotations. For a full description of their properties refer to the excellent online article by Weisstein [124].

The following subsection describes the observation model, along with the enhancement employed within this thesis. The observation model, along with the process model, describes the underlying visual characteristics that enable the Kalman filter to perform depth estimation.

6.2.3 Observation model

In this study, observations assume that the visual cues underlie a linear perspective projection. In other words, if an image plane is defined to be at the origin of a coordinate system with normal vector parallel to the z-axis, and given a point $\mathbf{p}$ where

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

and given that the center of projection lies on the z-axis at $z = -f$, then $\mathbf{y}$ is the linear perspective projection of $\mathbf{p}$ onto the image plane such that

$$\mathbf{y} = \begin{bmatrix} y_x \\ y_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} \left(\frac{1}{1 + \frac{p_z}{f}} \right) .$$

This projection is illustrated in Figure 47 below:

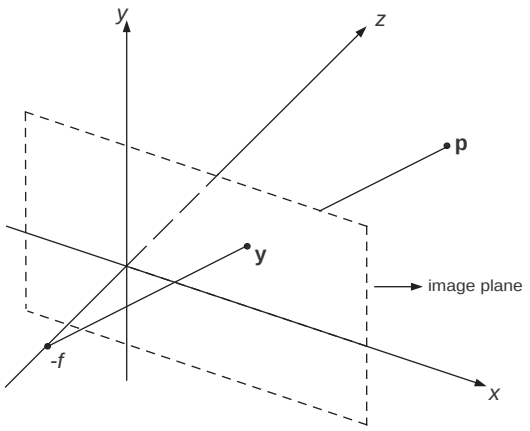


Figure 47: Linear perspective projection of $\mathbf{p}$ onto $\mathbf{y}$ in image plane with a focal length of f

The inverse process is of interest for reconstruction purposes. The equation is given as:

$$\mathbf{p} = \begin{bmatrix} y_x(1 + \frac{p_z}{f}) \\ y_y(1 + \frac{p_z}{f}) \\ p_z \end{bmatrix} . \tag{7}$$

This indicates that it is sufficient to know the focal length f, the projected image coordinates y_x and y_y, as well as the depth parameter p_z to fully reconstruct $\mathbf{p}$. For the purposes of this

thesis the focal length is assumed to be constant. Furthermore, the image point coordinates are obtained from image data and motion estimation—thus the only unknown that is required is the depth p_z of $\mathbf{p}$.

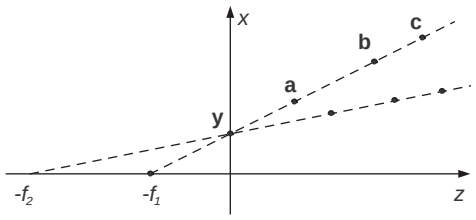


Figure 48: Different points **a**, **b** and **c** yield the same observation **y**. Similarly different focal lengths f_1 and f_2 can yield the same observation.

Unfortunately, as Figure 48 demonstrates, depth parameters cannot be uniquely estimated but only relative to the depth of other points. This implies that a reconstruction of a set of points from an image sequence is only accurate up to a scaling factor. This is not a detrimental flaw in itself, as for many applications it is sufficient to reconstruct a scaled model of the scene. Nonetheless, if at least one absolute reference point is known, then the appropriate scaling factor can be computed and applied to the set of reconstructed points to yield an accurate reconstruction.

When performing three-dimensional reconstruction it is important to distinguish between the camera coordinate space (*ccs*) and object coordinate space (*ocs*). The above equations describe observations in camera coordinate space; however, for reconstruction purposes it is necessary to compute results in object coordinate space. Consider Figure 49 below. It indicates the relation between object and camera coordinate space.

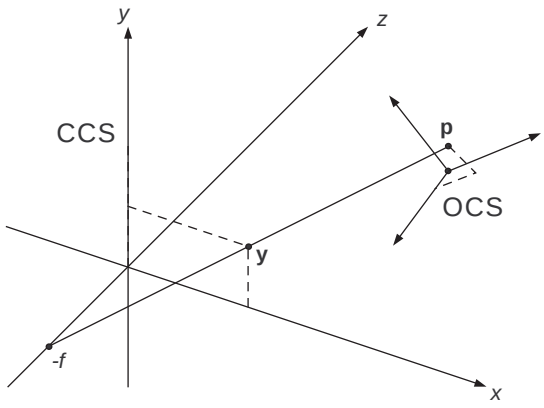


Figure 49: Camera and object coordinate spaces

Given that the origins of the two coordinate systems are displaced through the vector $\mathbf{t}$ and that the orientation of the object coordinate system is related to the camera coordinate system through the rotation matrix $\mathbf{R}$, then the conversion from object coordinate space to camera coordinate space is given by the equation

$$\mathbf{p}^{ccs} = \mathbf{R}\mathbf{p}^{ocs} + \mathbf{t}$$

and conversely the inverse relationship is given by

$$\begin{aligned} \mathbf{p}^{ocs} &= \mathbf{R}^{-1}(\mathbf{p}^{ccs} - t) \\ &= \mathbf{R}^{-1} \begin{bmatrix} y_x(1 + \frac{p_z}{f}) - t_x \\ y_y(1 + \frac{p_z}{f}) - t_y \\ p_z - t_z \end{bmatrix} \end{aligned} \tag{8}$$

where we substitute (7) for $\mathbf{p}^{ccs}$.

The relative orientations of the coordinate systems may change over time; however, the initial reference rotation may be chosen freely. Given that any initial choice will suffice, the most prudent choice is to align the two coordinate spaces with each other:

$$\mathbf{R}_0^{-1} = \mathbf{I}$$

Furthermore, the translation vector $\mathbf{t}_0$ is initialised based on the initial observations of feature points

$$\mathbf{t}_0 = \begin{bmatrix} \overline{y}_{x,0} \\ \overline{y}_{y,0} \\ 0 \end{bmatrix}$$

where $\overline{\mathbf{y}}_0$ is the mean of initial two-dimensional observations. This is reasonable as the sample mean can be expected to approximate the distribution mean.

Finally, by adding the initial conditions into (8) and writing it in time-dependent form, the perspective projection camera model is derived:

$$\mathbf{p}_i^{ocs} = \mathbf{R}_i^{-1} \begin{bmatrix} y_{x,0}(1 + \frac{s_i}{f}) - \overline{y}_{x,0} \\ y_{y,0}(1 + \frac{s_i}{f}) - \overline{y}_{y,0} \\ s_i \end{bmatrix} \tag{9}$$

where $s_i = p_{z,i}^{ccs}$.

Typical observation model

Equation (9) serves as the *typical* observation model for the unscented Kalman filter. During each cycle of the Kalman filter the observation model iterates through each projected sigma point and applies the camera and structural data local to that sigma point to the observation model. Each sigma point represents a sample, a variation of the full system state—similar but

slightly displaced from the mean; Figure 50 illustrates this concept: the dots represent states, the striped bars indicate the actual values of the state. As the figure shows the sigma points are variations on the original state.

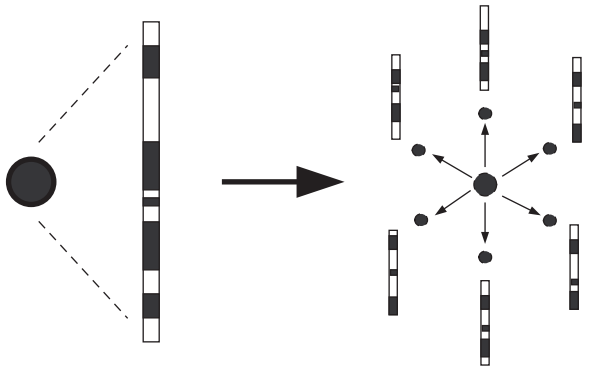


Figure 50: The progression from single state description to a sigma point description of a system

Each sigma point, with its associated state, possesses a local variation of the motion state of the system. The typical observation model makes use of this localised data to compute depth cues. However, the localised motion states can describe slightly different estimates of the local camera which may produce smaller or larger variations in the final structural estimate. Program 6 summarises the idea.

Program 6 Typical observation model

```
m = number_of_sigma_states
n = number_of_points_in_sigma_state

i = 0
while i < m do begin
  camera = extractCameraFromSigmaState(i)
  j = 0
  while j < n
    result[j, i] = computeStructuralEstimate(j, i, camera)
    j = j + 1
  end
  i = i + 1
end
```

Normally this does not pose a problem, as only a single reconstruction is made and any slight shifts due to localised camera variation are averaged into the structure. However, in this thesis a piecewise reconstruction is performed where individual pieces are computed independently of each other; as a result it is important that individually reconstructed pieces are compatible to ease the reintegration into a unified structure. Figure 51 demonstrates the difference in

Program 7 Enhanced observation model

```
m = number_of_sigma_states
n = number_of_points_in_sigma_state
camera = null

i = 0
while i < m do begin
  camera = camera + extractCameraFromSigmaState(i)
  i = i + 1
end
camera = camera / i

i = 0
while i < m do begin
  j = 0
  while j < n
    result[j, i] = obtainStructuralEstimate(j, i, camera)
    j = j + 1
  end
  i = i + 1
end
```

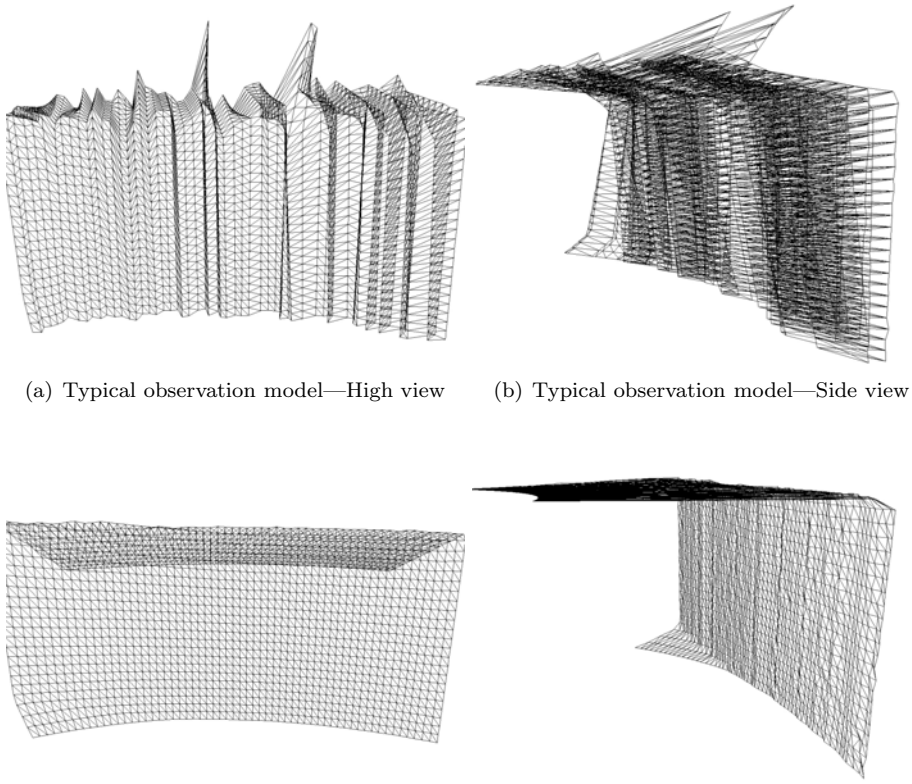
reconstruction from the typical and the enhanced observation models.

Enhanced observation model

The enhanced model is a relatively simple extension of the typical observation model. It was developed from the realization that the Kalman filter consists of two distinct modeling steps, a process prediction step and an observation prediction step. The information that is manipulated in each of these steps is distinct: the prediction step models the motion of the system, leaving the structural data untouched. The observation step makes use of the predicted motion state and the measurements obtained from the system to estimate the structure of the system without changing the predicted motion state.

The motion state of the system remains unchanged during the observation step. Furthermore, each sigma point may possess a distinct camera model. After the observation step a Kalman gain is computed and an improved prediction of the system's state is produced. During this final step the sigma point states are collapsed into a single state again which represents the best estimate of the system at that point in time.

There is, however, no need to adhere strictly to this sequence of events—in fact, before the Kalman gain is computed, the best estimate for the camera model is not the localised motion state of individual sigma points. The best estimate is a fusion of known local motion models across all sigma points.



(a) Typical observation model—High view (b) Typical observation model—Side view

(c) Improved observation model—High view (d) Improved observation model—Side view

Figure 51: Poor and good piecewise compatability

In other words, the enhanced observation model proceeds to compute a premature estimate of the system's motion model as an initialization step. This estimate is computed as the mean of the individual sigma point motion models. The enhanced observation model then proceeds to compute the depth estimates of sigma point states similarly to the typical observation model, though it makes use of the averaged motion model rather than local motion models. Program 7 on page 90 illustrates the algorithmic flow of the enhanced observation model.

6.2.4 Process model

Typically the process model adopted in a structure-from-motion Kalman filter describes the underlying structure and motion characteristics used in the reconstruction process. Usually the majority of data filtered describes the structural estimates, whilst the remaining data models the translation, translational-velocity, orientation and angular velocity of the camera—this thesis utilizes such a structure and camera model.

As described in Section 6.2.1, the environment is assumed to be static. This implies that specific structural process modeling is unnecessary—instead the Kalman filter employed in this

thesis acts purely on depth and camera data. In other words, the motion model declares that the current structural estimate is the best estimate—making changes to the estimate would lessen the quality of the estimate. Accordingly, given that the structural input data is defined as $\mathbf{s}$, such that

$$\mathbf{s} = [s_0, s_1, \ldots, s_k, \ldots, s_{m-1}]^{\mathrm{T}}$$

where m indicates the number of features considered and $s_k = p^{ocs}_{z,k}$, then the state transition for the structure is given simply as

$$\mathbf{s}_i = \mathbf{s}_{i-1} + Q_{i-1}$$

where Q is the process covariance.

Along with the structural part of the process model is a second part that describes camera motion in the system. The camera system utilized makes use of a simple dynamics model that incorporates position, orientation, velocity and angular velocity, such that:

$$\begin{aligned}
\mathbf{v}_i &= \mathbf{v}_{i-1} + Q^{\mathbf{v}}_{i-1} \\
\mathbf{p}_i &= \mathbf{p}_{i-1} + \mathbf{v}_i \Delta t + Q^{\mathbf{p}}_{i-1} \\
\mathbf{a}_i &= \mathbf{a}_{i-1} + Q^{\mathbf{a}}_{i-1} \\
\mathbf{A}_i &= \mathbf{A}_{i-1} + 0.5 \times \mathbf{A}_{i-1} \times \widetilde{\mathbf{a}}_i^T + Q^{\mathbf{A}}_{i-1}
\end{aligned}$$

where t represents the timestep, $\mathbf{v}$ is the velocity, $\mathbf{p}$ is the position, $\mathbf{a}$ is the angular velocity, and $\mathbf{A}$ is the orientation. $\widetilde{\mathbf{a}}$ denotes the skew matrix of $\mathbf{a}$. $\mathbf{v}$, $\mathbf{p}$ and $\mathbf{a}$ are vectors, $\mathbf{A}$ is a quaternion. Program 8 details the process model.

The combination of the reconstruction process using slivers, along with the underlying observation and process models used by the Kalman filter, enable the reconstruction of depth estimates in a scene. The experimental results of these reconstructions are presented in the following chapter.

> *To put it boldly, it is the attempt at a posterior reconstruction of existence by the process of conceptualization.*
>
> Albert Einstein

Program 8 Process model

```
m = number_of_sigma_states

i = 0      % structural update
while i < m do begin
  result.struc[i] = current.struc[i] + current.strucQ[i]
  i = i + 1
end

i = 0      % velocity update
while i < m do begin
  result.vel[i] = current.vel[i] + current.velQ[i]
  i = i + 1
end

i = 0      % position update
while i < m do begin
  result.pos[i] = current.pos[i] + result.vel[i] + current.posQ[i]
  i = i + 1
end

i = 0      % angular velocity update
while i < m do begin
  result.angvel[i] = current.angvel[i] + current.angvelQ[i]
  i = i + 1
end

i = 0      % orientation update
while i < m do begin
  result.orient[i] = current.orient[i] + current.orientQ[i]
  normalize(result.orient[i])
  skew = obtainSkewMatrix(result.angvel[i])
  result.orient[i] = result.orient[i] + 0.5 * result.orient[i] * skew
  normalize(result.orient[i])
  i = i + 1
end
```

Chapter 7

Results

> *Results! Why, man, I have gotten a lot of results. I know several thousand things that won't work.*
>
> Thomas A. Edison

Edison is a man who did indeed know several thousand things that would not work. Nonetheless his career as an inventor is blessed with well in excess of a thousand patents in the United States of America, and several more in the United Kingdom, France and Germany.

This chapter offers a description of experimental findings produced by the structure-from-motion algorithm for enclosed environments developed in this thesis. The chapter is structured to present specific and detailed results for a cuboidal room, allowing reconstructed properties to be clearly identified; the next section covers complex reconstructed environments, demonstrating feasibility of the approach for scenes with varying properties. This is followed by a discussion on limitations of the environment reconstruction showcased. Finally a section is presented that demonstrates the reconstructor's ability to infer depth data from a real sample.

7.1 Cuboidal room

The first, and most thorough, test of the structure-from-motion paradigm presented in this thesis uses a simulated cuboidal room. Such a room offers significant advantages for the purpose of evaluating the reconstructed scene: it showcases the algorithm's ability to estimate depth parameters along

- flat surfaces,
- parallel and orthogonal surfaces,
- and sudden transitions between surfaces.

These geometric characteristics are particularly taxing to estimate accurately and results may be readily inspected visually—as the human eye is well trained to spot abberrations in flat surfaces and right angles. Figure 52, below, captures a sample of the scene that is reconstructed.

Reconstruction is limited to approximately a quarter of the full scene—as results can be more readily interpreted and the computational time required before results are available is significantly reduced. This allows faster testing and validation of results, which in turn allows testing of a greater diversity of parameters. Experimental results from such a partial reconstruction can be extended to a complete reconstruction without loss of generality. To ease understanding of the methods and parameters used in the reconstruction process a short summary is presented below that lists all primary reconstruction elements:

- The raw input data consists of a simulated virtual environment that makes use of Quake 3 maps for rendering purposes as described in Chapter 3.
- The raw input data is motion tracked using a block-matching algorithm described in Section 4.2.
- The overall reconstruction is performed using many smaller reconstructions. The reconstructed pieces—vertical slivers—are combined into a single reconstruction afterwards, as described in detail in Section 6.2.1.
- Compatibility between slivers is ensured by using an enhanced observation model (Section 6.2.3), as well as performing the same amount of reconstruction (same number of frames) for each piece and maintaining the same visual and motion parameters for the camera during the reconstruction of each piece.
- Camera motion is limited to horizontal translation. This ensures that features tracked are not lost prematurely. Furthermore the constrained motion allows the greatest possible displacement from initial the position which aids depth estimation (page 84).
- Each piece to be reconstructed is tracked for 60 frames—the horizontal speed of the camera is selected such that no features are lost from sight within those 60 frames.
- The unscented Kalman filter is initialized with an initial camera motion estimate of $(1,0,0,0),(0,0,0),(0,0,0),(0,0,0)$; the camera orientation, angular velocity, position and translational velocity respectively. The initial structure is initialized as $(1,1,1,...,1,1)$. Recall that only one parameter per feature is required (see Section 6.2.2).
- The covariance matrices are initialized as diagonal matrices with a constant value along the main diagonal as follows: state covariance 0.0005, model covariance 0.001, and observation covariance 0.0001.
- Each reconstructed piece consists of 32 feature points. In the wireframe images that are presented in this chapter each vertical reconstructed line represents such a piece. The horizontal connection between these pieces follows a simple rule that matches neighbouring points with each other.

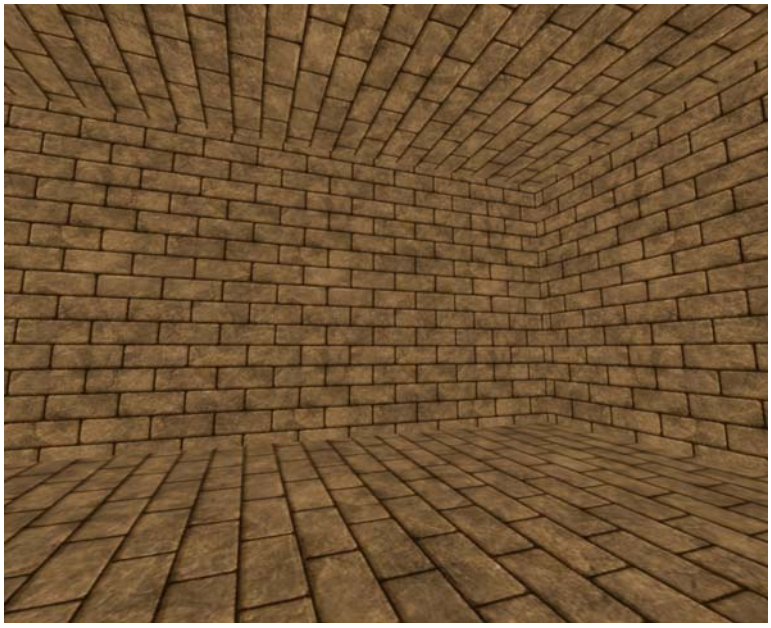


Figure 52: Screenshot of original scene

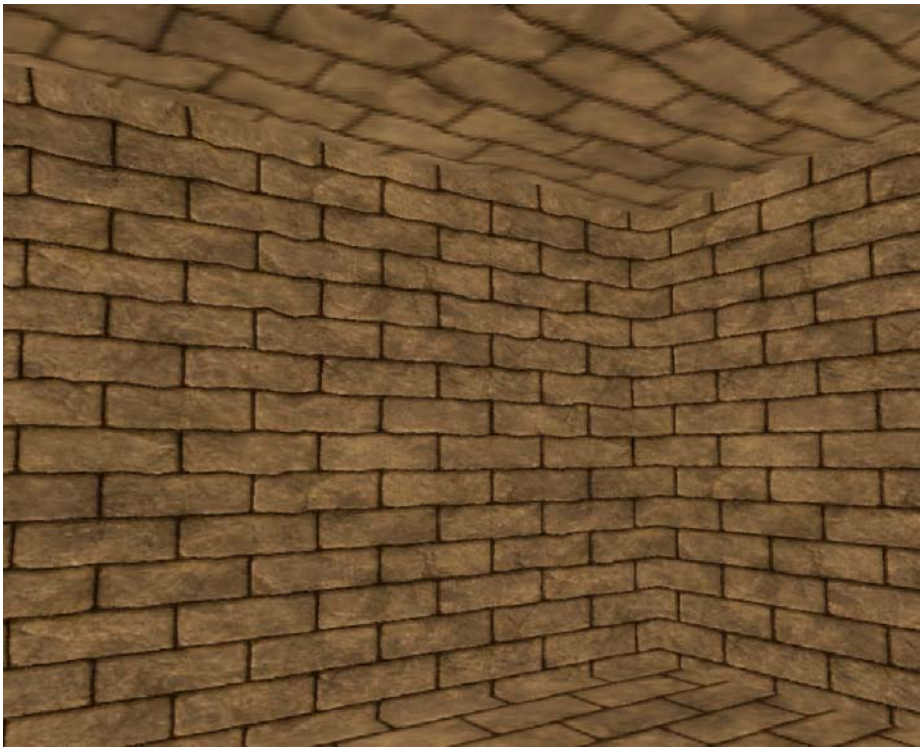
It should be noted that the texture data in the scene is repetitive and offers only a limited degree of distinct features for the purposes of a dense optical flow solution—as dense flow models do not have the freedom to choose feature rich sections of the texture such as T-junctions. The texture data, however, possesses adequate localised variation to allow tracking of arbitrary points at a sufficient degree of accuracy for the purposes of depth reconstruction.

Figures 53a and 53b respectively display close and distanced screenshots of the reconstructed scene. The scenes are re-textured by projecting an environmental map onto structural estimates. The environmental map is generated during a scene traversal by accumulating pixel-sized slivers of observation and compiling the results into textural data sets. Figure 54 displays part of such an environment map.

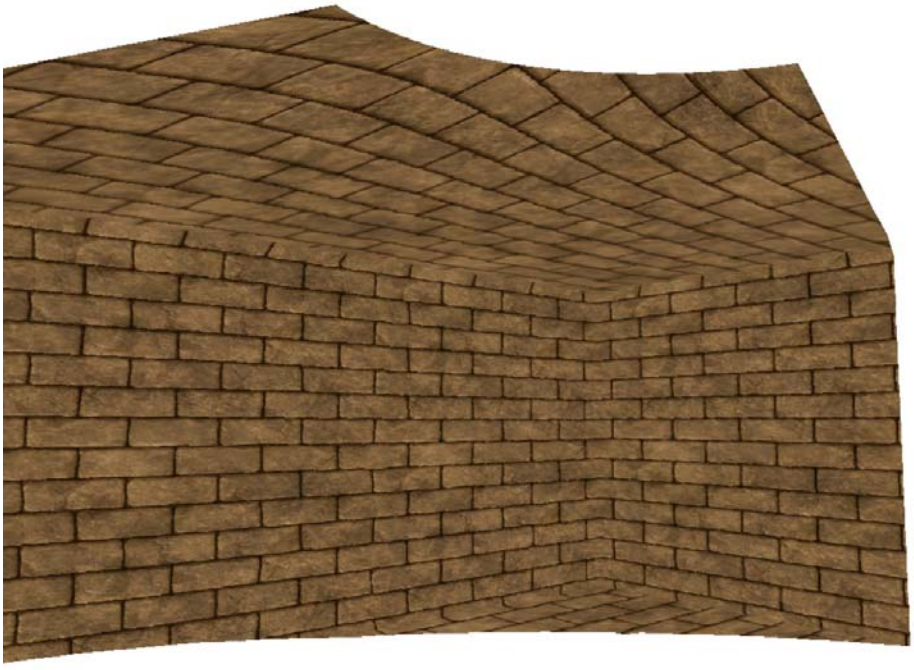
7.1.1 Application of texture

Mapping the textural environment onto the reconstructed structure is straightforward, as it merely requires a radial application of texture coordinates to structure coordinates. However, the technique of creating an environment map and projecting it onto the structural estimate offers only limited accuracy. The texture portrays the scene reasonably well when observed from the source of reconstruction. As Figure 55 demonstrates, however, the texture may exhibit distortions and poor alignment when viewed from alternate viewing angles.

An optimal texturing solution would perform a piecewise re-texturing of the scene using only



(a) Close-up



(b) Distanced

Figure 53: Screenshots of reconstructed scene

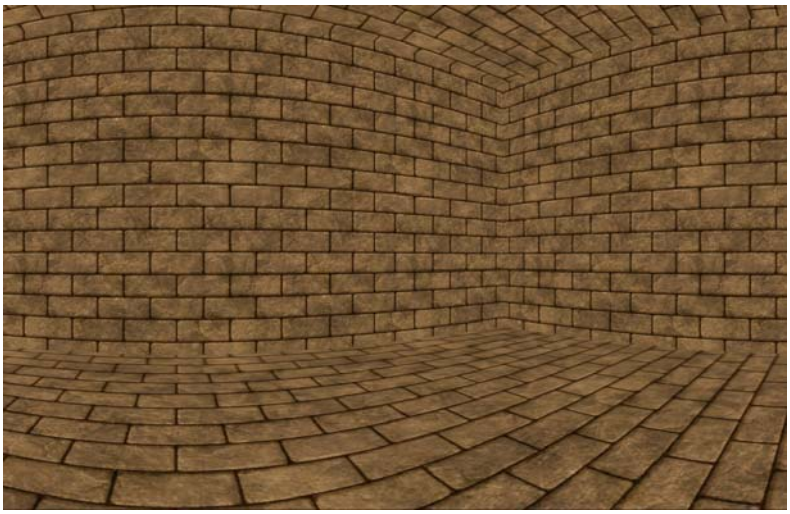


Figure 54: Partial environment map

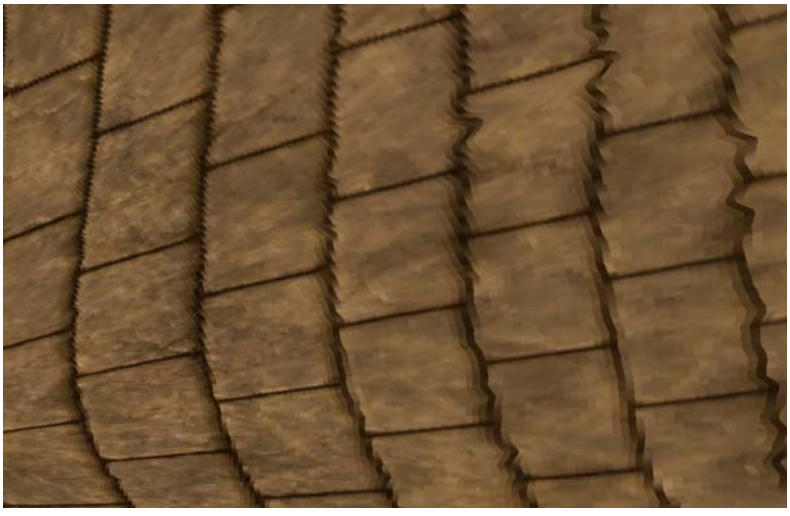
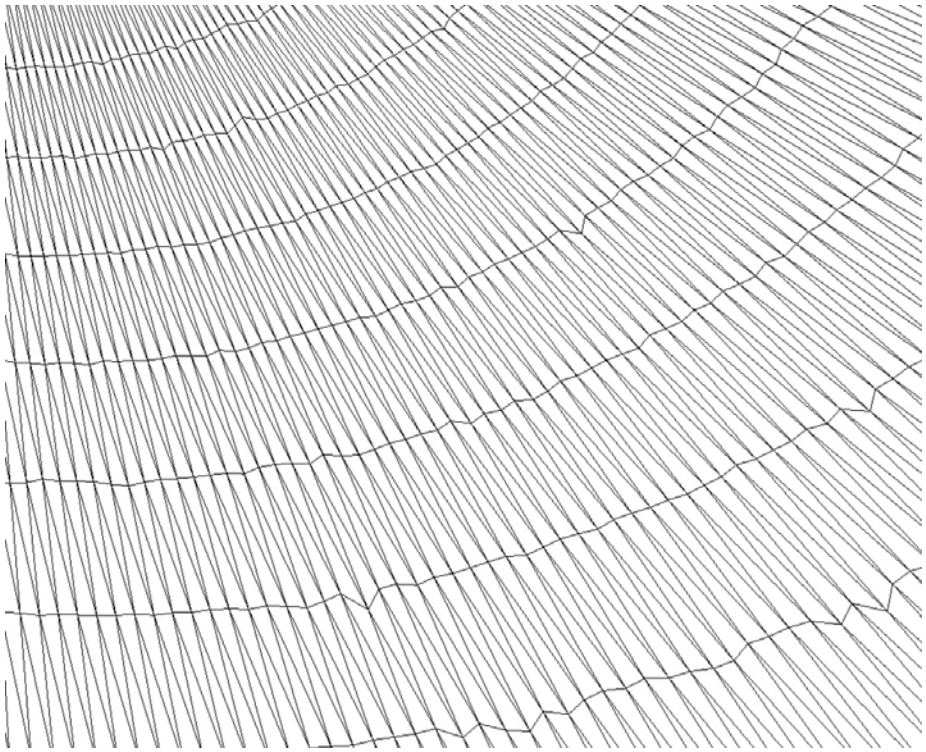


Figure 55: Poor texture alignment and distortion

orthogonal texture data and distortion-free projection onto the structure. Such a process is arduous and generally requires a human operator to achieve good results.



(a) Plan view

(b) Level

Figure 56: Accurate floor and ceiling estimates

7.1.2 Floor and ceiling

Figures 56a and 56b demonstrate a high degree of accuracy in the estimation of floor and ceiling sections. The top-down plan view of the ceiling in Figure 56a displays an even distribution of reconstructed points with only subtle noise introduced by a couple of deviations from the norm. Figure 56b reinforces that impression with a leveled view of the ceiling which shows no obvious reconstruction errors. The nearly perfectly planar surface of the ceiling lends testimony to the accuracy of the reconstruction process.

The high accuracy of ceiling and floor sections may be attributed to the surfaces being parallel to the viewing direction: individual flaws and noise in depth estimates are less obvious as the errors present themselves in shifts in depth. Since both the floor and ceiling are parallel to

the depth axis it is to be expected that their reconstructed counterparts are perceived to be accurate.

Features reconstructed along planes that are parallel to the viewing axis are, subjectively, less prone to errors as the errors present themselves in shifts along the viewing axis. In other words, although the reconstruction is not inherrently stronger in such planes, the results appear more accurate as the errors are less obvious.

This interpretation of accuracy is demonstrated in Figure 57, which displays a section of a wall—orthogonal to the depth axis. The reconstruction of the wall segment is good; however, compared to floor and ceiling, the estimates suffer from some degree of noise, sufficient to be readily perceived by inspection. Assuming that noise is evenly distributed along all areas of reconstruction this confirms that depth estimates are perceived as more or less accurate depending on their angle towards the depth/viewing axis.

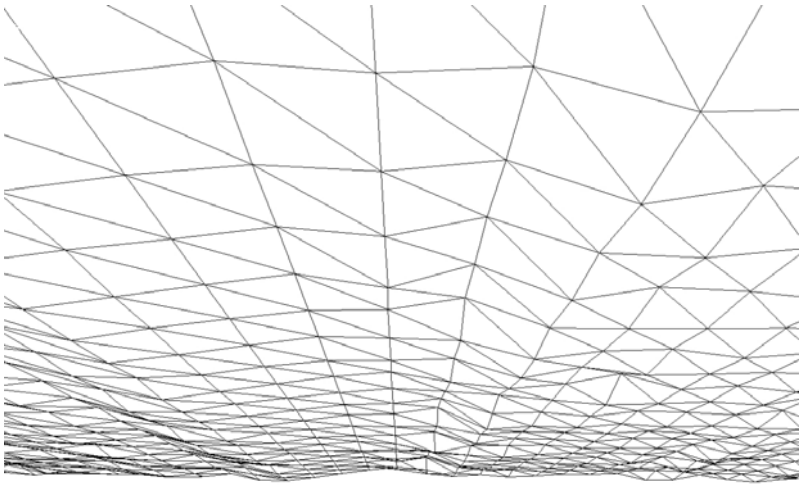


Figure 57: Slightly erratic reconstruction of wall segments orthogonal to the depth axis

It should be noted that the perceived accuracy of depth estimates is inversely proportional to the perceived accuracy of texture data. Consider Figure 53: errors in texture alignment and distortion are most obvious in floor and ceiling segments and least prominent in wall sections. The relationship between perceived error in depth estimate and texture data serves to emphasize that errors exist in all scene sections that are reconstructed; however, the perception of these errors—or the way they are expressed—may vary.

Figure 58 depicts a corner section of the scene, note the corner in the upper right. The reconstruction accurately captures the abrupt angle transition as two wall sections and the ceiling meet—indicating the ability of the estimator to compensate for acute transitions in scenes. In contrast, the estimation of wall sections is not perfectly straight but follows a gentle curve, as shown in Figure 59 along the top. Figure 60 displays a full view of the reconstruction.

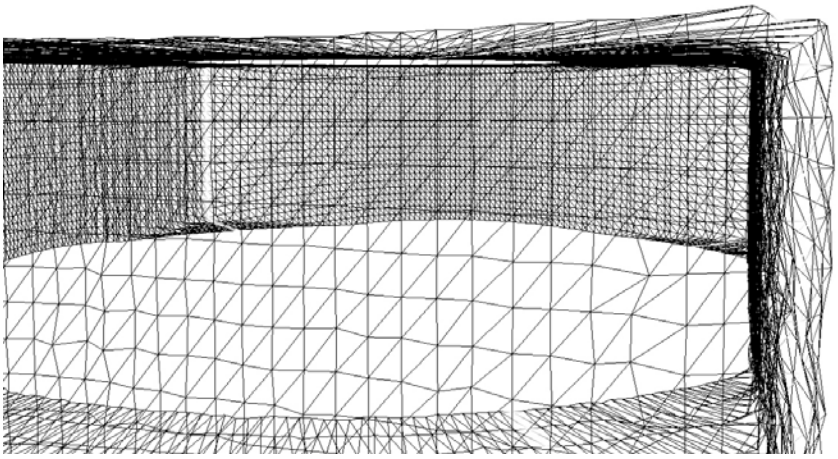


Figure 58: Corner—abrupt and accurate transition between wall, ceiling and floor elements

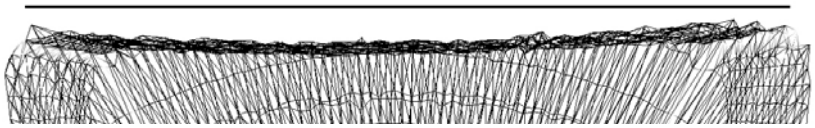


Figure 59: Curvature along wall segment as viewed from above

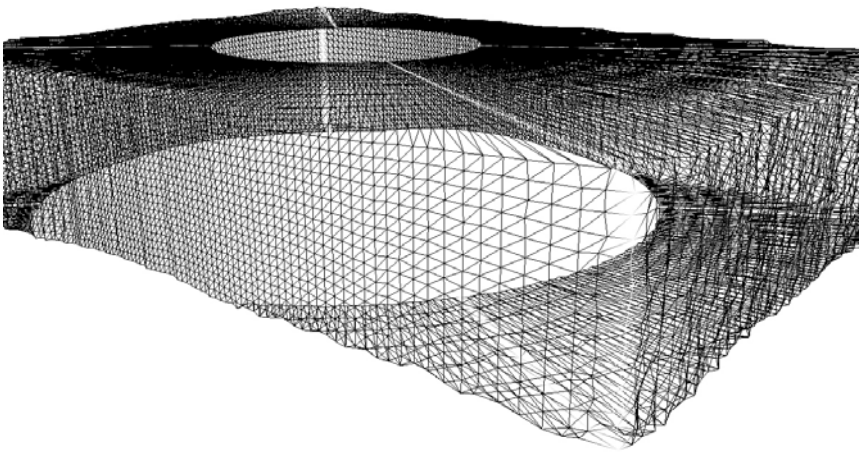


Figure 60: Wireframe of reconstructed scene

7.1.3 Objective accuracy

The discussion of reconstructive accuracy to this point has emphasised subjective and perceived accuracy. However, making use of a virtual environment has several advantages—one of which is that accurate structural data is known and can be compared to the reconstruction. This subsection offers an objective measure of accuracy for the reconstruction of the cuboidal room.

The cuboidal room has the benefit that true and estimated structure can readily be compared. For this purpose the reconstructed data is rotated to align its principle axis with that of the display, as shown in Figure 61. This is not an essential step, but it makes subsequent comparisons simpler.

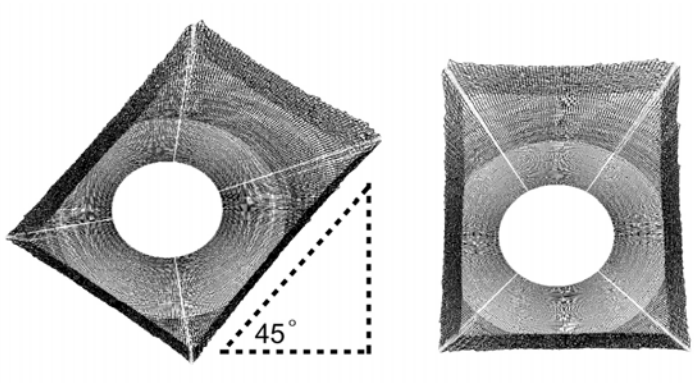


Figure 61: Relative error computation, aligning reconstruction by rotation

Only wall elements are taken into account for the objective accuracy results. Floor and ceiling are omitted as they are structurally nearly perfectly accurate—whereas walls display the greatest deviation from the desired norms. The reconstruction method used only allows estimation upto a scaling constant. To compensate for this, the error computation computes a relative error by dividing the observed deviations by the diagonal of the reconstructed model, as illustrated in Figure 62.

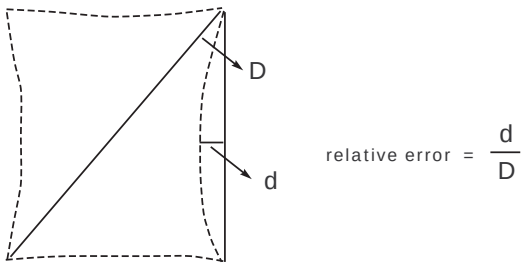


Figure 62: Relative error computation

The objective reconstruction error graph is presented in Figure 63 below:

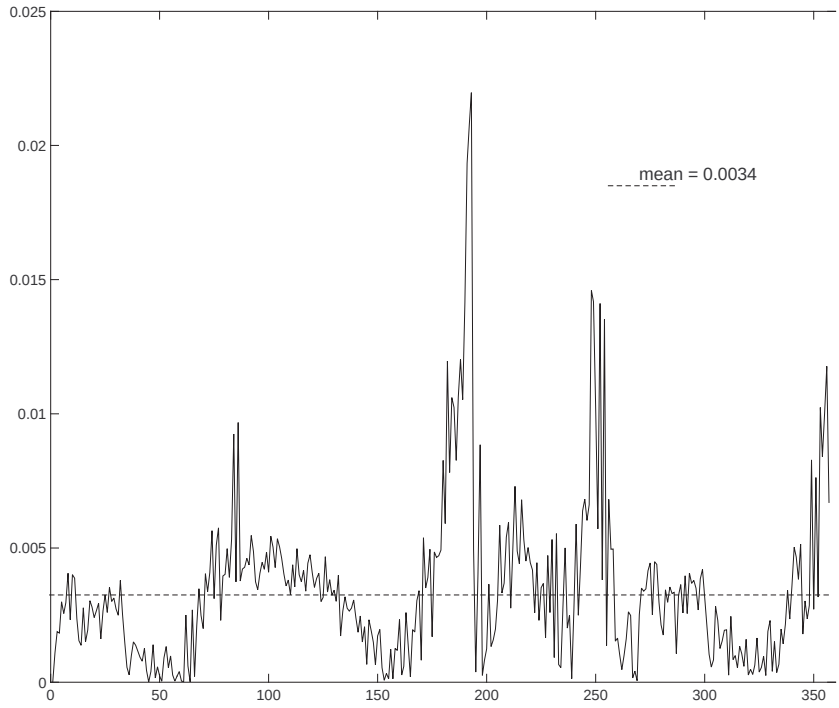


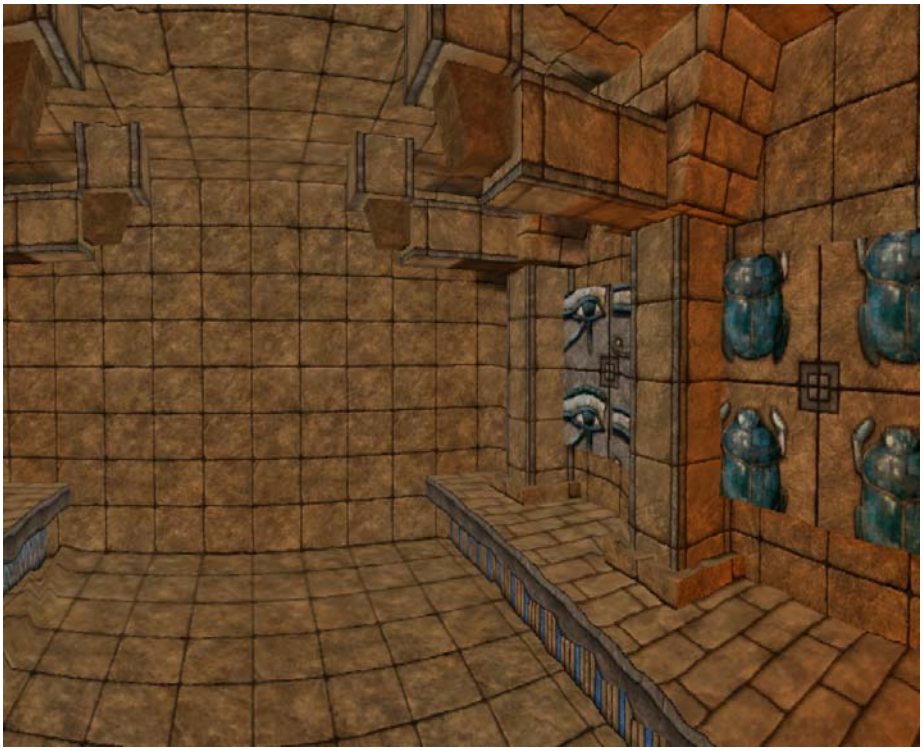
Figure 63: Relative error graph

The horizontal axis represents an element of each of the 360 reconstructed sections along with its relative error. It is readily apparent that the reconstruction is of an exceptional quality: the mean deviation from the true value is as little as 0.34%. The largest error is 2.25% of the diagonal of the whole scene.

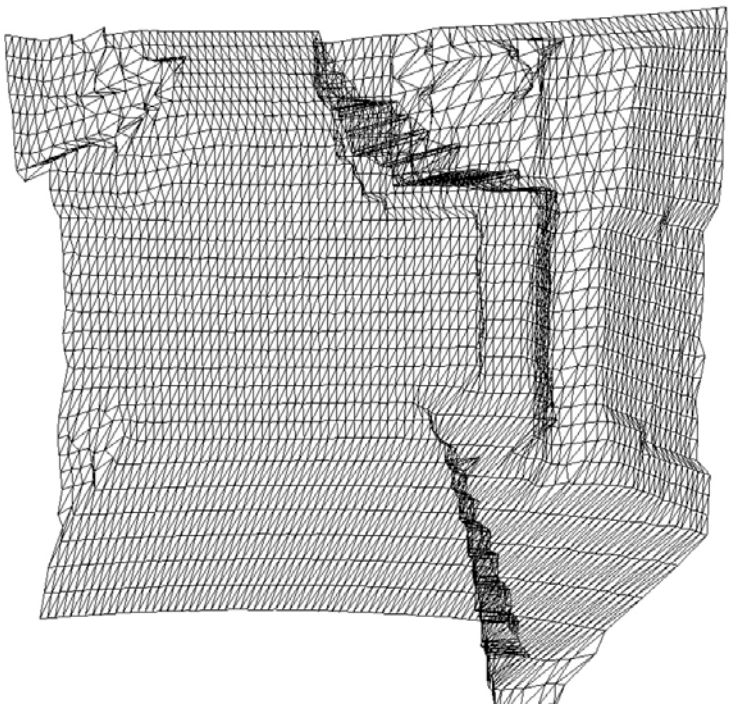
7.2 Other scenes

This section demonstrates the estimator in a variety of scenes. These scenes showcase enclosed environments of a complex nature and illustrate the ability of the reconstruction process to infer depth in scenes of quasi real-world complexity. The scenes are shown in two parts: a re-textured, in-perspective image showing the scene from the position of the camera and an off-perspective wireframe. The wireframed images are displayed from a displaced perspective that allows the wireframes to emphasize the structure of the reconstructed scene. A collage of reference images is shown in Figure 69.

It should be noted that the re-textured scene is not as fully detailed and accurate as the images may suggest—the texture itself contains many visual cues that the human eye is trained to interpret. Nonetheless, when comparing the wireframe with the textured environment it is



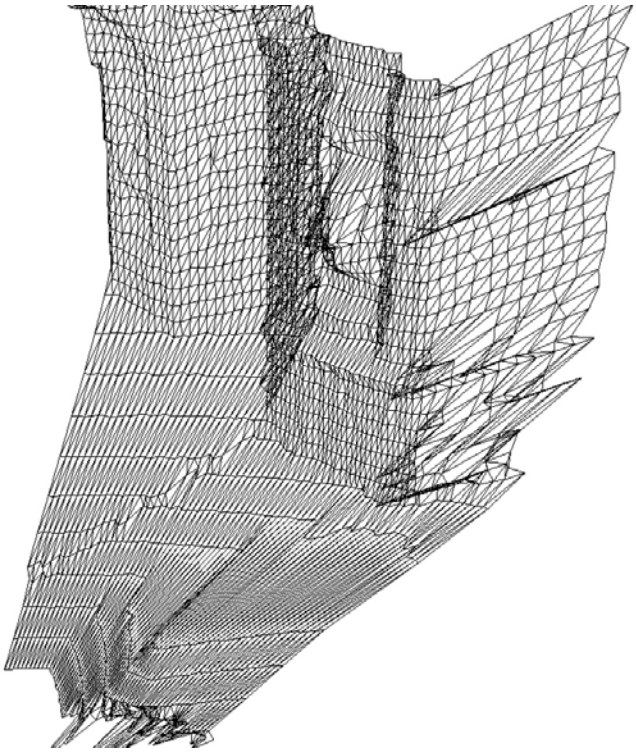
(a) Textured



(b) Wireframe

Figure 64: Egyptian temple

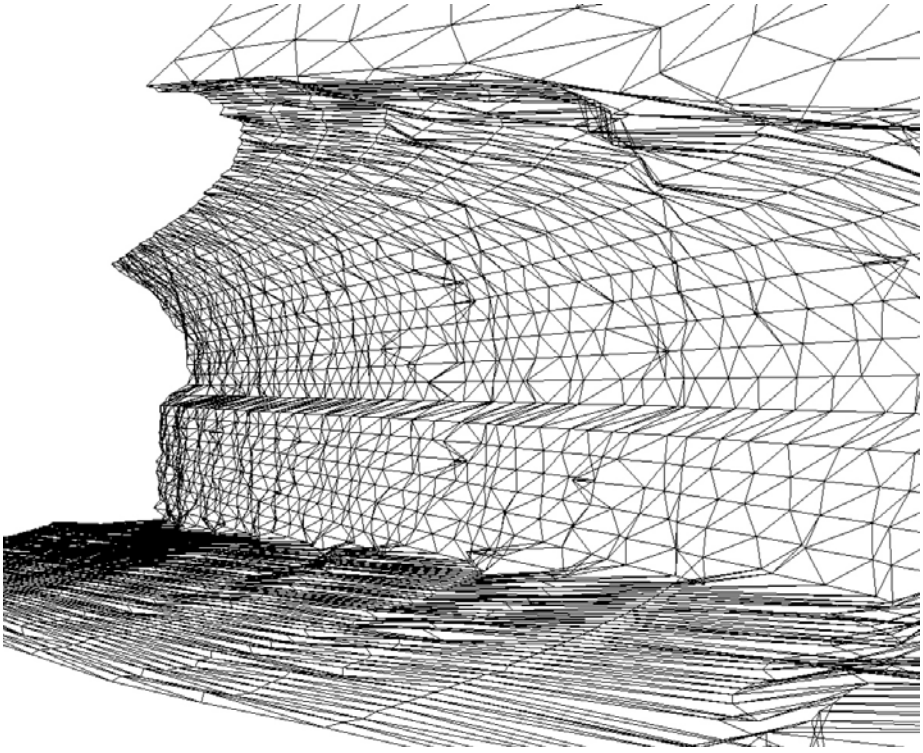
(a) Textured



(b) Wireframe

Figure 65: Wall with Lion

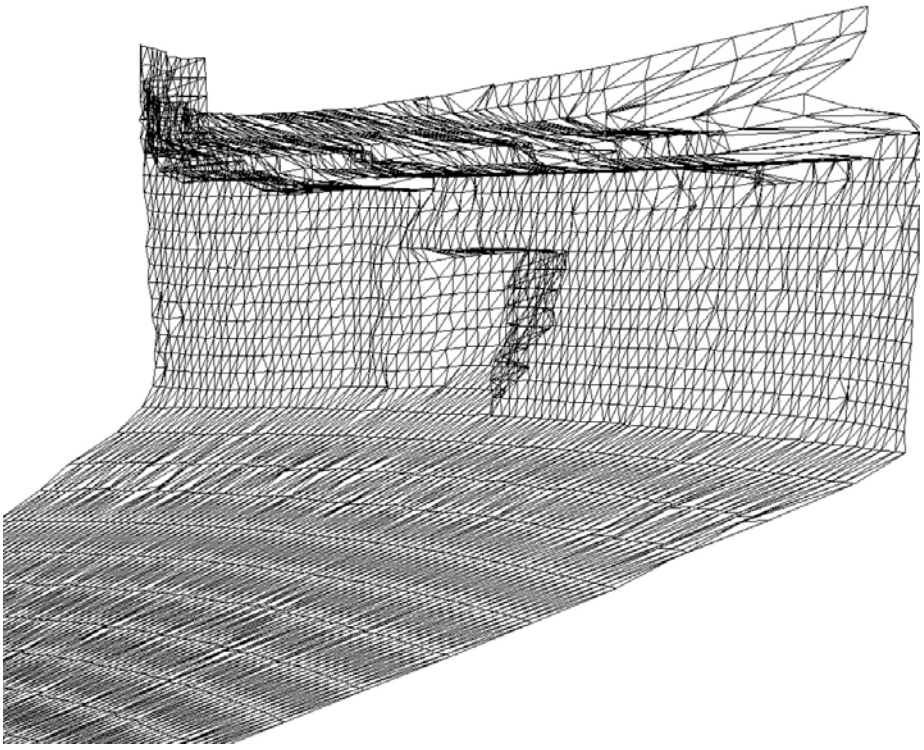
(a) Textured



(b) Wireframe

Figure 66: Arching Wall

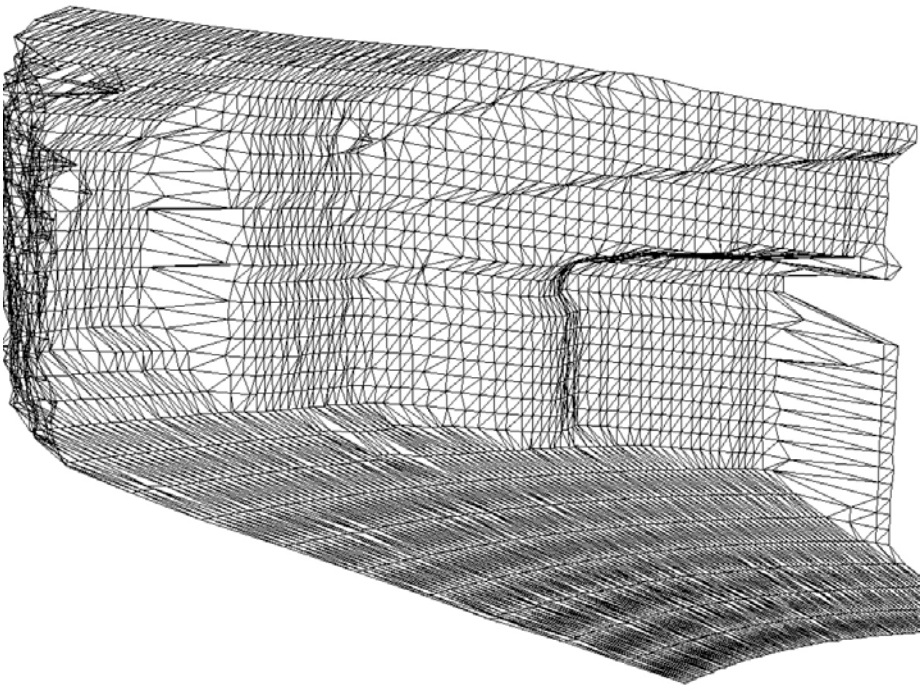
(a) Textured



(b) Wireframe

Figure 67: Entrance with Stairs

(a) Textured



(b) Wireframe

Figure 68: Room with Depth Details

Figure 69: Reference images from the virtual environment renderer

clear that considerable structural detail has been recovered from the input stream.

- Figure 64 demonstrates reconstruction in an environment rich in depth contrast.
- Figure 65 is an example of the estimator's ability to determine subtle depth variations.
- Figure 66 shows the reconstructor's capacity to extract smooth and curved transitions from input data.
- Figure 67 exemplifies indistinct and poorly lit environments.
- Figure 68 reconstructs a scene with diverse depth features.

7.3 Rate of reconstruction

This section describes the speed at which different aspects of reconstruction are computed. Two primary concerns exist in this regard: the time required to compute one frame of reconstruction and the number of frames required for the Kalman filter to converge.

All references to execution speed are relative to the workstation used for experiments in this thesis, an AMD Athlon64 3200 clocked at 2GHz and equipped with 1GB of memory.

7.3.1 Per frame

The bulk of computational time while processing a frame is spent within three systems:

- Virtual environment rendering and acquisition of render results
- Motion tracking of render data
- Reconstruction motion data using a Kalman filter

Table 2 summarises the computational times required for a complete cycle:

	Time in ms	Percentage
Rendering and acquisition	10.993	10.5
Motion tracking	66.118	63.2
Reconstruction	27.580	26.3

Table 2: Summary of time spent in reconstruction processes

It is clear that motion tracking requires the largest amount of computational resources whereas the actual reconstruction is fast. The Kalman filter used in this study only processes a relatively small set of features during each frame due to piecewise reconstruction—typical Kalman-based reconstructors make use of a larger amount of feature points and correspondingly spend more time performing reconstruction computations.

7.3.2 Convergence

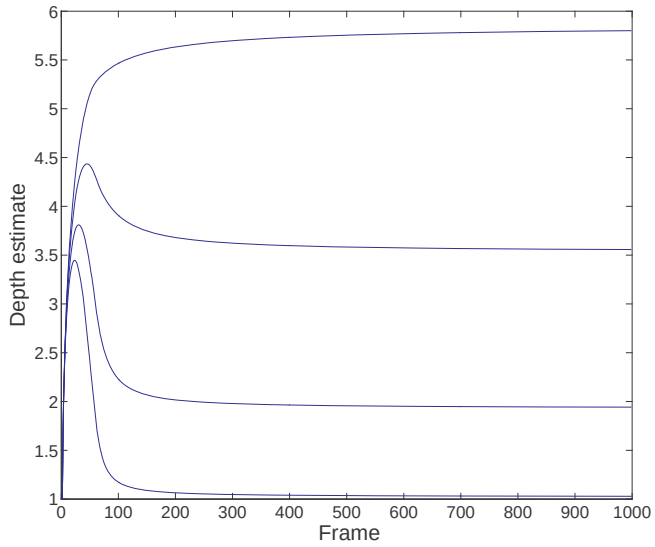


Figure 70: Depth estimates over time, z-coordinate in object coordinate system

As Figure 70 demonstrates, the reconstruction model rapidly converges after an initial period of uncertainty. The graph depicts the convergence paths of four features from a reconstruction process.

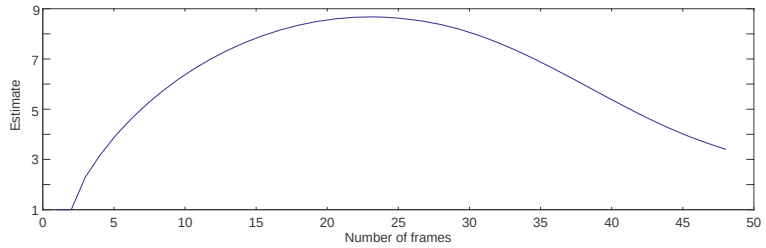
7.3.3 Convergence rate: frames vs displacement

In Figure 70 a good early estimate for the final convergence of features is around frame 100. This is not a surprise as this appears to be typical for Kalman filter based structure-from-motion solutions. See for example Rautenbach [90].

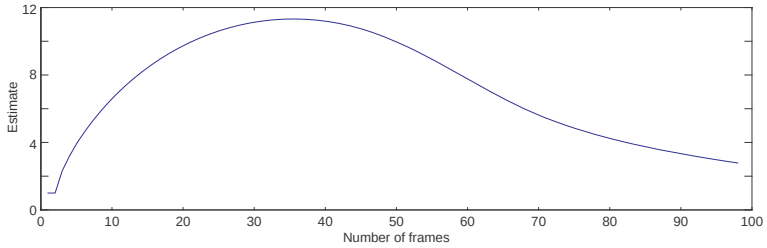
The use of a virtual rendering environment, however, allows great flexibility in reconstruction experiments; and a series of such experiments suggests that the observation that it takes Kalman filter-based approaches around 100 frames to reach a good estimate is an artifact of a deeper insight.

Observe the graphs in Figure 71. Each subfigure demonstrates the estimate convergence for the same scene using varying frame rates. The frame rate determines the horizontal translation of the camera between successive frames. The total translation of the camera in each of the graphs in Figure 71 is the same.

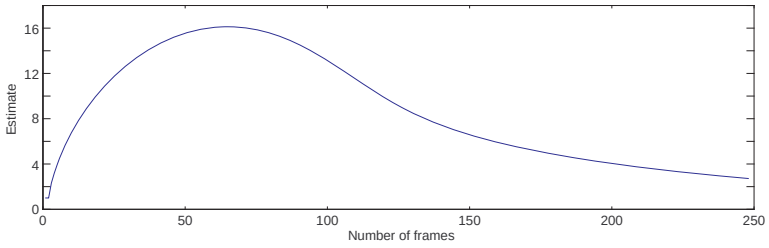
It is important to note that the convergence of the estimate proceeds the fastest in Figure 71a. The first indication of this is the maxima of the graphs: in each graph from 71a to 71e the



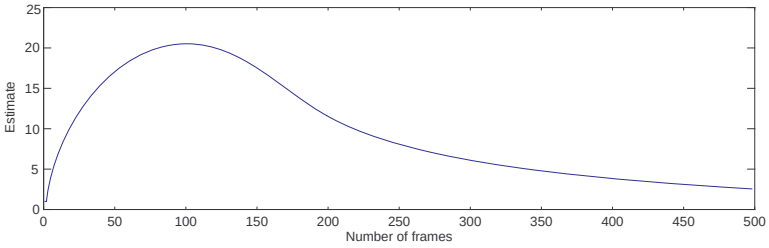
(a) Reconstruction with step size 2.5



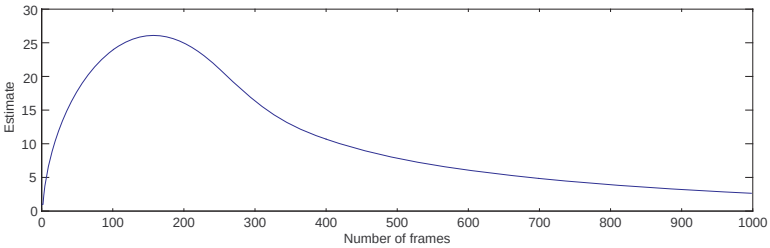
(b) Reconstruction with step size 1.25



(c) Reconstruction with step size 0.5



(d) Reconstruction with step size 0.25



(e) Reconstruction with step size 0.125

Figure 71: Convergence using different displacements

maximum is achieved at a later frame number, as early as frame 23 in 71a and as late as 160 in 71e.

A casual glance would suggest that the last graph, 71e, has achieved the most accurate overall convergence—as would be expected from the experiment that had the largest number of frames available to achieve convergence. However, a more thorough investigation reveals that each of the experiments depicted by the graphs has approximately achieved the same final estimate. Figure 72 overlays the graphs of Figure 71 into a single graph and connects the endpoints of the individual graphs to demonstrate that their final estimate—for the number of frames under consideration—is quasi-equivalent.

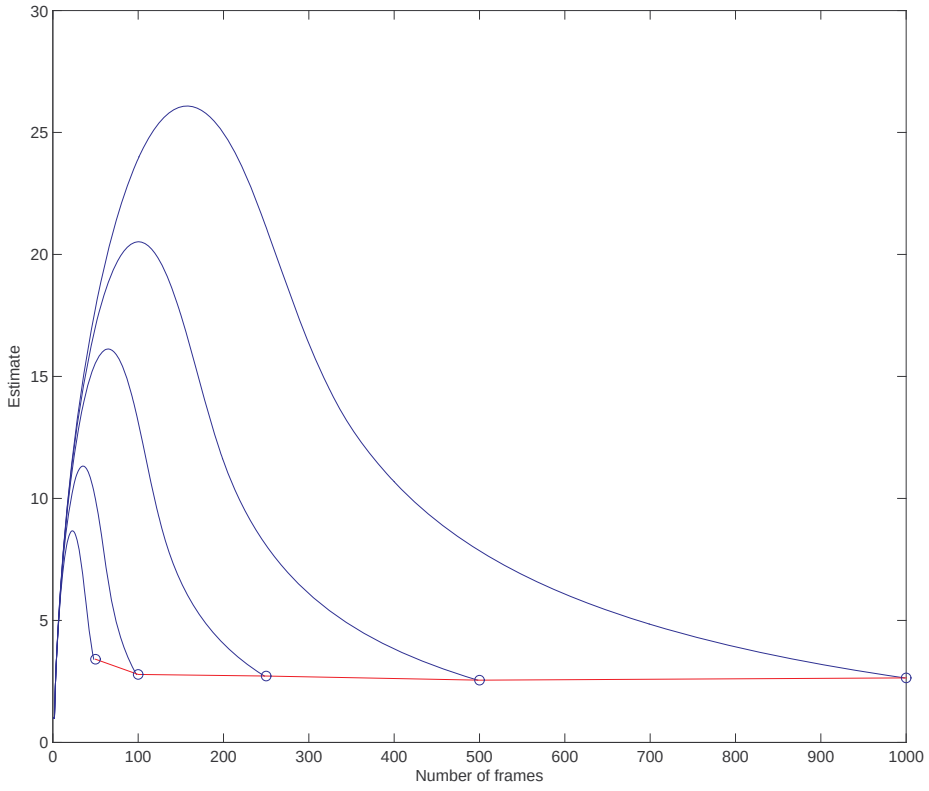


Figure 72: Near equivalent convergence

The above experiments serve to emphasize the role that inter-framerate of displacement plays in aiding the Kalman filter to achieve rapid convergence of reconstruction. To summarise, the result can be interpreted as follows: a larger number of frames provides the Kalman filter with more information with which to achieve convergence; however, a larger displacement between frames offers the Kalman filter richer information, that is to say, information that makes it easier to distinguish features and therefore accelerates the convergence of the reconstructive process.

7.4 Limitations

The reconstruction method described in this study offers good depth estimating capabilities for environment reconstruction. However, it should be noted that the reconstructor has limits in the amount of details that can be reconstructed, an inability to recover from feature loss, and only considers single viewpoint reconstruction.

7.4.1 Detail reconstruction

Although Figure 65 shows good detail in the reconstruction of the lion's head embedded in the wall, the detail in the input stream exceeds the ability of the estimator to recover depth details. This particular limitation can be overcome by increasing the resolution of reconstruction, in other words increasing the number of features tracked.

There is, however, a diminishing return: increasing the number of features significantly increases the computational time required to reconstruct a scene. Furthermore it is not possible to have more features than the number of pixels available. Finally—in the case of sparse trackers such as Kanade-Lucas-Tomasi that track the best features found—the confidence in the accuracy of the tracked features decreases as less optimal features must be chosen.

This limitation is not unique to the algorithm presented in this text, but is common to all reconstruction methods that are subjected to noise. No generic solutions exist; however, it is usually possible to increase the definition of input data and use better motion tracking to find a level of reconstruction that is deemed acceptable to the application in question.

7.4.2 Feature loss

The methods employed in the reconstruction algorithm presented in this text are designed to minimize feature loss during reconstruction. Unfortunately feature loss cannot be entirely eliminated without dynamically adapting camera motion whenever the motion estimator reports loss of feature points, or implementing a method to recognize feature loss and recover from it if features are recovered at a later stage.

The former solution cannot be employed in all applications—the latter, however, could be implemented generically. A viable option would be to make use of a particle filter-based motion tracking solution. Such solutions have been shown to be able to recover lost features in tracking applications [46, 47].

Although such a solution would be able to recover lost feature points, the method is itself subject to limitations: features that are lost must actually reappear in the input stream at a later stage for the method to be useful, and features that reappear should not be distorted beyond the ability of the motion tracker to recognise them.

(a) Initial view

(b) Distortion

Figure 73: Distortion due to loss of features

Figure 73, above, illustrates the problem of fatal feature loss. During reconstruction the camera sweeps sideways to such an extent that the entrance to the passage obscures scene elements further down the passage. Features inside the passage are lost in the process and are thus poorly reconstructed. This is particularly evident in the erratic reconstruction of the stairs in the scene.

7.4.3 Single viewpoint reconstruction

The implementation used for the algorithm described in this text settles for the reconstruction of the input scene from a single viewpoint. This implies that the reconstructed scene cannot recover structural data of scene elements that are obscured by other scene elements.

Such limits to the reconstruction are evident in, for example, the wireframe of Figure 68: the stretched triangles in the mid-left and far right of the image typically indicate the presence of scene sections that are not visible from the current viewpoint. For example, the stretched triangles in the mid-left of Figure 68 represent the continuation of the stairs-passage that is partially visible in the textured image. The solution has been outlined previously, in Section 6.2.1: reconstructions from multiple viewpoints are computed and merged into a single structure using constructive solid geometry.

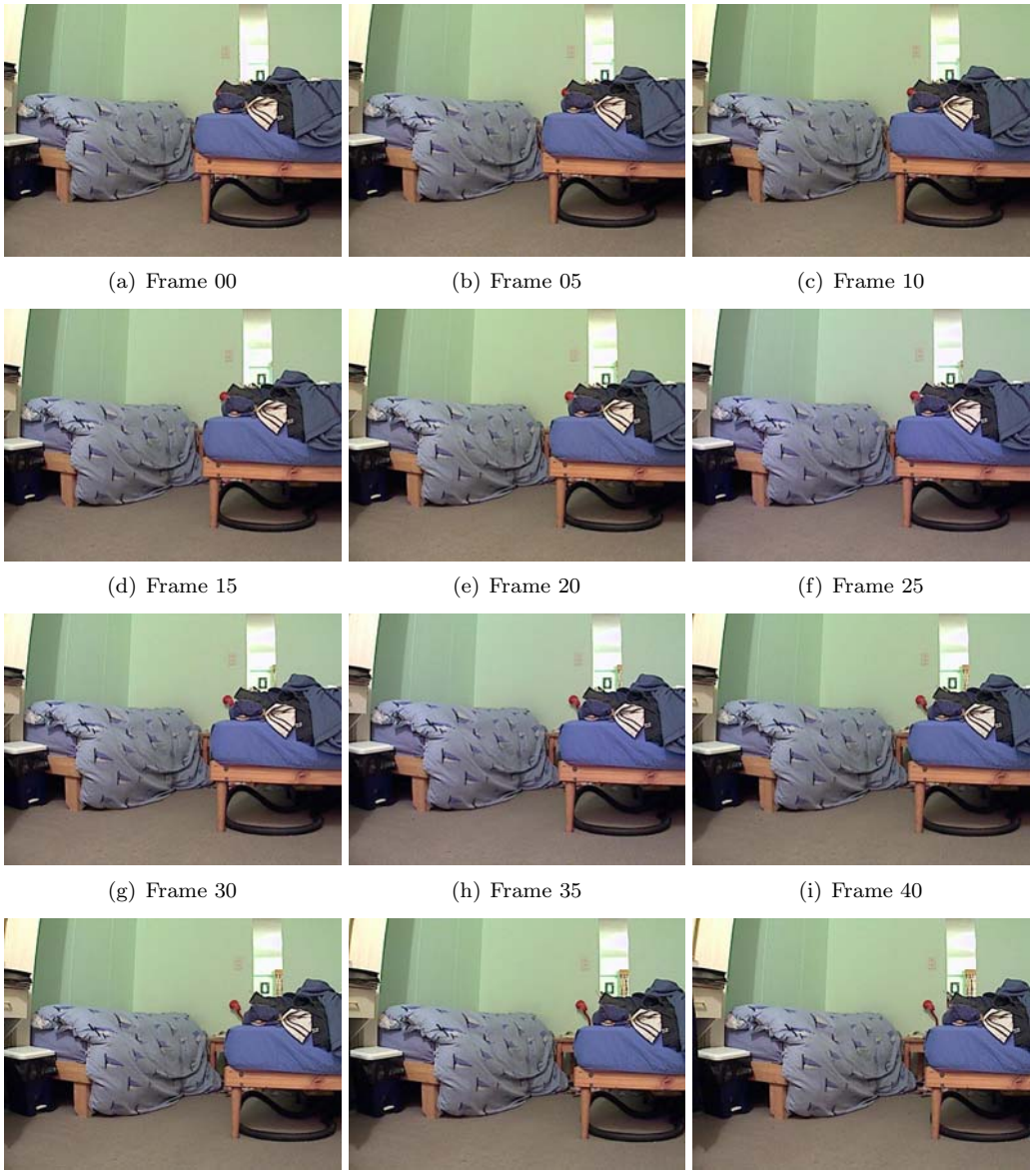
7.5 Real samples

Before concluding the results chapter we present two short demonstrations of the Kalman filter performance on real samples. The first sample features a portion of a bedroom; the video shows how the view changes as the camera slowly moves sideways. The second sample features a small Lego structure; the video shows how the structure is moved by hand.

Unlike the simulated, virtual environment examples, the motion tracking performed in the real samples makes use of the Kanade-Lucas-Tomasi tracker as implemented by the Open Computer Vision library [100]—an open-source solution to a variety of vision related algorithms. The reason for this is that the video quality in both real samples is poor, and therefore an accurate dense tracking of motion is not viable.

7.5.1 Bedroom

Figure 74 below displays frames from a short video clip captured with a web camera. The image quality of the footage is rather poor and contains large featureless sections such as the walls and floor. The image sequence contains a significant amount of noise and abrubt changes in ambient hue. Furthermore the clip contains only 56 frames.



(a) Frame 00 (b) Frame 05 (c) Frame 10

(d) Frame 15 (e) Frame 20 (f) Frame 25

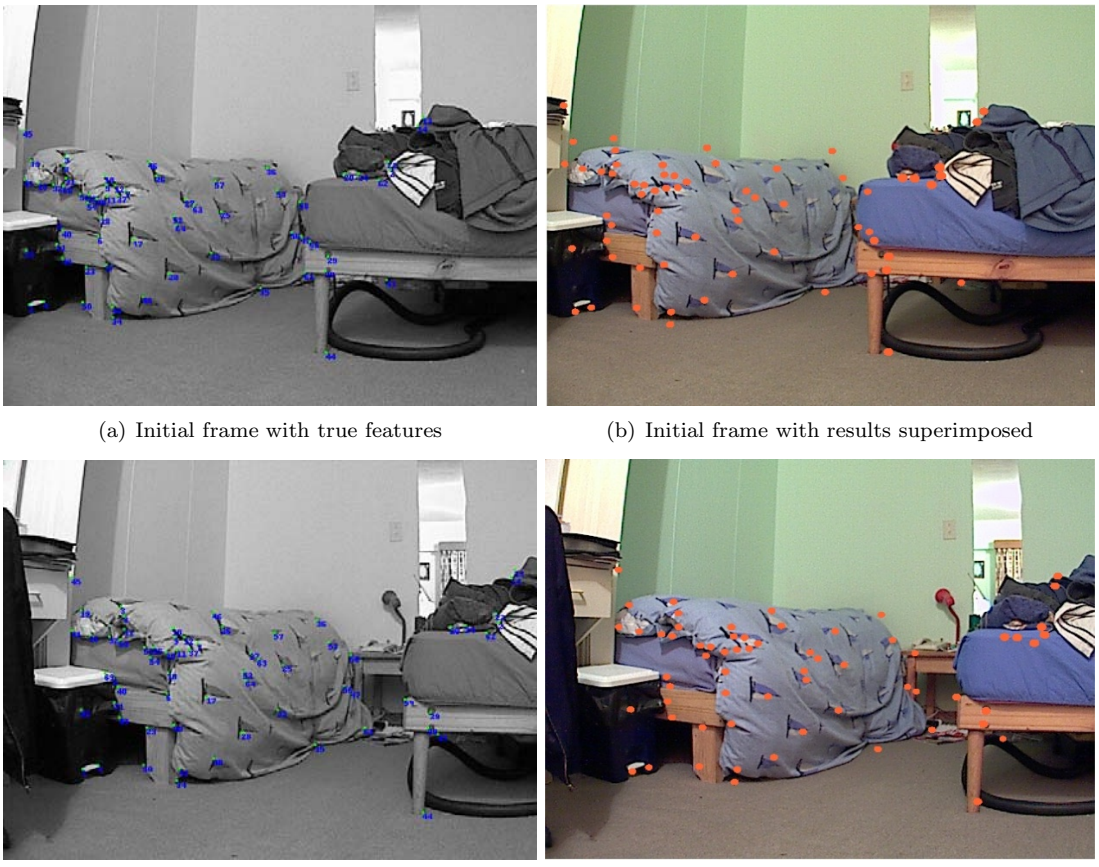
(g) Frame 30 (h) Frame 35 (i) Frame 40

(j) Frame 45 (k) Frame 50 (l) Frame 55

Figure 74: Bedroom sample

The motion data from the sample was input to the Kalman reconstructor that is used within this thesis. The resulting structural data possess little correlation and cannot be readily merged into a single structural frame. Instead each point is independent of all other points making it difficult to visualise the resulting three-dimensional reconstruction.

This makes it difficult to confirm the quality of the reconstruction using images—nonetheless Figure 75 below attempts to demonstrate the quality of the reconstruction by comparing tracked and reconstructed features in the initial and final frames. Figure 75 is intended to show the correlation of features as their position changes from the initial to the final frame. Figures 76 and 77 magnify the information shown in Figure 75 to allow easier observation of individual features.



(a) Initial frame with true features

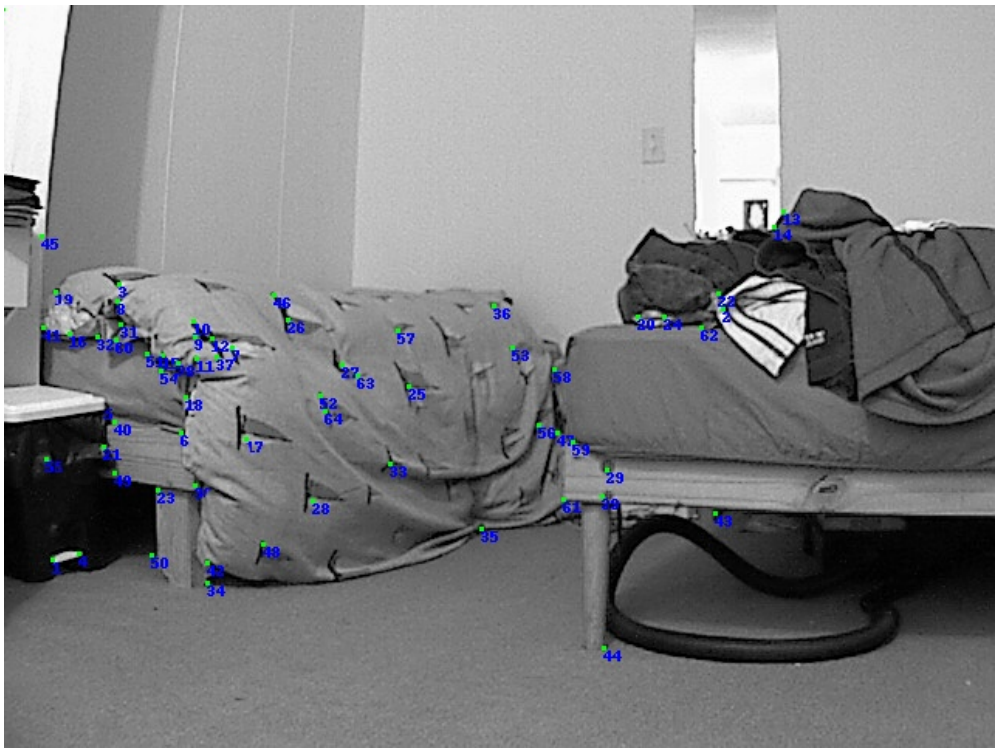
(b) Initial frame with results superimposed

(c) Final frame with true features

(d) Final frame with results superimposed

Figure 75: Comparative images of reconstruction, first and final frame

A single image pair comparing real—meaning true features—and superimposed—meaning reconstructed—data does not truly convey any meaningful data regarding the quality of the reconstruction; it is necessary to make use of at least two independent viewpoints to discern the relative displacement of tracked and reconstructed features. That is why two image sets are provided, corresponding to the initial view and the final view respectively.



(a) Initial frame with true features

(b) Initial frame with results superimposed

Figure 76: Comparative images of reconstruction, initial frame

(a) Final frame with true features

(b) Final frame with results superimposed

Figure 77: Comparative images of reconstruction, final frame

The figures demonstrate the quality of the reconstruction—the superimposed reconstructed points match the real features well and visually the result matches expectations. Comparing Figures 76 and 77, the reconstructed features accurately follow the displacement of the real features. Figure 75 shows a more compact form of the prior two figures to allow an easier appreciation of the way displaced features are matched.

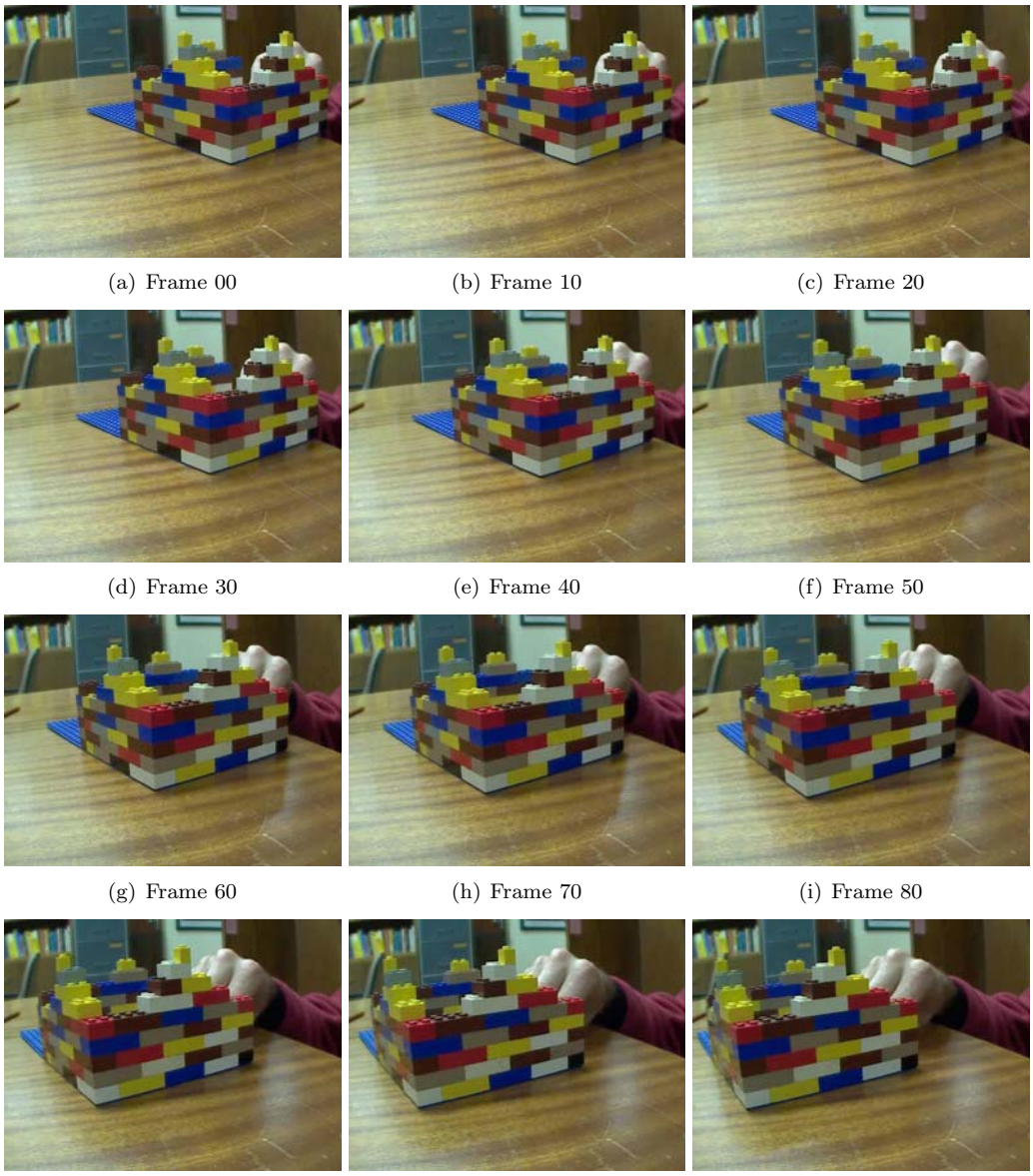
The quality of the reconstruction is noteworthy when compared to the same reconstruction using the typical observation model (Figure 78). It is immediately obvious that the typical model has only barely begun to differentiate between the depth cues of the scene, whereas the enhanced model (Figure 75) has already achieved a very good estimate of the scene.

(a) Real initial frame (b) Superimposed initial frame (typical model)

Figure 78: Comparative images of reconstruction using typical model

7.5.2 Lego

Figure 79 shows selected frames from a short image sequence depicting a Lego structure that is pushed across a table top. Similar to the bedroom sample the image quality of the Lego sample is poor. The video clip contains 111 individual frames.



(a) Frame 00 (b) Frame 10 (c) Frame 20

(d) Frame 30 (e) Frame 40 (f) Frame 50

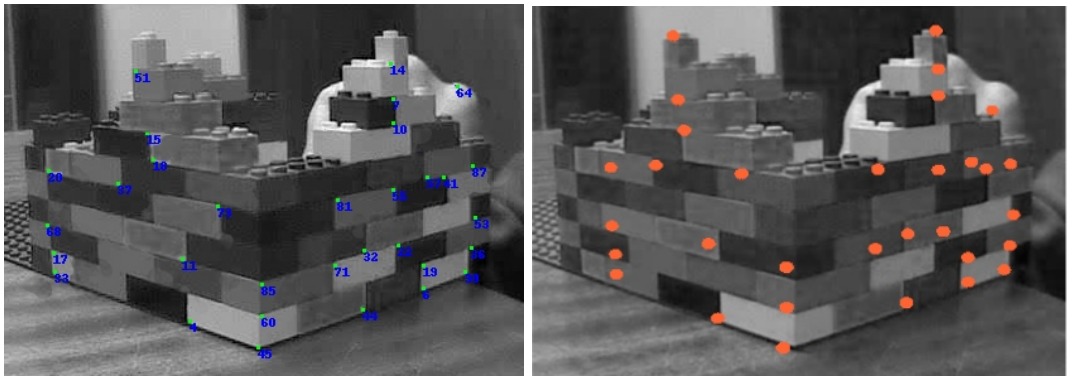
(g) Frame 60 (h) Frame 70 (i) Frame 80

(j) Frame 90 (k) Frame 100 (l) Frame 110

Figure 79: Lego sample

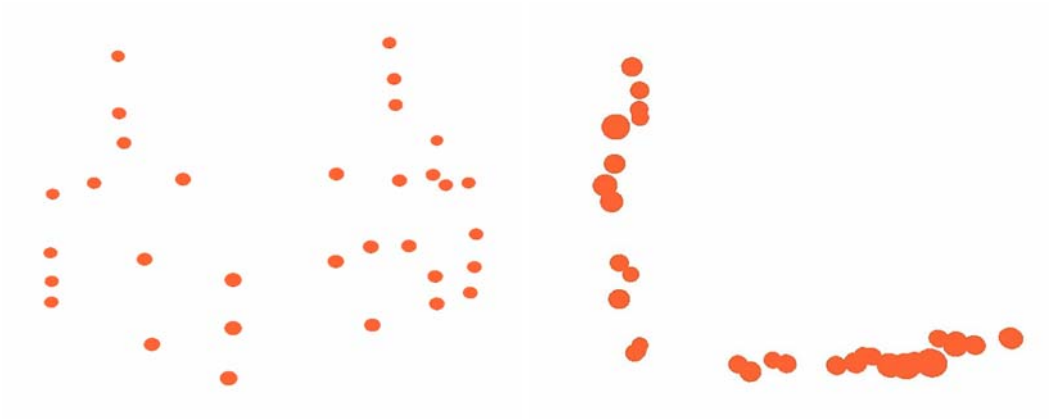
The motion data generated from the Lego sample was input to the Kalman filter used in this thesis. Unlike the bedroom sample the quality of the resulting structural information can be readily confirmed by inspection, as the resulting structure is easily identified.

Figures 80 and 81 below showcases the results obtained. The first figure demonstrates the correspondence between tracked and reconstructed features, and the second figure shows a front and top-down view of the reconstructed structure.



(a) Tracked features (b) Superimposed reconstructed features

Figure 80: Correspondence between tracked and reconstructed features



(a) Front view (b) Top-down view

Figure 81: Front and top-down views of reconstructed structure

A miracle is a portent that is not contrary to nature, but contrary to our knowledge of nature.

Augustinus

Chapter 8

Conclusion

Do not say a little in many words but a great deal in a few.

Pythagoras

This study has presented a method for the reconstruction of enclosed environments from image sequences. The method consists of a variety of elements, each of which addresses an aspect relevant to the problem of depth reconstruction from image sequences.

The first of these elements was the introduction of optimal luminance space in Chapter 2. The need for optimal luminance space follows from the fact that normal luminance computations cater to human perception rather than digital perception. This is an important point to consider as luminance space is often used in applications such as motion tracking that rely on accurate data. The chapter on optimal luminance space describes a metric for evaluating the amount of information in an image and presents an efficient heuristic for the computation of the optimal luminance space of an image. The heuristic presented was shown to retain a significantly larger amount of information than a normal luminance image and the approximation of optimal luminance was shown to be more efficient than a normal luminance computation.

In Chapter 3 the study proceeded to present a visual rendering tool based on the Quake 3 framework. This rendering tool served as the basis for the creation of scenes that are input to the reconstructor. Such an artificial environment renderer has significant advantages over real world samples: modern computational hardware and graphics algorithms are a flexible and cost-effective means for the acquisition of realistic environments. A virtual environment renderer allows the free manipulation of scene elements and camera motion—which offers substantial aid to experimentation and insight. For example, the use of a virtual environment led to the realization that feature displacement is a significant factor in the reconstruction of depth data from image sequences. It also allows the testing of the limits of the reconstruction algorithm in isolation of other issues such as tracker accuracy.

The text proceeded to describe the principles of motion estimation in Chapter 4, with specific reference to the implementation used in the study: a block-matching algorithm. This was followed by a chapter on Kalman filters, with an in-depth discussion of the unscented Kalman filter. It is shown that the type of nonlinearities encountered in this thesis are easily handled by the unscented Kalman filter.

Chapter 6 detailed the structure-from-motion framework adopted in this text. The primary problems that needed to be addressed in this study to successfully reconstruct enclosed environments were twofold: each image in the image sequence describing the environment only partially exposes the full scene that is being reconstructed, and a large data set needs to be reconstructed to create a meaningful representation of arbitrary environments.

These problems were solved by making use of a piecewise reconstruction whereby the environment is divided into vertical slivers that are reconstructed individually. The final scene is reconstructed by combining the results obtained from individual sliver reconstruction. This approach has the advantage that it addresses both the problems mentioned above; however, it introduces the new problem of ensuring compatibility between individual reconstruction elements.

This new problem was resolved by the introduction of a novel observation model. The enhanced observation model not only resolves the problem of compatibility between individual slivers, it additionally has the advantage that it is both more efficient and more accurate than the previously employed typical observation model—it achieves this result by unifying the differing camera estimates in individual sigma points into a single camera during the computation of the observation model.

Finally, empirical results were presented and analyzed. Several artificial scenes were successfully reconstructed using the methods described in this study—furthermore two real world samples were presented that demonstrated the ability of the reconstructor to cope with real world data. Additionally a series of experiments noted the significance of feature displacement to achieve convergence of the estimated structure; this is a significant insight as it moves the emphasis away from the number of frames required until convergence is achieved and instead places the focus on the displacement of features in image space.

Future work can capitalize on the insights described in this study: the solution provided is robust and can be implemented relatively cost effectively; however, the primary weakness of the structure-from-motion algorithm developed in this study is its reliance on good motion tracking data that can accurately track the motion of features as described in Section 6.2.1.

As the environment to be reconstructed is static it should be readily possible to make use of a light projection system to artificially increase the detail in scenes that have poor or no discernible features. Such an approach can be expected to yield good results in any enclosed real world environment.

A useful extension of the results offered in this study is the development of a constructive solid geometry system. Such a system can create an overall reconstruction from the depth data computed at a variety of locations, thereby allowing the reconstruction of complex and articulated structures.

In this study reconstructed data is textured using environment maps that are generated from the input provided by the virtual environment renderer. A promising avenue of research is investigating alternative texturing methods that may be able to improve on this solution.

This concludes the study of structure-from-motion for enclosed environments.

> *Begin thus from the first act, and proceed; and, in conclusion, at the ill which thou hast done, be troubled, and rejoice for the good.*
>
> Pythagoras

Appendix A: Software

The implementation of the structure-from-motion solution developed in this study consists of a number of applications. This appendix describes the programming environment and details the individual applications that were implemented. All applications were developed in Borland Delphi 2005 [16]. Three-dimensional graphics components made use of the DirectX 9 API [76].

Motion Estimation

Figure 82: Motion Estimation

The first application implemented focussed on motion tracking. The application possesses a main image and can cycle through a variety of distorted images. A feature window on the main image can be selected and the application estimates the location of the feature window on the distorted image using a block matching algorithm.

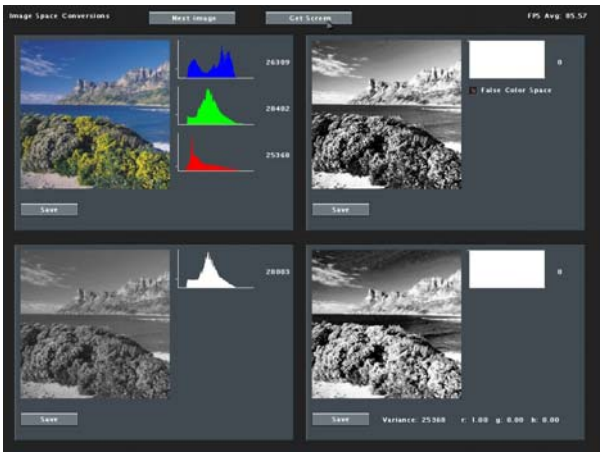


Figure 83: Image Space

Image Space

Experiments with the Motion Estimation application led to the realization that the concept of *optimal luminance space* might be worthwile exploring. The Image Space application was developed in response to this idea to investigate the properties of optimal luminance space. The code makes provision for a variety of classic and experimental image space transforms, including histogram equalization and weighted luminance computations.

Motion Data

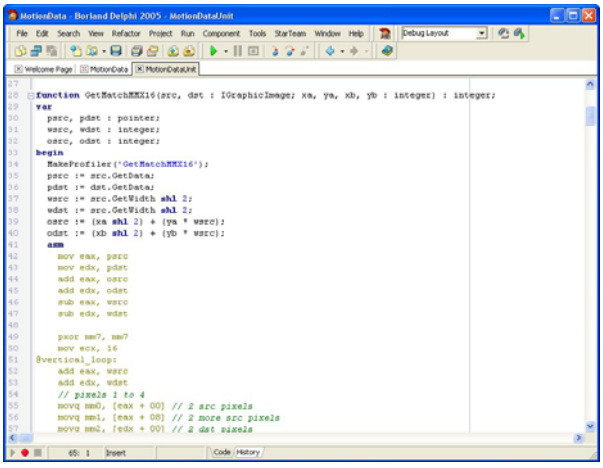


Figure 84: Motion Data

A dedicated test application was developed to create and compare different pixel matching routines. The best routine implemented makes use of MMX instructions.

Cube Remake

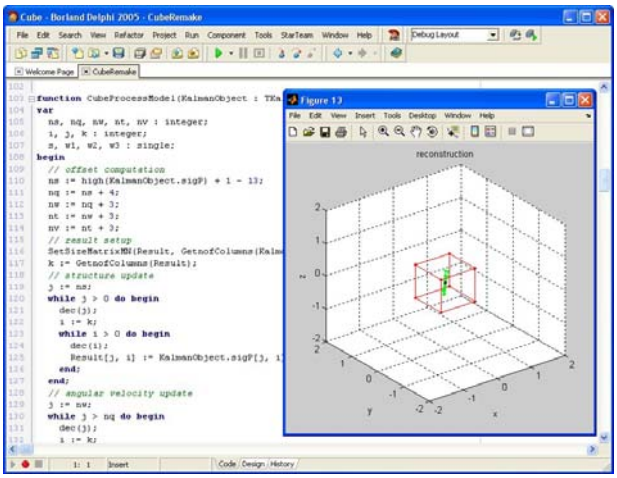


Figure 85: Cube Remake

A structure-from-motion implementation was developed for the Delphi programming language. The Cube Remake application code represents the basic Kalman filter implementation for this study.

BSP Render

Figure 86: BSP Renderer

An important step was the implementation of the Quake 3 BSP Renderer. The code generated for this application forms the visualization code for the virtual environment used in the Reconstruction application.

Reconstruction

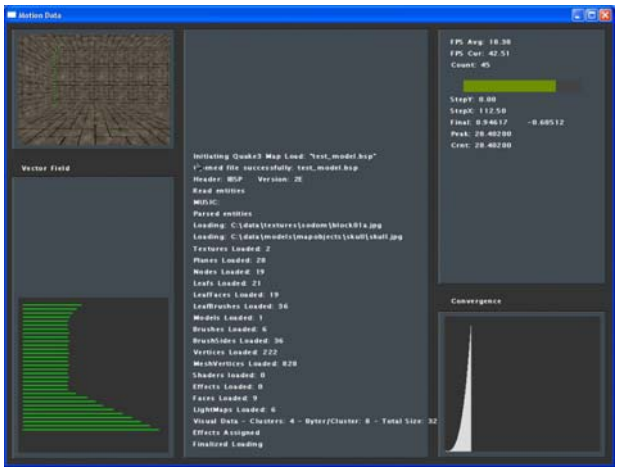


Figure 87: Reconstruction

The Reconstruction application represents the primary testing environment in this study. It includes a full implementation of the structure-from-motion system described in this thesis. It performs the following tasks: virtual environment rendering, motion tracking, and reconstruction of the structure from motion data. It is also possible to read in motion data from text files and perform structure-from-motion analysis using this data—this was used to perform the reconstruction of real samples.

Recon Viewer

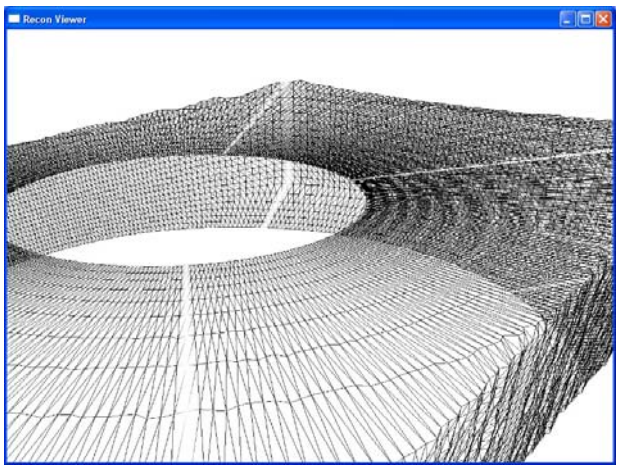


Figure 88: Recon Viewer

The Reconstruction application outputs a text file with resulting data—this data is interpreted by the Recon Viewer. The Recon Viewer generates a mesh of the reconstructed data and renders it.

Real Sphere

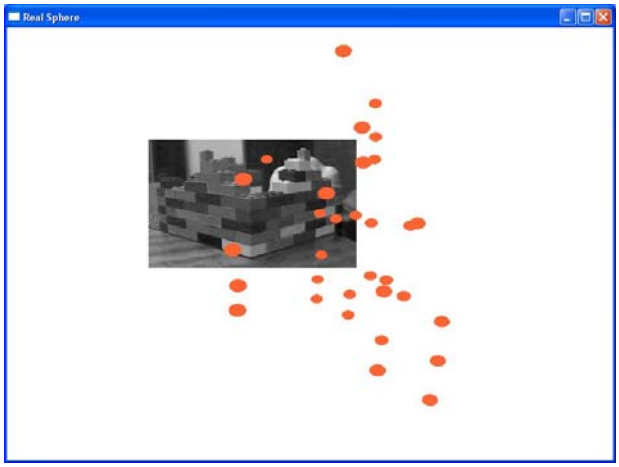


Figure 89: Real Sphere

Since the features tracked in real samples are chosen by the Kanade-Lucas-Tomasi feature tracker there is no readily available correlation between the reconstructed data points. The Real Sphere application was written to render the results of real samples; it does so by rendering orange spheres. Optionally an image can be loaded and blended into the background, this allows the manual correlation of the rendered spheres with the background image.

Bibliography

[1] 3D Studio Max. http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112.

[2] DivX. http://www.divx.com.

[3] I. Abuhadrous, F. Nashashibi, C. Laurgeau, and F. Goulette. Onboard real-time system for 3D urban environment reconstruction. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 479–483, June 2003.

[4] John Milligan Airey. *Increasing update rates in the building walkthrough system with automatic model-space subdivision and potentially visible set calculations*. PhD thesis, University of North Carolina, 1990.

[5] Luis Alvarez, Joachim Weickert, and Javier Sanchez. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision*, 39(1):41–56, 2000.

[6] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[7] Electronic Arts. Battlefield. http://www.battlefield1942.com/.

[8] Ali Azarbayejani and Alex Pentland. Recursive estimation of motion, structure and focal length. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 17(6), pages 562–575. IEEE, June 1995.

[9] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 434–441, 1998.

[10] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. Technical Report 299, Department of Computer Science, University of Western Ontario, July 1992.

[11] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[12] beepa. Fraps. http://www.fraps.com/.

[13] Dinkar N. Bhat and Shree K. Nayar. Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):415–423, April 1998.

[14] J. Bittner, V. Havran, and P. Slavík. Hierarchical visibility culling with occlusion trees. In *Proceedings of Computer Graphics International '98 (CGI'98)*, pages 207–219. IEEE, June 1998.

[15] Shawn J. Bohn and Erin N. Thornton. Environment reconstruction for robot navigation. In *Proceedings of SPIE*, volume 2217, pages 96–106, July 1994.

[16] Borland. Delphi. `http://www.borland.com/us/products/ide.html`.

[17] Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8–15, Santa Barbara, California, USA, 1998.

[18] Ted Camus. Real-time quantized optical flow. In *IEEE Conference on Computer Architectures for Machine Perception*, Como, Italy, 1995.

[19] Edwin Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, 1974.

[20] K.M. Cheung, S. Baker, and T. Kanade. Shape-From-Silhouette Across Time Part I: Theory and Algorithms. In *International Journal of Computer Vision*, volume 62, pages 221–247, May 2005.

[21] Norman Chin and Steven Feiner. Near real-time shadow generation using bsp trees. In *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 99–106, New York City, New York, USA, 1989. ACM Press.

[22] Alessandro Chiuso, Paolo Favaro, Hailin Jin, and Stefano Soatto. Structure from motion causally integrated over time. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24(4). IEEE, April 2002.

[23] R. Robert Clark, Michael H. Lin, and Colin J. Taylor. 3D environment capture from monocular video and inertial data. In *Proceedings of the SPIE on Three-Dimensional Image Capture and Applications VII*, volume 6056, pages 150–161, February 2006.

[24] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 1071–1076, Boston, Massachusetts, USA, June 1995. IEEE.

[25] Bob Crispen. VRML works. `http://vrmlworks.crispen.org/`.

[26] Charles Robert Darwin. *On the Origin of Species by Means of Natural Selection.* John Murray, London, 1859.

[27] Francis Decroos, Peter Schelkens, Freddy Stiens, Jan Cornelis, and Vassilios A. Christopoulos. Motion monitoring and classification with center-biased motion estimation. In *IEEE International Conference on Image Processing*, volume 1, pages 872–875, Vancouver, Canada, 2000.

[28] G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csobra. A solution to the simultaneous localisation and map building (SLAM) problem. In *IEEE Transactions on Robotics and Automation*, volume 17(3), pages 229–241, 2001.

[29] G. Dissanayake, S. B. Williams, H. F. Durrant-Whyte, and T. Bailey. Map management for efficient simultaneous localisation and map building (SLAM) problem. In *Autonomous Robots*, volume 12, pages 267–286, 2002.

[30] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. In *Statistics and Computing*, volume 10(3), pages 197–208, 2000.

[31] Eclipse Entertainment. Genisis3D. `http://www.genesis3d.com/`.

[32] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 124–133, New York City, New York, USA, 1980. ACM Press.

[33] Andrea Fusiello, Vito Roberto, and Emanuele Trucco. Symmetric stereo with multiple windowing. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(8):1053–1066, 2000.

[34] Dariu M. Gavrila and Larry S. Davis. Tracking of humans in action: a 3D model-based approach. In *ARPA Image Understanding Workshop*, pages 737–746, Palm Springs, California, USA, February 1996.

[35] Nikolaus Gebhardt. Irrlicht Engine. `http://irrlicht.sourceforge.net/`.

[36] Arthur Gelb, editor. *Applied Optimal Estimation*. M.I.T. Press, 1986.

[37] Donald B. Gennery. Visual tracking of known three-dimensional objects. In *International Journal of Computer Vision*, volume 7(3), pages 243–270, 1992.

[38] F. Glazer, G. Reynolds, and P. Anandan. Scene matching by hierarchical correlation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 432–441, Annapolis, 1983.

[39] Craig Gotsman, Oded Sudarsky, and Jeffrey A. Fayman. Optimized occlusion culling using five-dimensional subdivision. *Computers & Graphics*, 23(5):645–654, 1999.

[40] John Guild. The colorimetric properties of the spectrum. In *Philosophical Transactions of the Royal Society of London*, volume A230, pages 149–187, 1931.

[41] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–152, Manchester University, United Kingdom, August 1988. The University of Sheffield Printing Unit.

[42] Eugene Hecht. *Optics*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 2nd edition, 1987.

[43] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[44] Carine Hue, Jean-Pierre Le Cadre, and Patrick Perez. A particle filter to track multiple objects. In *IEEE Workshop on Multi-Object Tracking*, pages 61–68, Vancouver, Canada, July 2001. IEEE.

[45] id Software. Quake 3: Arena – GNU GPL, August 2005. News: `http://linuxgames.com/news/feedback.php?identiferID=7750&action=flatview`, Source: `ftp://ftp.idsoftware.com/idstuff/source/quake3-1.32b-source.zip`.

[46] Matthew Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 343–356, Cambridge, United Kingdom, 1996.

[47] Matthew Isard and Andrew Blake. CONDENSATION – conditional density propation for visual tracking. In *International Journal of Computer Vision*, volume 29(1), pages 5–28, August 1998.

[48] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29:1799–1808, December 1981.

[49] Kirk M. Joubert. 3D model reconstruction using photoconsistency. MSc Thesis, University of Cape Town, 2007.

[50] Simon J. Julier. A skewed approach to filtering. In *AeroSense: The 12th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, Orlando, Florida, USA, 1998. SPIE. Signal and Data Processing of Small Targets.

[51] Simon J. Julier. The scaled unscented transformation. Technical report, IDAK Industries, 901 Missouri Boulevard, 179 Jefferson City, MO 65109, 1999.

[52] Simon J. Julier and Jeffrey K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, University of Oxford, Department of Engineering Science, Robotics Research Group, Oxford, OX1 3PJ United Kingdom, November 1996.

[53] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, Orlando, Florida, USA, 1997. SPIE.

[54] Simon J. Julier and Jeffrey K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In *Proceedings of the IEEE American Control Conference*, pages 887–892, Anchorage, Alaska, USA, 8–10 May 2002. IEEE.

[55] Simon J. Julier and Jeffrey K. Uhlmann. The scaled unscented transformation. In *Proceedings of the IEEE American Control Conference*, pages 4555–4559, Anchorage, Alaska, USA, 8–10 May 2002. IEEE.

[56] Simon J. Julier, Jeffrey K. Uhlmann, and Hugh F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the 1995 American Control Conference*, pages 1628–1632, Seattle, Washington, USA, June 1995.

[57] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[58] Rudolph Emil Kalman and Richard Snowden Bucy. New results in linear filtering and prediction theory. *Transactions of ASME, Journal of Basic Engineering*, 83(D):95–108, 1961.

[59] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.

[60] Kenichi Kanatani. Motion segmentation by subspace separation and model selection. In *IEEE International Conference on Computer Vision*, volume 2, pages 586–591. IEEE, 2001.

[61] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion-compensated interframe coding for video conferencing. In *IEEE National Telecommunication Conference*, pages G5.3.1–G5.3.5, New Orleans, Louisiana, USA, November 1981.

[62] Vladlen Koltun, Yiorgos Chrysanthou, and Daniel Cohen-Or. Virtual occluders: an efficient intermediate PVS representation. In Bernard Peroche and Holly E. Rushmeier, editors, *Rendering Techniques 2000: Proc. 11th Eurographics Workshop Rendering*, pages 59–70. Springer, 2000.

[63] Vladlen Koltun, Yiorgos Chrysanthou, and Daniel Cohen-Or. Hardware-accelerated from-region visibility using a dual ray space. In Steven J. Gortler and Karol Myszkowski, editors, *Rendering Techniques 2001: Proc. 12th Eurographics Workshop Rendering*, pages 205–216. Springer, 2001.

[64] Steen Kristensen and Henrik Iskov Christensen. Continuous reconstruction of scene objects. In *Proceedings of SPIE '93 Sensor Fusion VI*, pages 272–281, Boston, USA, 1993.

[65] Jean Baptiste Lamarck. Philosophie Zoologique. 1809.

[66] Nobel Lectures. *Physics 1922 – 1941*. Elsevier Publishing Company, Amsterdam, 1965.

[67] R. Li, B. Zeng, and M. L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4:438–442, August 1994.

[68] Hongche Liu, Tsai-Hong Hong, Martin Herman, Ted Camus, and Rama Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding*, 72(3):271–286, 1998.

[69] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[70] David Luebke and Chris Georges. Portals and mirrors: simple, fast evaluation of potentially visible sets. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 105–106. ACM SIGGRAPH, April 1995.

[71] Dana Mackenzie. Ensemble Kalman filters bring weather models up to date. *SIAM*, 36(8), October 2003.

[72] D. F. Malan, W. H. Steyn, and B. M. Herbst. Remote satellite position and pose estimation using monocular vision. In *Proceedings of the 5th International Symposium of the Internatinal Academy of Astronautics*, pages 161–168, Berlin.

[73] J. Malo, F.J. Ferri, J. Albert, and J.M. Artigas. Splitting criterion for hierarchical motion estimation based on perceptual coding. *Electronics Letters*, 34(6):541–543, 1998.

[74] Keith McLaren. The development of the CIE 1976 (L*a*b*) uniform colour-space and colour-difference formula. *Journal of the Society of Dyers and Colourists 92*, pages 338–341, 1976.

[75] Francois G. Meyer, R. Todd Constable, Albert J. Sinusas, and James S. Duncan. Dense nonrigid motion tracking from a sequence of velocity field. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–844. IEEE, 1996.

[76] Microsoft. Directx 9.0c. `http://msdn.microsoft.com/directx/`.

[77] Amar Mitiche and Patrick Bouthemy. Computation and analysis of image motion: a synopsis of current problems and methods. *Int. Journal of Computer Vision*, 19(1):29–55, July 1996.

[78] Toshihiko Morita and Takeo Kanade. A sequential factorization method for recovering shape and motion from image streams. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19(8), pages 858–867. IEEE, 1997.

[79] Rama Chellappa Namrata Vaswani. A particle filtering approach to abnormality detection in nonlinear systems and its application to abnormal activity detection. In *3rd International Workshop on Statististical and Computational Theories of Vision*, 2003.

[80] Shree K. Nayar. Shape from focus. Technical Report CMU-RI-TR-89-27, Robotics Institute, Carnegie Mellon University, November 1989.

[81] Sergiu Nedevschi, Silviu Bota, Tiberiu Marita, Florin Oniga, and Ciprian Pocol. Real-Time 3D Environment Reconstruction Using High Precision Trinocular Stereovision. In *IEEE-TTTC International Conference on Automation, Quality & Testing, Robotics AQTR 2006*, pages 25–28, Cluj-Napoca, Romania, May 2006.

[82] John Oliensis. A critique of structure-from-motion algorithms. In *Computer Vision and Image Understanding*, volume 80(2), pages 172–214. CVIU, 2000.

[83] M. Oren and Shree K. Nayar. Generalization of the Lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, April 1995.

[84] S. Panzieri, F. Pascucci, and R. Setola. Interlaced extended kalman filter for real time navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2780–2785, August 2005.

[85] Andrew Mark Peacock. *Information fusion for improved motion estimation*. PhD thesis, University of Edinburgh, May 2001.

[86] L.-M. Po and W.-C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:313–317, June 1996.

[87] Charles Poynton. A guided tour to color space. In *Proceedings of the SMPTE Advanced Television and Electronic Imaging Conference*, pages 167–180, San Francisco, USA, February 1995.

[88] Gang Qian, Rama Chellappa, and Qinfen Zheng. Robust structure from motion estimation using inertial data. In *Journal of the Optical Society of America A*, volume 18(12), December 2001.

[89] David Radd. What hollywood could learn from the gaming industry, May 2006. `http://biz.gamedaily.com/industry/adwatch/?id=12877`.

[90] Pieter Albertus Rautenbach. Facial feature reconstruction using structure from motion. Master's thesis, University of Stellenbosch, South Africa, December 2004.

[91] James M. Rehg and Takeo Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In J. Eklundh, editor, *Proceedings of Third European Conference on Computer Vision*, volume 2, pages 35–46, Stockholm, Sweden, 1994. Springer-Verlag.

[92] Ioannis M. Rekleitis. *Cooperative Localization and Multi-Robot Exploration*. PhD thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, February 2003.

[93] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House, 2004.

[94] Carl Scheffler, Konrad Scheffler, and Christian W. Omlin. Articulated tree structure from motion – a matrix factorization approach. In *14th Annual Symposium of the Pattern Recognition Association of South Africa*, Langebaan, South Africa, 2003.

[95] Stanley F. Schmidt. Applications of state space methods to navigation problems. In C. T. Leondes, editor, *Advances in Control Systems*, pages 293–340, New York City, New York, USA, 1966. Academic Press.

[96] Vassilis E. Seferidis and Mohammad Ghanbari. General approach to block-matching motion estimation. *Optical Engineering*, 32(7):1464–1474, 1993.

[97] Vassilis E. Seferidis and Mohammad Ghanbari. Generalised block-matching motion estimation using quad-tree structured spatial decomposition. *IEE Proceedings - Vision, Image, and Signal Processing*, 141(6):446–452, 1994.

[98] Barry Sherlock. The unscented Kalman filter. Private communication. University of North Carolina.

[99] Jianbo Shi and Carlo Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, 1994.

[100] Open source. OpenCV. `http://sourceforge.net/projects/opencvlibrary`.

[101] A. James Stewart. Hierarchical visibility in terrains. In Julie Dorsey and Philipp Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 217–228, New York City, New York, 1997. Springer Wien.

[102] Wolfgang Straßer. *Schnelle Kurven- und Flächendarstellung auf graphischen Sichtgeräten*. PhD thesis, Technische Universität Berlin, 1974.

[103] Steve Streeting. OGRE. `http://www.ogre3d.org/`.

[104] Peter Swerling. First order propagation in a stagewise smoothing procedure. *Annals of Mathematical Statistics*, 29(2):585–588, 1958.

[105] Lassi Tasajärvi, Bent Stamnes, and Mikael Schustin. *Demoscene: the Art of Real-Time*. Even Lake Studios, 2004.

[106] Leonid Taycher, John W. Fisher III, and Trevor Darrell. Recovering articulated model topology from observed motion. In *Advances in Neural Information Processing Systems*, pages 1311–1318, 2002.

[107] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):61–69, July 1991.

[108] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[109] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. In *International Journal of Computer Vision*, volume 9(2), pages 137–154, November 1992.

[110] Shimon Ullman. Maximizing rigidity: The incremental recovery of 3-d structure from rigid and nonrigid motion. In *Perception*, volume 13, pages 255–274, 1984.

[111] S. Uras, F. Girosi, A. Verri, and V. Terre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–87, 1988.

[112] Valve. Half Life 2. `http://half-life2.com/`.

[113] Valve. Source SDK. `http://msdn.microsoft.com/coding4fun/half-life/`.

[114] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The unscented particle filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, May 2000. Available: `http://cslu.cse.ogi.edu/publications/ps/merwe00.pdf`.

[115] Rudolph van der Merwe and Eric Wan. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC)*.

[116] Rudolph van der Merwe and Eric Wan. *Kalman Filtering and Neural Networks*, chapter 7 - The Unscented Kalman Filter. Wiley, 2001.

[117] Rudolph van der Merwe and Eric Wan. Sigma-point kalman filters for probabilistic inference in dynamic state-space models. In *Workshop on Advances in Machine Learning*, Montreal, Canada, June 2003.

[118] Barry-Michael van Wyk. Verifying stereo vision using structure from motion. MSc Thesis, Applied Mathematics, University of Stellenbosch, 2007.

[119] Chris Venter and Ben M. Herbst. Structure from motion estimation using a non-linear kalman filter. In *Proceedings of the 12th Annual Symposium of the Pattern Recognition Association of South Africa*, pages 65–70, Franschhoek, 2001.

[120] B. Wade. The BSP Tree FAQ, 2001. `ftp://ftp.sgi.com/other/bspfaq/faq/bspfaq.html`.

[121] Dirk Wolfram Wagener. Feature tracking and pattern registration. Master's thesis, University of Stellenbosch, South Africa, November 2003.

[122] Alfred Russel Wallace. On the tendency of varieties to depart indefinitely from the original type. 1858.

[123] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Intelligent Robots and Systems*, pages 3834–3839, August 2005.

[124] Eric W. Weisstein. Quaternion, 2007. From Mathworld—A Wolfram Web Resource. `http://biz.gamedaily.com/industry/adwatch/?id=12877`.

[125] W. David Wright. A re-determination of the trichromatic coefficients of the spectral colors. In *Transactions of the Optical Society London*, volume 30, pages 141–164, 1928.

[126] Annie Yao and Andrew Calway. Uncalibrated narrow baseline augmented reality. In G. M. Cortelazzo and C. Guerra, editors, *First International Symposium on 3D Data Processing Visualization and Transmission*, pages 182–185. IEEE, June 2002.

[127] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. Technical report, School of Computer Science, University of Central Florida, Orlando, FL 32816.