LLNL-TH-515171

LAWRENCE LIVERMORE NATIONAL LABORATORY

Error Detection, Factorization and Correction for Multi-View Scene Reconstruction from Aerial Imagery

M. Hess-Flores

November 21, 2011

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Mauricio Hess Flores
December 2011
Electrical and Computer Engineering

Error Detection, Factorization and Correction for Multi-View Scene Reconstruction from Aerial Imagery

<u>Abstract</u>

Scene reconstruction from video sequences has become a prominent computer vision research area in recent years, due to its large number of applications in fields such as security, robotics and virtual reality. Despite recent progress in this field, there are still a number of issues that manifest as incomplete, incorrect or computationally-expensive reconstructions. The engine behind achieving reconstruction is the matching of features between images, where common conditions such as occlusions, lighting changes and texture-less regions can all affect matching accuracy. Subsequent processes that rely on matching accuracy, such as camera parameter estimation, structure computation and non-linear parameter optimization, are also vulnerable to additional sources of error, such as degeneracies and mathematical instability. Detection and correction of errors, along with robustness in parameter solvers, are a must in order to achieve a very accurate final scene reconstruction. However, error detection is in general difficult due to the lack of ground-truth information about the given scene, such as the absolute position of scene points or GPS/IMU coordinates for the camera(s) viewing the scene.

In this dissertation, methods are presented for the detection, factorization and correction of error sources present in all stages of a scene reconstruction pipeline from video, in the absence of ground-truth knowledge. Two main applications are discussed. The first set of algorithms derive total structural error measurements after an initial scene structure computation and factorize errors into those related to the underlying feature matching process and those related to camera parameter estimation. A brute-force local correction of inaccurate feature matches is presented, as well as an improved conditioning scheme for non-linear parameter optimization which applies weights on input parameters in proportion to estimated camera parameter errors. Another application is in reconstruction pre-processing, where an algorithm detects and discards frames that would lead to inaccurate feature matching, camera pose estimation degeneracies or mathematical instability in structure computation based on a residual error comparison between two different match motion models. The presented algorithms were designed for aerial video but have been proven to work across different scene types and camera motions, and for both real and synthetic scenes.

Kenneth I. Joy, PhD, Chair
Dissertation Committee Chair

Error Detection, Factorization and Correction for Multi-View Scene Reconstruction from Aerial Imagery

By

MAURICIO HESS FLORES
B.S. (Universidad de Costa Rica) 2002
M.S. (Universidad de Costa Rica) 2004

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Kenneth I. Joy, PhD, Chair

Mark A. Duchaineau, PhD

Owen T. Carmichael, PhD

Committee in charge

2011

Error Detection, Factorization and Correction for Multi-View Scene Reconstruction from Aerial Imagery

I dedicate this document and all of my efforts to grow personally and professionally to my family, who have always supported me unconditionally.

Acknowledgments

First, I would like to give my most sincere thanks to Ken Joy and Mark Duchaineau for all their help, time, valuable discussions and for giving me the opportunity to work on this relevant, modern and challenging project, both at UCDavis and at LLNL. Many thanks to the Lawrence Scholar Program at LLNL, for financing my studies since 2009. To Daniel Knoblauch, for the great teamwork through all the summers we worked together and the papers that resulted from our collaboration. Also to Peter Lindstrom at LLNL for his valuable help, ideas and code contribution, and to Owen Carmichael and Michael Goldman for all their helpful suggestions and discussions.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Contents

List of Figures

List of Tables

Abstract

Scene reconstruction from video sequences has become a prominent computer vision research area in recent years, due to its large number of applications in fields such as security, robotics and virtual reality. Despite recent progress in this field, there are still a number of issues that manifest as incomplete, incorrect or computationally-expensive reconstructions. The engine behind achieving reconstruction is the matching of features between images, where common conditions such as occlusions, lighting changes and texture-less regions can all affect matching accuracy. Subsequent processes that rely on matching accuracy, such as camera parameter estimation, structure computation and non-linear parameter optimization, are also vulnerable to additional sources of error, such as degeneracies and mathematical instability. Detection and correction of errors, along with robustness in parameter solvers, are a must in order to achieve a very accurate final scene reconstruction. However, error detection is in general difficult due to the lack of ground-truth information about the given scene, such as the absolute position of scene points or GPS/IMU coordinates for the camera(s) viewing the scene.

In this dissertation, methods are presented for the detection, factorization and correction of error sources present in all stages of a scene reconstruction pipeline from video, in the absence of ground-truth knowledge. Two main applications are discussed. The first set of algorithms derive total structural error measurements after an initial scene structure computation and factorize errors into those related to the underlying feature matching process and those related to camera parameter estimation. A brute-force local correction of inaccurate feature matches is presented, as well as an improved conditioning scheme for non-linear parameter optimization which applies weights on input parameters in proportion to estimated camera parameter errors. Another application is in reconstruction pre-processing, where an algorithm detects and discards frames that would lead to inaccurate feature matching, camera pose estimation degeneracies or mathematical instability in structure computation based on a residual error comparison between two different match motion models. The presented algorithms were designed for aerial video but have been proven to work across different scene types and camera motions, and for both real and synthetic scenes.

Kenneth I. Joy, PhD, Chair
Dissertation Committee Chair

Chapter 1

Introduction

The field of computer vision has been steadily expanding during the past years due to its many applications in components of our modern lifestyles, such as in medicine, autonomous navigation, robotics, virtual reality, security and many more. It is a large area of research, which involves concepts from many different disciplines such as mathematics, computer science and even the psychology of perception.

One important research area in computer vision is *multi-view scene reconstruction*, which during the past years has seen an increase in applications involving for example security, industrial design, virtual environment creation and enhancement, mobile robotics and reconstruction of cultural heritage sites. Specifically, the work on multi-view reconstruction from aerial imagery presented in this dissertation is part of a larger project known as the *VidCharts* project, which is being carried out at Lawrence Livermore National Laboratory (LLNL). This project devises advanced algorithms for large-scale video processing, organization and interaction. Dr. Mark Duchaineau of the Computation Directorate serves as the Principal Investigator, and the work in this dissertation is part of an on-going collaboration between LLNL and the Institute for Data Analysis and Visualization (IDAV) at UCDavis, under Prof. Kenneth Joy.

The particular application currently under development consists of the scenario where an aircraft is flying around urban environments, carrying an array of sensors which collects images at high frame rates as the aircraft circles around the scene over and over again, such that image data collection is massive. Once enough data has been collected, a semantic analysis of the scene becomes possible, and activity happening in the scene can be inferred. Two main stages are involved in this process. In the initial state, basically no information at all is known about either the cameras acquiring imagery, for example their positions and orientations, or the scene being viewed, for example its three-dimensional structure. This implies that all computational resources and processing should be directed towards gathering information necessary to apply algorithms which allow for the camera poses and structure to be initially obtained. In the steady state, after enough information has been collected and a certain accuracy has been achieved as far as the camera positions, orientations and the computed structure, less and less resources would be needed for this purpose and could be used on other tasks which would further help infer information about the scene. For example, if the reconstruction is accurate enough, it becomes possible to estimate what the next image should look like at the pixel level, which allows for data compression. With enough scene information, elements such as lighting conditions or the texture properties of the buildings could be inferred. Additionally, moving objects with respect to the static scene such as cars and people could be identified and tracked. At a certain point, given large amounts of information, even higher levels of semantic scene understanding could be achieved. As a particular example, if two cars where to collide it would be possible to detect this, given that all static objects and buildings in the scene would be known and all moving objects would have been identified and tracked.

To achieve such a level of scene understanding, it is clear to see that an immense amount of data must be collected and processed, and at least part of the computation must be done on-board, in real-time, where limitations exist on the size, weight and power of the computational resources that can be carried aboard the aircraft. To deal with these challenges, VidCharts project members at LLNL have been tackling issues such as pre-processing, obtaining accurate image correspondences, large-format video transforms and encoding, motion detection and tracking, and 3D scene reconstruction. Pre-processing of images is a necessary first step. Since images are acquired from separate cameras, which even having the same specifications have slight mechanical variations, makes it necessary to undergo several pre-processing steps before forming one coherent large-scale image, which can be used as input to processes such as camera pose estimation and scene reconstruction. The array of images must be stitched together to form a 'mosaic', assuring that effects such as vignetting, which is reduction in the brightness or saturation of an image near the edges with respect to the center, have been corrected. Resolution enhancements, known as super-resolution, can also be applied to the resulting images. The underlying engine behind all of these operations is to obtain sub-pixel accurate 'dense image correspondences'. For example, the registration of successive frames in a video sequence into a common coordinate system at the pixel level in itself is what enables applications such as dense 3D reconstruction of the viewed scene, segmentation of background and moving objects and data compression. Duchaineau's algorithm [15] provides an accurate and real-time implementation to perform the dense correspondence process, and is the main component of the VidCharts effort.

Given the context under which scene reconstruction fits into the greater scheme of the VidCharts project, the rest of this Introduction will describe the steps necessary to achieve scene reconstruction and previous work from the literature on the subject, followed by descriptions of novel algorithms that were developed to deal with the shortcomings of algorithms from the literature. These algorithms will be described in detail in the remaining chapters of this document.

Though there are variations in the literature, the principal and most common steps in scene reconstruction are camera calibration, feature extraction and matching, epipolar geometry estimation, pose estimation, structure estimation and bundle adjustment, and each of these concepts will now be described. Even though the ultimate goal is to reconstruct from large arrays of cameras, the work in this document will deal with the specific case of one camera viewing the scene.

Camera calibration refers to the process of recovering the camera's intrinsic parameters. These include the focal length, the principal point, which corresponds to the image plane intersection of the camera's principal ray, and pixel skew, a measure of pixel orientation. All of these parameters are described in detail in Appendix A. Usually, some information about these parameters is known before-hand from manufacturer specification sheets, for example the focal length. If the information is unknown, it must be obtained through a process known as self-calibration [30]. For the particular case of this dissertation, it was always assumed that reasonably accurate initial values were available, and later fine-tuned through a posterior bundle adjustment.

Feature matching involves recovering the pixel coordinates of image features that correspond to the same scene point in a set of images. Sparse feature matching is used for sparse features such as corners or scale-invariant features. The work of Shi-Tomasi [80] and Tomasi-Kanade [92] in the early 90's was pivotal as far as detecting good point matches and tracking them throughout a video sequence. Currently, the most commonly-used feature matching algorithms are the 'Scale-Invariant Feature Transform' (SIFT) by Lowe [49] and the 'Speeded-Up Robust Features' (SURF) by Bay et al. [5], along with Tola et al.'s 'DAISY' [91] algorithm. On the other hand, dense feature matches, known as dense correspondences, can also be obtained for all or most pixels in a set of images. Dense correspondence algorithms make use in many cases of the optical flow between images [36, 6]. Other methods, like [58], examine the implications of shape on the process of finding dense correspondences and half-occlusions, which are points visible in one of the two cameras, for a stereo pair of images. An overview and comparison of different dense correspondence algorithms is given in [77].

The dense correspondence algorithm used in this dissertation is a direct method solving correspondences coarse-to-fine on $4-8$ mesh image pyramids, with a 5×5 local affine motion model as outlined by Duchaineau et al. [15]. The algorithm guarantees that every destination pixel is only used once and if possible every pixel gets a correspondence pixel in the destination frame. All the correspondences are calculated without any knowledge of the camera pose or epipolar constraints. This leads to a very flexible but still reliable correspondence calculation. This algorithm is further described in Chapter 2.

Once feature matching has been performed, the reconstruction of a scene through stereo vision techniques has traditionally made use of a mathematical approach which establishes the intrinsic geometrical and mathematical relationship between pairs of views, known as the 'epipolar geometry'. Given as few as two views of a scene, epipolar geometry describes the relationship between independent pairs of image feature matches, for example the pixel coordinates of a given corner of a building as seen through two different camera positions, and the point in 3D space towards which the light rays from the two camera centers through those pixel positions are directed. Details on epipolar geometry can be found in Appendix B.

The mathematical description of the epipolar geometry can be encapsulated in its entirety in a 3×3 rank-two matrix known as the 'fundamental matrix' F, which was originally introduced into the research community by Luong [50]. There exist both direct methods to solve for its entries as well as non-linear methods, as described in Appendix B. The direct methods involve setting up linear equations using information from the input feature matches and then solving the system through singular-value or eigenvalue decompositions. These include the *5-point* [84], *6-point* [63, 61], *7-point* [30] and *8-point* [27] algorithms. As was proven in Hartley's work [27], the input feature match data should be normalized to avoid numerical instabilities in the solutions. The survey by Rodehorst [75] has a very good summary on the advantages and disadvantages of these direct methods, where it is proven that the *5-point* method [84], which is also the most recent, performs the best and avoids the degenerate cases which may occur with other methods. Non-linear methods [30] include the 'algebraic minimization method', 'minimization of the epipolar distance' and the 'gold-standard method', all of which minimize cost functions based on entries of the fundamental matrix, or its calibrated version, the 'essential matrix'. The 'Random Sample Consensus' (RANSAC) algorithm [18] can be coupled with any of these methods to help obtain more robust estimates for F. Using the computed epipolar constraints in a process known as 'guided matching', more matches can be generated across the images to obtain a dense set of matches [30]. An issue with such constrained matches is that they depend directly on the quality of the estimated epipolar geometry, making them mathematically valid but not necessarily correct in reality.

The importance of the essential matrix lies in the fact that it can be decomposed, using for example Singular Value Decomposition, into the camera's 'pose' or 'extrinsic parameters'. Pose consists of relative translation and relative rotation between the cameras. It can only be recovered directly from the essential matrix since it takes into account the camera's intrinsic parameters such as its focal length, skew and principal point, and not from the fundamental matrix. Pose estimation based on the essential matrix is explained in Appendix C. A more detailed mathematical derivation can be found in the work of

Spetsakis et al. [82] and Weng et al. [100]. A complete overview of several different pose estimators is given in Rodehorst et al. [75], though there are a large number of algorithms in the literature for recovering the camera extrinsics and also its intrinsics. One interesting approach determines pose from optical flow [9, 10], by use of a differential epipolar equation that relates optical flow to both the intrinsics and extrinsics of the camera. Yet other algorithms deal with situations where information is missing, such as the focal length [66]. Nistér's approach [57] estimates pose by applying the RANSAC [18] algorithm directly on the pose parameters. If both feature matches and the corresponding 3D structure are available, poses for two or any number of cameras viewing this structure can be recovered using the 'Direct Linear Transformation' algorithm [89]. Yet other methods estimate pose through other techniques, such as using vanishing points [11] and vanishing lines [99].

Once the pose has been recovered, 'triangulation' is typically used to compute 3D positions from feature matches, intrinsics and extrinsics. The most common method is 'linear triangulation', which involves solving a 4×4 system directly for each point. An 'optimal triangulation' method [29] has been proven to give more accurate results, though it is considerably slower. The state-of-the-art method is Lindstrom's 'fast triangulation' [45], which outperforms all others as far as speed and robustness. All of these algorithms are discussed in Appendix D.

In the discussion up until now, pose and structure estimation has only been discussed for the simplest two-view case. However, there exist techniques where the pose and structure can be computed directly from three or even more images at once. In Avidan et al. [3], a threading function operates on two consecutive fundamental matrices and connects them using the 'trifocal tensor', which describes the epipolar geometry of three views. The threading operation guarantees that consecutive camera matrices are consistent with a unique 3D model, without ever recovering a 3D model. A classic approach known as 'Tomasi-Kanade Factorization' [93] solves for both structure and camera motion, where a factorization method uses a $2F \times P$ measurement matrix of the image coordinates of P points tracked through F frames. Under orthographic projection this matrix is of rank-three, and the Singular Value Decomposition technique is used to factor the measurement matrix into two matrices which represent object shape and camera motion, respectively.

Once feature matches, poses and scene structure have been computed for multiple frames, a common post-processing step is to carry out a 'bundle adjustment' parameter optimization, where the total reprojection error of all computed 3D points in all cameras is minimized using non-linear techniques. A sparse implementation [47] has been developed, which speeds up processing though this is an inherently slow process. Details on this process can be found in Appendix E, and also in several publications [30, 17, 98]. Some variations to the standard method exist. For example, in Zhang et al. [106] a basis of equations for formulating an improved cost function is presented. It involves less unknowns than the ones used in standard bundle adjustment by eliminating the camera orientation parameters through algebraic manipulation.

A number of algorithms exist to carry out pose and structure estimation sequentially for long image sequences, given initial estimates. Our implementation of such a system is described in Chapter 6, but there are many ways to do this in the literature. For example, some recent publications exploit the properties of 'particle filters' for pose estimation and reconstruction. In Pupilli et al. [69], a particle filter provides recursive approximations to the posterior density for the pose parameters of a hand-held camera. In Qian et al. [70], the structure-from-motion problem is addressed using 'Sequential Monte Carlo' methods. A new algorithm based on random sampling is derived to estimate the posterior distributions of camera pose and scene structure, and can handle issues such as erroneous feature tracking, occlusions and moving objects. Many methods exist which are not based on such filters. In [71], SSD tracking is combined with incremental structure computation into a system computing both motion and structure on-line from video. In combination, the structure estimation and tracking benefit each other, resulting in both better structure and more robust tracking. In Nistér [57], a system capable of performing robust live ego-motion estimation for perspective cameras is presented. The system is powered by RANSAC [18] with preemptive scoring of the motion hypotheses. Of special interest is the approach in [107], where an incremental motion estimation algorithm to deal with long image sequences is proposed. It applies to a sliding window of triplets of images, but also uses those points shared only by two views. The problem is formulated as a series of local bundle adjustments in such a way that the estimated camera motions in the whole sequence are consistent with each other. The computational gain achieved makes it considerably faster than global bundle adjustment.

A number of algorithms are capable of reconstructing long image sequences but in a non-sequential manner. In Fitzgibbon et al. [20], a hierarchical processing of images is used, which uses image triplets and associated trifocal tensors as primitives. This is shown to optimally distribute error over the sequence. Its major contribution is that it can deal with closed sequences, where a part of the scene is revisited later on in the sequence, using additional constraints available for these cases. Nistér [55] also reconstructed scenes using a hierarchy of trifocal tensors from an uncalibrated frame sequence. In Martinec et al. [52], a new technique for estimating a multi-view reconstruction given pairwise Euclidean reconstructions up to rotations, translations and scales is presented. The partial reconstructions are glued by first estimating camera rotations consistent with all reconstructions, then modifying all pair-wise reconstructions according to the new rotations and refined by bundle adjustment, and finally the rotations are used to estimate both camera translations and 3D points.

Methods also exist for sequential image acquisition from an un-calibrated hand-held camera or rig [68, 2]. Reconstruction from turn-table sequences has also been shown [19]. Pollefeys et al. [67] perform urban 3D reconstruction from stereo pairs mounted on a car. The 3D reconstruction is performed based on a plane-sweep stereo algorithm with multiple viewing directions. The most modern methods, such as those presented in Snavely et al. [81] and Goesele et al. [22], can reconstruct scenes sparsely from images at the Internet scale. With poses extracted from such methods, dense patch-based methods such as 'Patch-Based Multi-View Stereo' (PMVS) [21] can produce quasi-dense reconstructions from the same set of images. There are a number of publications with the purpose of evaluating the different reconstruction methods from the literature. Seitz et al. [79] introduced a comparison and evaluation platform for reconstructions from stereo, termed the Middlebury Stereo Evaluation. Strecha et al. [85] provide a similar framework. Both evaluations provide publicly-available datasets for testing, and allow the submission of results to allow for a global comparison.

Yet another area of active research involves how to visualize the obtained 3D scene models. For example, Remondino's work [73, 74] provides good overviews of the post-processing following the acquisition of 3D points, such as creating triangle meshes from the computed point clouds. Modern software packages used to visualize such models using texturing, lighting and other user-defined viewing conditions are also discussed [74].

Even though many different approaches to the 3D scene reconstruction problem have been proposed, and many clear advances have been made during the past decades since the problem was first studied, the main issues to this day involve reconstruction accuracy and complexity. Such issues must be dealt with in order to achieve the desired semantic analysis we want to achieve for our aerial imagery application. For instance, dense correspondence and feature matching algorithms, from which pose estimation and triangulation algorithms directly depend on, have not been able to flawlessly deal with changing image conditions such as occlusions, moving objects and lighting changes. No technique so far, either direct or non-linear, has been capable of extracting error-free poses from images, though the newest methods, such as the *5-point* algorithm, are capable of avoiding degeneracies that were the downfall to previous epipolar geometry and pose estimation algorithms. Bundle adjustment, which has been proven to be necessary to avoid error accumulation in parameters to the extent possible, is inherently very computationally-expensive. Though it can be sped up through sparse techniques, it is by far the main bottle-neck if real-time reconstruction of large scenes is intended. Overall, there are a number of error sources at each stage of a reconstruction pipeline that can affect the quality of the final multi-view reconstruction. Unfortunately, detection of such errors is very difficult unless there is ground-truth knowledge involved, for example absolute poses obtained through GPS and IMU measurements, that can be used for error detection and correction. Furthermore, the two issues of accuracy and computational expense are bound to one another, since lower absolute errors would by itself allow for expensive optimization processes such as bundle adjustment to converge faster to an optimal solution.

In light of the above, the objective of this PhD dissertation is to present a generalized error analysis framework for scene reconstruction from aerial video, consisting of methods for the detection, factorization and correction of error sources present in all stages of a reconstruction pipeline, and in the absence of ground-truth knowledge. The main goal is to reduce errors at the different stages since this not only improves reconstruction quality, but also helps alleviate the computational expense associated with processes such as bundle adjustment. Though the presented algorithms were designed for sequential scene reconstruction from aerial video, they have been proven to work across different scene types and camera motions, and for both real and synthetic scenes. Furthermore, because of their nature, the methods are general enough that they can be applied in conjunction with many different types of scene reconstruction algorithms besides the reconstruction pipelines which will be described in this document.

The research that was performed can be divided into two main parts, which follow a chronological order in the way they were actually performed. A scene reconstruction pipeline using unconstrained dense correspondences was initially implemented, as described in Chapter 2. Even for a reduced amount of images, errors in the dense correspondence process cause pose and structure estimation inaccuracies that accumulate over time in sequential reconstruction. An example of this is shown in the left image of Fig. 1.1, which shows a dense reconstruction achieved from a few images of the *Walnut Creek* dataset. The left-most buildings were reconstructed inaccurately mainly due to occlusions and repetitive patterns, as seen visually and also evidenced by high reprojection errors. Error accumulation would cause the reconstruction to degenerate further if more images were added to the reconstruction. Due to the computational expense and error accumulation, a multi-view reconstruction pipeline was proposed that made use of accurate sparse feature matching instead of dense correspondences as input, as described in Chapter 6. The goal of this pipeline is to obtain accurate camera poses throughout a video sequence and create as additional output a sparse point cloud representing scene structure. The pipeline operates by obtaining an initial robust two-view reconstruction and sequentially adding-in new frames using the 'Direct Linear Transformation' algorithm [89] embedded in RANSAC [18], while bundle-adjusting after every step. An example of this is shown in the right image of Fig. 1.1, which shows a full orbital reconstruction of the synthetic *Lubbock Canyon* dataset. The estimated camera positions are rendered as yellow points, and it can be seen that they form almost a perfect circle, matching the true camera trajectory. The poses and structure were

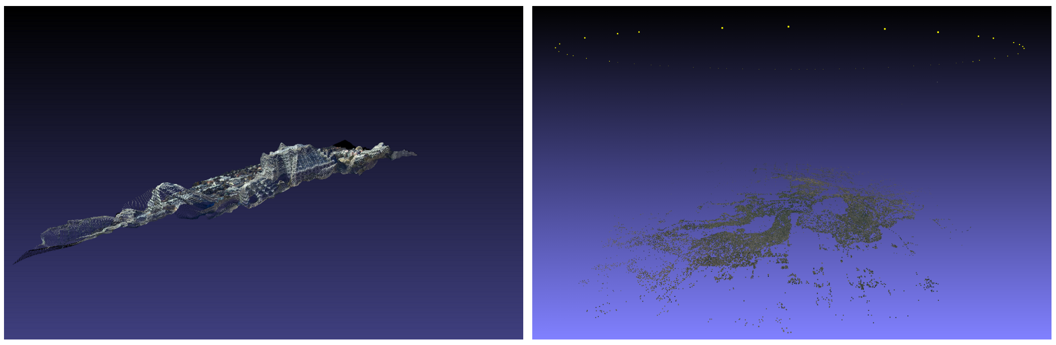


Figure 1.1: Scene reconstruction from dense correspondences (left), which is prone to inaccuracies due to issues such as occlusions and repetitive patterns, as shown for the *Walnut Creek* dataset. Higher accuracy in pose and structure estimation can be achieved using sparse feature matching as shown for the *Lubbock Canyon* dataset (right), but semantic analysis is no longer possible due to incompleteness. Camera positions for this case are rendered in yellow.

very accurately estimated, with a very low average reprojection error. Unfortunately, a reconstruction this sparse is not appropriate for the semantic analysis that should be achieved in our intended application. Current methods for upgrading to a dense reconstruction, such as 'Patch-Based Multi-View Stereo' (PMVS) [21] offer promise as far as achieving very accurate dense reconstructions in general, but still suffer from incompleteness and inaccuracy in certain cases. Initially, our main focus has been on reducing errors as much as possible across all stages of the pipeline, but the end goal is to achieve dense, error-free reconstruction. There is high hope that a method such as the one introduced in Chapter 7 may eventually lead to this goal.

In this context, two main applications will be discussed for dealing with scene reconstruction errors in the absence of ground-truth. The first set of algorithms derive total structural error measurements after an initial scene structure computation and factorize errors into those related to the underlying feature matching process and those related to camera parameter estimation. These algorithms were originally designed for the dense correspondence scenario but can be used as-is with sparse feature matching. Another application of the proposed generalized error analysis framework is in reconstruction pre-processing, in the scenario of sequential reconstruction from sparse feature matching. An algorithm was developed that detects and discards frames that would lead to inaccurate feature matching, camera pose estimation degeneracies or mathematical instability in structure computation based mainly on a residual error comparison between two different match motion models. The main algorithms that were developed are summarized below, and will be described in detail in the remaining chapters of this document, which each include an overview of the associated state-of-the-art. This includes an entirely new methodology for scene reconstruction error detection and correction, which we have termed 'parallax orbits'.

The work in Hess-Flores et al. [32] presents a novel method to detect and correct inaccuracies in a set of unconstrained dense correspondences between two images. Details can be found in Chapter 3. Starting with a set of dense correspondences [15], an initial pose estimate and dense 3D scene reconstruction are obtained and bundle-adjusted. Reprojection errors are then computed for each correspondence pair, which is used as a metric to distinguish high and low-error correspondences. An affine neighborhood-based coarse-to-fine iterative search algorithm is then applied only on the high-error correspondences to correct their positions.

Such an error detection and correction mechanism is novel for unconstrained dense correspondences, for example not obtained through epipolar geometry-based guided matching [30], since that forces reprojection errors to be zero even when matches are incorrect and thus masks any errors. Results on real and synthetic imagery indicate that correspondences in regions with issues such as occlusions, repetitive patterns and moving objects can be identified and corrected, such that a more accurate set of dense correspondences results from the feedback-based process, as proven by more accurate pose and structure estimates. Such an error detection for dense correspondences without knowledge of ground-truth had not been achieved in the literature. An important motivation for using feedback after bundle adjustment is to avoid applying the correction mechanism to all available correspondences, which would result in $10 - 20x$ slower processing times during this phase. While it is not the objective here to explicitly solve for the occlusion problem in reconstruction or detect moving objects, the end goal is to achieve the best possible correspondence accuracy in such problem areas, even if it implies a higher computational expense, which makes it important to apply only where necessary.

The method discussed in Hess-Flores et al. [32] mentioned above and in Chapter 3 yields good error detection and correction results for dense correspondences, pose and structure estimation, but the main drawback of the method is that applying bundle adjustment is very expensive. As will be described, it is possible to obtain a measure of the error in the dense correspondences, and additionally in camera parameters, by using a very simple measure of ray divergence when attempting scene reconstruction. Details on this algorithm can be found in Knoblauch et al. [41], and in Chapter 4. Such ray divergence is a function of both the quality in the given unconstrained dense correspondences as well as in the estimated camera parameters. Total reconstruction error is obtained by measuring ray divergence for each dense correspondence pair; under perfect conditions such rays should intersect exactly in space but this is generally not the case. The set of ray divergences provides an error map without requiring ground-truth information or making any assumptions about the scene, which is the main novelty. Additionally, an error separation is introduced, such that errors related to the dense correspondences can be separated from errors related to camera parameter inaccuracies. The total error map consists of a smooth global error superimposed by high-frequency errors. Considering that the two main error sources are camera-parameter inaccuracies and inaccurate dense correspondences, the important assumption is made that camera parameters introduce a smooth overall error, and can be modeled by a B-spline surface, while correspondence errors show up as local, high-frequency errors and are modeled as the difference between the total error map and the B-spline surface. A further analysis of the two types of errors based on signal processing signal-to-noise-ratio theory allows for a more systematical decision on which of the two processes, dense correspondence computation or camera calibration, have a greater error and must be improved.

Furthermore, using the error measure for camera-related parameters, it is then shown how it can be used to improve the convergence properties of the bundle adjustment process. This line of work is described in Hess-Flores et al. [33], and also in Chapter 4. In this case, a set of very accurate yet sparse feature matches is used as input, such that the total ray divergence error is assumed to correspond to camera-parameter inaccuracies. It is known that divergences vary smoothly, and based on their histogram a set of weights can be used in bundle adjustment to improve its convergence. A proof of the validity of ray divergence as a measure of camera parameter uncertainty is provided, since it correlates well with Beder et al.'s confidence ellipsoid roundness measure for computed 3D scene points [7] in the case when image feature covariances are set to identity, which is a reasonable assumption given very accurate input feature matches. It is proven that this novel weighting scheme outperforms other common bundle adjustment weights such as image feature covariances [8, 105], and coupled with its inexpensive computation it becomes very suitable for general multi-view pose estimation and reconstruction applications. The entire procedure is first derived for the two-view case, but also shown how this can easily be extended to multiple views.

The algorithms described in Chapters 3 and 4 describe error detection, factorization and correction techniques, demonstrating the approaches mainly for the two-view case. However, in the extension to multiple views an essential pre-processing step is how to choose *which* frames to use if reconstructing from a large number of images or from streaming video, which becomes possible if using accurate sparse feature matching. This process is known in the literature as 'frame decimation', and there are a surprisingly small number of algorithms to do this in the literature. The goal of frame decimation is not only to reduce the computational load but also discard frames that could possibly lead to inaccurate feature matching, pose degeneracies and mathematical instability in structure computation. To this end, a method is presented for non-parametric sequential frame decimation algorithm for image sequences in low-memory streaming environments. A detailed explanation is found in Knoblauch et al. [40], and also in Chapter 5. Previous approaches from the literature apply a global frame decimation, which takes into account the entire set of frames and is therefore intractable for streaming video, and/or rely on thresholds that make them scene-dependent. The novel algorithm eliminates these issues, and is a key component of the sequential sparse reconstruction pipeline described in detail in Chapter 6, which operates with decimated keyframes. A frame decimation cost function was designed that ensures at most three frames are kept in memory at any given time, and has one global maxima representing a good keyframe at each evaluation step. The cost function is essentially a weighted version of Torr's 'geometric robust information criterion' (GRIC) [94], which provides residual error relationships between epipolar geometry and homography fitting, such that pose degeneracies and structural instability are detected when homography fitting is more accurate than epipolar geometry fitting. Weighting is based on information from feature matching, and favors shorter baselines and with a larger coverage of image area. The approach was tested with different real and synthetic datasets consisting of different scene types and camera motions, and provides keyframes resulting in sequential multi-view reconstructions with very low and constant reprojection errors, while reducing the amount of frames typically to $20\% - 40\%$ of the original number.

The most recent line of work in this dissertation, and with the potential to be the most relevant, is a novel method that exploits the strong constraint imposed by the path of a moving camera to allow for a fundamentally different way of detecting and concurrently correcting errors in the different scene reconstruction processes. The main insight behind this method is that parallax movement corresponding to a feature track should ideally be a scaled version of the camera trajectory when projected onto a plane. This constraint had not been explicitly taken into account in the existing scene reconstruction literature, and it is shown how it is a valid constraint that obeys epipolar geometry criteria. It is discussed how this principle can be used to concurrently improve camera parameters, scene structure and also the feature tracks themselves using a very efficient, more accurate, faster and more complete alternative to traditional bundle adjustment, which can be performed in one simple, non-iterative step. The theory behind this novel constraint is detailed in Chapter 7.

The inputs are a sparse reconstruction and feature tracks for a set of images, along with the corresponding camera projection matrices. Such information can be obtained for example with the pipeline described in Chapter 6. The first step is to pick an anchor frame, such that scene reconstruction will be performed with respect to this particular frame, and using only those feature tracks visible in that frame along with their associated cameras. Next, a ‘reconstruction plane’ is chosen. Rays are shot from all cameras through the pixel positions for each track and intersected with this plane. Each feature track projected on the plane is referred to as a *parallax orbit* or *parallax path*. If all parallax orbits are translated to a common origin, and the same is done for the mirrored projection of the camera path on the plane, it can be proven that each parallax orbit on the plane is a scaled version of the camera path on the plane. Furthermore, it can be proven that assuming such a common origin, parallax orbit positions for all feature tracks seen by a particular camera must lie on the same line on the reconstruction plane, such that reprojection error is zero along these lines. Traditional bundle adjustment searches for such lines, but without using the intersecting parallax orbits to help guide the optimization, and this is the key advantage of the method, which is based on two fundamental constraints imposed by incorporation of camera trajectory information. This new methodology is still in its infancy as of the completion of this dissertation, and much work remains to be done to prove the generality of the framework, but there is hope that it can have a major impact on the future accuracy and speed of multi-view reconstruction.

In summary, the following is a list of the topics to be covered in the remainder of this dissertation. A detailed description of how to achieve scene reconstruction for the simplest case, two views, is described in Chapter 2. The procedure is demonstrated mainly for the case when dense correspondences are used as input. Chapter 3 describes the iterative algorithm that was developed to detect and correct dense correspondence inaccuracies based on feedback from pose and structure estimation between two views. Next, the new algorithm that was proposed for improving the performance of bundle adjustment based on reconstruction ray divergence is described in detail, in Chapter 4. The last chapters in the dissertation deal with the extension to sequential multi-view reconstruction, making use of robust sparse feature tracking instead of dense correspondences. One key element in this case is frame decimation, which filters frames that are ill-posed for pose and structure estimation as well as feature matching prior to being transferred to the reconstruction pipeline. A novel algorithm for sequential frame decimation is described in Chapter 5. Chapter 6 describes in detail the sequential multi-view pose and sparse structure estimation pipeline that was developed. Finally, the fundamental concepts in the new method based on parallax orbits is described in Chapter 7. Finally, conclusions of the work presented in the dissertation are provided in Chapter 8.

Chapter 2

Two-View Pose and Structure Estimation

As was first mentioned in the Introduction, scene reconstruction from video sequences is important in many modern applications, such as in security, virtual reality, robotics, industrial design, mobile robotics, reconstruction of heritage sites and many others. In particular, dense reconstruction from aerial video sequences allows information to be gathered such that a semantic analysis of the content of a scene becomes possible. Additionally, the registration of successive video frames into a common coordinate system, necessary for reconstruction, also allows for the segmentation of background and moving objects, as well data compression.

For example, if the reconstruction is accurate enough, and information about the next positions of the cameras can be inferred from all the information collected and processed before, it should become possible to estimate what the next image should look like, at a pixel level. At a certain point, other information such as lighting conditions, the texture properties of the buildings, etc. could already be known as well. Since the scene at some point would be known well enough, any moving objects, such as cars and people, could more easily be identified and even tracked. Then even higher levels of scene understanding could eventually be achieved.

A dense scene reconstruction is key in order to achieve such a semantic analysis. To give an example, a dense reconstruction from images of the *Walnut Creek* dataset is shown in Fig. 2.1. It can be seen that it is possible to achieve dense reconstruction, but a closer look reveals that there are regions in the images that were reconstructed inaccurately. This in itself summarizes the main issue that has been encountered in the literature for scene reconstruction, and introducing novel ways to detect and correct the issues that result in inaccurate reconstructions is the main goal of the present document. It is important to note at this point that a dense reconstruction is strictly necessary in order to achieve a semantic understanding of the viewed scene. If 'dense correspondences', consisting of pixel-to-pixel matches between pairwise frames as described in Section 2.1.2 are used as input, these are affected by issues such as occlusions, moving objects and repetitive patterns. The present chapter, along with Chapters 3 and 4, deal with reconstruction from dense correspondences and solutions for issues that arise in this scenario. If more robust but sparse features are used for a sparse reconstruction, the scene might not be adequately sampled though more accuracy is obtained. Sparse reconstruction is discussed in Chapters 5, 6 and 7. Since sparse approaches can be used to obtain robust pose and initial structure estimates, an additional way of performing reconstruction is to apply a final dense reconstruction method starting from the sparse set of points, as discussed in Chapter 6. The objective of the present chapter is to provide a detailed look into a reconstruction pipeline based on dense correspondences, and for the simplest case of two views. This explains some of the fundamental concepts in reconstruction that are applied throughout the rest of the document. Section 2.1 will describe such a two-view reconstruction pipeline in detail, followed by quantitative and qualitative results in Section 2.2. Finally, Section 2.3 discusses how to achieve a three-view reconstruction from dense correspondences, along with the issues present when extending to more views.

2.1 Two-View Reconstruction Pipeline from Dense Correspondences

There are a number of algorithms in the literature that can be used to achieve reconstruction from two or more images, but the basic steps, and those which are used in

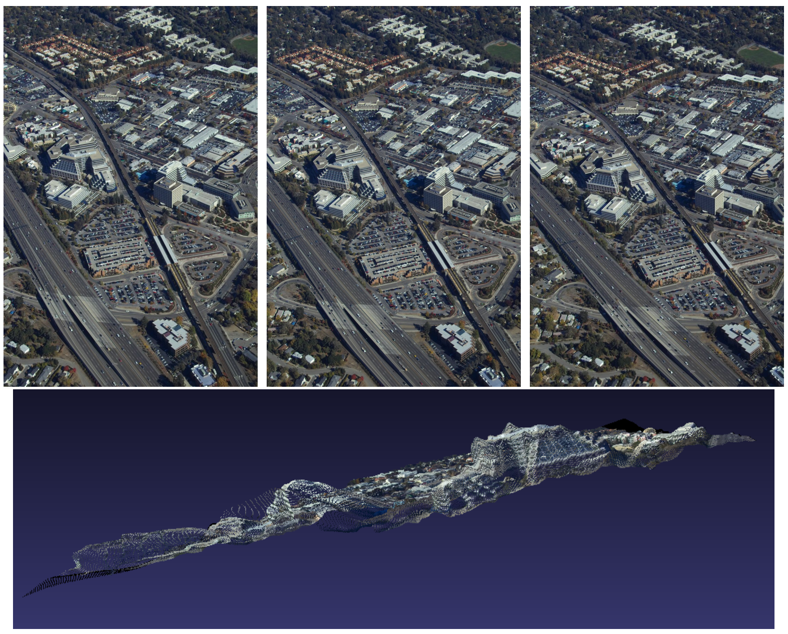


Figure 2.1: Dense three-view reconstruction (bottom) from frames $1623 - 1625 - 1628$ of the *Walnut Creek* dataset (top), based on dense correspondences.

our two-view reconstruction pipeline, are the following [30]:

- Camera calibration: described in Section 2.1.1.
- Feature matching: described in Section 2.1.2.
- Epipolar geometry estimation: described in Section 2.1.3.
- Pose estimation: described in Section 2.1.4.
- Structure computation: described in Section 2.1.5.
- Incorporation of additional views: described in Section 2.1.6.
- Bundle adjustment: described in Section 2.1.7.

The following subsections will describe each of these steps in the order in which they are actually performed in a two-view reconstruction pipeline, followed by results and analysis in Section 2.2.

2.1.1 Camera Calibration

The goal of camera calibration is to obtain the camera's 'intrinsic parameters', consisting of the camera's focal length, principal point and pixel skew. These parameters can be assumed known or estimated. Normally these values can be obtained from the specification sheet for a given camera, or from EXIF tags found in images acquired from the camera. In most cases this information is available, and we start out with this information for our two-view and multi-view reconstructions. If unknown, there are *self-calibration* methods in the literature to obtain the intrinsic parameters using only feature matches. A detailed description of the parameters involved, camera geometry and methods for self-calibration can be found in Appendix A.

2.1.2 Feature Matching - Dense Correspondences

Feature matching is the process of locating the pixel coordinates of a specific scene point, such as the corner of a building, in multiple images. There are both sparse and dense methods to achieve matching. Sparse methods such as *SIFT* [49] and *SURF* [5] obtain a sparse set of scale-invariant features based on neighborhood gradient magnitude and direction information. A high-dimensional feature vector is created for obtained keypoints, for example 128D for SIFT and 64D for SURF. Feature matching then comes down to performing an Approximate Nearest Neighbors search between feature vectors of two images. Sparse feature matching is discussed in further detail in Chapter 6. Since in this chapter we will describe dense reconstruction from dense correspondences, this process will now be described in detail.

The computation of dense image correspondences has been of great importance recently in several Computer Vision applications. For example, the dense registration of successive frames in a video sequence into a common coordinate system at the pixel level enables applications such as dense 3D reconstruction of the viewed scene, segmentation of background and moving objects and data compression.

The goal of a dense correspondence process is to output a 2D coordinate for each source pixel indicating its forward correspondence into the target domain. There are different approaches for this in the literature. Harris and Stephens [24] introduced a motion analysis algorithm based on corners and edges, which is only suitable for specific objects of interest that are to be tracked. Other approaches base the correspondence search on epipolar constraints as shown in [68]. To exploit the epipolar constraints the camera poses have to be known in advance or have to be calculated with a subset of reliable correspondences. A number of other approaches are based on optical flow [6]. A great overview of the available methods can be found in the work of Hirschmüller and Scharstein [35, 77].

For the VidCharts project, Mark Duchaineau's dense correspondence algorithm is used [15], which is part of the *LibGen* library and is a general-purpose unconstrained dense correspondence optical flow-based solver. There are several reasons for starting out with a general-purpose dense correspondence algorithm. By not using epipolar constraints as in guided matching, it allows for errors in the dense correspondences to be unmasked in later stages. Additionally, it is a more general approach that adequately samples the scene; for example sparse feature matchers could fail to find a significant amount of features in regions with little intensity variation, whereas dense correspondences could still be obtained.

The warping function for this algorithm is a bijection, such that there are no folds or gaps in the mesh of warped samples. A coarse-to-fine image pyramid is built as a diamond hierarchy, to improve prediction quality and remove artifacts. The algorithm is iterative with two phases applied at successively finer resolution levels [15]. Each iteration has a matching phase followed by a straightening phase. The resulting transformation for level 'i' is used as a starting prediction for level 'i + 1'. There can be hundreds or thousands of iterations at each level. The first phase, the matching phase, consists of gradient descent. For each source pixel, a local gradient is computed at its current position in the target image. This is used to make a linear prediction of the direction and distance to move the source pixel in the target image to match its intensity. For robustness, the step size is clamped to a fraction of a target pixel. As the gradient magnitude becomes small, the gradient direction becomes more noise than signal, so such pixels are disqualified, as well as those that go out of bounds of the target image, from motion during the matching phase. Next, a straightening phase is applied. Whereas each source pixel moves independently in the matching phase, in this phase information about the current locations of source pixel

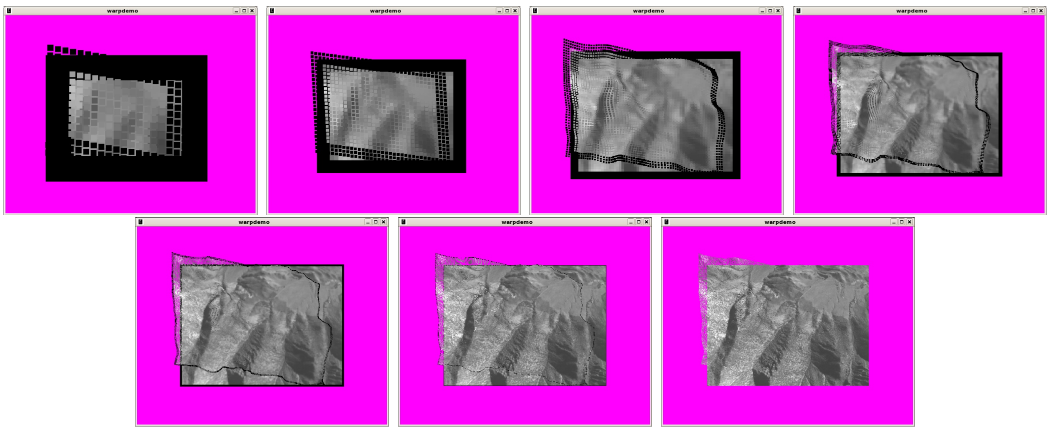
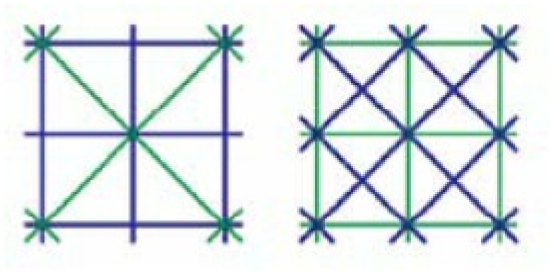


Figure 2.2: Dense correspondences - optical flow on a coarse-to-fine image pyramid [15]. Applications include mosaicing, super-resolution, mover detection and dense reconstruction. The process is shown for frames $1 - 3$ of the *Palmdale* dataset.

neighborhoods is used to locally regularize the warp. For each source pixel, a local 5×5 neighborhood of locations is used to compute an affine transformation from the source to target image. The new target location of each source pixel then becomes a weighted average of its target locations as predicted by all the local affine transformations to which it contributes.

As far as performance [15], the matching and straightening phases achieve 145 million and 68 million pixel iterations per second, respectively. This is 230 times that of the un-optimized CPU implementation, or 3000 times if some simplifications are made, which yields 7 frames per second on 1024×1024 images. The hierarchical process is shown in Fig. 2.2. The dense correspondence process is iterative, and for accurate results, typically hundreds or even thousands of iterations must be run at each level of the diamond hierarchy. Fig. 2.3 displays the diamond hierarchy layout when advancing to a finer-resolution scale, and also shows an example *warpcontrol* file to specify the amount of iterations and commands at each level, where i stands for gradient descent, v for a global affine fit and 5 to applying affine-fitting in local 5×5 neighborhoods around the current pixel being analyzed.



```
warpcontrol {
  start_level { 12 }
  level12 { loops { 1000 } commands { i 5 i 5 i v } }
  level10 { loops { 1000 } commands { i 5 i v } }
  level8 { loops { 500 } commands { i 5 } }
  level6 { loops { 400 } commands { i 5 } }
  level4 { loops { 300 } commands { i 5 } }
  level2 { loops { 200 } commands { i 5 } }
  level0 { loops { 100 } commands { i 5 } }
}
```

Figure 2.3: Diamond hierarchy for dense correspondences [15] (top), and a sample *warpcontrol* file for specifying the number of iterations and commands to control the warp process at each level (bottom).

2.1.3 Epipolar Geometry Estimation

The 'epipolar geometry' between two views describes the intrinsic projective geometry between the two views. This relationship is encapsulated by the 3×3 'fundamental matrix', F. The importance of this matrix is that it contains information about the relative rotation and translation, or pose, between a pair of cameras. There are a number of algorithms in the literature to compute the fundamental matrix, and this is discussed in detail in Appendix B. For the results and tests shown in this chapter, the *Normalized 8-point* algorithm [27] was used. Either dense correspondences or sparse feature matches can be used to compute the F matrix, and any of the methods found in the literature can be embedded in *RANSAC* [18] to improve estimation robustness.

2.1.4 Pose Estimation

The process of pose estimation yields a relative rotation and translation between two cameras. Assuming known camera intrinsic parameters, the 'essential matrix' E can be computed from the fundamental matrix F. The E matrix can then be decomposed into relative rotation R and relative translation T, recovered as a unit direction vector, using

Singular Value Decomposition. The first camera is normally assumed to be in the origin of the world coordinate system. Four possible (R, T) pairs are recovered, and a 'depth test' must be used to determine which of the four possible (R, T) pairs is correct, such that depths must be positive in both cameras for the correct pair. This procedure for recovering pose is described in detail in Appendix C, though there are many other methods, as discussed in the same Appendix.

2.1.5 Structure Computation

The process of structure computation yields the 3D position for a scene point corresponding to a feature match or correspondence. The most common method in the literature for this is 'linear triangulation', where a system of equations based on both relative pose and feature match positions is solved to obtain a homogeneous 3D position (X, Y, Z, W), such that the final coordinates in 3D space are $(X/W, Y/W, Z/W)$. This process is described in detail in Appendix D, along with other methods available in the literature.

2.1.6 Incorporation of Additional Views

The incorporation of additional views, known commonly as '3D-2D registration', can be performed in a number of ways, but the 'Direct Linear Transformation' is the most common and direct method. Given a set of previously-computed scene points and also matches between the previous camera and the current camera, for example the third with an initial two-view reconstruction, a linear system can be solved to compute the 'projection matrix' for the new (in this case, third) camera. Projection matrices are further described in Appendix C. The absolute rotation and translation for that camera with respect to the first can be recovered using QR-decomposition of the projection matrix. Just like with fundamental matrix estimation, estimation of the projection matrix can also be embedded in RANSAC [18]. The DLT method is described in Appendix F.

2.1.7 Bundle Adjustment

The objective of 'bundle adjustment' is to perform a non-linear optimization of all estimated parameters, such as pose and structure, such that reprojection error with respect

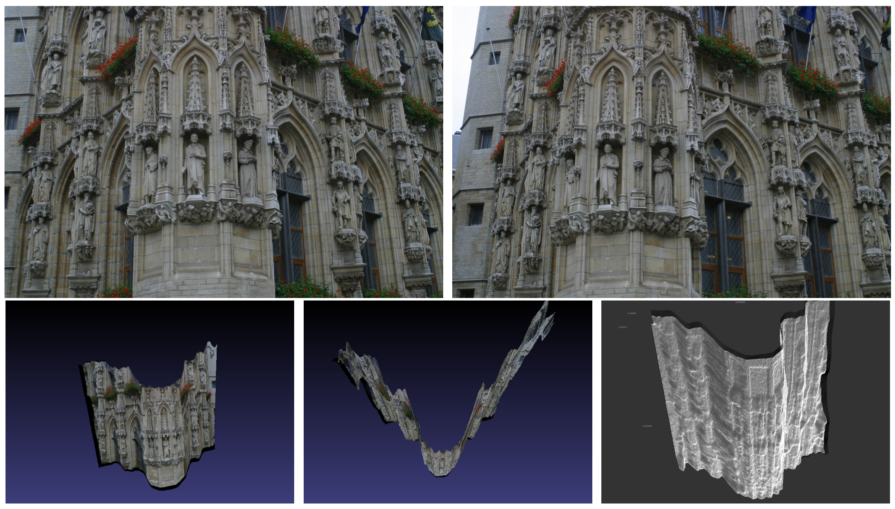


Figure 2.4: Dense reconstructions (bottom) between frame pair $0 - 1$ (top) from the *Leuven City Hall* dataset [85].

to the input feature matches is minimized. Intrinsic parameters and radial distortion can also be included as part of the optimization. Details can be found in Appendix E.

2.2 Tests and Results

Using the described pipeline, dense two-view scene reconstructions, without incorporation of additional views at first, were achieved for pairwise frames in different datasets. Fig. 2.4 shows dense reconstruction snapshots between frames $0-1$ of the *Leuven City Hall* dataset [85]. Fig. 2.5 shows dense reconstruction snapshots between frames $0 - 1$ of the *Rocks 2* dataset [60]. Fig. 2.6 shows pairwise reconstructions between frames $0 - 4$ and frames $4 - 7$ of the *Megascene1* dataset.

Yet another representation of a scene that can be achieved after pose and structure estimation is via a 'depth map', with respect to a chosen camera. For example, Fig. 2.7 shows depth maps obtained with respect to the first camera of a pairwise reconstruction, for the *Coneland* and *Walnut Creek* datasets.

Besides visual results, a more exhaustive analysis of the mathematical behavior seen in pose and structure estimation was performed, to obtain a better understanding of mechanisms that cause failures. This is very important considering that pairwise pose

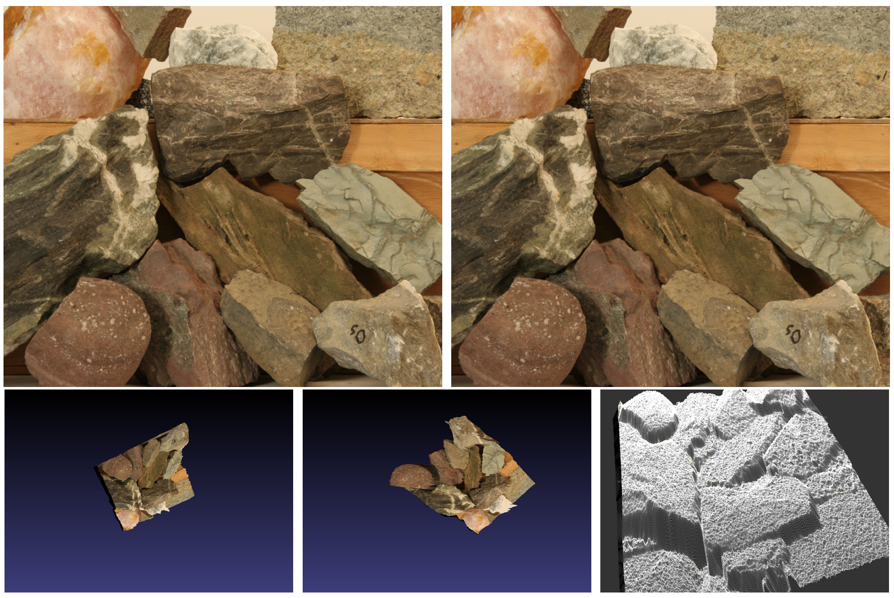


Figure 2.5: Dense reconstructions (bottom) between frame pair $0-1$ (top) from the *Rocks 2* dataset [60].

and structure estimation is the building block for multi-view reconstruction. To this end, tests were performed on both real and synthetic scenes to aid in testing the mathematical behavior and stability of two-view reconstruction. All code for the described two-view reconstruction pipeline was written in the *C* language. For certain matrix operations, such as the Singular Value Decomposition and obtaining eigenvalues and eigenvectors, functions from the *Numerical Recipes for C* library were used. In order to check the numerical values being obtained, another version of the pipeline was written but now using Intel's *OpenCV* computer vision library. Comparisons between the two libraries showed very slight numerical differences in the obtained results.

A series of tests was designed to obtain a better understanding of the underlying factors that affect the *Normalized 8-point* algorithm [27] from dense correspondences, pose estimation using the computed fundamental matrix and the subsequent linear triangulation of 3D positions. The tests can be grouped in three main categories, and each will now be described:

- Test by varying the amount of iterations per level used to create correspondences

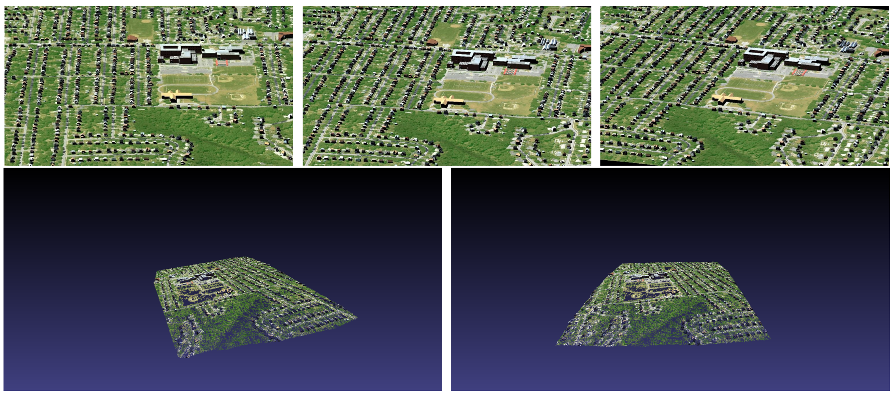


Figure 2.6: Dense reconstructions between frame pair $0-4$ (bottom left) and frame pair $4-7$ (bottom right) from the *Megascene1* dataset. Frames $0-4-7$ are displayed from left to right along the top row.

- Test by varying the amount of correspondences used to compute F
- Test by adding noise to good correspondences to analyze robustness

2.2.1 Test by Varying the Amount of Iterations Per Level Used to Create Correspondences

As was previously mentioned, dense correspondences were obtained using a program included in Mark Duchaineau's *LibGen* library, where the coarse-to-fine optical flow-based algorithm works by iterating at the different levels of the resolution pyramid [15]. This can either be done manually ('warpdemo' program) or automatically ('programwarp'), by specifying the amount of iterations per level with which the program should run. If a higher number of iterations is used, in theory the quality of the correspondences should improve, which was the hypothesis to be tested. A higher number of iterations, however, implies an increased processing time. This particular test consisted on creating correspondences for an image pair (frames $1623-1628$) from the *Walnut Creek* dataset, using the default iterations per level, and then $2x$, $3x$, $4x$ and $5x$ these amounts per level.

The following observations were made. First, visual results improve as the amount of iterations is increased, though the change isn't too noticeable after a certain amount of iterations is used. For example, $3x$, $4x$ and $5x$ iterations produce basically no difference

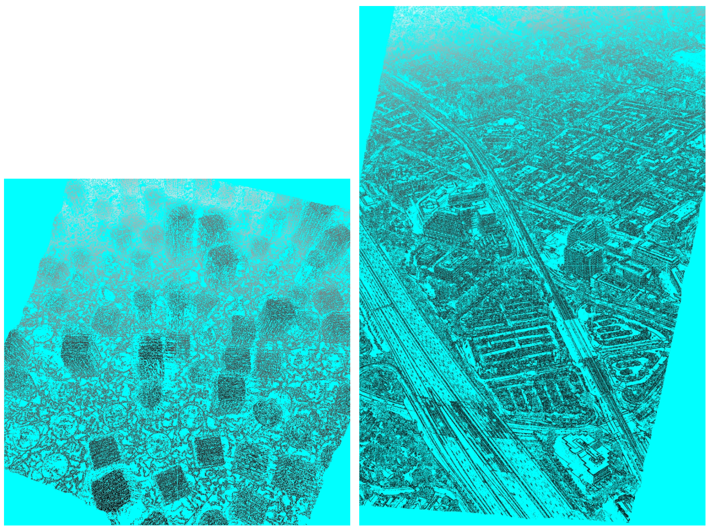


Figure 2.7: Depth map for the reconstruction between frame pair $1 - 20$ from the *Coneland* dataset computed with respect to frame 1 (left), and depth map for the reconstruction between frame pair $1623 - 1628$ from the *Walnut Creek* dataset computed with respect to frame 1623 (right). Darker colors indicate lower depth values.

in the visual or numerical results obtained. Based on the above result, correspondences were generated using $5x$ iterations for every possible image combination in the six-image $1623 - 1628$ burst. This assured that good correspondences were being used and this allowed the behavior of the *Normalized 8-point* algorithm to be directly analyzed. Good 3D results were obtained for all images except for consecutive images, where the short 'baseline', or separation between cameras, caused linear triangulation to fail, which is a known problem with the method. In order to devise a metric to determine when the baseline is too small, a numerical analysis was made of the intermediate results obtained in the reconstruction process. The following data was analyzed for each possible two-view reconstruction in the six-image burst:

- Condition number of the data matrix A used to extract the fundamental matrix F.
- *Frobenius norm* of F.
- Average value of W from all computed 3D points (X, Y, Z, W).

Fig. 2.8 visually shows these trends, and from them the following conclusions can be reached. First, the condition number of the data matrix A to solve for the fundamental matrix decreases as the baseline increases. Lower condition numbers are more numerically stable than higher ones. Furthermore, a wider baseline also correlates with a higher Frobenius norm for F and higher average values for the homogeneous coordinate W in each obtained (X, Y, Z, W) 3D point. There are some fluctuations given that different images are used but the general trend holds true. These results indicate that the *Normalized 8-point* algorithm and/or the linear triangulation process does not deal well with small baselines. As far as linear triangulation, it has been proven that a small baseline makes it difficult to triangulate points since not enough displacement is available and this manifests as near-singular data matrices when solving for 3D points. As for the *Normalized 8-point* algorithm, the fundamental matrices show a lot of numerical instability and the method suffers from 'degeneracies', such that the epipolar geometry can sometimes be undefined or not uniquely determined. These conditions are further discussed in Chapter 5. Since direct methods like this one are used to seed non-linear methods such as the 'gold-standard' method, erroneous results can be obtained when seeding with a bad starting point. Thus, in general the more-robust *5-point* algorithm [84] should be used, since it deals much better with degeneracies and is the state-of-the-art fundamental matrix estimation algorithm. Furthermore, a more systematic method we devised to determine when the baseline is too small, and which also detects degeneracies, is described in Chapter 5.

2.2.2 Test by Varying the Amount of Correspondences Used to Compute F

In this test, the fundamental matrix F was computed by randomly choosing 100, 1000, 10000 and 100000 correspondences out of the 288447 total correspondences available for *Walnut Creek* image pair $1623 - 1628$. This particular image pair was chosen due to the very good visual results obtained in the 3D reconstruction. In general, as the number of correspondences used to estimate F increases, a number of effects are seen. First, the value of W in each (X, Y, Z, W) 3D point increases, as it did in the iteration test, while

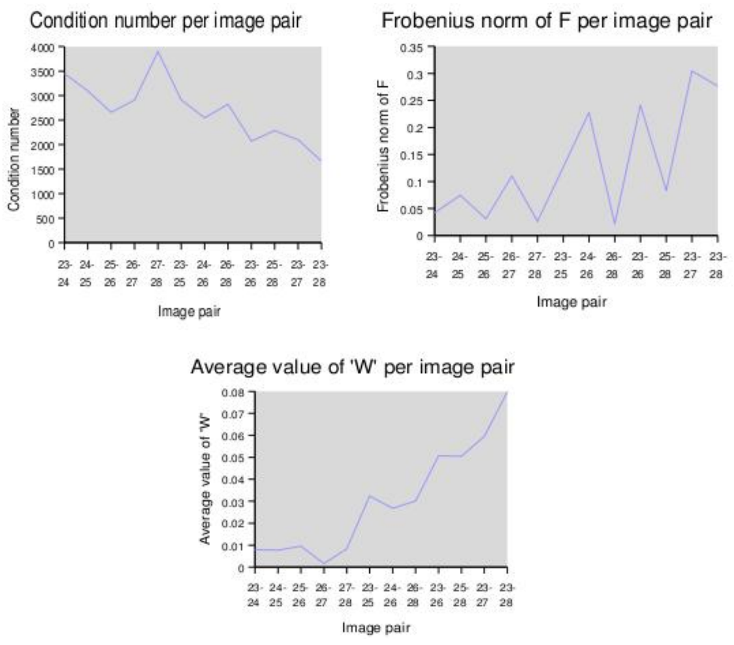


Figure 2.8: Trend charts for values obtained in pose and structure estimation, for varying baselines in *Walnut Creek* dataset two-view reconstructions.

X, Y and Z themselves remain fairly constant. What this basically does is change the scale of the reconstructed points. Also, the translation in the X and Y directions becomes more dominant with respect to the Z-translation, which is what is expected due to the type of camera movement for this particular dataset, and visually-better 3D structures are observed.

2.2.3 Test by Adding Noise to Good Correspondences to Analyze Robustness

In this test, random noise of up to 2 pixels and then up to 5 pixels was added to a subset of 100, 1000, 10000 and 100000 correspondences out of the 288447 total correspondences available for *Walnut Creek* image pair $1623 - 1628$, to see how this affected the 3D results and estimated fundamental matrix. Subjectively good 3D results were obtained for these levels of noise, though numerical values do change, indicating that estimation of the epipolar geometry and 3D points is robust even in the presence of some noise, given that the noise is added to correspondences which would otherwise be very accurate and also using a large-enough baseline.

2.2.4 Results After Bundle Adjustment

An analysis was also performed to numerically analyze the effects of applying bundle adjustment on a dense two-view reconstruction. With synthetic data, it is possible to analyze if this process actually improved the initial pose and structure estimates, since the real values are known. However, for real scenes such as *Walnut Creek*, only the reprojection error output from the 'Generic SBA' program used for bundle adjustment [47] (see Appendix E for further details) and visual inspections on the final 3D points can tell if the process fine-tuned the initial estimates. For example, SBA was applied on the pose and structure estimates for the $1623 - 1628$ frame pair reconstruction. After 41 iterations, the total reprojection error was reduced from $2.57852e + 06$ pixels to 0.156807 pixels, in 147.61 seconds, which is very time-consuming despite the sparse implementation of the algorithm. In order to visually get a sense of how bundle adjustment can affect the quality of a reconstruction, Fig. 2.9 shows the reconstructions obtained before and after applying bundle adjustment on the pairwise pose and structure corresponding to frame pair $15 - 17$ of the *Palmdale* dataset. Notice how the reconstruction after bundle adjustment shows smoother terrain transitions.

2.3 Extension to Three or More Views

To achieve three-view reconstructions, the 'trifocal tensor' is a common tool used in the literature [3]. However, given a previously-computed structure between two frames and matches with respect to a newly-added image, we have found that the 'Direct Linear Transformation' (DLT) embedded in RANSAC provides very good results. This process is detailed in Appendix F. Yet another way to achieve it using two previously-computed pairwise reconstructions that share a frame in common is by applying trinocular constraints on the magnitudes of the translation vectors recovered between the three cameras. This method, however, has not provided results as accurate as those obtained using DLT. The

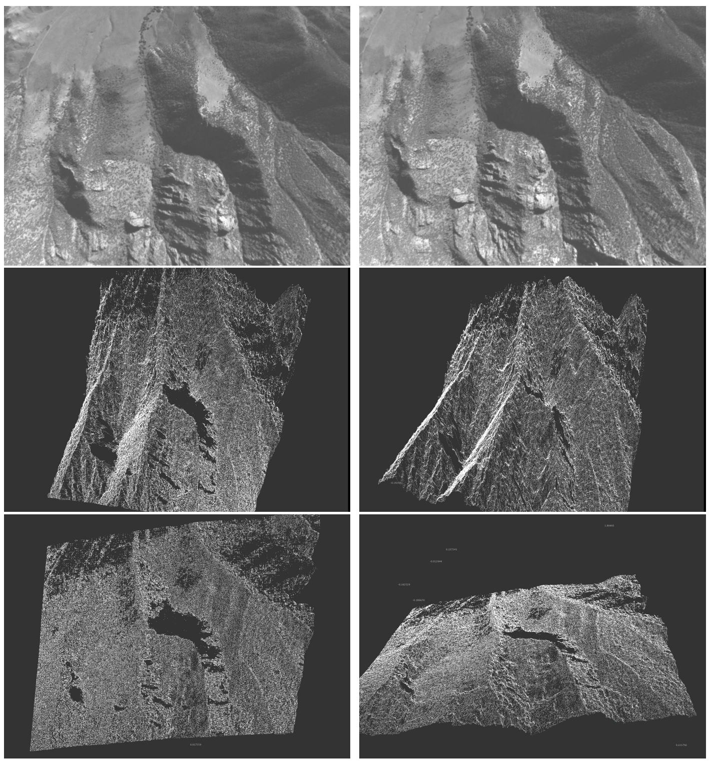


Figure 2.9: Dense reconstructions of frames $15 - 17$ from the *Palmdale* dataset before (middle) and after (bottom) applying bundle adjustment. The two frames $15 - 17$ are shown on the top row.

following figures show snapshots of three-view reconstructions obtained using this procedure. Fig. 2.10 shows snapshots of dense reconstructions between frames $0 - 4 - 7$ and $4 - 7 - 9$ of the *Megascene1* dataset. Fig. 2.11 shows snapshots of dense reconstruction between frames $1 - 15 - 20$ of the *Coneland* dataset.

Even though visually the obtained three-view reconstructions in general appear to be correct, and reprojection errors are low, there are a number of issues with this sequential dense reconstruction approach when attempting reconstructions longer than three views, even when using robust methods such as RANSAC for pose and projection matrix estimation. Dense correspondences suffer from occlusions, texture-less regions and repetitive patterns that cause inaccuracies, such that only small amounts of images could be reliably processed before pose and structure estimation falls apart due to accumulated error. Furthermore, bundle adjustment optimization of parameters, which has been proven in the

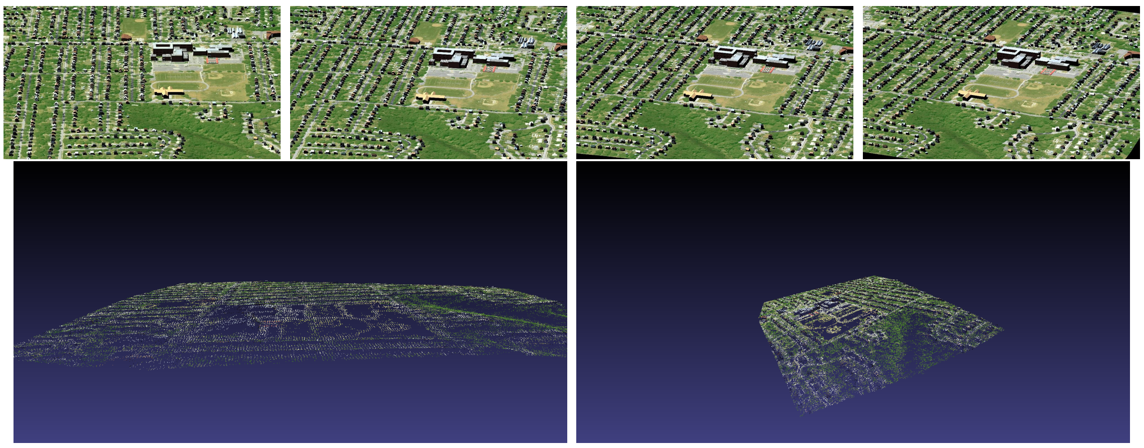


Figure 2.10: Dense reconstruction of frames $0-4-7$ (bottom left) and frames $4-7-9$ (bottom right) from the *Megascene1* dataset, with input frames along the top row.

literature to be necessary for accurate results, is extremely expensive and time-consuming for the dense case. The complexity of bundle adjustment is $O(MN^3)$, where 'M' is the number of feature points and 'N' is the number of frames. Since bundle adjustment should ideally be applied after addition of every new image for sequential dense reconstruction, scalability becomes a major issue. The dense correspondence algorithm [15] has been proven to perform in real-time and is parallelizable, but that does not necessarily apply to the rest of the pipeline, specially in the case of bundle adjustment, and with very large image sizes and high frame rates. Every feature must be tracked through every image, which is very difficult because of the mentioned issues, and yet errors in any of the steps must be identified and corrected since they will always propagate, affecting the reconstruction over time, specially in such a sequential reconstruction approach. Due to the mentioned issues, much of our early work was devoted towards detecting and correcting the main sources of errors present in dense correspondences, attempting to improve parameter estimation robustness, and seeking more efficient ways to achieve reconstruction from dense correspondences, as will be detailed in Chapters 3 and 4.

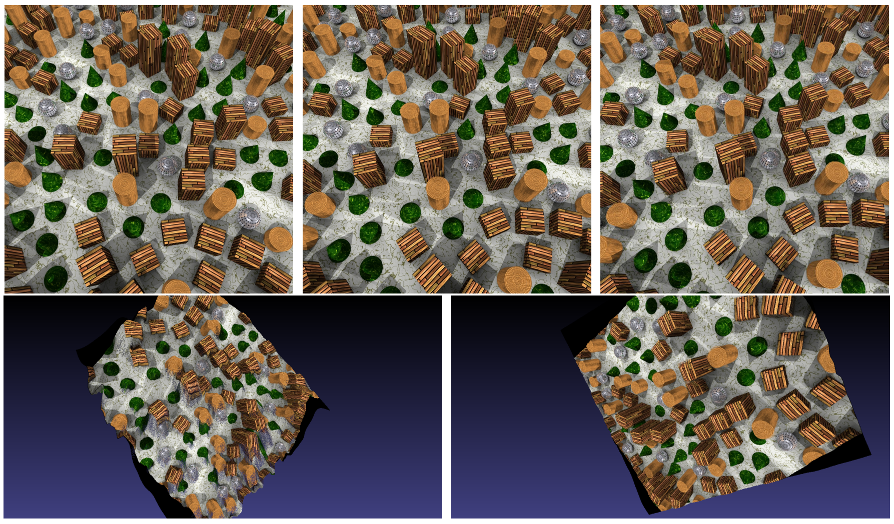


Figure 2.11: Dense reconstructions (bottom) of frames $1-15-20$ from the *Coneland* dataset, shown along the top row.

Chapter 3

Dense Correspondence Error Detection and Correction

In Chapter 2, it was discussed how scenes can be reconstructed starting from a set of unconstrained dense correspondences. It was also mentioned that long reconstructions could not be achieved this way because of pose and structure inaccuracies resulting from using inaccurate dense correspondences, which are affected by issues such as occlusions, texture-less regions, moving objects and repetitive patterns.

To this end, in this chapter a novel method to detect and correct inaccuracies in a set of unconstrained dense correspondences between two images is presented. In summary, starting with a robust, general-purpose dense correspondence algorithm, an initial pose estimate and dense 3D scene reconstruction are obtained and bundle-adjusted. Reprojection errors are then computed for each correspondence pair, which is used as a metric to distinguish between high and low-error correspondences. An affine neighborhood-based coarse-to-fine iterative search algorithm is then applied only on the high-error correspondences to correct their positions. Such an error detection and correction mechanism is novel for unconstrained dense correspondences, for example not obtained through epipolar geometry-based guided matching. Results indicate that correspondences in regions with issues such as occlusions, repetitive patterns and moving objects can be identified and corrected, such that a more accurate set of dense correspondences results from the feedback-based process, as proven by more accurate pose and structure estimates. The complete derivation can be found in Hess-Flores et al. [32], but the method will be explained here in detail.

3.1 Introduction

Dense correspondences suffer from inaccuracies in their estimation that arise whenever there are certain conditions such as occlusions and moving objects present in the scene, and also in regions with little texture or repetitive patterns. Such conditions do not necessarily affect algorithms for sparse feature matching, but certain applications strictly call for the use of dense correspondences, and any of these adverse conditions ultimately affect quality in such applications. To this end, a novel method for detecting and correcting inaccurate dense correspondences will now be described, giving the proof of concept for the case of two input images.

The detection and correction mechanism is enabled by feedback after estimating camera poses and scene structure from the two views and applying bundle adjustment. Using reprojection error after bundle adjustment as the metric to distinguish between high-error and low-error correspondences, an affine neighborhood-based coarse-to-fine iterative algorithm is applied to correct high-error correspondences. The main assumption is that the input dense correspondence set must be unconstrained; for example it cannot have been generated from techniques such as guided matching [30] for the algorithm to work. The reprojection error metric used to detect errors has no meaning for correspondences constructed assuming a perfect fit of these to a given epipolar geometry, as will be detailed later. An important motivation for using feedback after bundle adjustment is to avoid applying the correction mechanism to all available correspondences, which would result in $10 - 20x$ slower processing times during this phase. While it is not the objective of the work presented here to explicitly solve for the occlusion problem in reconstruction or detect moving objects, the end goal is to achieve the best possible correspondence accuracy in such problem areas, even if it implies a higher computational expense, which makes it important to apply only where necessary. Experimental results on real and synthetic data sets indeed show an overall improvement in the accuracy of the dense correspondence set after applying the procedure.

Error detection for dense correspondences has been done in the past, but either under simplifying assumptions or with respect to ground-truth data. In [102], matching errors are identified and corrected, but only one specific scene type is handled. The algorithm in [53] evaluates matching algorithms by introducing an error surface from matching errors. In both cases, the simplifying assumption of searching for disparity along scan lines is made. An exhaustive overview and evaluation of dense correspondence algorithms is given in [77], though the comparisons are done with respect to ground-truth values. As for error correction, an algorithm known as optimal triangulation [30] makes an attempt to correct correspondences based on the pre-computed epipolar geometry between the images. However, such a correction, while mathematically correct and obtained by minimizing a geometrically meaningful criterion, does not necessarily produce matches that are correct in reality; it also reduces reprojection error after reconstruction to zero, thus preventing error detection using such a criteria.

An initial reconstruction of the scene from the two input views is needed as part of the algorithm, and this was described in detail in Chapter 2, but a brief overview of the steps taken here is now given. In general, a reconstruction pipeline consists of obtaining matches (correspondences) between the images, then computing the relative camera poses between them and finally computing the structure of the scene. The matches used for the initial pose estimation can either be sparse features, for example corners, or dense correspondences, which assign a correspondence in a destination image to each source image position, and can be computed through a variety of methods [77]. For two views, the epipolar geometry between them, encapsulated by the fundamental matrix F [30], can be computed from the initial matches. This matrix can be computed through direct methods, such as in [84, 30], as well as through non-linear methods [30]. The RANSAC algorithm can be coupled with these methods to help obtain more robust estimates for F. Using the computed epipolar constraints, more matches can be generated across the images to obtain dense correspondences (details can be found in [30]). Again, an issue with such constrained correspondences is that the new matches depend directly on the quality of the estimated epipolar geometry, making them mathematically valid but not necessarily correct.

Once matches are available, either sparse or dense, the relative pose (rotation and translation) between the cameras viewing the scene can be computed. Several methods exist, and an overview of different pose estimators is given in [75]. In the particular case that the F matrix is available or has been computed from matches, and if the camera's intrinsic parameters (such as the focal length, skew and principal point) are assumed known, the essential matrix E can be computed and decomposed into the relative rotation and translation. Finally, the scene's 3D structure can be obtained using the available sparse or dense matches. Typically, linear or optimal triangulation [30] is applied on each correspondence pair to generate a 3D position corresponding to the scene structure. Once pose and structure estimates are available, a common fine-tuning step for both estimates is to carry out a *bundle adjustment*, where the total reprojection error of all computed 3D points in all cameras is minimized using non-linear techniques [30]. Fortunately, sparsity in the data has allowed for great speed-ups in this process [47].

By coupling the use of unconstrained dense correspondences in a bundle-adjusted reconstruction pipeline, a novel mechanism to identify the most inaccurate dense correspondences and correct them using an iterative method can be achieved. The entire procedure will be described in detail in Section 3.2, followed by experimental results (Section 3.3) and conclusions (Section 3.4).

3.2 Proposed Algorithm

3.2.1 Pose and Structure Estimation Based on Dense Correspondences

The first step in our algorithm is to compute unconstrained dense correspondences between two images, for which the sub-pixel accurate direct method [15] described in Chapter 2 was used. As mentioned earlier, dense correspondences are prone to errors resulting from occlusions, moving objects, texture-less regions and repetitive patterns. For now, the next steps of pose and structure estimation must proceed despite these errors, but it will be explained in Section 3.2.2 how these issues can be respectively detected and corrected through a novel mechanism based on feedback.

The first step in estimating the relative pose between the two cameras is to estimate the 3×3 fundamental matrix F, which encapsulates the epipolar geometry between the two views. The direct and robust 5-point method [84] embedded in RANSAC is currently being used. It is important to mention that even though a large amount of correspondences are available for estimating F, only a small number are actually needed. Even if the minimal amount is used, the use of RANSAC coupled with random sampling ensures that a reliable F can be estimated in a computationally-efficient yet accurate manner. Now, the essential matrix E is obtained from the fundamental matrix, assuming known intrinsic parameters for the camera, and factorized into the rotation and unit translation (R, t) pair representing the pose. To obtain the scene structure as a set of 3D points for each correspondence pair, linear triangulation was used. A dense scene structure must be computed, since it will be used as part of the error detection and correction mechanism based on feedback that will be described later on.

The objective of the next step, bundle adjustment, is to adjust pose and structure estimates in such a way that the total reprojection error of the 3D points with respect to their corresponding 2D correspondences in each camera is minimized [30]. Details on the bundle adjustment process can be found in Appendix E. An implementation that exploits the sparse block structure of the normal equations solved at each iteration to greatly speed up the process was used [47]. Bundle adjustment must be applied to the entire structure, in order to allow for detection of high-error correspondences, as outlined next.

3.2.2 Outlier Correspondence Detection and Correction

Once bundle adjustment has been applied on the structure and two cameras, all correspondences are now classified based on the reprojection error of the 3D point each pair generated; those classified as having low reprojection errors will be referred to as *inliers*, and high-error ones as *outliers*. Since bundle adjustment is the maximum-likelihood estimator for zero-mean Gaussian noise, the optimized pose and structure estimates, plus the known intrinsic parameters, allow for the 'unmasking' of errors purely in the correspondences in this step. If very erroneous initial pose and structure estimates arise from a very inaccurate input dense correspondence set, optimization may actually guide the estimates away from the global optimum in such cases, thus failing to unmask pure correspondence errors, but

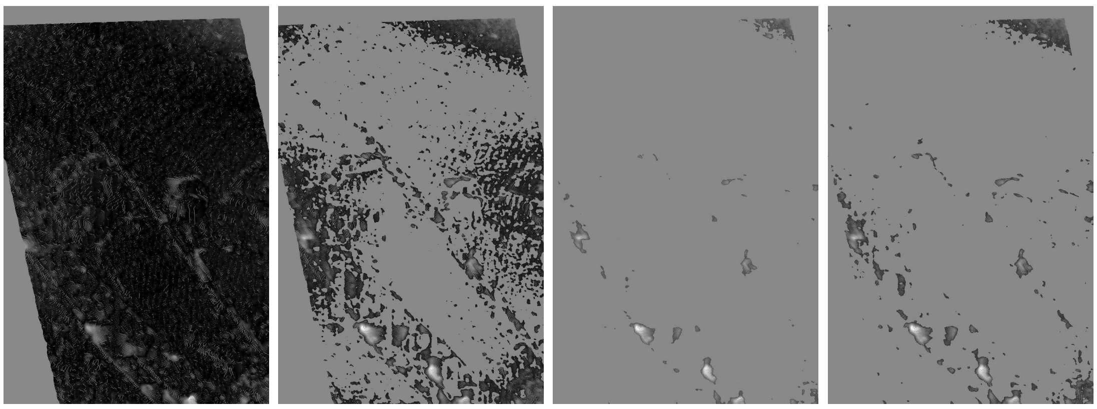


Figure 3.1: Applying a low threshold to detect outliers from a set of correspondences (left) results in unnecessary processing (middle left), while a high threshold erroneously yields very few outliers (middle right). An appropriate threshold must identify only the problematic regions (right).

it is assumed that a reasonable amount of correspondences are accurate enough such that initial pose and structure estimates are in the vicinity of their optimal values.

The reprojection error for the i_{th} correspondence pair is taken as the sum of the absolute values of the errors obtained by reprojecting its resulting 3D point into each individual image. Then, a threshold on the reprojection error r_i (Eq. 3.1) given optimized cameras $\hat{a}_j$ and structure $\hat{b}_i$ is established, such that correspondence pairs whose error is above the threshold are deemed outliers, while the rest are inliers. Without this threshold, or with a low one, the procedure described in the next section, whose processing time is linear in the amount of pixels, would be applied to nearly every pixel in the image, which is expensive. On the other hand, a higher threshold would imply faster processing, but with the downfall that some correspondences with relatively substantial errors are left uncorrected. This is shown in Fig. 3.1, which shows the effect of the used threshold on the number of detected outliers. The algorithm should solely detect high-error correspondences in problematic regions. An analysis of the reprojection error histograms for different data sets reveals that the curves gradually taper off as the reprojection error grows. This observation is key towards determining an appropriate threshold. From visual observation of the detected outliers using different thresholds for different real and synthetic data sets, along with the corresponding histogram information, it was determined that a threshold t of an average (as defined in Eq. 3.2) plus 1.5 standard deviations (Eq. 3.3) of the reprojection errors results in an appropriate outlier detection.

$$r_i = \sum_{j=1}^{2} |d(Q(\hat{a}_j, \hat{b}_i), x_{ij})| \tag{3.1}$$

$$\mu_r = \frac{1}{N} \sum_{i=1}^{N} r_i \tag{3.2}$$

$$t = \mu_r + 1.5\sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - \mu_r)^2} \tag{3.3}$$

Next, for a given outlier correspondence pair, the objective is to correct the position of the match in the target image to the information in the source image, while keeping the position in the source image fixed, to find a better match than the one currently available. The algorithm works on a coarse-to-fine resolution pyramid, where a fixed amount of iterations, typically hundreds, is applied per resolution level, such that the pixel count doubles at each level. After constructing the hierarchy, a sub-pixel accurate iterative, three-phase algorithm is applied at successively finer levels. Each iteration consists of perturbation, matching (based on gradient descent) and affine-fitting phases. The resulting transformation for level i of the hierarchy is used as a starting prediction at level $i + 1$.

Starting at the coarsest level, a fixed-size image chip from the source image is centered at the start position on the target image. The first phase of one iteration, perturbation, consists of adding noise to the source image chip in order to avoid local minima which could possibly occur in the next phase, which is based on gradient descent. In this matching phase, for each pixel of the source image chip, a local gradient is computed at its current position in the target image. This gradient is used to make a linear prediction of the direction and distance to move the source pixel in the target image to match its intensity [15]. Each pixel moves independently in this phase. For robustness, the movement step size is only a fraction of a pixel, and further modified according to the magnitude of the gradient. As the gradient magnitude becomes small, as determined by an adaptive threshold, the gradient direction becomes more noise than signal, and such pixels are eliminated from use in the next phase. In the final phase, a least-squares fit is applied to find an affine transformation to be applied to the source image chip. Only those pixels inside

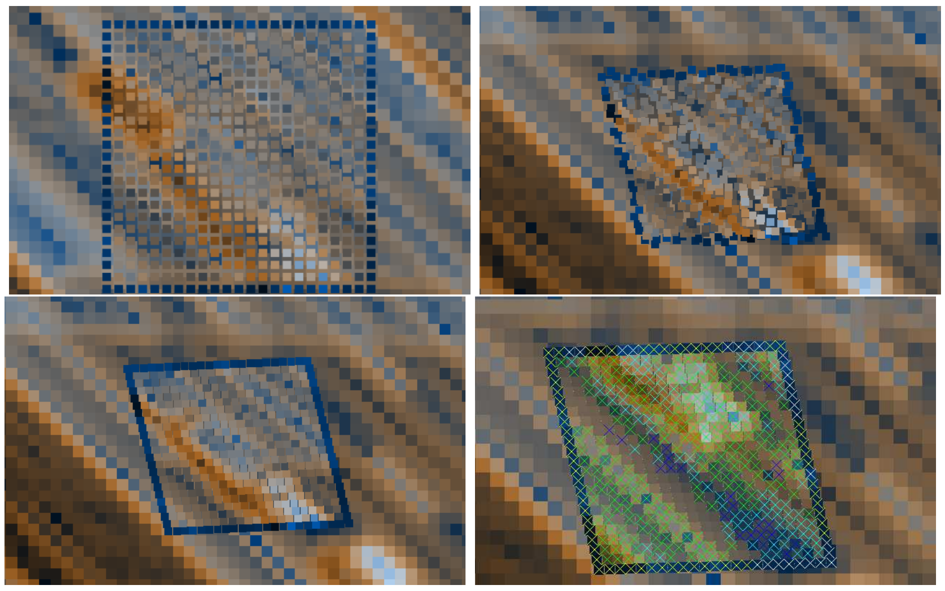


Figure 3.2: Affine correction process (see text for details).

the chip that were not eliminated during the matching phase are used. The three-phase process is iterated a number of times at this coarsest level first and then at successively higher resolutions, resulting in a new and more accurate correspondence position in the target image once completed. The process is illustrated in Fig. 3.2. For an aerial view of a small section of a road with vehicles, the upper left image shows the initial position of the image chip, where gradients are color-coded such that the largest gradients are displayed in lighter colors. Results of the three-phase algorithm are also illustrated for a given iteration: the upper right image shows the result of noise perturbation followed by matching, where the image depicts via tilts in the pixels the direction and also the movement of each individual pixel, and the lower left image shows the affine fit computed from this information. The lower right image shows marked with an 'X' those pixels that were eliminated in the matching phase. Though this correction process is expensive, the goal is to achieve more accurate correspondences by taking into account the actual structure of the neighborhood around a given point, which is more strict than using just the pure epipolar constraint, which could be geometrically but not physically correct.

To determine the most appropriate fixed neighborhood size, the improvement percentage in the average reprojection error for detected outliers with respect to the average obtained before correction was tested for different sizes. It was concluded that similar results are obtained, which is quite remarkable and indicates that the correction process is very robust even when using a relatively small neighborhood. For the *Aerial Views I* dataset [60], though slightly better results are obtained for a large 59×59 neighborhood (3.04% improvement), an 11×11 size (2.9% improvement) was chosen as it yields good results with only a fraction of the processing time. Results were actually worse (1.9% improvement) for a 35×35 neighborhood.

3.3 Results

In this section, the results of the presented approach are analyzed. Both real and synthetic data sets were used to test the algorithm. Tests on real imagery included an aerial imagery dataset, *Walnut Creek*, and two publicly-available data sets: *Aerial Views I* [60] and *Rocks 2* [35]. Fig. 3.3 shows the resulting 3D structure obtained after correction using an image pair from the *Walnut Creek* data set. Fig. 3.4 shows reprojection errors after bundle adjustment, color-coded such that white means a high error and black a low one, over a uniform gray background, for the same image pair. It is clear that the higher errors in general are seen on structures that tend to have plain or repetitive patterns, for example on highways and train tracks (circled in red), near occlusion edges (green) and near the edges of the image (blue). After applying the proposed method, it can be seen that reprojection errors in these areas are generally lower, and the reconstruction is very accurate as seen in Fig. 3.3 for such areas. The highest remaining errors are seen near occlusion edges, which makes sense since there is information missing in such areas (as opposed to textureless regions, which can potentially be matched with enough neighborhood information). A synthetic scene, *Coneland*, was also used to test the proposed algorithm. Fig. 3.5 shows results of the outlier detection and correction from two images of this dataset.

A ground-truth evaluation of the proposed algorithm was also performed. Table 3.1 shows the pose errors with respect to the ground-truth values when using the original set of dense correspondences versus the modified set after applying the proposed algorithm for the *Coneland* data set. Translational error is obtained as the angle in degrees of the

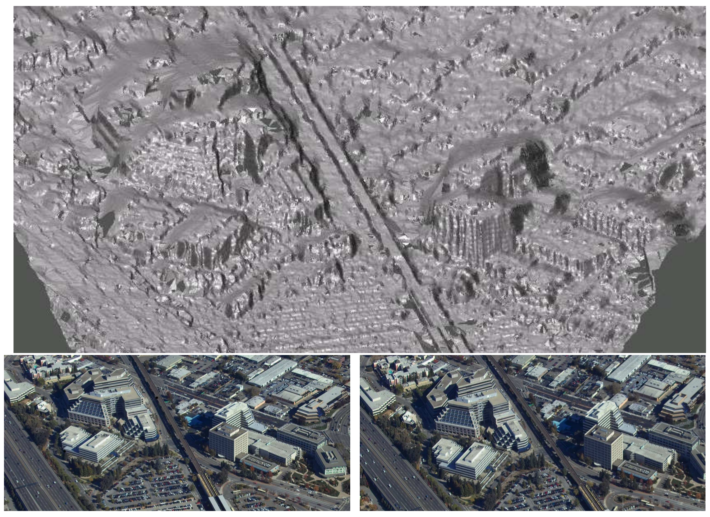


Figure 3.3: Closeups of a reconstruction (top) and its two input images from the *Walnut Creek* dataset (bottom), after outlier correction.

dot product between the ground-truth translation and each estimate. Rotational error is obtained as the angle for the quaternion corresponding to the difference rotation matrix between ground-truth and estimated rotations for each case. It can be seen that pose estimates improve, even though the robust RANSAC is used to estimate F, showing that an overall more accurate set of correspondences is indeed achieved. Table 3.2 shows the outlier percentage and outlier reprojection error improvement percentage when applying the algorithm for some of the test data sets. At first glance the improvements may seem small, but when dealing with sub-pixel accuracy even small errors can result in large structural inaccuracies, so in practice the improvement is substantial.

One possible improvement for the algorithm is to use adaptive neighborhood sizes for the outlier correction process, based on intensity variation statistics for a given chip position. Using larger chips could potentially yield more accurate results in texture-less regions. The use of hardware solutions, such as using GPU's to speed up expensive processes, must also be further analyzed.

Table 3.1: Pose errors $\triangle_R$ for rotation and $\triangle_T$ for translation (in degrees) with respect to ground-truth values using original and modified dense correspondences, for the *Coneland* dataset.

Correspondences	$\triangle_R$	$\triangle_T$
Original	9.818953°	2.443838°
Modified	0.167859°	0.418460°

Table 3.2: Outlier percentage, average outlier reprojection error μ_E (in pixels) before correction and error improvement percentage $\triangle_E$ for tested datasets.

Dataset	Outliers	μ_E	$\triangle_E$
Walnut Creek	5.442%	2.072	10.956%
Aerial Views I	8.475%	11.392	3.234%
Coneland	5.753%	2.759	3.404%

3.4 Conclusions

A new method for detecting and correcting outlier dense correspondences between two images was presented. Initial estimates for the pose and scene structure are obtained from the given dense correspondences, assuming known camera intrinsic parameters, and are then bundle-adjusted. The resulting reprojection errors per correspondence pair are used as a metric to separate high-error and low-error correspondences. Then, an affine neighborhood-based iterative algorithm operating on a coarse-to-fine resolution pyramid is used to correct outlier correspondences. Results on both real and synthetic scenes show that a more accurate set of dense correspondences is obtained after applying the proposed method, which results in an improvement in pose and structure estimates.

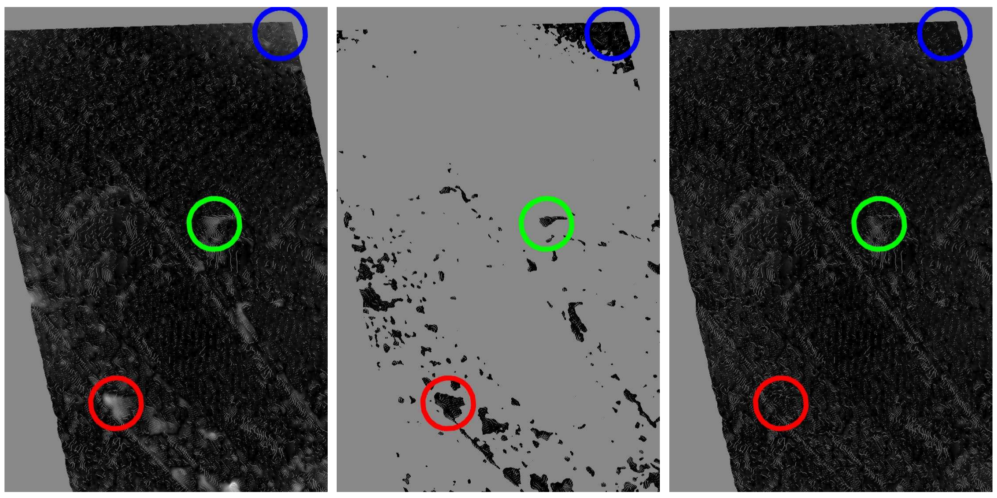


Figure 3.4: Reprojection errors after bundle adjustment (left) for an image of the *Walnut Creek* data set. The detected and corrected outliers are shown (middle), along with errors for the resulting set of correspondences (right).

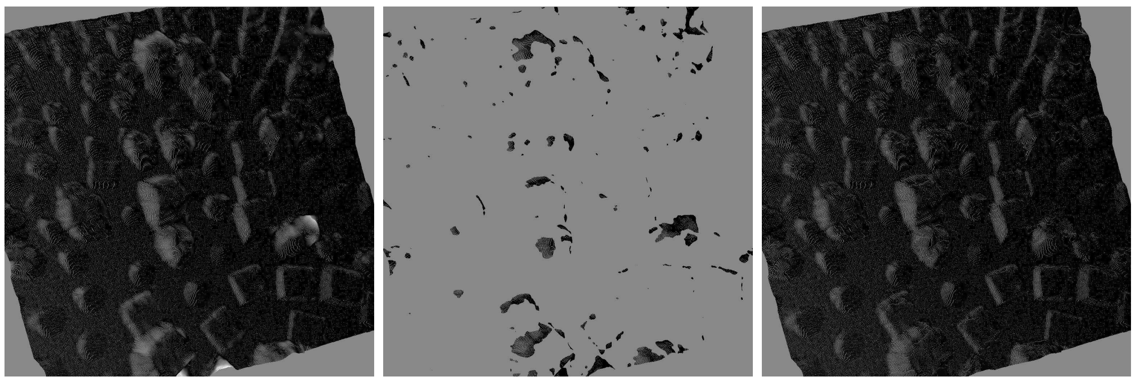


Figure 3.5: Reprojection errors after bundle adjustment (left) for an image of the *Coneland* dataset. The detected and corrected outliers (middle) are shown, along with the errors for the resulting set of correspondences (right).

Chapter 4

Ray Divergence-Based Bundle Adjustment Conditioning

Chapter 3 discussed a method to correct a set of dense correspondences between two views by using feedback from the pose and structure estimation process after bundle adjustment. While good results were obtained, the main drawback of the method is that applying bundle adjustment is very expensive.

As will be described in this chapter, it is possible to obtain a measure of the error in the dense correspondences, and additionally in camera parameters, by using a very simple measure of ray divergence when attempting scene reconstruction. Details on this algorithm can be found in Knoblauch et al. [41]. Such ray divergence is a function of both the quality in the given unconstrained dense correspondences as well as in the estimated camera parameters. The set of ray divergences provides an error map without requiring ground-truth information or making any assumptions about the scene, which is the main novelty. Additionally, an error separation is introduced, such that errors related to the dense correspondences can be separated from errors related to camera parameter inaccuracies. A further analysis of the two types of errors based on signal processing theory allows for a more systematical decision on which of the two processes, dense correspondence computation or camera calibration, has a greater error and must be improved. Furthermore, using the error measure for camera-related parameters, it is then shown how it can be used to improve the convergence properties of the bundle adjustment process. In this case, a set of very accurate yet sparse feature matches is used as input, such that the total ray divergence error is assumed to correspond to camera-related errors. It can be shown that divergences vary smoothly, and based on their histogram a set of weights can be derived and used in bundle adjustment to improve its convergence. It is proven that this novel weighting scheme outperforms other common bundle adjustment weights such as image feature covariances, and coupled with its inexpensive computation, it becomes very suitable for general multi-view pose estimation and reconstruction applications. Details on this process can be found in Hess-Flores et al. [33], but will be explained in detail here. Since the work in Hess-Flores et al. [33] encompasses the results from our earlier work in Knoblauch et al. [41], this algorithm will be described in detail in this chapter, with Section 4.2.1 dedicated to describing the initial work in Knoblauch et al. [41] in more detail.

4.1 Introduction

During the past years there has been a surge in the amount of work dealing with multi-view reconstruction of scenes, for industry and in many other modern applications, as has been mentioned previously. State-of-the-art algorithms [81] provide very accurate matching, camera poses and scene structure, based on sparse features such as those obtained with the SIFT [49] or related algorithms. These recent algorithms are capable of reconstructing large scenes from even unstructured image sets, obtained for example from the Internet. Pollefeys et al. [67] perform urban scene reconstruction from stereo pairs mounted on a car, where the 3D reconstruction is performed based on a plane-sweep stereo algorithm with multiple viewing directions. There are many other approaches for scene reconstruction from images, using either pre-calibrated cameras [103] or estimating camera pose for each frame [25] as part of the reconstruction.

In all such scenarios, camera parameters such as location, orientation and intrinsics may be available or accurately estimated for some of the cameras but not all. This could also be the case even in structured sets of images acquired with the same camera. Because of this reason, despite very accurate feature matching, the accuracy of a multi-view reconstruction still relies on accurate camera parameter calibrations. This creates a great need to identify where and why errors are present in these parameters, specifically without the need to know ground-truth, since this is not always available. In the absence of ground-truth data, multi-view algorithms usually resort to bundle adjustment [47, 98] to reduce reprojection error, which is the most meaningful geometric measure of accuracy in the lack of any ground-truth. However, this can be an expensive element in a scene reconstruction pipeline for high numbers of scene points and cameras, despite recent and efficient sparse implementations such as *SBA* [47], and must be used wisely. Furthermore, it requires a good enough starting point close to the global minimum for convergence.

The main goal of the line of work presented in this chapter is to show how simple *ray divergence* when attempting scene reconstruction is an inexpensive yet powerful tool that can aid in bundle adjustment convergence for multi-view stereo. Ray divergence is defined as the shortest distance between rays emanating from each respective camera center and through each pixel position of a given feature track, as will be further described in Section 4.2.1. This work is partially inspired by our recent algorithm [41], which measures per-correspondence ray divergence when attempting scene reconstruction from a set of initial unconstrained dense correspondences and then decomposes the total error map into errors related to camera parameters and correspondence errors. To our knowledge there had been no other previous work on such an error factorization without using ground truth knowledge. The ray divergence metric relies on the input feature matches being unconstrained, which is what allows for measuring geometric errors. As mentioned previously, using matches generated for example through epipolar geometry-based guided matching would yield no reconstruction error, since these are generated such that they lie on the same epipolar plane with the point they represent in 3D space.

As far as other previous work on camera parameter error analysis, it has been done for the most part with respect to ground-truth values, such as the methodology to test the accuracy of camera pose estimation presented in Rodehorst et al. [75]. The work in Zhao et al. [109] deals with how extrinsic and intrinsic calibration inaccuracies contribute towards depth estimation errors, but for the specific case of a stereo camera pair with a known baseline and other relative positioning information. Benchmarks also exist for reconstruction accuracy [79, 85], though the analysis is done versus ground-truth values, and our algorithm is based on ray divergence rather than the accuracy of exact recovered positions.

In our algorithm, we compute ray divergence per feature track and use it as a joint measure of all camera parameter inaccuracies, without the need for ground-truth knowledge and prior to actually computing the 3D structure. We start out as presented in our initial error factorization algorithm [41], which will be explained in more detail in Section 4.2.1, first computing ray divergences for all available feature matches but with the important difference that robust SIFT features instead of dense correspondences are used, keeping in mind that such feature matches are also unconstrained and therefore it is possible to extract a geometric error unlike in guided matching. We also assume that these feature matches are highly accurate, and this is generally true since sparse SIFT matches are less prone to mismatching due to occlusions, repetitive patterns and texture-less regions than dense correspondences. To further ensure that we have very accurate matches, epipolar geometry-based RANSAC outlier removal [30] is applied prior to computing ray divergences. This in turn allows us to assume that the total ray divergence error corresponds only to camera-related inaccuracies, such that we can avoid the error factorization in Knoblauch et al. [41] to obtain camera parameter errors.

As will be discussed, the validity of ray divergence as a measure of camera parameter uncertainty can be proven, since it correlates well with Beder et al.'s confidence ellipsoid roundness measure for computed 3D scene points [7] in the case when image feature covariances are set to identity. Furthermore, since ray divergence encodes camera inaccuracy information, we show how it can be used in weighted bundle adjustment to improve its convergence properties. It is shown how this scheme outperforms weighting based on more-expensive image feature covariance metrics [8, 105] or Beder et al.'s confidence measure. The entire procedure is first derived for the two-view case, but later shown how this can easily be extended to multiple views. In summary, our algorithm presents a very practical and inexpensive way to measure camera parameter uncertainty in the absence of ground-truth information and use that uncertainty to improve bundle adjustment conditioning. The entire procedure will be described in detail in Section 4.2, followed by experimental results (Section 4.3) and conclusions (Section 4.4).

4.2 Proposed Algorithm

Our analysis will begin with the two-view case, where it is first shown in Section 4.2.1 how to compute ray divergence, and in Section 4.2.3 how to set up weighted bundle adjustment based on ray divergence values. The extension to multiple views will be outlined in Section 4.2.4.

4.2.1 Two-View Ray Divergence Calculation for Dense Correspondences

The first step in our algorithm [33] is to compute ray divergence per feature match, similarly to Knoblauch et al. [41], except we start with sparse SIFT features [49] instead of dense correspondences. However, to better understand the factorization process, a detailed analysis of the work in [41] will now be performed. This paper introduces a simple error evaluation based on ray divergence. The main contribution is the separation of the error into the two main error sources, the camera parameter error and the correspondence error, without the prerequisite of ground-truth data.

In the case of *perfect* feature matches or correspondences, camera intrinsics and extrinsics and no radial distortion, rays starting from each camera center and through the respective image plane feature location should intersect at an exact position in 3D space, but due to any inaccuracies this generally will not occur. We define ray divergence as the shortest distance between such rays, as depicted on the left image of Fig. 4.1.

Ray directions D_i for the two cameras are calculated per Eq. 4.1, with x_i and y_i being the pixel coordinates in each image. The absolute orientation R_i and position C_i for each of the two cameras is computed by factorizing the essential matrix, which can be computed from feature matches using *N-point* algorithms [30], as explained in Appendix B. The cameras' intrinsic parameters, such as focal length and principal point, with no pixel skew, are assumed to be at least roughly known in order to create each 3×3 matrix K_i, which are equal if the same camera is being used.

$$D_i \;=\; R_i * K_i^{-1} * \begin{pmatrix} x_i \; y_i \; 1 \end{pmatrix}^T \tag{4.1}$$

Given the camera center locations C_i, the shortest distance between the two rays corresponds to the Euclidean distance between the nearest distance points P_i on each ray as shown in Eq. 4.2, with t_i defining the distance to move along each ray. Finally, the ray divergence d can be obtained from $d_i = |P_1 - P_2|^2$.

$$P_i \quad = \quad C_i + t_i * D_i \tag{4.2}$$

It is important to note that the error is directional. In order to get a direction, the cross product of the two direction rays is taken and the resulting vector direction is considered to be the positive direction. The error calculation is performed for every dense correspondence pair across a pair of images. As shown in [41], the total error map consists of a smooth global error superimposed by high frequency errors. The two main error sources are camera-parameter inaccuracies and inaccurate dense correspondences. The important assumption is made that camera parameters introduce a smooth overall error but correspondence errors show up as local, high-frequency errors. The absolute values and also direction of the total ray divergence error is taken into account as it is possible that crossing rays change their spatial order [41]. To separate the two error sources, the camera error is first estimated. For this, a least-squares B-spline approximation to the total ray divergence error map is computed. The B-spline used in the algorithm consists of a 5×5 support point grid, and is commonly referred to as a Bézier Curve. To filter the smooth camera error from the high-frequency correspondence error, a simple pixel-wise subtraction between the total error map and the smooth B-spline surface, corresponding to the camera-parameter errors, yields the correspondence error map.

To show how errors in camera parameters affect ray divergence in a global, smooth manner, a total ray divergence map was obtained for a few test sequences. This error comprises any inaccuracies with the camera poses, intrinsics or radial distortion. Each was factorized into camera-parameter error maps, modelled as smooth B-spline surfaces, and correspondence error maps, composed by the remaining high-frequency components. The resulting camera-parameter error maps are shown in Fig. 4.1.

Signal-to-noise ratio (SNR) analysis

As discussed in the previous section, the camera error is modelled as a B-spline surface, which is a deterministic function. The correspondence error, on the other hand, is the

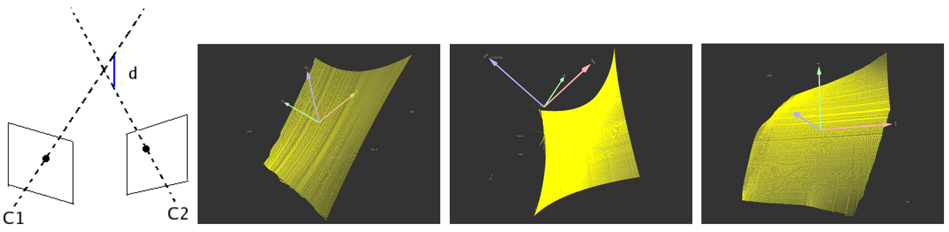


Figure 4.1: Concept of ray divergence d (left), and sample dense camera parameter error maps for image pairs from different datasets (right), to depict their smooth variation.

difference between the overall error and the camera error surface and is non-deterministic. Thus, it can be viewed as noise. For a given image, noise can either be independent or dependent on the image data. It will be assumed that the two errors are independent, since only a subset of the correspondences chosen by RANSAC are used to compute the camera pose, which are not necessarily representative of the entire set of correspondences. If the correspondence error is modelled as independent additive noise, the overall error $f(i,j)$ is the sum of the signal (camera error) $s(i,j)$ and the noise (correspondence error) $n(i,j)$, as shown in Eq. 4.3.

$$f(i,j) = s(i,j) + n(i,j) \tag{4.3}$$

The noise n is approximately zero-mean Gaussian and described by its variance σ_n^2. The relationship between the two sources of error can be described by the signal-to-noise ratio (SNR), which is given by Eq. 4.4, where σ_s and σ_n are the standard deviations of the camera error and correspondence error, respectively.

$$SNR = \frac{\sigma_s}{\sigma_n} \tag{4.4}$$

If there were a dependency between the two sources of error, the noise could be modelled with a multiplicative or non-linear model, but such models are mathematically more complicated, which further justifies assuming independence. Assuming that the 'signal' is the camera error, and that the correspondence error is the 'noise', a high SNR indicates numerically that the camera error is the dominant one, and that some algorithm should be applied to overcome this deficiency. On the other hand, an SNR smaller than one suggests that the correspondences are the main error source and that the main focus should be on correspondence improvement. An SNR around one shows that both error sources have around the same influence.

The signal-to-noise ratio (SNR) representation allows to obtain valuable information. For example, it tells if the correspondences or the camera error are dominant and suggests which of these inputs should be fixed first. In the case of dominant correspondence errors, the worst correpondences based on the error map should be revisited. If the camera error is dominant, the pose estimation should be refined. It also allows for comparison across sequences, where SNR values indicate for which types of scenes (for example flat versus heavily textured) different dense non-epipolar correspondence algorithms are more likely to fail. To prove the validity of the proposed SNR approach, histograms for both the camera error and the correspondence error can be obtained for different sequences. As expected, the camera error histogram approximately follows a smooth distribution. This is inherently true because B-spline surfaces by definition are smooth, and differentiable at any point. The correspondence error histogram, on the other hand, follows an approximately Gaussian distribution, of type $N(\alpha, \sigma^2)$. This has been verified for the given test sequences.

Results summary for dense correspondences

Tests were conducted on a machine with Quad Core CPU @2.66 Mhz and 4 GB of RAM. All results were achieved in a few seconds depending on the size of the input images, as described in [41]. Different real and synthetic datasets were tested. As described in [41], for aerial imagery datasets it is seen that the largest correspondence errors appear in occlusion areas and in areas where there is not enough texture for the correspondence algorithm to lock down the best correspondences. There are also high errors where moving objects appear, such as on roads. The problem is that these objects move from one frame to the other and therefore the correspondences are incorrect. These results demonstrate that problem areas are found by the introduced correspondence error map.

Tests performed with synthetic imagery, where perfect camera positions and correspondences are known, were aimed at proving that the assumption of the smooth camera error is correct and that the extraction of correspondence errors results in a reliable error map. The algorithm was also tested with perfect camera poses but using computed dense correspondences, and it was seen that the main error source by far is due to inaccurate correspondences, as expected and supported by the low SNR value of 0.26, which supports the similarity of the total error map and the correspondence error map. If using both computed camera poses and dense correspondences, the resulting correspondence error map is up to normalization just like the one with the perfect camera poses, which shows a correct separation of camera error from correspondence error, and the high SNR of 4.83 implies that the camera error is dominant in this case. Yet another test was performed to prove that if using perfect correspondences the total error should correspond to the camera error. This turned out to be true, as seen by the high SNR value of 35.7, which implies that essentially all the error is in the camera parameters.

4.2.2 Two-View Ray Divergence Calculation for Sparse Feature Matches

As mentioned, to use ray divergence for bundle adjustment conditioning, as will be detailed in the rest of this chapter, we will assume from now on that we count with a very accurate set of feature matches, and that ray divergence d is then computed for all available feature matches. In Knoblauch et at. [41], the resulting set of divergences corresponds to the total reconstruction error which is a function of both feature matching errors and camera-related errors, but the main difference now is that we assume the entire error corresponds to the cameras given that feature matching is very accurate. This turns out to be a good approximation even if there are small matching errors. Matches will never be perfect in reality, but we filter bad matches through RANSAC on the epipolar geometry, using a $3.84\sigma^2$ inlier threshold on Sampson error [30], or assuming that the set of initial features has been evaluated using Zeisl's metric [105] such that only those considered reliable for matching are left.

Therefore, we can say that ray divergence d_i for a given feature match is a function of relative rotation between the two cameras R_{rel}, relative translation T_{rel}, intrinsic parameters for the two cameras K_1 and K_2, and radial distortion, which we'll represent as distorted pixel coordinates (x_{ri}, y_{ri}), such that $d_i = f(R_{rel}, T_{rel}, K_1, K_2, x_{ri}, y_{ri})$.

Starting with sparse features, a smooth but sparse set of surface points is obtained as shown in Fig. 4.2 for the *Palmdale* dataset, which shows grayscale-coded ray divergence values for all available matches. In general, it has been observed that the highest diver-

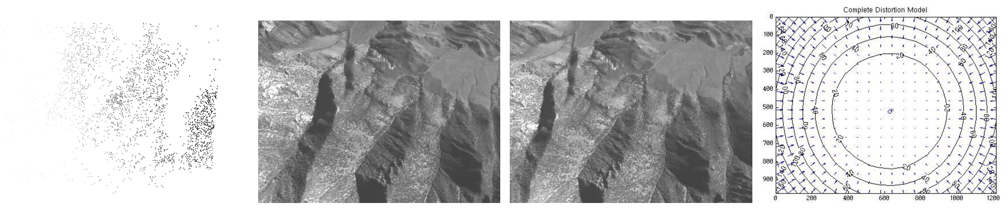


Figure 4.2: Ray divergences (left) for the set of matches from a pair of *Palmdale* dataset images (middle), displayed such that lighter colors indicate higher divergences. The true radial distortion map for the used camera, in pixels, is also displayed (right).

gences tend to occur towards the edges of images, as seen in Fig. 4.2, where most matches are on the left-hand side of the images, in part because of radial distortion, and it becomes clear that we want such matches to have less of an influence in bundle adjustment because of their higher ray divergence, as discussed further in Subsection 4.2.3.

4.2.3 Bundle Adjustment Weighting with Ray Divergences

Now that ray divergences have been computed, and assuming that these are a function mainly of camera parameter inaccuracies, it will be shown how these values can be used as input weights to bundle adjustment in order to improve its convergence properties. However, one further step before applying bundle adjustment is to obtain initial estimates for the scene's structure. We use Lindstrom's triangulation algorithm [45] due to its superior accuracy and speed with respect to standard linear triangulation [30].

Weighted bundle adjustment

The objective of bundle adjustment is to adjust pose and structure estimates in such a way that the total reprojection error of the 3D points with respect to their corresponding 2D feature track positions in each camera is minimized [30]. The cost function which is traditionally minimized can be expressed as the sum of squares of the reprojection error between each 3D point and the feature matches which yielded it, as explained in detail in Appendix E. The *SBA* implementation [47] was used to perform the bundle adjustment, since it exploits the sparse block structure of the normal equations solved at each iteration to greatly speed up the process.

The Levenberg-Marquardt algorithm is based on solving the ‘augmented normal equations’ at each iteration. In *weighted* bundle adjustment, each input feature is weighted differently with the objective of improving convergence by giving less weight to those features that are more likely to be inaccurate. In practice, these weights are implemented as covariances. The normal equations have the form shown in Eq. 4.5, but when using weighted bundle adjustment, the equations change to the form shown in Eq. 4.6, where Σ corresponds to a block-diagonal matrix consisting of 2×2 covariance matrices for each input feature, J is the parameter Jacobian matrix, δ_p the parameter update step, μ the damping term and ϵ the error vector.

$$(J^T J + \mu I)\delta_p = J^T \epsilon \tag{4.5}$$

$$(J^T \Sigma_x^{-1} J + \mu I)\delta_p = J^T \Sigma_x^{-1} \epsilon \tag{4.6}$$

Comparison with reconstructed point confidence ellipsoid roundness

Before proceeding, we wish to analyze the validity of ray divergence as a measure of camera errors, such that it can aid in bundle adjustment. Beder et al. [7] present an algorithm to determine the best initial pair for a multi-view reconstruction. Their analysis is based on computing a confidence ellipsoid for each computed 3D scene point X, such that its roundness measures the quality of each obtained point. For two views, the covariance matrices of image feature matches x' and x” are given by C' and C” respectively. Then, the covariance matrix C_{XX} of the distribution of the scene point coordinates X is proportional to the upper left 4×4 sub-matrix $N^{-1}_{1:4,1:4}$ for the inverse of the 5×5 matrix N given by Eq. 4.7. The A and B matrices encode information related to the projection matrices for the two cameras, the image coordinates of the feature match yielding the scene point, and the 3D point coordinates.

$$N = \begin{pmatrix} A^T \left(B \begin{pmatrix} \text{C'} & 0 \\ 0 & \text{C"} \end{pmatrix} B^T \right)^{-1} A & X \\ X^T & 0 \end{pmatrix} \tag{4.7}$$

Now, if the homogeneous vector $X = [X_0^T, X_h]^T$ is normalized to Euclidean coordinates, the covariance matrix of the distribution of the Euclidean coordinates is given by Eq. 4.8, where J_e corresponds to the Jacobian of a division of X_0 by X_h.

$$C^{(e)} = J_e C_{XX} J_e^T \tag{4.8}$$

Finally, if we perform the singular value decomposition of the matrix $C^{(e)}$, the roundness R of the confidence ellipsoid is obtained as the square root of the quotient of the smallest singular value λ_3 and the largest singular value λ_1, per $R = \sqrt{\frac{\lambda_3}{\lambda_1}}$. The value of R lies between 0 and 1, and only depends on the relative geometry of the two poses, the feature positions and the 3D point; radial distortion is not modeled.

Something very important to note here is that image feature covariances [8, 105] are defined completely by the intensity variations in local neighborhoods and thus may look rather random to visual inspection, with no clear pattern as the image is traversed, as seen on the right in Fig. 4.2. On the other hand, the surface of ray divergences has a much smoother shape, which is a function of all camera parameter inaccuracies. So if we filter out all features that have high image covariances, matches obtained between remaining 'good' features are still bound to the information ray divergence provides, in order to know if they are overall good or bad matches for reconstruction purposes. This is the power of using ray divergence to weight bundle adjustment, since it provides information beyond just the feature matching uncertainty. For example, two *perfect* matches could still yield a non-zero ray divergence due to camera inaccuracies. Therefore, using ray divergence or even the values provided by Beder et al.'s metric [7], though more expensive to compute and not inclusive of radial distortion, provide a stronger constraint towards weighting bundle adjustment than image-based covariances [8, 105]. The right side of Fig. 4.3 shows the result of applying Zeisl's image covariance metric [105] on a select group of SIFT features, displayed as ellipses with size proportional to covariance values. The left side shows the smooth transitions in values for Beder et al.'s confidence ellipsoid roundness [7] using identity image feature covariances, and the middle shows its correlation with ray divergence. Though it is not an exact correlation because of differences near the edges of images, where the behavior is slightly different, the bulk of points show a very good correlation (a coefficient of 0.93 for the main linear part of this particular plot), such that higher divergences, in absolute value, exhibit lower roundness.

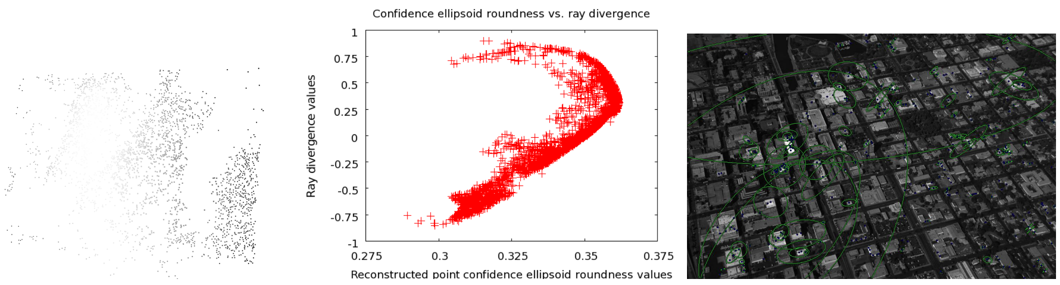


Figure 4.3: Reconstructed point confidence ellipsoid roundness values using identity image feature covariances (left) for the set of matches from a pair of *Palmdale* dataset images, where lighter colors indicate lower confidence values. The middle image shows its correlation with ray divergence. The right image displays Zeisl's covariance metric values [105] for SIFT features in a *Stockton* image as green ellipsoids.

Gaussian weighting

A close look at a ray divergence histogram reveals a smooth curve, typically reaching a maximum near zero. If we assume that the probability $p(d)$ that a given feature match exhibits a ray divergence d is given by Equation 4.9, where μ_d corresponds to the mean ray divergence and σ_d to its standard deviation for a given two-view set of feature matches, we can essentially assume that ray divergence histogram values follow a Gaussian probability density function (pdf) and use these values as weights for bundle adjustment. The average and standard deviation are computed directly from the ray divergences for the available set of feature matches. Since these weights must be input as 2×2 covariance matrices, we assume an isotropic probability distribution and set the diagonal elements with equal pdf-based values, while setting the remaining two elements to zero. It is very important to note that we want to penalize low pdf values since these correspond typically to higher divergences. Therefore, we 'invert' the pdf values and place this number along the diagonal; their original values are obtained again later from matrix inversion while solving the augmented weighted normal equations. This results in higher covariances providing lower weights.

$$p(d) \quad = \quad \frac{1}{2\pi\sigma_d^2} e^{\frac{|d-\mu_d|^2}{2\sigma_d^2}} \tag{4.9}$$

The advantages of using Gaussian values as weights is that positive weights are always obtained, no matter what the divergence values are or if they show zero-crossings. The area under the computed Gaussian curve is always unity, by definition, and this is helpful towards mathematical stability since very large variations between the smallest and largest assigned weights is not typical. Also, exponentials are much cheaper to compute than for example a singular value decomposition, as needed in Beder et al.'s algorithm [7]. Finally, ray divergence transitions are smooth such that high ray divergences should be assigned higher covariances than lower ones.

4.2.4 Extension to Multiple Views

The extension to multiple views is rather simple, and is based directly on the two-view case. In a sequential multi-view pipeline, since covariances have to be specified as 2×2 covariance matrices for each feature of a given feature track, for each feature in a new image we simply assign the Gaussian-based weight corresponding to the ray divergence for the feature's match to the prior image. Average and standard deviation are obtained from the set of pairwise matches between the two most recent images, in order to compute the pdf prior to computing each individual weight. Covariances for the features in the very first image can be initialized to identity, or by computing them from images [8, 105] for better initial accuracy. This way of chaining pairwise consecutive estimates works well no matter what the number of frames as long as pairwise ray divergence estimates are obtained for 'good' baselines, neither too small nor too large, which can usually be achieved through a prior *frame decimation* [40]. This process will be explained in detail in Chapter 5. An analysis of this baseline effect on divergences is discussed in Section 4.3. For non-sequential cases, the average of all ray divergence values for all matches to a given feature could potentially be used, though we have yet to test this case.

4.3 Results

The algorithm was tested on real scenes such as *Stockton*, *Palmdale*, *castle-P19* [85] and *Medusa* [68], as well as synthetic scenes such as *Megascene1* and *Coneland*. All tests

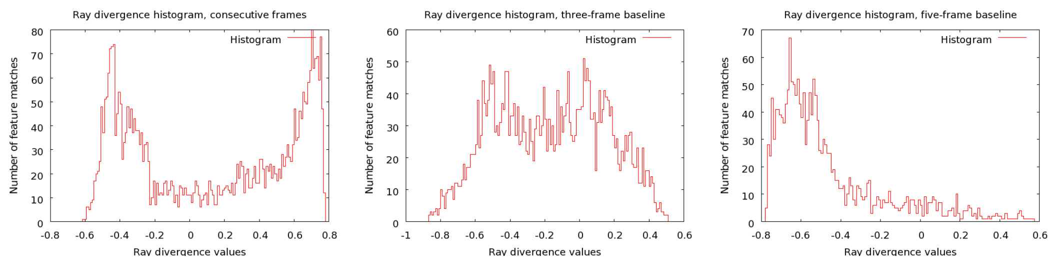


Figure 4.4: Ray divergence histograms at increasing baselines from left to right, for pairwise frames from the *Stockton* dataset.

were conducted on a single-core *Intel Xeon* machine at 2.80GHz with 1 GB of RAM, on one thread. For all tests, we assume that the same camera is used per dataset and have initial values available for the focal length and principal point, though these in some cases were inaccurate. Images were not undistorted prior to testing, and were acquired sequentially.

One important initial experiment consisted in analyzing the behavior of ray divergence given different baselines. For this, we started out with one frame of the *Stockton* sequence and then obtained ray divergences at different baselines from that particular frame. In Fig. 4.4, results show that Gaussian fitting works well for 'good' baselines, which are typically achieved by applying frame decimation [40] or other choosing algorithms [81] such that the baseline is not too small since linear triangulation instability and pose estimation degeneracies occur or too large since feature matching errors are more likely. This was also verified in several other datasets. The middle image shows the most smooth histogram, and that is where frame decimation picked the best keyframe. In general, with good baselines ray divergence histograms are smooth and can generally be approximated well by Gaussian fitting. With other baselines, ray divergences would not be suitable for Gaussian fitting and therefore for bundle adjustment, since the values are more heavily affected by noise. A good frame decimation is key to our algorithm's success. Table 4.1 shows the reprojection error and processing time results for different baselines, where it is shown that the chosen frame decimation keyframe at a baseline of three frames yielded the lowest reprojection error and processing time per point.

In the next experiment, we compared processing times and reprojection errors obtained using weighted bundle adjustment under four different conditions: bundle adjust-

Table 4.1: Number of points, final total reprojection error R (pixels), bundle adjustment iterations I, processing time t in seconds and min/max ray divergence for Gaussian-pdf ray divergence-weighted bundle adjustment at different baselines, for the *Stockton* dataset. The best results were obtained for the keyframe selected by a previous frame decimation, which has a three-frame baseline.

Baseline	Points	R	I	t	min_d	max_d
Consecutive	3605	0.049	150	4.24	-0.606	0.774
3 frames	3369	0.013	33	0.83	-0.863	0.508
5 frames	1831	0.200	73	0.87	-0.774	0.561
8 frames	476	0.111	30	0.09	-0.297	0.537

ment weighted by image feature covariances [8], by confidence ellipsoid roundness with and without including image feature covariances, and based on ray divergences. This was only performed on 'good' two-view baselines, obtained with prior frame decimation. Table 4.2 shows the results for some test datasets. Average values for all test parameters were obtained across pairwise frame analysis for all consecutive pairs of each dataset. Unweighted bundle adjustment was not compared, since the comparison would not be direct. Time is consumed by the *SBA* software [47] to read-in covariance data, and there is matrix inversion for covariance matrices and multiplication of these with Jacobian matrix elements at each iteration, so processing times are typically higher when using covariances. Even so, our bundle adjustment weighting outperforms unweighted bundle adjustment as far as final reprojection error in almost every case, as seen on the right in Fig. 4.6 where NBA represents the unweighted case. It can be seen that ray divergence-based weighting outperforms every other type of weighting in just about every category, though it's slightly slower and with a higher reprojection error than the more-expensive $UIBA$ in a few cases. However, overall our weighting scheme provides the best combination of processing time, final reprojection error and computational complexity in computing weights. As far as complexity, Beder's algorithm ($UWBA$ and $UIBA$) for example includes the inversion of a 5×5 matrix and two singular value decompositions of a 4×4 matrix and a 3×3 matrix, whereas ray divergence

Table 4.2: Iterations I, final total reprojection error R (pixels) and processing time t (seconds) in (I, R, t) format obtained using bundle adjustment under four different weighting schemes: image feature covariances (CBA), reconstructed point confidence ellipsoid roundness with ($UWBA$) and without ($UIBA$) including image feature covariances, and Gaussian-pdf with ray divergences ($RDBA$).

Dataset	CBA	$UWBA$	$UIBA$	$RDBA$
Stockton	$43, 0.621, 0.90$	$40, 0.171, 0.84$	$37, 0.072, 0.79$	$38, 0.015, 0.78$
Palmdale	$23, 4.687, 0.45$	$22, 1.692, 0.38$	$20, 0.831, 0.41$	$22, 0.113, 0.37$
castle-P19	$150, 281.13, 0.99$	$150, 4150, 0.95$	$150, 1046.1, 0.88$	$97, 90.036, 0.62$
Dinosaur	$26, 2.631, 0.06$	$22, 0.286, 0.05$	$24, 0.09, 0.05$	$24, 0.162, 0.05$
Megascene1	$49, 12.14, 0.04$	$42, 0.179, 0.03$	$45, 0.074, 0.03$	$46, 0.124, 0.04$
Coneland	$150, 28052, 1.10$	$150, 1880.38, 0.99$	$115, 599.88, 0.79$	$126, 81.86, 0.90$

computation does not involve SVD or inversions at all. The feature covariance method CBA is also more expensive, requiring multiple exponential evaluations for each covariance matrix, whereas our method computes a single exponential value.

Having proven that the algorithm performs very well on pairwise reconstructions, it was also applied as explained in Section 4.2.4 to perform multi-view reconstructions using our sparse multi-view reconstruction pipeline, which is described in Chapter 6. Fig. 4.5 shows on the top row sparse reconstructions that were obtained while applying sequential multi-view reconstruction, bundle-adjusting with each added image using ray divergence-based weighting. These high-quality sparse reconstructions allow for other algorithms to be applied, such as dense reconstructions with the PMVS algorithm [21] as shown on the bottom row of Fig. 4.5. Fig. 4.6 shows the effect on scene reconstruction of using original distorted images versus versions that were undistorted using parameters recovered per our weighted bundle adjustment, for the *Palmdale* dataset.

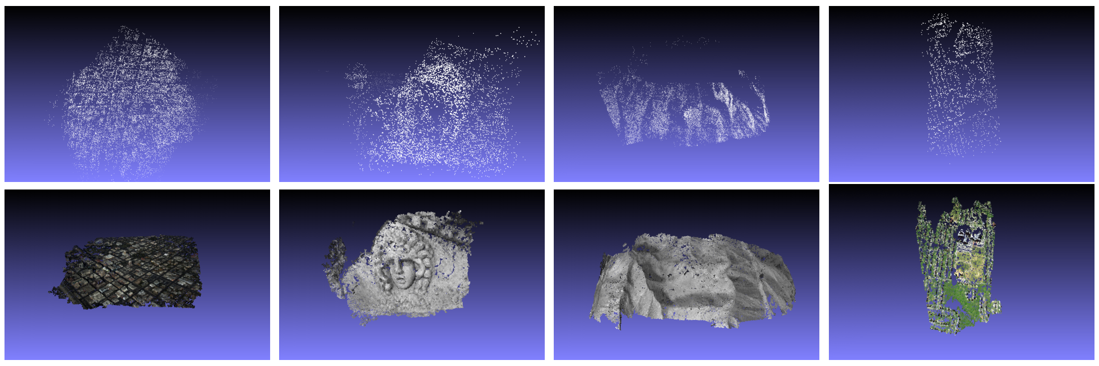


Figure 4.5: Top row: sparse multi-view reconstructions for the *Stockton* (left), *Medusa* (middle left), *Palmdale* (middle right) and *Megascene1* (right) datasets. Their respective dense reconstructions using the PMVS algorithm [21] are shown on the bottom.

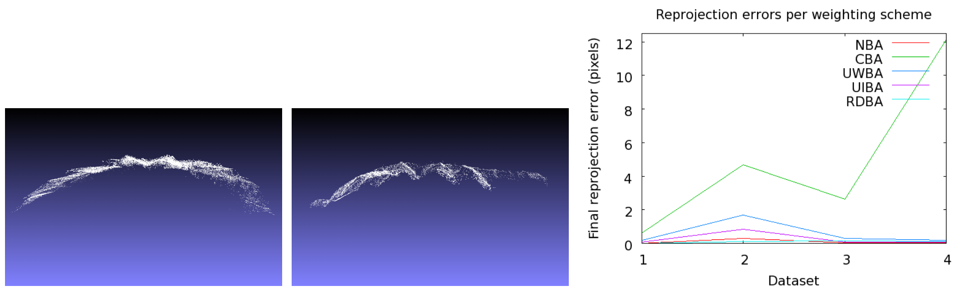


Figure 4.6: Side view of a multi-view reconstruction showing the effect of using distorted images (left) versus undistorted images per our algorithm (middle), for the *Palmdale* dataset. Total reprojection errors are lower than with other weighting schemes (right), as shown for a few datasets.

4.4 Conclusions

An algorithm that makes use of scene reconstruction ray divergence for weighting bundle adjustment and improving its convergence properties was introduced. It was shown that ray divergence, which is a function of all camera parameter inaccuracies, is more efficient to compute and outperforms other weighting schemes such as those based on image feature covariances. There is no dependence on ground-truth information, and results show an improved convergence on different real and synthetic scene types.

Chapter 5

Non-Parametric Sequential Frame Decimation

The previous chapters discussed methods for detecting and correcting errors in dense feature matching, camera parameter estimation and structure estimation. Even though those methods help to alleviate some of the issues seen with scene reconstruction in general, in the specific case of using dense correspondences, because of sequentially adding-in error-prone correspondences the resulting pose and structure estimation errors could be reduced mainly by using dense bundle adjustment, and this makes the reconstruction process intractable for scenarios where real-time processing of very large images and image sets is required.

For this reason, we began working in a fundamentally different way. Instead of starting from dense correspondences, it was decided to work with a very sparse yet accurate set of feature matches to obtain accurate camera extrinsic and intrinsic information, along with a sparse scene structure, and leave the dense reconstruction process as the very last step. The remaining chapters in this thesis will describe this framework in detail, keeping in mind that the algorithms already presented in Chapters 3 and 4 could still be used within this framework.

A very essential pre-processing step for multi-view reconstruction, and one which had not been dealt with previously in our work due to the fact that small amounts of images were being used, is how to choose *which* frames to use if reconstructing from a large number of images or from streaming video. Such extended reconstructions become possible if using accurate sparse feature matching. This process is known in the literature as 'frame decimation', and there are a surprisingly small number of algorithms to do this in the literature. The goal of frame decimation is not only to reduce the computational load but also discard frames that could possibly lead to inaccurate feature matching, pose degeneracies and mathematical instability in structure computation. To this end, a method is presented for non-parametric sequential frame decimation for image sequences in low-memory streaming environments. A detailed explanation is found in Knoblauch et al. [40]. The main contribution of this work is the introduction of a sequential low-memory work-flow for frame decimation in embedded systems where memory and memory traffic are limited. Such an online pre-processing filter removes frames that are ill-posed for reconstruction before streaming. The method will now be explained in detail.

5.1 Introduction

There exist a great number of different algorithms to achieve multi-view reconstruction in the literature, as was shown in the Introduction [81, 68, 55]. Basically all algorithms begin by extracting feature matches [49, 5] between frame pairs, to then compute the epipolar geometry, poses and scene structure. However, as part of the automation of such algorithms, it has become a challenge to find 'good' image pairs for pose and structure estimation. There are a few algorithms in the literature to achieve such 'frame decimation', as listed in Knoblauch et al. [40]. Nistér [56] proposed a frame decimation algorithm based on global motion estimation between frames and a sharpness measure to remove redundant frames. Ahmed et al. [1] proposed a frame decimation algorithm based on number of correspondences, the 'geometric robust information criterion' ($GRIC$) [94] and a point-to-epipolar line cost between frame pairs. Royer et al. [76] introduced a simple sequential frame decimation algorithm for robotic applications. Their frame decimation decision is based on the number of available correspondences between keyframes, and tries to decimate as many frames in-between keyframes without going below an empirically chosen number of correspondences. Torr et al. [95] use their previously introduced GRIC approach to improve the correspondence track extraction over several frames by analysing if the epipolar geometry or a homography is a better motion model for the given frames. The active search algorithm proposed by Davison [14] performs a global analysis of frames to decide which one adds the most new information to a multi-view reconstruction. All of these approaches either analyze frames in a global manner, after they have all been acquired, and/or rely on thresholds that make them scene-dependent. It is clear that for streaming environments buffering many frames in memory in order to perform such a global analysis is not suitable. Also, we seek a general algorithm that can work on any type of scene, and independently of the camera movement.

To this end, the main contribution of this work is to provide a low-memory, sequential and threshold-independent approach to frame decimation, such that good 'keyframes' for pose and structure estimation are filtered to a sequntial multi-view reconstruction pipeline, and the rest are discarded. The cost function used for frame decimation is based on a weighted version of Torr's GRIC criterion, as will be explained in the remainder of the chapter, where weights are derived from an analysis of feature matching and such that one global maxima is obtained at each keyframe evaluation step.

5.2 Proposed Algorithm

Fig. 5.1 shows the flow chart for the proposed approach [40]. Given a keyframe k, the first step is to obtain feature matches with respect to the present candidate frame, $k+i$. A cost function, fG, is then evaluated. Feature matches are obtained based on SURF [5] features. The cost function continues to be evaluated for subsequent candidate frames as long as the cost function's value increases, such that $fG(k, k+(i-1)) <= fG(k, k+i)$. When a frame pair for which the value of fG decreases with respect to the last, and is a positive value, the *previous* frame is chosed as the next keyframe. The first suitable frame pair for the sequence is initialized using Beder and Steffen's algorithm [7]. The process then repeats, and all extracted keyframes can be used for pose and structure estimation, while discarding all others. Notice that at most three frames, being the current keyframe and

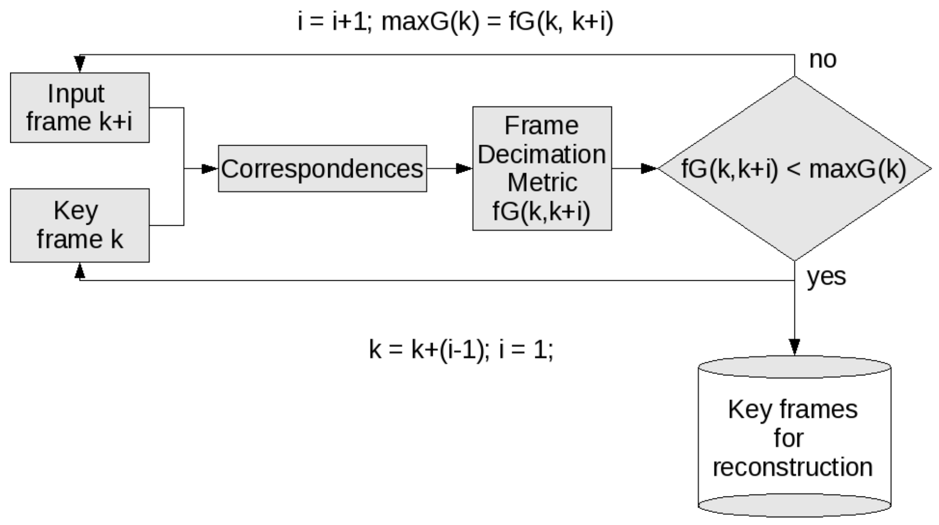

i = i+1; maxG(k) = fG(k, k+i)
no
Input frame k+i
Key frame k
Correspondences
Frame Decimation Metric fG(k,k+i)
fG(k,k+i) < maxG(k)
yes
k = k+(i-1); i = 1;
Key frames for reconstruction

Figure 5.1: Flow chart for the proposed sequential frame decimation algorithm [40].

current and last candidate frames need to be kept in memory.

5.2.1 Camera Pose Degeneracy and Structural Instability Detection

Because of the way the cost function is designed, as will be explained, the framework filters frames that can cause pose estimation degeneracies as well as avoiding small baselines, where linear triangulation suffers from mathematical instability due to decomposition of near-singular matrices. At the same time, the baseline is kept small enough to allow for accurate feature matching. With longer baselines, occlusions and lighting changes for example can have a negative impact on any matching process.

There are two cases for pose estimation degeneracy, known as the motion degeneracy and the structure degeneracy. The motion degeneracy arises when the relative movement between frames consists of only a rotation and no translation. In such cases, the epipolar geometry between the views is undefined. The structure degeneracy occurs when matches between frames correspond to scene points that lie on the same plane in space. In this case the epipolar geometry is not unique, as there is a two-parameter family of possible solutions. However, in both of these cases a homography mapping between the two frames can still be achieved, and used to describe the relative change in the scene. Notice also that in these same scenarios where homography fitting is good, numerical errors in linear triangulation for structure computation are more likely to occur. A metric known as Torr's 'geometric robust information criterion' (GRIC) [94] is a useful tool that allows a comparison between the quality of an epipolar geometry fit to the data versus a homography fit. The GRIC criterion is based on the goodness of fit of a model and also on its parsimony, which is basically a penalty favoring lower-dimensional models. The lower the value for GRIC, the better the model fits the data, be it an epipolar geometry fit or a homography fit. Therefore, the relative comparison between the two model fits, which we call relative GRIC or *relGRIC*, provides information about the quality of a frame pair for pose and structure estimation. The expression for $GRIC(X)$ is defined by Eqs. 5.1 and 5.2, while *relGRIC* is defined as in Eq. 5.3, where H stands for a 3×3 homography matrix and F to a 3×3 fundamental matrix for epipolar geometry.

$$GRIC(X) = \sum_i \rho(e_i^2)_i + \lambda_1 dn + \lambda_2 k \tag{5.1}$$

$$\rho(e_i^2)_i = min(\frac{e_i^2}{\sigma^2}, \lambda_3(r-d)) \tag{5.2}$$

$$relGRIC(F,H) = \frac{GRIC(H) - GRIC(F)}{GRIC(H)} \tag{5.3}$$

The goodness of fit consists of the sum of squared residuals e_i of either F or H with respect to the input feature matches. The parsimony is based on d, the number of dimensions modeled ($d = 3$ for F and $d = 2$ for H), k, the number of degrees of freedom in the model ($k = 7$ for F and $k = 8$ for H), r, the dimension of the input data, which corresponds to $r = 4$ in the case of 2D correspondences, σ^2 is the variance of the residual errors, and similarly to Ahmed et al. [1] we set $\lambda_1 = log(r)$, $\lambda_2 = log(rn)$ and λ_3 corresponds to a limit for the residual error.

If the value of *relGRIC* is low or negative, it means that a homography fit has lower errors, and in these cases pose degeneracies or structural instability can occur. Therefore, a frame pair can be considered 'good' for pose and structure estimation the higher the value for *relGRIC*.

5.2.2 Weighting Based on Feature Matches

The value of *relGRIC* provides information about pose degeneracy and structural instability, and its value tends to increase with the baseline. However, this is true as long as there are enough good matches available to reliably compute the epipolar geometry, and this is not possible with very large baselines, where less good matches are obtained due mainly to occlusions and lighting changes. To take this important factor into account, a weighting term *cW* for *relGRIC* had to be created, which takes into account the amount and image layout of the matches used for estimating *relGRIC*. A good measure for the size of the baseline is the ratio between the number of feature matches of the source keyframe with the present frame, N_C, and the number of features in the source keyframe, N_F. To take into account only good matches, N_C was replaced by N_I , which is the number of inlier matches from the RANSAC-based fundamental matrix calculation. The expression for *cW* is shown in Eq. 5.4.

$$cW = \frac{N_I}{N_F} \tag{5.4}$$

As the baseline grows, the value for *cW* tends to decrease. To additionally ensure that the matches cover as much of the scene as possible, such that a better representation of the scene can be obtained, another weight was introduced. This weight *aR*, shown in Eq. 5.5, is a ratio between the feature match area *cA* and image size *iA*, such that *cA* is the area of the axis-aligned bounding box of all inlier matches. This ratio should remain as high as possible in order to better represent the scene.

$$aR = \frac{cA}{iA} \tag{5.5}$$

5.2.3 Frame Decimation Cost Function

The *GRIC* value and feature match-based weighting can be combined to obtain the final cost function for decimation. Keeping in mind that we seek positive values for *relGRIC* as more suitable for pose and structure estimation but without using too large of a baseline, where feature matching becomes more inaccurate, the cost function fG is simply a multiplication of the two terms, as shown in Eq. 5.6.

$$fG = (cW * aR) * relGRIC(F, H) \tag{5.6}$$

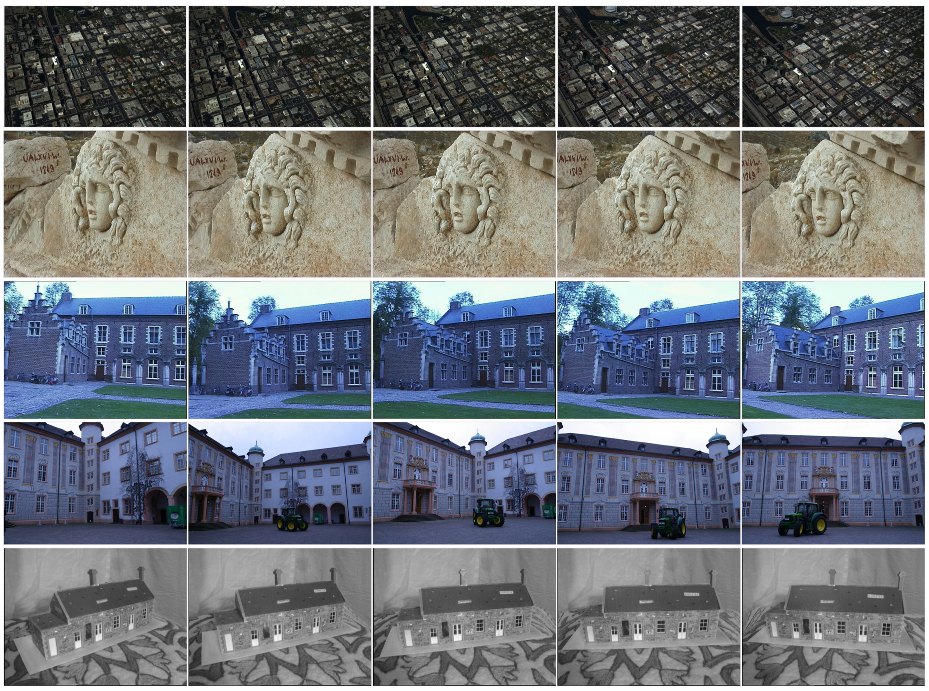


Figure 5.2: First five keyframes extracted by the proposed frame decimation [40] for different datasets. From top to bottom: *Stockton*, *Medusa* [68], *Leuven Castle* [68], *castle-P30* [85] and *Model House* [19].

It can be seen that fG will have a high value if $relGRIC$ has a high value and the match-based weighting is also high, but will decrease as the baseline grows. This provides an adequate baseline size that minimizes errors from feature matching and pose estimation.

5.3 Results

The proposed frame decimation algorithm was tested on real and synthetic scenes of different types and with different camera motions. Detailed results are given in Knoblauch et al. [40], but the main results will be summarized. Examples of decimated sequences can be seen in Fig. 5.2. All the different scene types and camera motions are decimated well by the proposed algorithm, including the case of the *Model House* sequence [19], which had been spaced before-hand to allow for good baselines. In this case, decimation did not discard any frames, as expected due to the good nature of the input frames for pose and structure estimation.

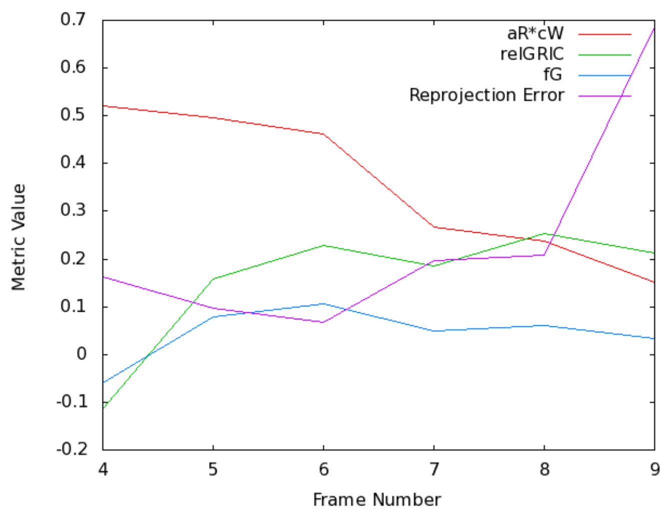


Figure 5.3: Cost function fG [40] and its components (aR*cW, *relGRIC*) vs. reprojection error for pairwise reconstructions with respect to frame $k = 0$ of the *Stockton* dataset.

An evaluation of the cost function fG was carried out, as shown in Fig. 5.3, where fG, *relGRIC* and the match-based weights are plotted against reprojection error for a set of frames from the *Stockton* dataset with respect to the first keyframe, at $k = 0$. The value for fG is highest at frame $k + 3$, which is also where the lowest reprojection error is seen, and that is where the next keyframe was chosen to be. Additionally, it can be seen that frame $k + 1$ has a negative fG value and is therefore prone to structural instability, as evidenced by the high reprojection error. Also, the highest value for *relGRIC* is obtained at frame $k+5$, but this frame was not chosen as the keyframe since the match-based weighting, which is monotonically decreasing, makes the final fG value lower than at frame $k + 3$, and this way a large baseline is avoided. After performing frame decimation for the entire sequence, 38 percent of the frames remained and the rest were discarded.

A similar analysis was performed for the *Medusa* dataset [68], where 23 percent of frames were extracted. More frames are decimated in this scene than for *Stockton* since the camera movement seems to stall at times with respect to the distance to the scene, with very small consecutive baselines during those moments. In general, comparing values of *relGRIC* and fG shows that with small baselines, such that a large portion of the scene is covered by each and every frame, the value of *relGRIC* is the dominant part of the cost

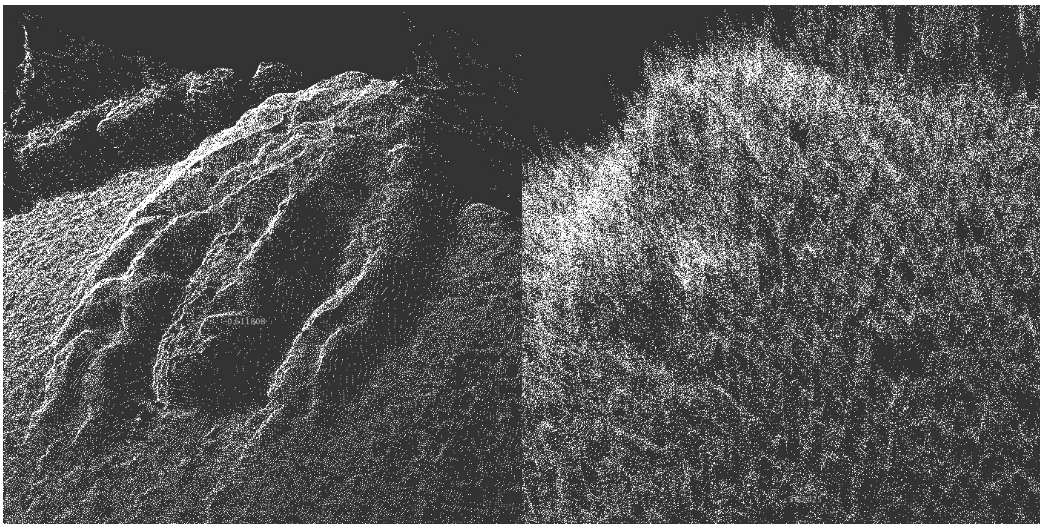


Figure 5.4: Reconstructions between chosen keyframe pair $1-13$ (left) and low reprojection error pair $1-5$ (right) for the *Medusa* dataset.

function fG. However, the match-based weights have a much greater influence over the value of fG when there are larger baselines between consecutive frames and therefore less overlap between frames.

A further analysis of results for *Medusa* shows that low reprojection errors were obtained for frames where the baseline with respect to the current keyframe was very small and therefore the resulting value for fG was negative. This can happen with small baselines since structure computation suffers from numerical instability in such cases, especially when using linear triangulation, but low reprojection errors are possible since noisy computed positions can still lie along rays that reproject close to the input feature matches. This effect can be seen in Fig. 5.4, where the reconstructions between frames $1-13$, where the keyframe was chosen, and $1-5$ both have low reprojection errors but the reconstruction between frames $1-5$ is very noisy. This makes it clear that chosen keyframes should yield low reprojection errors with respect to the past keyframe but that reprojection error by itself should not be used as the cost function for the frame decimation decision.

Another experiment consisted in analyzing whether or not the amount of obtained keyframes is independent of the input frame rate. To this end, the *Medusa* [68] video sequence was decoded into 5, 10, 15 and 20 fps and keyframes were extracted for each case.

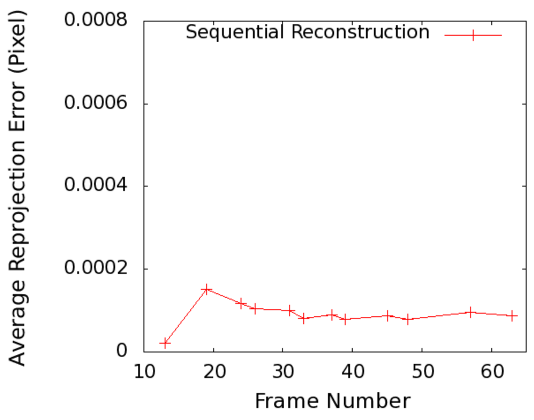


Figure 5.5: Reprojection error during sequential reconstruction with decimated frames. The crosses represent keyframes and the corresponding reprojection error with every newly-added keyframe is plotted [40].

Results show that the number of keyframes remains nearly constant, with variations due to the fact that decimation is non-unique (locally optimal) and also because frames that would be possible keyframes are not even present in the lower decodings.

Finally, to show that keyframes chosen with the proposed decimation are well-suited for pose and structure estimation, starting with a two-view reconstruction, keyframes were sequentially added into the reconstruction and the average reprojection error after each addition was evaluated. Results show that the average reprojection error in pixels after addition of every extracted keyframe remains very small and constant, as shown in Fig. 5.5, where keyframes are represented by crosses and remaining frames were discarded. This shows that degenerate camera poses, structural instabilities and large baselines were all avoided.

5.4 Conclusions

Our work in Knoblauch et al. [40] introduced a non-parametric sequential frame decimation algorithm for scene reconstructions in low-memory streaming environments, which reduces the number of input images for pose and structure estimation by discarding frames that lead to pose degeneracies, structural instabilities or inaccurate feature matching.

The main contribution is a sequential algorithm based on evaluating a cost function which reaches a global maximum representing the next keyframe at each evaluation step. At most three frames are needed in memory at a time when making the decimation decision, which is independent of thresholds or assumptions about the scene. The cost function is a weighted version of Torr's 'geometric robust information criterion' (GRIC), which compares residual error between epipolar geometry and homography fitting to a set of feature matches. The approach was proven to perform well with different scene types and camera motions, as evidenced by a reprojection error analysis using extracted keyframes.

Chapter 6

Multi-View Pose and Sparse Structure Estimation

Chapter 2 described in detail the process of two-view reconstruction from dense correspondences. Chapter 5 described a procedure to obtain keyframes that are appropriate for pose and structure estimation from a video stream. Assuming that a set of keyframes has been extracted, and using the building blocks of scene reconstruction detailed in Chapter 2 as well as the Appendices, the goal of this chapter is to describe the implementation of a multi-view sequential reconstruction pipeline. The major difference with respect to the procedure described in Chapter 2 is that sparse yet very accurate feature matching is used instead of dense correspondences, allowing for much longer reconstructions and with much faster processing times, since bundle adjustment acts over a much more reduced set of structure parameters and the input values to bundle adjustment are closer to their global maximum given the better initial pose and structure estimates obtained from the accurate feature matches.

There are a number of possible ways to achieve a multi-view reconstruction in the literature, and many methods are presented in the Introduction. Some representative examples are given here. For short sequences, one possible method is to compute all pairwise poses for each camera with respect to the first, apply 'N-view' linear triangulation and a final global bundle adjustment. For longer sequences, it is possible to obtain subsequence

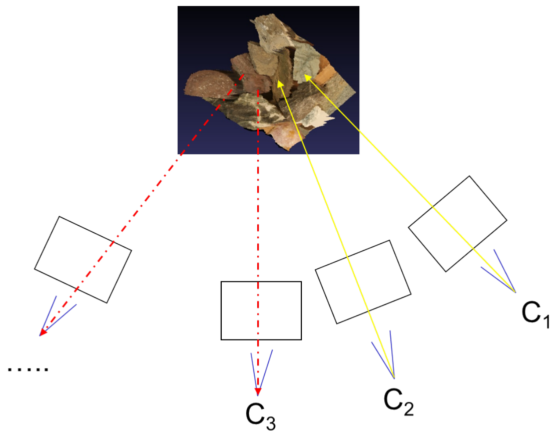


Figure 6.1: Sequential addition of new cameras and points to an existing reconstruction, applying bundle adjustments for fine-tuning.

reconstructions, and finally merge all into one through similarity transformations. The use of Kalman and particle filters has also been explored, and this process is explained later on in this chapter. Another method is to begin with an initial reconstruction involving only two or three images, and then sequentially add-in new cameras and points, applying bundle adjustments for fine-tuning. New projection matrices and poses can be initialized using the 'Direct Linear Transformation', as explained in Appendix F. This process is illustrated in Fig. 6.1. Due to the very good results we obtained in initial tests, this is the general methodology we chose for multi-view reconstruction. The steps involved are detailed in Section 6.1, and the obtained results are shown in Section 6.2. Finally, Section 6.3 discusses other methods in the literature for sequential multi-view pose and structure estimation.

6.1 Multi-View Sparse Reconstruction Pipeline

A complete and automated sequential reconstruction pipeline will now be described. The building blocks for reconstruction, such as for example pose estimation (Appendix C), structure computation (Appendix D), incorporation of new views (Appendix F) and bundle adjustment (Appendix E) are all explained in more detail in the Appendices and also in Chapter 2. Therefore, here only the main steps are listed. There are two main components, the first which corresponds to computing an initial two-view scene reconstruction, and the second is the incorporation of new images into the initial reconstruction.

Initial two-view reconstruction:

1. Using the proposed frame decimation algorithm (Chapter 5), find the best next keyframe starting from the first image of the sequence.
2. Compute SIFT [49] matches between the image pair.
3. Compute the epipolar geometry using the *Normalized 8-point* algorithm embedded in RANSAC with outlier removal (Appendix B).
4. Compute the relative pose between the two cameras (Appendix C) and an initial scene reconstruction using 'fast triangulation' [45].

For each new keyframe, starting from the third:

1. Compute SIFT features for the current keyframe and inlier matches with respect to the previous keyframe.
2. Initialize new feature tracks or continue existing ones with the new matches.
3. Compute the projection matrix for the new keyframe using DLT with 3D-2D matches (Appendix F).
4. Recompute the entire structure to take into account the new keyframe, using *N-view* linear triangulation (Appendix D).
5. Apply bundle adjustment to fine-tune all current pose and structure estimates (Appendix E). Poses can be obtained from projection matrices using RQ-decomposition. Optimization can also include intrinsics and radial distortion (Appendix A). If enough matches can be obtained, this procedure allows for processing of hundreds of images or more. An accurate initial estimate of the intrinsics is essential. Such information is usually extracted from image meta-data or from the camera specifications whenever available.

An essential component of such a pipeline is the feature detection and matching algorithm used, since this is the fundamental building block of the pipeline. Pose and structure estimation as well as incorporation of new images and bundle adjustment all rely on accurate feature matching to produce accurate results themselves.

There are a few current methods for feature detection and matching which provide very good results and are widely used by the Image Processing and Computer Vision research communities. For sparse feature detection and matching, Lowe's 'Scale Invariant Feature Transform' (*SIFT*) algorithm [49] has proven to be the most widely-used and was the first of its kind. Algorithms inspired by SIFT soon followed, such as 'Speeded-Up Robust Features' (*SURF*) by Bay et al [5] and more recently 'Gradient Location and Orientation Histogram' (*GLOH*) [54], which is an extension of SIFT designed to increase robustness and distinctiveness. For dense matching, a number of algorithms were described in the Introduction, and the algorithm we use in this project [15] was explained in detail in Chapter 2, but there a number of new and successful dense algorithms which are worth mentioning, such as *SIFT-Flow* [46], which performs dense matching of SIFT descriptors, *DAISY* [91], which is a dense descriptor for wide-baseline stereo, and *PatchMatch* [4], which matches image patches based on the Approximate Nearest Neighbors algorithm.

Due to the success we have had using it as the building block for our pipeline, the SIFT algorithm will now be described in more detail. The algorithm's purpose is for extracting distinctive invariant features from images, which then allows for reliable matching between different views of a scene. The features are invariant to image scale and rotation, and provide robust matching across a large range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination [49].

There are four basic steps involved, which will be summarized here but details can be found in Lowe [49]. The first is scale-space extrema detection, where a search over all scales and image locations is implemented by using a difference-of-Gaussians function to identify potential interest points. Next is keypoint localization, where at each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability. In the next step, orientation assignment, one or more orientations are assigned to each keypoint location based on local image gradient directions.

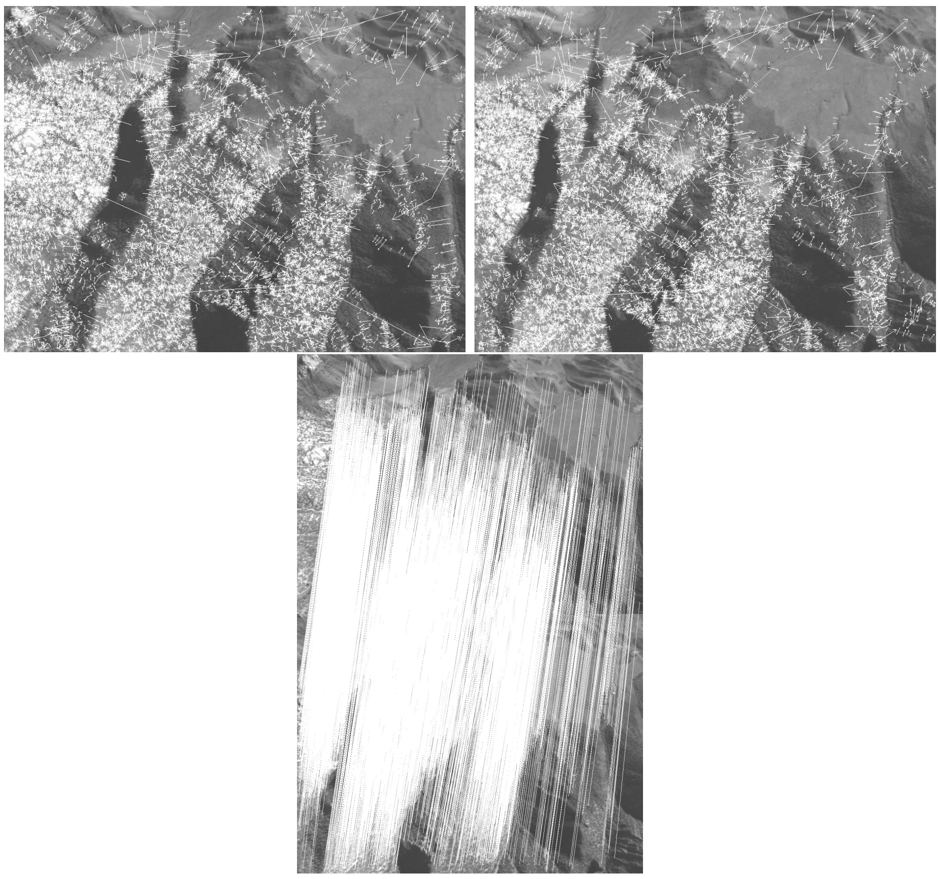


Figure 6.2: SIFT features (top) and matches (bottom) between frames $1-3$ of the *Palmdale* dataset, where matching positions are displayed as lines between matching features in the two images, which are stacked vertically.

All future operations are performed relative to the assigned orientation, scale, and location for each feature, to provide invariance to these. Finally, a local image descriptor is created. The local image gradients are measured at the selected scale in the region around each keypoint, and transformed into a representation that allows for significant local shape distortion and change in illumination. The concept of SIFT matching is illustrated in Fig. 6.2.

6.2 Tests and Results

The described pipeline was applied to a number of real and synthetic scenes. Sample images of real scenes are shown in Fig. 6.3 and Fig. 6.4 shows sample synthetic scenes.

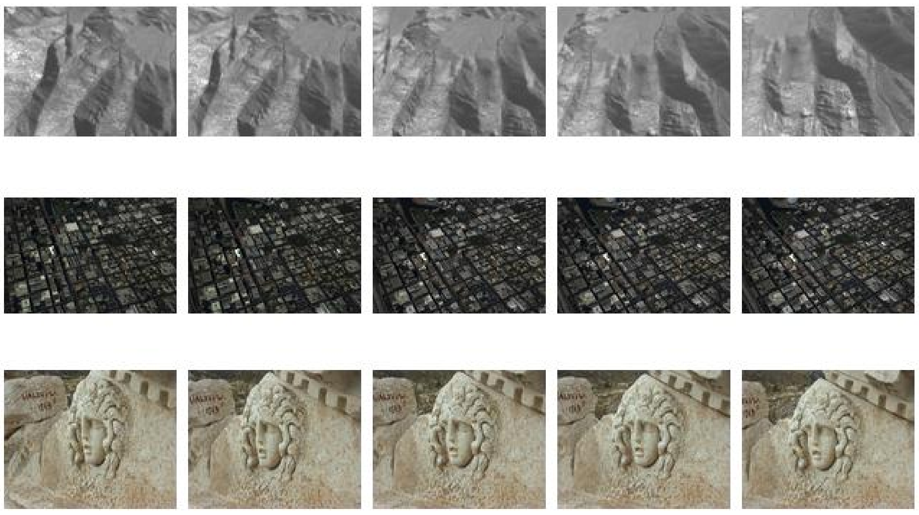


Figure 6.3: Sample datasets depicting real scenes: *Palmdale* (top), *Stockton* (middle) and *Medusa* [68] (bottom).

A series of screen shots show sparse reconstructions obtained for some of these datasets using the proposed pipeline. Figs. 6.5 and 6.6 show results for the *Dinosaur* dataset [60]. Figs. 6.7, 6.8 and 6.9 show results for the *Medusa* dataset [68].

An image coordinate can be modified to take into account radial distortion, as explained in Appendix A. Using the distortion parameters computed from bundle adjustment, undistorted images can be created, as shown in Fig. 6.10. Fig. 6.11 shows the effect on a multi-view reconstruction of using original, distorted images versus their undistorted counterparts, for the *Palmdale* dataset.

Even though good results were obtained with the proposed pipeline, there are a number of potential improvements that can be applied. One such improvement is to use an initial reconstruction based on a trifocal tensor instead of a two-view reconstruction since this adds robustness to the initial set of points as more images are taken into account, though this takes away the benefits of fast triangulation [45], which is an algorithm for two views. Another potential improvement is to replace SIFT or SURF feature detection and matching with the previously-mentioned DAISY [91] algorithm, a SIFT-inspired dense wide-baseline feature matcher. Currently, the minimum amount of images a point should be visible in to be included in the current reconstruction is set at two, but more accurate

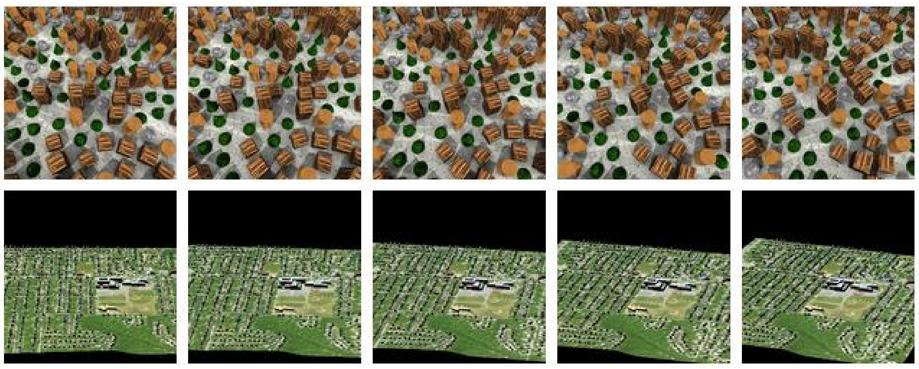


Figure 6.4: Sample datasets depicting synthetic scenes: *Coneland* (top) and *Megascene1* (bottom).

initialized points are achieved if seen by a greater number of views, so such a control over the number of views for initialization will be included. A multi-view extension of our dense correspondence error detection and correction algorithm [32] will be analyzed, in order to get rid of or correct reprojection error or positional outliers at each step, but now based on sparse feature matching instead of dense correspondences. Additionally, computation can be saved by not re-computing points already seen by a large amount of cameras, as well as applying selective instead of global bundle adjustment at each step. Certain existing points in the structure could be unaffected by the incorporation of a new view, either because no information is available in it to affect the point, or because the current position is stable enough to not be affected. Recognising these cases would save a lot of re-computation which is currently being done.

The sparse structure and poses given by this pipeline can be used as input for dense reconstruction algorithms such as the novel 'Patch-Based Multi-View Stereo' (*PMVS*) [21]. Figs. 6.12, 6.13, 6.14 and 6.15 show reconstructions obtained with PMVS [21] for the *Stockton*, *Medusa*, *Megascene1* and *Palmdale* datasets, respectively.

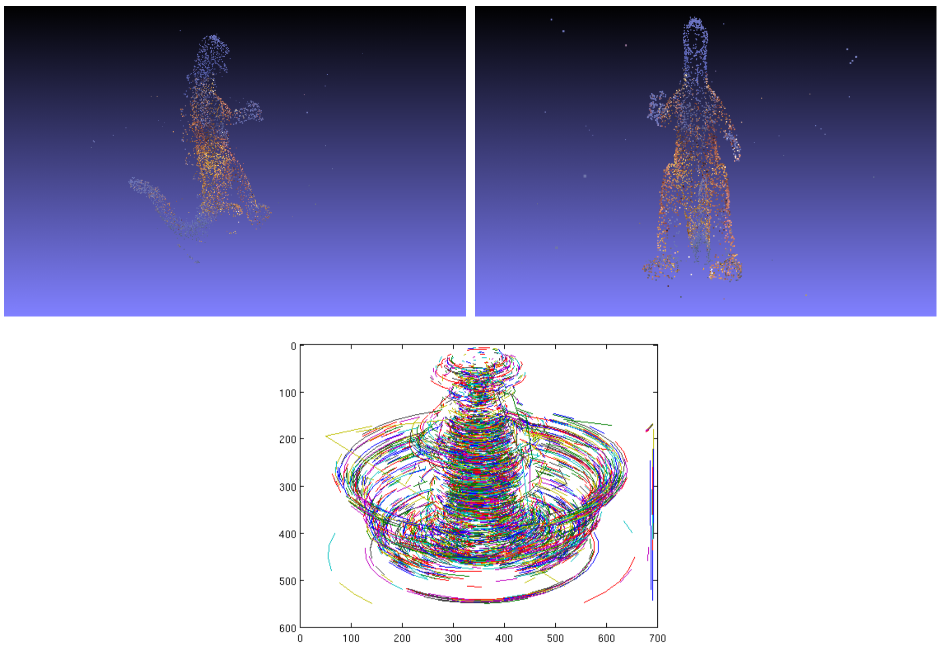


Figure 6.5: Rendered structure (top) and correspondence track summary (bottom) for the *Dinosaur* dataset [60].

6.3 Other Methods for Sequential Pose and Structure Estimation

It is clear from experiments run so far that standard bundle adjustment, even in its sparse form, is very slow and computationally-intensive, and not a viable option for error reduction if real-time processing of large images is intended. Also, standard bundle adjustment has typically been used as a final post-processing step to obtain optimal poses and structure when all the data is already available. This is clearly not an option with the enormous amount of data that can be obtained from the intended aerial imagery application. Some of the recent algorithms for sequential pose and structure acquisition instead make use of probability theory to try to achieve accurate pose and structure estimates. Specifically, *Kalman filters* and *particle filters* can be used to probabilistically model camera poses, and can be used to update the probability of being in a certain state based on acquired new observations.

The 'Kalman filter' is a recursive filter that is capable of estimating the state of

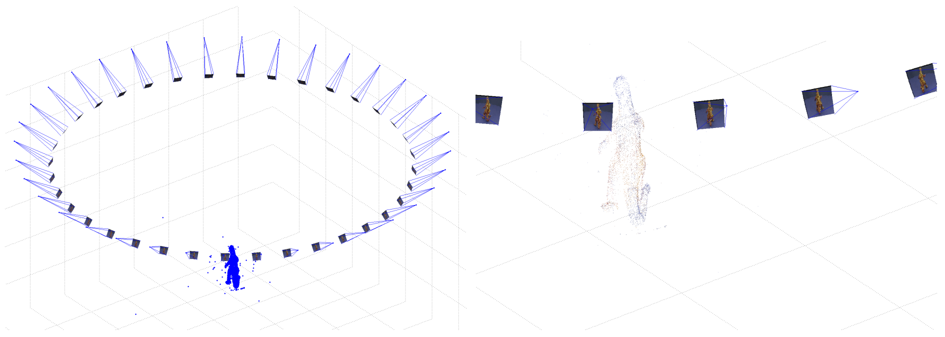


Figure 6.6: Rendered cameras and structure for the *Dinosaur* dataset [60].

a dynamic system from a series of incomplete and possibly noisy measurements. The filter consists of a set of equations for a predictor-corrector estimator that seeks to minimize the error covariance of parameter estimates. In the particular case of scene reconstruction, the Kalman filter can be used for estimating the pose parameters of rotation R and translation T if accurate initial estimates are known, even if noise and occlusions affect the feature matches from which the poses could be estimated. For example, the motion of a camera could have a state vector which consists of the six parameters of the camera pose, plus its translational and angular velocities. Any measurements, such as image location of feature match positions, are assumed to be related to the state by a linear or non-linear measurement model. At each time step, the Kalman filter first makes an estimate of the current state, *a priori*, along with its covariance matrix, which is a measure of uncertainty in the state variables. This estimate is then refined by incorporating measurements to yield the *a posteriori* estimate and its covariance matrix. Noise, initial uncertainty and posterior distributions are all assumed to be Gaussian. One problem with Kalman filters for this application involves the chosen motion model. If the model is too simple, for example by assuming constant velocity, the results obtained could be inaccurate. Furthermore, if measurements are assumed to be independent when they are really not, for example image projections of points that all belong to the same plane in space, this can result in a false reduction in the covariance of the *a posteriori* state estimate, which makes the estimate unreliable and renders the process inaccurate.

A more general formulation known as 'particle filters' was created, which repre-

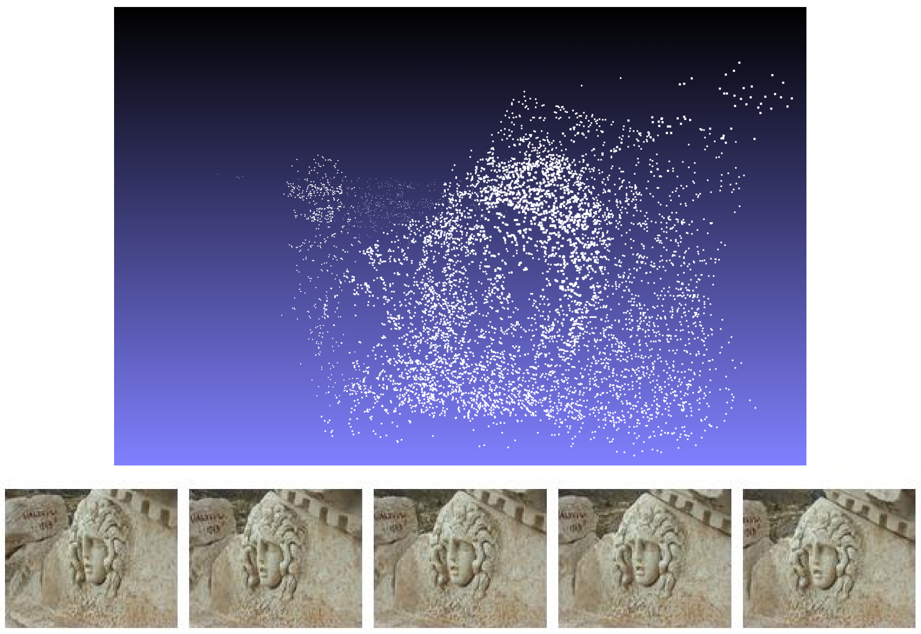


Figure 6.7: Multi-view reconstruction (top) of the *Medusa* dataset [68], with sample images on the bottom.

sents the state density as a mixture of Gaussians that can approximate any density, instead of restricting posterior state distribution to be strictly Gaussian as in Kalman filters. The resulting set of 'particles', which are in essence weighted hypotheses, allow for keeping track of several hypotheses over time, and this is what makes particle filters more robust in general than Kalman filters. One major issue with particle filtering is that a large number of particles may be needed if the motion is not well-defined, which makes the process slow. Furthermore, the mean or median of particle values is used to make the state estimate, which may not be an accurate estimate. However, the use of particle filters does have some advantages, since they do not require linearization of the relation between the state and the measurements, and additional information can be input into the system as part of the state vector, for example if GPS data from the set of cameras is available.

Several publications exploit the properties especially of particle filters for pose estimation and scene reconstruction. In Pupilli et al. [69], a particle filter provides recursive approximations to the posterior density for the pose parameters of a hand-held camera. Each particle represents a pose, and these are weighted according to their likelihood, which

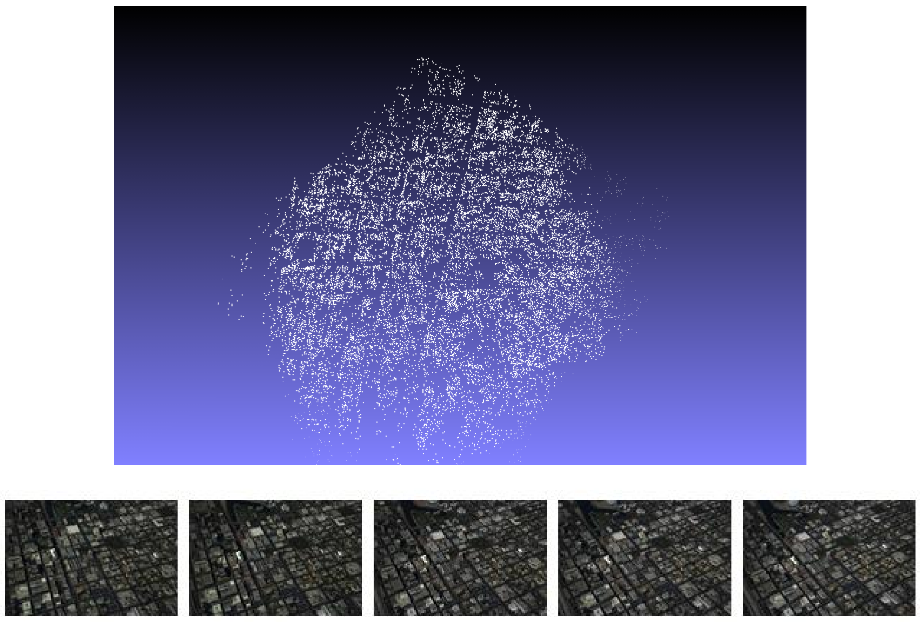


Figure 6.8: Multi-view reconstruction (top) of the *Stockton* dataset, with sample images on the bottom.

is computed by a process similar to template matching, where the reprojections of a set of points into the image plane for a given pose are compared with respect to a template. In Qian et al. [70], the structure from motion problem is addressed using ‘Sequential Monte Carlo’ methods. A new algorithm based on random sampling is derived to estimate the posterior distributions of camera poses and scene structure. Experimental results show that issues such as erroneous feature tracking, occlusions and moving objects can be well-modeled and addressed using the proposed method.

Other algorithms in the literature also deal with the processing of long sequences, but not necessarily using filters, and were all analyzed initially to see if any helpful ideas could be gathered towards our pipeline implementation. In Fitzgibbon et al. [20], instead of sequential processing of the images, a hierarchical processing is used, building from image triplets and associated trifocal tensors. This is shown to optimally distribute error over the sequence. Its major contribution is that it can deal with closed sequences, where a part of the scene is revisited later on in the sequence, using additional constraints available for these cases. However, hierarchical processing with streaming large images might affect real-time operation, so this approach might not be suitable for our application. In Martinec et al. [52], a new technique for estimating a multi-view reconstruction given pair-wise Euclidean reconstructions up to rotations, translations and scales is presented. The partial reconstructions are glued by first estimating camera rotations consistent with all reconstructions, and then pair-wise reconstructions are modified according to the new rotations and refined by bundle adjustment while keeping the corresponding rotations constant. Finally, the refined rotations are used to estimate both camera translations and 3D points. This approach requires multiple point cloud reconstructions to be estimated, which could lead to inaccuracies when registering them together, as opposed to obtaining one initial reconstruction which could then be fine-tuned incrementally. In Rachmielowski et al. [71], SSD tracking is combined with incremental structure computation into a system computing both motion and structure on-line from video. In combination, the structure estimation and tracking benefit each other, resulting in both better structure and more robust tracking. By makine use of the 3D structure, the method can manage visibility constraints, add new image patches to track as they come into view and remove ones that are occluded or fail. In Nistér [57], a system capable of performing robust live ego-motion estimation for perspective cameras is presented. The system is based on RANSAC with pre-emptive scoring of the motion hypotheses. In the approach of Zhang et al. [107], a novel incremental motion estimation algorithm to deal with long image sequences is proposed. It applies to a sliding window of triplets of images, but also uses those points shared only by two views. The problem is formulated as a series of local bundle adjustments in such a way that the estimated camera motions in the whole sequence are consistent with each other. The computational gain achieved makes it considerably faster than global bundle adjustment.

The above approaches all present different ways to deal with sequential scene reconstruction, but in most cases it is clear that real-time performance is not possible with large images and high frame rates unless some fundamental modifications were performed. For example, hardware acceleration on a GPU of such algorithms, as well as in our own pipeline, is an important topic that should be taken into account.

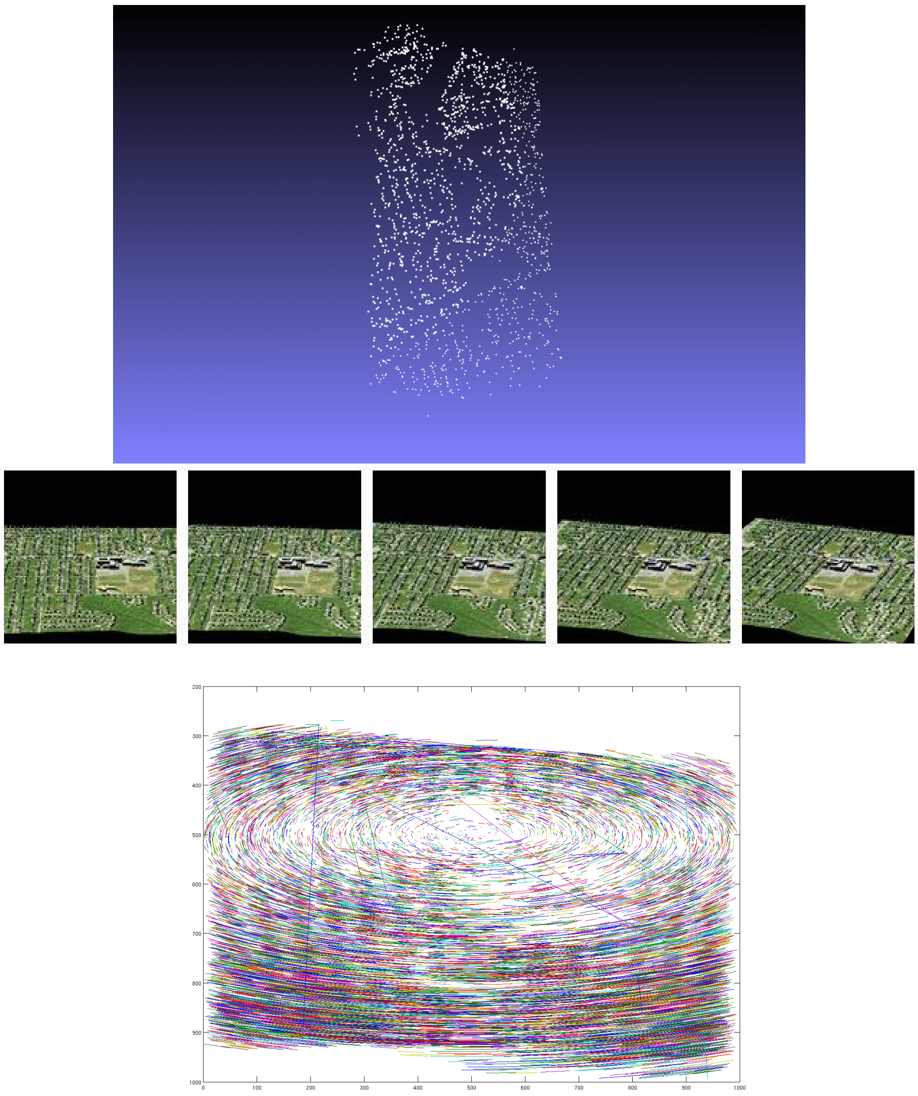


Figure 6.9: Multi-view reconstruction (top) of the *Megascene1* dataset, with sample images (middle) and feature track summary (bottom).

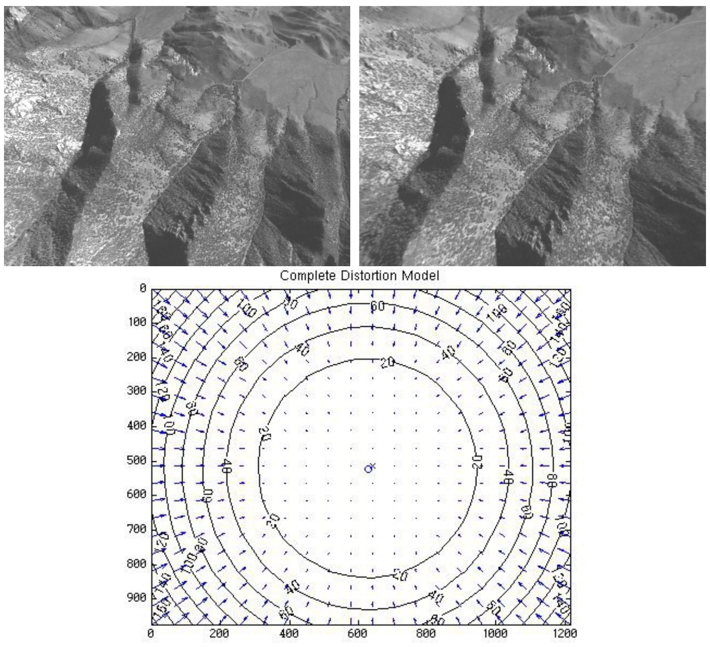


Figure 6.10: Original, distorted *Palmdale* image (top left) and its undistorted version (top right), with the computed distortion map on the bottom.

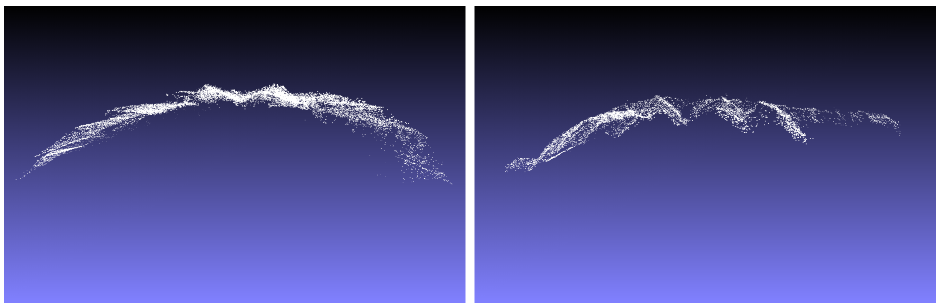


Figure 6.11: Effect on a multi-view reconstruction of using original, distorted images (left) versus their undistorted counterparts (right), for the *Palmdale* dataset.

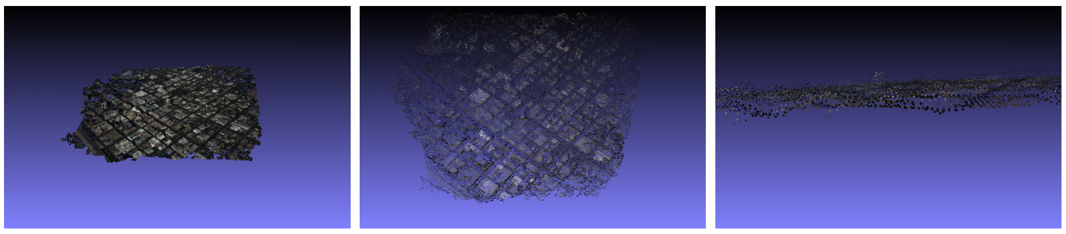


Figure 6.12: Patch-Based Multi-View Stereo (PMVS) [21] applied on the sparse reconstruction pipeline output for the *Stockton* dataset.

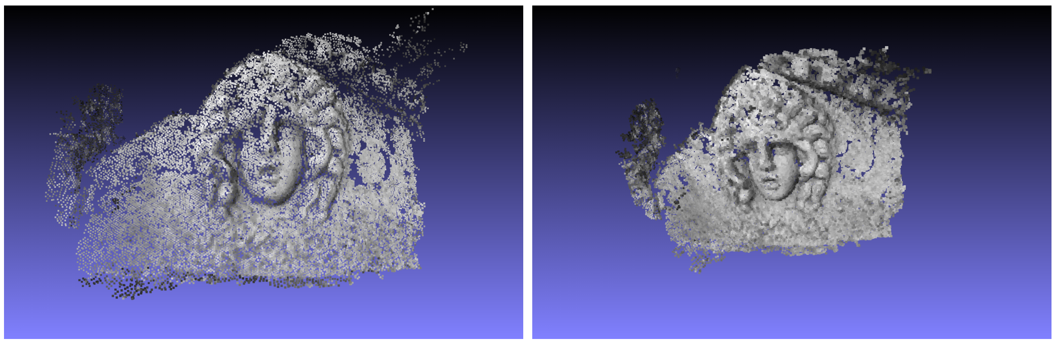


Figure 6.13: Patch-Based Multi-View Stereo (PMVS) [21] applied on the sparse reconstruction pipeline output for the *Medusa* dataset.

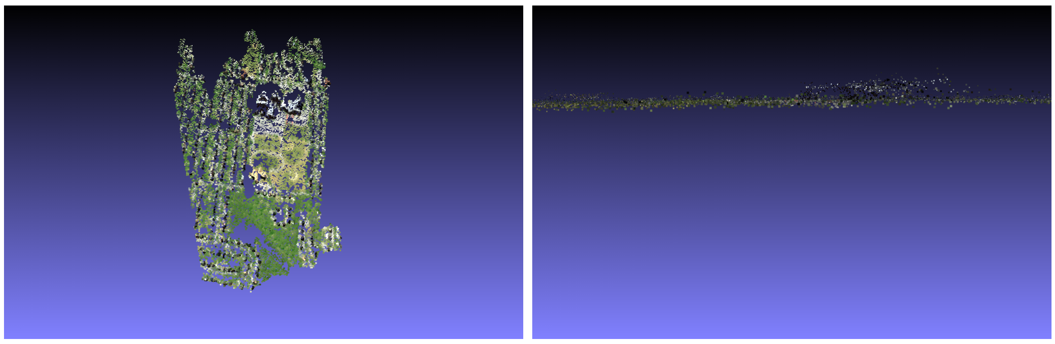


Figure 6.14: Patch-Based Multi-View Stereo (PMVS) [21] applied on the sparse reconstruction pipeline output for the *Megascene1* dataset.

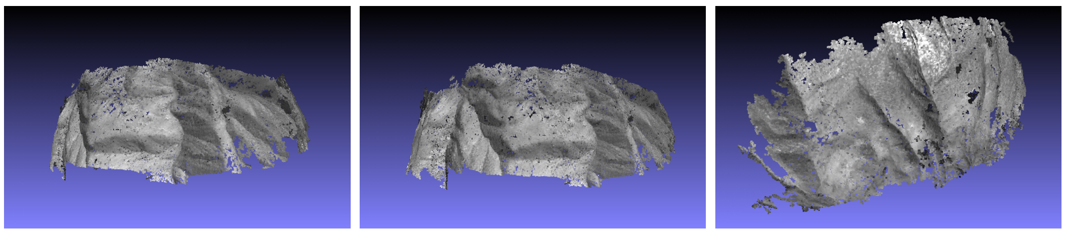


Figure 6.15: Patch-Based Multi-View Stereo (PMVS) [21] applied on the sparse reconstruction pipeline output for the *Palmdale* dataset. The entire dataset (652 images) was reconstructed, with decimation filtering through roughly one out of every 10 frames.

Chapter 7

Parallax Orbits

As has been described throughout this document, the main issue with all reconstruction algorithms is the inaccuracy in structure computation due to inaccurate feature matching and camera parameters. Such inaccuracies are alleviated by bundle adjustment, but this can be a very expensive step in a reconstruction pipeline and relies on somewhat accurate pose and structure estimates.

The goal of the present chapter is to present the theory behind a novel method that exploits the strong constraint imposed by the path of a moving camera to allow for a new way of performing bundle adjustment. The main insight behind this method is that parallax movement corresponding to a feature track should ideally be a scaled version of the camera trajectory when projected onto a plane. It is discussed how this principle can be used to concurrently improve camera parameters, scene structure and also the feature tracks themselves using a very efficient, more accurate, faster and more complete alternative to traditional bundle adjustment, which can be performed in one simple, non-iterative step.

As input, we assume that we have a sparse reconstruction of a scene, based on sparse feature tracks, and some information on the camera projection matrices or intrinsic and extrinsic calibration data, such as the focal length, skew and principal point as well as relative or absolute position and orientation. Such information can be obtained with our pipeline as described in Chapter 6, or with software packages such as *Bundler* [81], which is also based on SIFT feature detection and matching [49], but a short summary of the process is now given. Once matches are available, either sparse or dense, the relative pose between the cameras viewing the scene can be computed. In the particular case that the fundamental matrix F is available or has been computed from matches, and if the camera's intrinsic parameters are assumed known, the essential matrix E can be computed and decomposed into the relative rotation and translation. Finally, the scene's 3D structure can be obtained using the available sparse or dense matches. Once pose and structure estimates are available, bundle adjustment is applied to fine-tune all estimates. All of these steps are explained in detail in Chapters 2 and 6, and also in the Appendices. The *Bundler* package assumes the general case of unordered image sets, but in our framework this as well as ordered image sets can be used. In the ordered case, to ensure at least somewhat accurate pose and structure estimates, the use of frame decimation ensures the use of good image pairs for pose and structure estimation [40], as detailed in Chapter 5.

Once the mentioned inputs are available, the first step in our algorithm is to pick an *anchor frame*, such that scene reconstruction will be performed with respect to this particular frame. Now only those feature tracks that are visible in this frame need to be kept, along with just the cameras associated with these tracks. Next, a *reconstruction plane* must be chosen, and conditions that this plane must meet will be described in Section 7.1.2. Rays are then shot from all available cameras for each track and intersected with this plane. Each feature track projected on the plane will be referred to as a *parallax orbit*, or equivalently *parallax path*. If all parallax orbits are translated to a common origin, and the same is done for the mirrored projection of the camera path on the reconstruction plane, it can be shown that each track's projection on the plane is an *exact* but *scaled* version of this path. Furthermore, it can be proven that assuming such a common origin, parallax orbit positions for all feature tracks seen by a particular camera must lie on the same line on the reconstruction plane. These are the two fundamental constraints imposed by incorporating camera trajectory information into parameter optimization. The next sections discuss how these constraints can be used in practice to improve parameter estimates. A detailed introduction to the concept of parallax orbits is given in Section 7.1. The application of parallax orbits to solve the bundle adjustment problem in a fundamentally different way is detailed in Section 7.2.

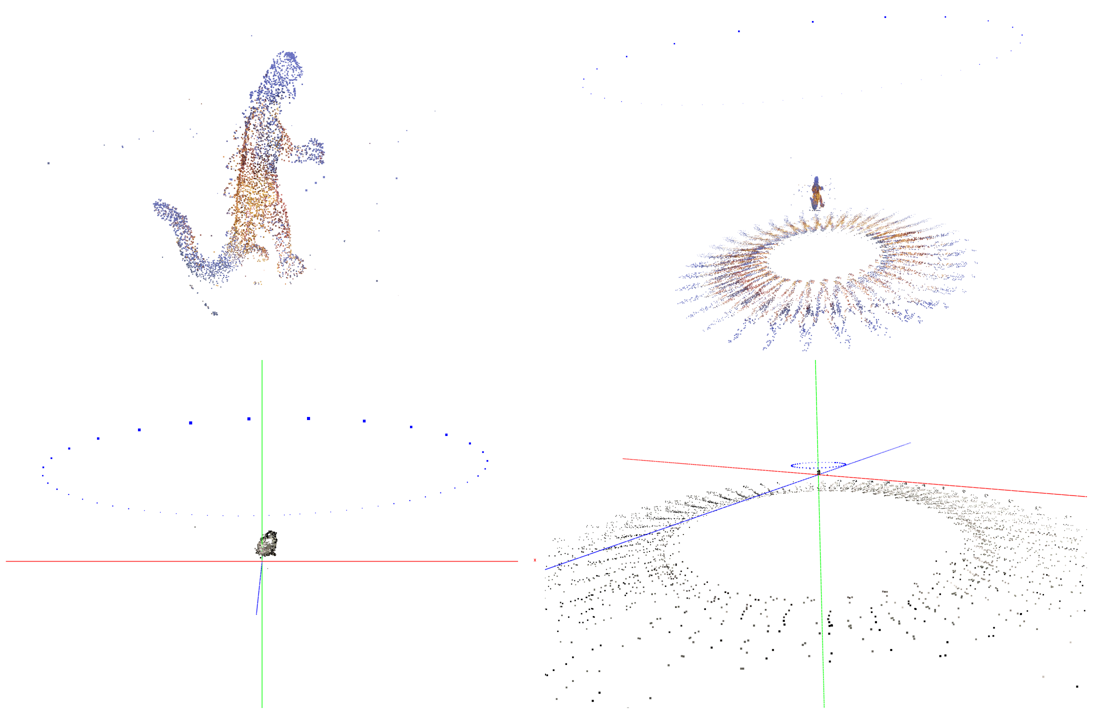


Figure 7.1: Sparse ground-truth reconstruction of the *Dinosaur* dataset (upper left), and ray intersections with the plane $Z = -1$ (upper right). The same is shown for the *dinoRing* dataset [79] on the bottom.

7.1 Introduction to Parallax Orbits

In order to understand the concept of parallax orbits, it will first be assumed that we have a set of images for which *perfect* feature matches and exact knowledge of each camera's intrinsics, extrinsics and radial distortion is available. In this scenario, rays emanating from each camera center and through the respective feature match position on each image plane should intersect at an *exact* position in 3D space. Now assume that these rays continue on in space and eventually intersect some plane. We define the intersections each camera's rays create on that plane, which will be defined from now on as the *reconstruction plane* π, as *replicas*. To visualize this concept, Fig. 7.1 shows a sparse reconstruction of the *Dinosaur* dataset on the left, and ray intersections with the plane $Z = -1$ on the right. Each replica visually resembles a 2D projection of the 3D object on the plane. If a different plane is used, different replica shapes are obtained on that particular plane, but it will be shown later that what matters is the *relative* position of ray-plane intersections and not

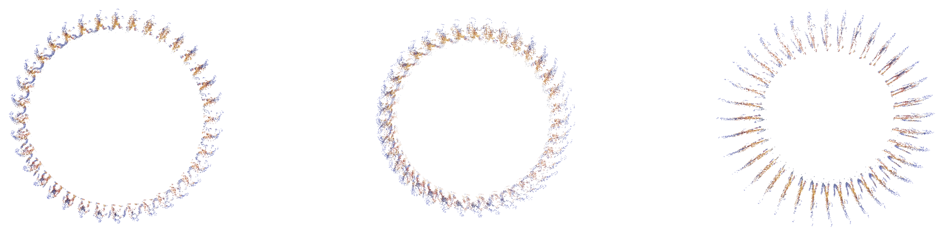


Figure 7.2: Ray-plane intersections for the *Dinosaur* dataset, at planes $Z = -1$ (left), $Z = 0$ (middle) and $Z = 5$ (right).

the actual shape of the replicas. The effect of plane position on replica shape is shown in Fig. 7.2. Again, some conditions do exist on what plane can be used, and this will be discussed further in Sec. 7.1.2.

7.1.1 Ray-Plane Intersection Calculation

As mentioned previously, it is assumed that 3×4 projection matrices for each camera and a set of sparse feature tracks are available as inputs. In our coordinate representation, we define each projection matrix as shown in Eq. 7.1, where K corresponds to each respective camera's 3×3 intrinsic calibration matrix, R is its absolute orientation matrix and T its absolute position. We work directly with P, but $[K, R, T]$ can be easily extracted from P using QR-factorization [30]. If these were not available but a set of feature tracks and sparse 3D structure was, the Direct Linear Transformation algorithm embedded in RANSAC can be used to robustly extract these [30], as discussed in Appendix F.

$$P = K[R|T] \tag{7.1}$$

Each camera center $C = (X, Y, Z, W)$ can be computed from $PC = 0$ as the null-space of P using Singular Value Decomposition, but for improved numerical stability we use the expressions shown in Eq. 7.2, where p_i corresponds to the i_{th} column of P [30].

$$\begin{aligned} X &= \quad det([p_2, p_3, p_4]) \\ Y &= -det([p_1, p_3, p_4]) \\ Z &= \quad det([p_1, p_2, p_4]) \\ W &= -det([p_1, p_2, p_3]) \end{aligned} \tag{7.2}$$

For a given feature in an image, a ray starting from the respective camera center C and through its position x on the image plane can be computed parametrically per Eq. 7.3 [30]. The right pseudo-inverse $P+$ of a projection matrix P is computed from $P+ = P^T(PP^T)^{-1}$. Since a ray can be defined with two points, one will always be the camera center and the other a point X in space defined by the parameter λ.

$$X(\lambda) = (P+)x + \lambda C \tag{7.3}$$

Given this general ray equation, to compute the intersection between a ray emanating from a camera center and a plane (A, B, C, D), we compute the value of the parametric distance along the ray, which will be referred to as t, for which the intersection is achieved. Let the ray $R(t) = R_0 + tR_d, t > 0$, such that $R_0 = [X_0, Y_0, Z_0]$ corresponds to the camera center C coordinates and $R_d = [X_d, Y_d, Z_d]$ is some point along the ray. If the plane is defined as $Ax + By + Cz + D = 0$, then $A(X_0 + X_d t) + B(Y_0 + Y_d t) + (Z_0 + Z_d t) + D = 0$, which yields the value for t shown in Eq. 7.4.

$$t \quad = \quad \frac{-(AX_0 + BY_0 + CZ_0 + D)}{AX_d + BY_d + CZ_d} \tag{7.4}$$

This ray-plane intersection is then performed for all available feature match positions across all cameras.

7.1.2 Reconstruction Plane Conditions

The chosen reconstruction plane must comply with a series of criteria. First, it cannot intersect the visual hull made up of the scene and cameras. It must also be placed such that the camera path can be projected onto the plane. We found that the best solution to both requirements is to first obtain a best-fit plane to the camera path, and place the reconstruction plane parallel to the best-fit plane at a position that is somewhat outside of the scene-camera visual hull.

7.1.3 Choosing an Anchor Frame

Since the amount of images in a sequence could be arbitrarily long, anchor frames should be chosen such that there is overlap between them and information for the entire scene is contained in the set of anchors. For now, focus will be on explaining the algorithm for one particular anchor frame only.

Given an anchor frame, the first action to be taken is to choose only feature tracks that span that particular anchor, out of the complete set of available tracks, along with the respective projection matrices for cameras whose tracks span the anchor. Additionally, the computed scene structure must only span these cameras. To explain with an example, take for instance the *Dinosaur* dataset [60]. This dataset consists of 36 equally-spaced frames, corresponding to turn-table views of a plastic dinosaur. If the very first frame is chosen as the anchor frame, and using the provided ground-truth feature tracks, it can be seen that these span at most 11 frames, so the algorithm would proceed with projection matrices and tracks that span only those specific frames.

7.1.4 Initial Parallax Orbit Computation

After choosing an anchor frame, parallax orbits are computed at the anchor frame for each feature track. As was described in Section 7.1, if rays are shot through each camera center and through each feature of a track, a parallax orbit, or path, is formed on the reconstruction plane. This can be seen on the left-hand side of Fig. 7.3 for all ground-truth feature tracks visible in the first frame, which was chosen as the first anchor frame. The right-hand side shows a top view of the parallax orbits formed for a chosen set of those feature tracks, where each path has the color of the corresponding 3D scene point. The outermost green track corresponds to the projection of each camera center on the plane, reflected such that it has the same orientation as the parallax orbits. From this it is clear that each parallax orbit is a scaled version of the camera path, simply located at different positions and of different lengths depending on the length of each feature track. This is the

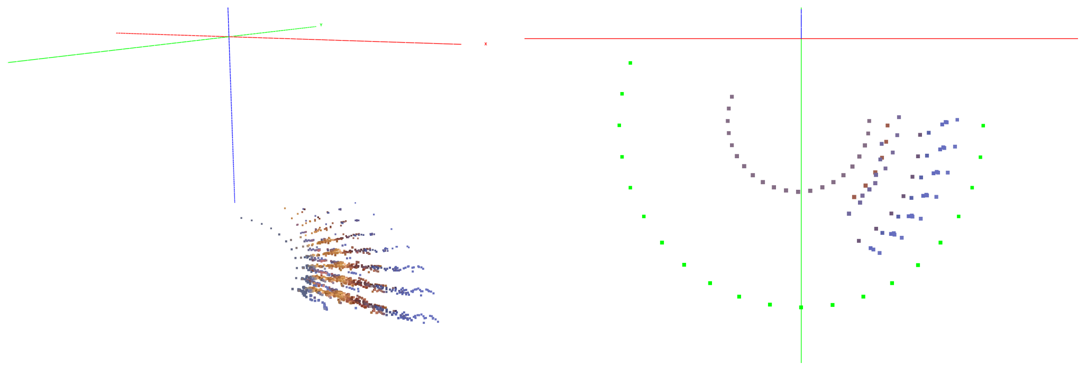


Figure 7.3: Parallax orbits for an anchor frame of the *Dinosaur* dataset, at plane $Z = -1$ (left). A top view of the paths obtained for a chosen set of feature tracks is shown on the right.

key observation towards understanding our algorithm, as each scale *defines* the $3D$ position along each ray to the anchor.

7.1.5 Translation of Parallax Orbits to the Anchor Position

Now that everything has been projected onto the reconstruction plane, we can work in a 2D coordinate system with (x, y) coordinates. In this system, the first step is to translate all parallax orbits on this plane such that the ray-plane intersections of each path *at the anchor frame* lie on the same origin. In this representation, shown in Fig. 7.4, it becomes much more clear to see that the parallax orbits follow the shape of the reflected camera path *exactly*, but at different scales.

7.1.6 Zero Reprojection Error Lines

A closer analysis of Fig. 7.4 reveals another very important concept: in the position-invariant space achieved after translating all parallax orbits and the reflected camera path to a common origin, ray-plane intersections for all features seen in the same camera along with the projected and reflected camera center all lie on the *same line*. In general, if this is achieved for a given set of cameras and feature tracks, the reprojection error for the $3D$ positions yielded by those features in those cameras is 0. However, notice that those features could be moved *along* that same line, still yielding zero reprojection error but now with incorrect scaling, which manifests as an inaccurate $3D$ structure. The very power of

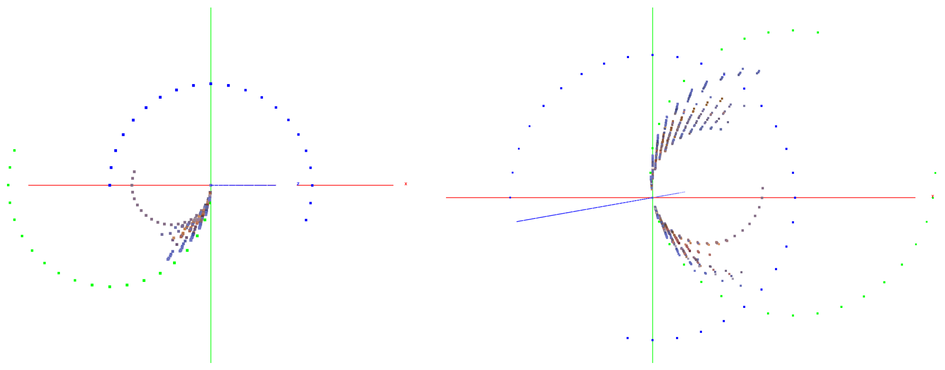


Figure 7.4: Parallax orbits and reflected camera paths translated to a common origin on the reconstruction plane, using the first (left) and 18*th* (right) frames as anchors, for the *Dinosaur* dataset, at plane $Z = -1$. The original projected camera paths are shown in blue, with their reflected versions in green.

our technique lies in the fact that we can make use of this resulting grid structure instead of simple fitting on a line for parameter optimization, which is essentially what traditional bundle adjustment achieves while *blindly* searching for the best solution along this line. How this grid can be used will be discussed in Section 7.2, but first in Section 7.1.7 it will be analyzed how such a geometric construct meets epipolar geometry constraints, and is thus a valid construct for a novel type of bundle adjustment.

7.1.7 Relation with Epipolar Geometry

Finally, we wish to show how the described geometric construct is geometrically valid as it exactly matches epipolar geometry constraints. Since we count with projection matrices for each of the cameras, it is possible to extract pairwise fundamental matrices F between any camera pairs. For a pair of cameras, let P be the projection matrix for the first camera, P' for the second camera, $P+$ is the pseudo-inverse of P and C is the camera center for the first camera. Then it follows that the fundamental matrix between the two views is given by Eq. 7.5.

$$F = [P'C]_x P' P+ \tag{7.5}$$

If the epipole e' is known, then F can be computed from $F = [e']_x P' P+$, since $e' = P'C$. Yet another equivalent expression is $F = K^{-T} R K^T [K R^T t] x$, where $[K, R, t]$ correspond to

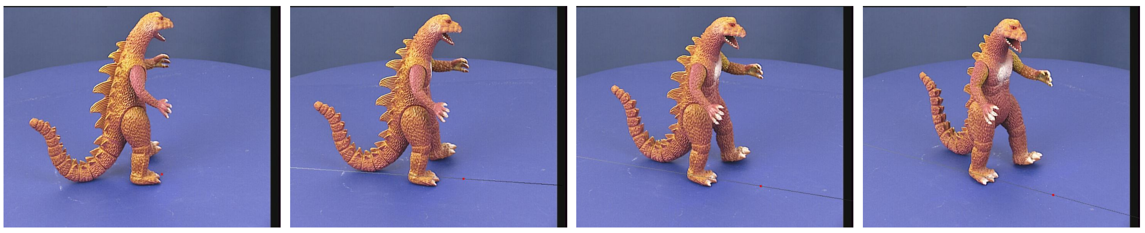


Figure 7.5: Anchor frame position in red (left) and epipolar lines for search in other images (right) for the *Dinosaur* dataset. Scales are color-coded such that black corresponds to '0' and white to '1'.

the camera's intrinsics and extrinsics as described previously. Another useful identity is that P' can be recovered from F and e' such that $P' = [[e']_x F|e']$, where $e = KR^T t$ and $e' = Kt$.

Now that the basic equations have been described, it will be analyzed how the parallax orbits construct relates to epipolar geometry. It is very important to mention that the line of search positions formed on each image plane, formed by reprojection of a given parallax orbit and which will be referred to as *locus*, corresponds to the epipolar line from the anchor camera to the current one. Furthermore, if a 'successful' match is found in the second image, the exact position of the match in a third image is given by the intersection of the locus with the corresponding epipolar line. Perfectly-defined epipolar lines can be obtained from the set of locus lines, which can be used for epipolar geometry searching, which makes this framework very useful for wide-baseline matching.

Any exact match (x, x') should meet the criteria $x' = (F_{N,1} * x_1)x(F_{N,N-1} * x_{N-1})$. An example of this is shown in Fig. 7.5, where the left-most image shows in red the feature that should be matched in the other images, and the remaining images show the corresponding epipolar line to search over, where scales are color-coded such that black is '0' and white corresponds to '1'. This search space can be further constrained for example through the use of homographies, but that is out of the scope of this introduction to the parallax orbits framework and we just limit the discussion here to showing that epipolar geometry constraints are actually met within this framework.

7.2 Bundle Adjustment Through Parallax Orbits

One major issue with bundle adjustment is that it basically searches ‘in the dark’ for the global optimum. Since no prior knowledge of the camera path is assumed, and this is not used as a constraint, there is always a chance of getting stuck in a local minimum. With the presented framework, we can ensure that the global minimum is always reached, one for which the reprojection error is zero and also such that all feature tracks are consistent with the camera path. Knowing the limitations of traditional bundle adjustment, in the next sections we discuss ways of replacing bundle adjustment for parameter optimization based on the parallax orbits framework for four different scenarios: perfect cameras and feature matches, perfect cameras and imperfect feature matches, imperfect cameras and perfect feature matches, and finally the general case of imperfect cameras and feature matches.

Perfect cameras and perfect feature matches

For perfect cameras and feature matches, the parallax orbits and camera path on the reconstruction plane form *identical* and scaled shapes. Furthermore, the relative position along a line drawn from the 2D origin on the plane to each projected camera position on the plane, which defines scale for a given parallax orbit, should be the same for all features of a given feature track with respect to their corresponding camera. This concept is illustrated at the top-left in Fig. 7.6, where every parallax path position snaps onto a perfect ‘grid’ over the reconstruction plane, where (x, y) are used as coordinates on this plane. For illustration purposes, each of these lines, along which reprojection error is zero, is shown in light green. The camera path on the reconstruction plane appears as a set of larger blue squares, and there are five tracks in total along five cameras C_1 to C_5, where parallax orbit positions for each track are drawn in a given color and at a given scale. To show these orbits continuously, each is rendered as a gray curve. Notice how each curve is a scaled version of the camera path.

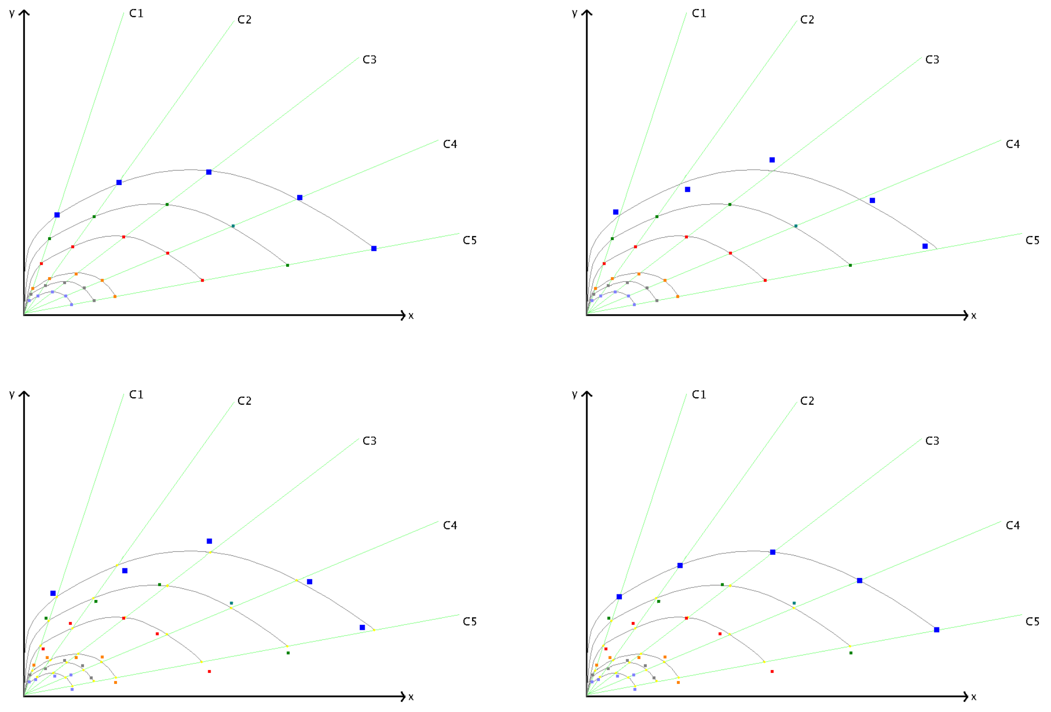


Figure 7.6: Parallax orbits and camera paths for perfect cameras and perfect feature matches (top left), imperfect cameras and perfect feature matches (top right), perfect cameras and imperfect feature matches (bottom left), and imperfect cameras and feature matches (bottom right).

Perfect cameras and imperfect feature matches

In this case, shown at the bottom-left in Fig. 7.6, the camera path falls exactly onto the perfect grid discussed in the perfect cameras and features case, but not the parallax orbits for feature matches. The positions where each should ideally latch onto are shown in yellow. We can show, using the *Dinosaur* dataset as an example, that it is possible to 'fix' feature tracks such that they can essentially become as accurate as ground-truth tracks. Fig. 7.7 shows a plot of ground-truth tracks in image space and SIFT-based tracks on the right. Notice how some of those tracks are clearly incorrect. Though the resulting $3D$ structure may look visually correct, its reprojection error is still not zero even after bundle adjustment. By moving feature track positions such that they lie on the perfect grid, the resulting reprojection error is essentially eliminated.

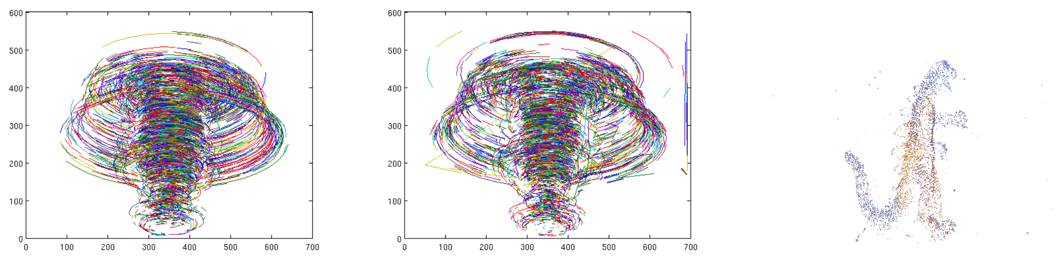


Figure 7.7: Image locations of ground-truth feature tracks (left) and SIFT-based feature tracks (middle) for the *Dinosaur* dataset. The visually-correct yet slightly inaccurate reconstruction obtained from SIFT features is also displayed (right).

Since we know the camera path, adjusting the feature matches involves two steps. First, since the light-green zero reprojection error lines are 'perfect' given the ground-truth camera path, we move each parallax orbit for a feature of a track the shortest distance such that it lies along its corresponding line for the given camera. Then, we move each along its respective line the shortest distance such that the camera path curve is exactly reproduced at a particular scale.

Imperfect cameras and perfect feature matches

This case is shown at the top-right in Fig. 7.6. Conversely to the case of perfect cameras and imperfect tracks, now the parallax orbits for feature matches lie correctly on the perfect grid and create perfect lines emanating from the anchor position and towards each camera. In this case, adjusting the cameras comes down to moving each camera's projection on the reconstruction plane the shortest distance such that it lies along the correct path, and then along each corresponding line the shortest distance such that the best-fit parallax orbit curve is met for the set of cameras. This can be done through standard quadratic fitting for example, but depends on the type of camera path.

Imperfect cameras and feature matches

This is the general case, and one that is most frequently encountered in real sparse reconstructions. This scenario is shown at the bottom-right in Fig. 7.6. In this case, we first have to find the best-fit zero reprojection error locus lines, and additionally the best-fit parallax orbits. Then we can snap each parallax orbit position and camera projection onto this grid to obtain the optimal positions. The initial locus line-fitting must be very robust. We use standard linear regression but embedded in RANSAC [18] for this purpose. Assuming a line $y = mx + b$ and N 2D positions, the slope m and intercept b can be computed from Eq. 7.6. Next, a best-fit *consensus* parallax orbit must be obtained. Finally, we perform snapping on the perfect grid as described in the previous sections.

$$\begin{aligned} m &= \frac{N\Sigma xy - (\Sigma x)(\Sigma y)}{N\Sigma x^2 - (\Sigma x)^2} \\ b &= \frac{\Sigma y - m(\Sigma x)}{N} \end{aligned} \tag{7.6}$$

7.2.1 Geometrical Constraints on Scale Space

A homography is a simpler model than the epipolar geometry model. It allows for a 2D prediction of a 3D movement, but since it doesn't correctly account for parallax like epipolar geometry does, it generally presents a residual error for a given feature match. However, residual measurements for a homography computed over the set of available feature matches can provide bounds on the expected image-to-image 2D movements of feature matches, to greatly constrain the multi-view chain of matches. Pairwise homography estimation over a set of feature tracks can greatly constrain the search space between scales ranging from 0 to 1, such that typically around 90% of scales are removed after this filter. Using the resulting homography prediction position which is closest to the locus line, searching involves simply moving along that line within the maximum residual distance, for searching only over those scales. For large baselines, the homography model becomes very inaccurate, so it should only be applied in a pairwise manner for closest consecutive views. In fact, to determine just how accurate the homography model is with respect to the more general epipolar geometry model, Torr's 'Geometric Robust Information Criterion' (GRIC) metric [94] can be used.

7.3 Future Work

The theory behind our novel method has been presented, but much work remains to be done with respect to testing these principles on a large number of real and synthetic scenes, depicting different camera motions, to see how general the method can be. If proven to work in all cases, we seek to achieve very accurate dense matching and reconstruction. Since our framework is occlusion-invariant, and uses information from all cameras and feature tracks, we can initialize new tracks by searching in scale space for the best scale, starting from a given position in the anchor frame. Once the correct scale has been found for a track, and by extension *all* corresponding feature track positions across all image planes, to find the 3D position for the corresponding scene point all that has to be done is to move along the ray from the anchor frame proportionally to the recovered scale. This is much faster and more accurate than applying for example multi-view linear triangulation [30]. The search in scale space could involve searching along the path for the position where intensities best agree (for example, with the lowest standard deviation), and can be aided by using resolution scale-space or feature descriptors, image patch-matching and also homographies since these help provide bounds on the scales to search over. For planar scenes the geometry can be recovered almost exactly without the need to search. An accurate and robust dense reconstruction can be achieved by applying this procedure for all available pixels of an anchor frame. Initial estimates show that 3D structure can be computed about 2000 times faster than computing pairwise optical flow-based dense correspondences. If used jointly with color segmentation, the joint analysis of computed tracks could make for a novel algorithm for obtaining accurate matches and structure over texture-less regions. Yet another consequence of our framework is that it allows for the auto-completion of tracks after occlusion: any discontinued paths that manifest as separate tracks can be joined by looking at which orbits have the same scale. Also, we are looking into the mathematical definition of a multi-view tensor based on the proposed principles. Looking further, the framework could also be potentially used for the compression of both images and structure parameters, by storing mainly scale-space information instead of explicit image plane or 3D positions.

Chapter 8

Conclusions

In this dissertation, a generalized error analysis framework was presented for scene reconstruction from aerial video, consisting of methods for the detection, factorization and correction of error sources present in all stages of a reconstruction pipeline, and in the absence of ground-truth knowledge. The presented algorithms were designed for sequential scene reconstruction from aerial video, but have been proven to work across different scene types and camera motions, and for both real and synthetic scenes. Furthermore, because of their nature the methods are general enough that they can be applied in conjunction with many different types of scene reconstruction algorithms besides the dense and sparse reconstruction pipelines described in this document.

Two main applications were discussed for dealing with errors in the absence of ground-truth. The first set of algorithms derive total structural error measurements after an initial scene structure computation and factorize errors into those related to the underlying feature matching process and those related to camera parameter estimation. Based on this novel detection of specific error sources in the reconstruction process, a brute-force local correction of inaccurate feature matches was presented, as well as an improved conditioning scheme for non-linear parameter optimization which applies weights on input parameters in proportion to estimated camera parameter errors. An overview will now be given of these methods.

A scene reconstruction pipeline using unconstrained dense correspondences obtained from a pair of stereo images was initially introduced. Even for a reduced amount of images, errors in the dense correspondence process cause pose and structure estimation inaccuracies that accumulate over time in sequential reconstruction. To this end, an algorithm for iterative dense correspondence error detection and correction through feedback from the bundle adjustment process was presented. The main goal was to detect and correct outlier dense correspondences between two images. Initial estimates for the pose and scene structure are obtained from the given dense correspondences, assuming known camera intrinsic parameters, and then bundle-adjusted. The resulting reprojection errors per correspondence pair are then separated into high-error and low-error correspondences based on a threshold computed from reprojection error statistics. Then, a brute-force affine search iterative algorithm operating on a coarse-to-fine resolution pyramid and inspired by the original dense correspondence algorithm is used to correct outlier correspondences. Results on both real and synthetic scenes show that a more accurate set of dense correspondences is obtained after applying the proposed method, which results in an improvement in pose and structure estimates.

Due to the computational expense and error accumulation in sequential reconstruction from dense correspondences, a multi-view reconstruction pipeline was proposed that made use of accurate sparse feature matching instead of dense correspondences as input. The goal of this pipeline is to obtain accurate camera poses throughout a video sequence and create as additional output a sparse point cloud representing scene structure. The pipeline operates by obtaining an initial robust two-view reconstruction and sequentially adding-in new frames using the Direct Linear Transformation algorithm embedded in RANSAC, while bundle-adjusting after every step.

Given that bundle adjustment is by far the most expensive process in the pipeline, an algorithm for bundle adjustment conditioning based on scene reconstruction ray divergence was designed. The proposed algorithm first computes ray divergences when attempting scene reconstruction triangulation from a set of sparse SIFT feature matches. Under ideal conditions, camera rays through a pair of feature matches should intersect in an exact position in space, but since in general this does not hold true due to errors in any of the underlying processes, those errors manifest as divergence between the camera rays. As demonstrated, such errors can be factorized as a smooth component due to camera parameter errors plus a high-frequency component representing matching errors. Assuming accurate feature matching, ray divergence errors are due mainly to camera parameter estimation inaccuracies. It was proven how ray divergences provide similar information to other reconstructed point uncertainty measurements, and thus is valid to use as a means for weighting bundle adjustment and improving its convergence properties. Due to its smooth variation across neighboring matches, from its histogram over the set of matches a set of weights can be derived, based on the Gaussian probability density function. It was shown that this novel weighting scheme is more efficient to compute and outperforms other weighting schemes such as those based on image feature covariances. As is the common theme throughout this dissertation, there is no dependence on ground-truth information, and results show an improved convergence on different real and synthetic scene types.

Finally, the concept of parallax orbits was introduced. Given an initial set of feature matches, poses and scene structure, an additional and strong camera path constraint allows for a concurrent error detection and correction in the pose estimation, feature matching and structure computation processes that essentially eliminates the need for traditional bundle adjustment parameter optimization. This constraint had not been explicitly taken into account in the existing scene reconstruction literature, and it was shown how it is a valid constraint that obeys epipolar geometry criteria. Though this new methodology is still in its infancy as of the completion of this dissertation, and much work remains to be done to prove the generality of the framework, there is hope that it can have a major impact on the future accuracy and speed of multi-view reconstruction.

Another application that forms part of the proposed generalized error analysis framework is in reconstruction pre-processing. An algorithm was presented that detects and discards frames that would lead to inaccurate feature matching, camera pose estimation degeneracies or mathematical instability in structure computation based mainly on a residual error comparison between two different match motion models. Known as frame decimation, such a filter is an essential component of a reconstruction pipeline such as the sparse pipeline described in this document, which operates based on frame decimation keyframes. Existing decimation algorithms in the literature decimate frames globally and cannot perform in environments such as in streaming aerial video, so the main contribution is the ability to decimate in a streaming fashion, and without the need for scene-dependent thresholds. To this end, a new frame goodness metric was introduced, which is designed to choose only one global maximum value at each keyframe evaluation, needing at most three frames in memory at a given time. The cost function is a weighted version of the GRIC criterion for residual error comparison between epipolar geometry and homography estimation. The GRIC component detects pose degeneracies and mathematical instability in structure computation, which occur with small baselines. Weighting is based on feature match layout and ratio with respect to the number of obtained features, and is meant to filter out very wide baselines. The algorithm was proven to perform well across different types of target scenes and camera movements.

References

[1] M.T. Ahmed, M.N. Dailey, J.L. Landabaso, and N. Herrero. Robust Key Frame Extraction for 3D Reconstruction from Video Streams. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 231–236, 2010.

[2] A. Akbarzadeh, J. M. Frahm, P. Mordohai, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistr, and M. Pollefeys. Towards Urban 3D Reconstruction from Video. In *in 3DPVT*, pages 1–8, 2006.

[3] S. Avidan and A. Shashua. Threading Fundamental Matrices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(1):73–77, 2001.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[6] S.S. Beauchemin and J.L. Barron. The Computation of Optical Flow. *ACM Comput. Surv.*, 27(3):433–466, 1995.

[7] C. Beder and R. Steffen. Determining an Initial Image Pair for Fixing the Scale of a 3D Reconstruction from an Image Sequence. In *DAGM-Symposium'06*, pages 657–666, 2006.

[8] M. J. Brooks, W. Chojnacki, D. Gawley, and A. van den Hengel. What Value Covariance Information in Estimating Vision Parameters? In *ICCV'01*, pages 302–308, 2001.

[9] M.J. Brooks, W. Chojnacki, and L. Baumela. Determining the Egomotion of an Uncalibrated Camera from Instantaneous Optical Flow. *Journal of the Optical Society of America*, 14:2670–2677, 1997.

[10] M.J. Brooks, W. Chojnacki, A. Van Den Hengel, and L. Baumela. 3D Reconstruction from Optical Flow Generated by an Uncalibrated Camera Undergoing Unknown Motion. *International Workshop on Image Analysis and Information Fusion, Adelaide*, 14:35–42, 1997.

[11] B. Caprile and V. Torre. Using Vanishing Points for Camera Calibration. *Int. J. Comput. Vision*, 4(2):127–140, 1990.

[12] L.F. Cheong and Y. Aloimonos. Iso-Distortion Contours and Egomotion Estimation. *International Symposium On Computer Vision*, 0:55, 1995.

[13] L.F. Cheong, C. Fermüller, and Y. Aloimonos. Effects of Errors in the Viewing Geometry on Shape Estimation. *Comput. Vis. Image Underst.*, 71(3):356–372, 1998.

[14] A.J Davison. Active Search for Real-Time Vision. In *ICCV 2005. Tenth IEEE International Conference on Computer Vision, 2005*, volume 1, October 2005.

[15] M. Duchaineau, J. Cohen, and S. Vaidya. Toward Fast Computation of Dense Image Correspondence on the GPU. In *Proceedings of HPEC 2007, High Performance Embedded Computing, Eleventh Annual Workshop*, pages 91–92, Lincoln Laboratory, Massachusetts Institute of Technology, 2007.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2nd edition, 2000.

[17] C. Engels, H. Stewenius, and D. Nistér. Bundle Adjustment Rules. In *PCV06*, 2006.

[18] M.A. Fischler and R.C. Bolles. Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 726–740, 1987.

[19] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D Model Construction for Turn-Table Sequences. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments, LNCS 1506*, pages 155–170. Springer-Verlag, June 1998.

[20] A.W. Fitzgibbon and A. Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 311–326, London, UK, 1998. Springer-Verlag.

[21] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[22] M. Goesele, N. Snavely, C. Curless, H. Hoppe, and S. M. Seitz. Multi-View Stereo for Community Photo Collections. In *Proceedings of ICCV 2007*, 2007.

[23] L. Goshen and I. Shimshoni. Balanced Exploration and Exploitation Model Search for Efficient Epipolar Geometry Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1230–1242, 2008.

[24] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[25] R. Hartley, R. Gupta, and T. Chang. Stereo from Uncalibrated Cameras. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 761–764. IEEE, 1992.

[26] R. I. Hartley. Self-Calibration from Multiple Views with a Rotating Camera. In *ECCV '94: Proceedings of the Third European Conference on Computer Vision (Vol. 1)*, pages 471–478, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[27] R. I. Hartley. In Defence of the 8-point Algorithm. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 1064, Washington, DC, USA, 1995. IEEE Computer Society.

[28] R. I. Hartley. Cheirality. In *ICCV '98*, 1998.

[29] R. I. Hartley and P. Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, 1997.

[30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[31] S. Hati and S. Sengupta. Robust Camera Parameter Estimation Using Genetic Algorithm. *Pattern Recogn. Lett.*, 22(3-4):289–298, 2001.

[32] M. Hess-Flores, M. A. Duchaineau, M. J. Goldman, and K. I. Joy. Iterative Dense Correspondence Correction through Bundle Adjustment Feedback-based Error Detection. In *VISAPP (1)'10*, pages 400–405. INSTICC Press, 2010.

[33] M. Hess-Flores, D. Knoblauch, M. A. Duchaineau, K. I. Joy, and F. Kuester. Ray Divergence-Based Bundle Adjustment Conditioning for Multi-View Stereo. In *PSIVT*, 2011.

[34] H. Hirschmüller and S. K. Gehrig. Stereo Matching in the Presence of Sub-Pixel Calibration Errors. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 437–444, 2009.

[35] H. Hirschmüller and D. Scharstein. Evaluation of Cost Functions for Stereo Matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 91–92, Minneapolis, MN, June 2007.

[36] B.K.P. Horn and B.G. Schunck. Determining Optical Flow: A Retrospective. pages 81–87, 1994.

[37] D.Q. Huynh, Y.S. Chou, and H.T. Tsui. Semi-Automatic Metric Reconstruction of Buildings from Self-Calibration: Preliminary Results on the Evaluation of a Linear Camera Self-Calibration Method. *ICPR*, 04:4599, 2000.

[38] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, July 2010.

[39] 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.

[40] D. Knoblauch, M. Hess-Flores, M. Duchaineau, K. I. Joy, and F. Kuester. Non-Parametric Sequential Frame Decimation in Low-Memory Streaming Environments. *7th International Symposium on Visual Computing, Las Vegas, Nevada*, pages 363–374, 2011.

[41] D. Knoblauch, M. Hess-Flores, M. Duchaineau, and F. Kuester. Factorization of Correspondence and Camera Error for Unconstrained Dense Correspondence Applications. *5th International Symposium on Visual Computing, Las Vegas, Nevada*, pages 720–729, 2009.

[42] D. Kong and H. Tao. A Method for Learning Matching Errors In Stereo Computation. In *British Machine Vision Conference (BMVC)*, 2004.

[43] M. S. Lew, T. S. Huang, and K. Wong. Learning and Feature Selection in Stereo Matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(9):869–881, 1994.

[44] H. Li and R. I. Hartley. Five-Point Motion Estimation Made Easy. In *ICPR (1)'06*, pages 630–633, 2006.

[45] P. Lindstrom. Triangulation Made Easy. In *CVPR*, pages 1554–1561, 2010.

[46] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT Flow: Dense Correspondence across Different Scenes. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 28–42, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] M.I.A. Lourakis and A.A. Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, August 2000.

[48] D. Lowe. Object Recognition from Local Scale-Invariant Features. In *ICCV*, pages 1150–1157. Published by the IEEE Computer Society, 1999.

[49] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal On Computer Vision*, 60(2):91–110, 2004.

[50] Q.-T. Luong. *Matrice Fondamentale et Auto-Calibration en Vision par Ordinateur.* PhD thesis, Universite de Paris-Sud, Orsay, 1992.

[51] Q.T. Luong and O.D. Faugeras. Determining the Fundamental Matrix with Planes: Instability and New Algorithms. In *CVPR93*, pages 489–494, 1993.

[52] D. Martinec and T. Pajdla. 3D Reconstruction by Gluing Pair-Wise Euclidean Reconstructions, or "How to Achieve a Good Reconstruction from Bad Images". In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 25–32, Washington, DC, USA, 2006. IEEE Computer Society.

[53] R. Mayoral and M. Aurnhammer. Evaluation of Correspondence Errors for Stereo. In *17th International Conference on Pattern Recognition (ICPR'04)*, volume 4, pages 104–107, 2004.

[54] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[55] D. Nistér. Reconstruction from Uncalibrated Sequences with a Hierarchy of Trifocal Tensors. In *Computer Vision - ECCV 2000*, volume 1842 of *Lecture Notes in Computer Science*, pages 649–663. Springer Berlin / Heidelberg, 2000.

[56] D. Nistér. Frame Decimation for Structure and Motion. In *3D Structure from Images (SMILE 2000)*, pages 17–34, London, UK, 2001. Springer-Verlag.

[57] D. Nistér. Preemptive RANSAC for Live Structure and Motion Estimation. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 199, Washington, DC, USA, 2003. IEEE Computer Society.

[58] A.S. Ogale and Y. Aloimonos. Shape and the Stereo Correspondence Problem. *Int. J. Comput. Vision*, 65(3):147–162, 2005.

[59] T. Okuma, K. Sakaue, H. Takemura, and N. Yokoya. Real-Time Camera Parameter Estimation from Images for a Mixed Reality System. *ICPR*, 04:4482, 2000.

[60] Oxford Visual Geometry Group. Multi-View and Oxford Colleges Building Reconstruction. `http://www.robots.ox.ac.uk/~vgg/data/data-mview.html`, August 2009.

[61] J. Philip. A Non-Iterative Algorithm for Determining all Essential Matrices Corresponding to Five Point Pairs. *Photogrammetric Record*, 15(88):589–599, 1996.

[62] J. Philip. Critical Point Configurations of the 5-, 6-, 7-, and 8-Point Algorithms for Relative Orientation. In *TRITA-MAT-1998-MA-13*, 1998.

[63] O. Pizarro, R. Eustice, and H. Singh. Relative Pose Estimation for Instrumented, Calibrated Platforms. In *VIIth Digital Image Computing: Techniques and Applications*, 2003.

[64] M. Pollefeys and L. Van Gool. A Stratified Approach to Metric Self-Calibration, 1997.

[65] M. Pollefeys, R. Koch, and L. Van Gool. Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters. In *ICCV*, pages 90–95, 1998.

[66] M. Pollefeys, R. Koch, and L. Van Gool. Self-Calibration and Metric Reconstruction Inspite of Varying and Unknown Intrinsic Camera Parameters. *Int. J. Comput. Vision*, 32(1):7–25, 1999.

[67] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *Int. J. Comput. Vision*, 78:143–167, July 2008.

[68] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59:207–232, 2004.

[69] M. Pupilli and A. Calway. Real-Time Camera Tracking Using a Particle Filter, 2005.

[70] G. Qian and R. Chellappa. Structure from Motion Using Sequential Monte Carlo Methods. *Int. J. Comput. Vision*, 59(1):5–31, 2004.

[71] A. Rachmielowski, D. Cobza, and M. Jagersand. Robust SSD Tracking with Incremental 3D Structure Estimation. In *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, page 12, Washington, DC, USA, 2006. IEEE Computer Society.

[72] S. Ramalingam, P. Sturm, and E. Boyer. A Factorization Based Self-Calibration for Radially Symmetric Cameras. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 480–487, Washington, DC, USA, 2006. IEEE Computer Society.

[73] F. Remondino. From Point Cloud to Surface: The Modeling and Visualization Problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Feb 2003.

[74] F. Remondino, S.F. El-Hakim, A. Gruen, and L. Zhang. Turning Images Into 3-D Models (Developments and Performance Analysis of Image Matching for Detailed Surface Reconstruction of Heritage Objects). *IEEE Signal Processing Magazine*, page 55, Jul 2008.

[75] V. Rodehorst, M. Heinrichs, and O. Hellwich. Evaluation of Relative Pose Estimation Methods for Multi-Camera Setups. In *International Archives of Photogrammetry and Remote Sensing (ISPRS '08)*, pages 135–140, Beijing, China, 2008.

[76] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest. Monocular Vision for Mobile Robot Localization and Autonomous Navigation. *International Journal of Computer Vision*, 74:237–260, 2007. 10.1007/s11263-006-0023-y.

[77] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal On Computer Vision*, 47(1-3):7–42, 2002.

[78] T. W. Sederberg and R. J. Meyers. Loop Detection in Surface Patch Intersections. *Computer Aided Geometric Design*, 5(2):161–171, 1988.

[79] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society.

[80] J. Shi and C. Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[81] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846, New York, NY, USA, 2006. ACM.

[82] M.E. Spetsakis and Y. Aloimonos. Optimal Visual Motion Estimation: A Note. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):959–964, 1992.

[83] M.K. Steliaros, R.A. Packwood, and G.R. Martin. Adaptive Block Matching Motion Compensation for Low Bit-Rate Video Coding. In *Proceedings of the First Advanced Digital Video Compression Enginering*, pages 81–88, Cambridge, UK, July 1996.

[84] H. Stewénius, C. Engels, and D. Nistér. Recent Developments on Direct Relative Orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[85] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In *CVPR'08*, pages –1–1, 2008.

[86] P. Sturm. Critical Motion Sequences for Monocular Self-Calibration and Uncalibrated Euclidean Reconstruction. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1100, Washington, DC, USA, 1997. IEEE Computer Society.

[87] P. Sturm. A Case Against Kruppa's Equations for Camera Self-Calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(10):1199–1204, 2000.

[88] P. Sturm. On Focal Length Calibration from Two Views. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, volume II, pages 145– 150, Dec 2001.*, 2001.

[89] I.E. Sutherland. Three-Dimensional Data Input by Tablet. *SIGGRAPH Comput. Graph.*, 8(3):86–86, 1974.

[90] R. S. Szeliski. Prediction Error as a Quality Metric for Motion and Stereo. In *ICCV '99: Proceedings of the International Conference on Computer Vision*, volume 2, pages 781–788, Washington, DC, USA, 1999. IEEE Computer Society.

[91] E. Tola, V. Lepetit, and P. Fua. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. In *PAMI*, volume 32, pages 815–830, May 2010.

[92] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[93] C. Tomasi and T. Kanade. Shape and Motion from Image Streams: a Factorization Method Parts 2,8,10 Full Report on the Orthographic Case. Technical report, 1992.

[94] P. H.S. Torr. Geometric Motion Segmentation and Model Selection. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998.

[95] P. H.S. Torr, A. W. Fitzgibbon, and A. Zisserman. The Problem of Degeneracy in Structure and Motion Recovery from Uncalibrated Image Sequences. *International Journal of Computer Vision*, 32:27–44, 1999. 10.1023/A:1008140928553.

[96] P.H.S. Torr and A.W. Fitzgibbon. Invariant Fitting of Two View Geometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):648–650, 2004.

[97] B. Triggs. Autocalibration from Planar Scenes. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 89–105, London, UK, 1998. Springer-Verlag.

[98] B. Triggs, P.F. McLauchlan, R. I. Hartley, and A.W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer-Verlag.

[99] L.L. Wang and W.H. Tsai. Computing Camera Parameters Using Vanishing-Line Information from a Rectangular Parallelepiped. *Mach. Vision Appl.*, 3(3):129–141, 1990.

[100] J. Weng, T.S. Huang, and N. Ahuja. Motion and Structure From Two Perspective Views: Algorithms, Error Analysis, and Error Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(5):451–476, 1989.

[101] F.C. Wu, Z.Y. Hu, and F.Q. Duan. 8-Point Algorithm Revisited: Factorized 8-Point Algorithm. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 488–494, Washington, DC, USA, 2005. IEEE Computer Society.

[102] Y. Xiong and L. Matthies. Error Analysis of a Real-Time Stereo System. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1087–1093, 1997.

[103] R. Yang and M. Pollefeys. A Versatile Stereo Implementation on Commodity Graphics Hardware. *Real-Time Imaging*, 11:7–18, February 2005.

[104] R. Zaibi, A. Enis Cetin, and Y. Yardimci. Small Moving Object Detection in Video Sequences. In *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, pages 2071–2074, Washington, DC, USA, 2000. IEEE Computer Society.

[105] B. Zeisl, P.F. Georgel, F. Schweiger, E. Steinbach, and N. Navab. Estimation of Location Uncertainty for Scale Invariant Features Points. In *BMVC09*, pages xx–yy, 2009.

[106] J. Zhang, M. Boutin, and D.G. Aliaga. Robust Bundle Adjustment for Structure from Motion. *Image Processing, 2006 IEEE International Conference on*, pages 2185–2188, Oct. 2006.

[107] Z. Zhang and Y. Shan. Incremental Motion Estimation Through Local Bundle Adjustment. Technical report, 2001.

[108] Z. Zhang and Y. Shan. Incremental Motion Estimation Through Modified Bundle Adjustment. In *Proceedings of the 2003 International Conference on Image Processing (ICIP 2003)*, volume 2, pages II–343–6 vol.3, Sept. 2003.

[109] W.Y. Zhao and N. Nandhakumar. Effects of Camera Alignment Errors on Stereoscopic Depth Estimates. *Pattern Recognition*, 29(12):2115–2126, December 1996.

The following appendices describe in detail some of the basic steps and mathematical formulations used throughout this document. The first appendices, corresponding to camera calibration (Appendix A), epipolar geometry estimation (Appendix B), pose estimation (Appendix C), triangulation (Appendix D), bundle adjustment (Appendix E) and the direct linear transformation (Appendix F), each describe essential components to a scene reconstruction pipeline. Finally, Appendix G describes the theory behind a novel algorithm for computing the relative pose between two cameras using only information available from a set of image patches.

Appendix A

Camera Calibration

The objective of this appendix is to derive the conversion between 3D scene points and positions in pixel space. This requires an explanation of both the concepts of perspective projection and the pin-hole camera model, as explained in Section A.1. The camera's intrinsic parameters will be described in Section A.2, and its radial distortion model is described in Section A.3.

A.1 Perspective Projection and the Pin-Hole Camera Model

A real camera can be modeled using the *pin-hole* camera model, and such that the screen coordinates of scene points are acquired using the perspective projection model. In the pin-hole camera model, the center of the camera, C, is considered to be a point (C_X, C_Y, C_Z) at the origin of a 3D coordinate system, which will be called from now on the *camera coordinate system.* The viewing direction of the camera is pointing along the Z-axis of this coordinate system, which is also known as the *principal axis.* The *image plane* is a rectangular region where the scene being viewed projects to in space, such that the plane is parallel to the XY plane of the camera coordinate system, and is a distance f, corresponding to the camera's *focal length f*, away from the camera center C. The *principal point* corresponds to the intersection of the principal axis and the image plane. These concepts are illustrated in Fig. A.1.

A light ray between the camera center C and an arbitrary position in 3D space

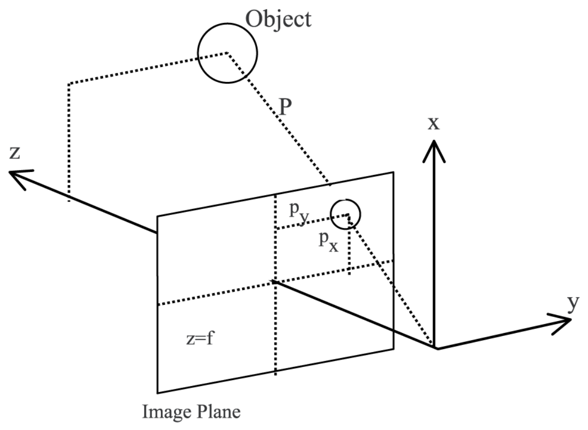


Figure A.1: Pinhole camera model and the concept of perspective projection.

and in front of the camera, $P = (X_{cam}, Y_{cam}, Z_{cam})$, gets projected on the image plane with 2D coordinates $(p_x, p_y) = (x_{cam}, y_{cam})$. The 3D coordinates of P are measured with respect to C. The 2D image plane coordinates are obtained using perspective projection, which simply applies the mathematical relationship between triangles given by Thales' theorem, as shown in Eqs. A.1 and A.2. The relationship can also be expressed as shown in Eq. A.3, where λ is simply a scale factor, and can be expressed in terms of the camera's focal length f and the depth of the 3D point, Z_{cam}, as shown in Eq. A.4. Furthermore, an equivalent matrix expression using homogeneous coordinates can be used to describe the perspective projection, as shown in Eq. A.5.

$$x_{cam} = \frac{fX_{cam}}{Z_{cam}} \tag{A.1}$$

$$y_{cam} = \frac{fY_{cam}}{Z_{cam}} \tag{A.2}$$

$$\lambda \begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix} = \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} \tag{A.3}$$

$$\lambda = \frac{Z_{cam}}{f} \tag{A.4}$$

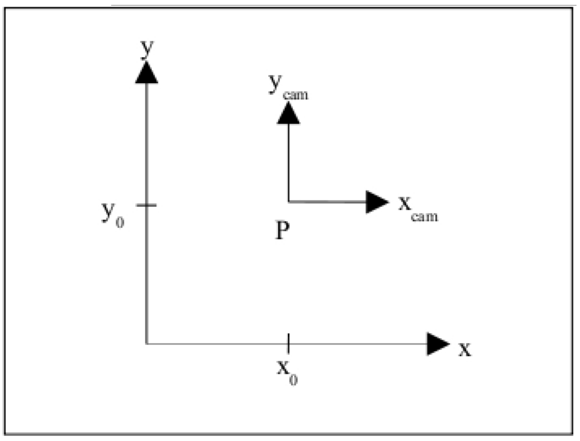


Figure A.2: Camera's principal point and coordinate system with respect to the image plane.

$$\lambda \begin{bmatrix} x_{cam} \\ y_{cam} \\ f_{cam} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix} \tag{A.5}$$

A.2 Intrinsic Parameters of the Camera

In real images, the origin of the image coordinates, for example the bottom-left corner of the image plane as seen in Fig. A.2, does not coincide with the principal point, and the scaling along each image axis is different, so the image coordinates must undergo a further transformation. The transformation consists of a 3×3 matrix known as the *camera matrix* K, which contains the camera's intrinsic parameters and provides the transformation between an image point and a ray in Euclidean 3D space. Let (x_0, y_0) correspond to the position of the camera's *principal point*, and let (x_{cam}, y_{cam}) be an image-plane position with respect to the camera's coordinate system, as shown in Fig. A.2.

The relation between camera coordinates and image pixel coordinates is given in Eqs. A.6 and A.7, where the parameters k_x and k_y have units of pixels/length. This same information can be expressed in matrix form as shown in Eq. A.8. The 3×3 matrix in Eq. A.8 corresponds to the upper-triangular camera matrix K. The 'focal distances' in the K matrix, corresponding to the $(1, 1)$ and $(2, 2)$ diagonal elements, are given by Eqs. A.9

and A.10. Dividing Eq. A.9 by Eq. A.10 results in the camera's *aspect ratio*, $\frac{\alpha_x}{\alpha_y}$. Lastly, the *pixel skew* s, which is an angular measure of the deviation of a pixel's shape from being rectangular, is normally assumed to be 0.

$$k_x x_{cam} = x - x_0 \tag{A.6}$$

$$k_y y_{cam} = y - y_0 \tag{A.7}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ f \end{bmatrix} \tag{A.8}$$

$$\alpha_x = f k_x \tag{A.9}$$

$$\alpha_y = f k_y \tag{A.10}$$

To give an example of a typical K matrix, the camera used for one of the test datasets being used, *Walnut Creek*, has the following specifications:

- Focal length: $f = 38mm$
- Pixel pitch (for both the 'x' and 'y' directions): 9.12×10^{-6} meters
- Image size: 1512×2268 pixels
- Principal point: $(756, 1134)$ (assumed to be right in the middle of the image)
- Skew s: assumed to be 0

These values result in the following K matrix:

$$K = \begin{bmatrix} 4166.6665 & 0 & 756 \\ 0 & 4166.6665 & 1134 \\ 0 & 0 & 1 \end{bmatrix}$$

A.2.1 What If the Intrinsic Camera Parameters Are Unknown?

A common problem in the literature deals with determining the camera's intrinsic parameters, most importantly the focal length, when these are unknown. Determining the camera intrinsic parameters by using only information from images taken by the camera is known as *self-calibration* [87, 64], and can be achieved if the type of motion of the camera is not a critical motion sequence [87], in which case degeneracies or ambiguous solutions are obtained.

One common approach in the literature is to use *Kruppa's equations* to help solve for the camera matrix K. These equations link the intrinsic parameters with the epipolar geometry between pairs of views [87]. By assuming constant intrinsic parameters, this and other methods based on the 'absolute conic' were developed. The absolute conic is the only conic which stays fixed under all Euclidean transformations, such that its position relative to a moving camera is constant. If the projection of the absolute conic can be determined, K can be recovered from the resulting matrix through Cholesky factorization [87, 64]. For example, a method for dealing with varying and unknown intrinsic parameters using this principle is detailed in [65]. In the work presented in this document, it is always assumed that the intrinsics are known with a fair degree of certainty, for example from camera specification sheets or EXIF tags in images. However, a generalization to try to also obtain these parameters analytically instead of assuming their values is a potential upgrade to our scene reconstruction pipelines.

A.3 Radial Distortion Model

An important aspect related to camera calibration is to take into account the *radial distortion* of the used camera(s) [38]. The effect this has on an image, specially if using what is known as a fish-eye lens, is that the acquired scene appears to radiate outwards in concentric circles instead of following a rectangular grid such as the shape of the image plane. Such a distortion pattern is seen on the right side in Fig. A.3 for the *Palmdale* dataset, which was acquired with a fish-eye lens. Since distortion values can sometimes be as high as dozens of pixels, this can seriously affect the quality of feature matching coordi-

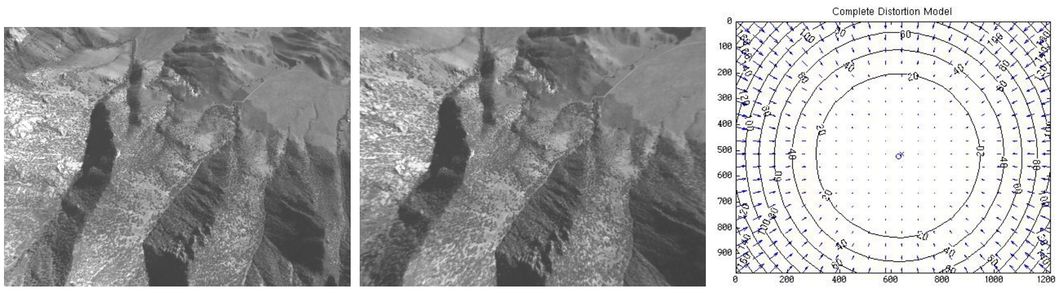


Figure A.3: Original, distorted *Palmdale* image (left) and its undistorted version (middle), with the computed distortion map on the right.

nates and therefore pose and structure estimation. It must be corrected for either during pre-processing of images or during bundle adjustment, as was done for the middle image in Fig. A.3 with respect to the original distorted image on the left, in order to achieve accurate matching results.

An image plane coordinate can be modified to take into account radial distortion as shown in Eq. A.11, where $r^2 = x^2 + y^2$ is the distortion radius for given (x, y) pixel coordinates, k_c is a 5×1 vector storing the image distortion coefficients and d_x is the tangential distortion vector measuring principal point decentering [38], as defined in Eq. A.12. The final pixel coordinates are then obtained per Eq. A.13.

$$x_d = (1 + k_c(1)r^2 + k_c(2)r^4 + k_c(5)r^6)x_{cam} + d_x \tag{A.11}$$

$$d_x = \begin{bmatrix} 2k_c(3)xy + k_c(4)(r^2 + 2x^2) \\ k_c(3)(r^2 + 2y^2) + 2k_c(4)xy \end{bmatrix} \tag{A.12}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{A.13}$$

Appendix B

Epipolar Geometry Estimation

Let x correspond to the 2D image position (x, y) of a pixel in an image of a scene, and let x' correspond to the (x', y') corresponding match coordinates in another image. The *epipolar geometry* describes the intrinsic projective geometry between the two views. This relationship is encapsulated by the *fundamental matrix* F, a rank-two 3×3 matrix such that $x'^T Fx = 0$. Fig. B.1 illustrates the concept of epipolar geometry.

Some definitions related to epipolar geometry will now be given. From Fig. B.1, it can be seen that both image points x and x' project to the same 3D point X in space, for rays starting from the respective camera centers C and C'. The 'epipolar plane' is the plane which intersects X, C and C'. The 'baseline' is the line segment in 3D space which connects the camera centers. The 'epipoles' e and e' are the image positions where the baseline intersects each image plane. Finally, the 'epipolar lines' are the segments that connect e with x and e' with x', respectively. Also, note that these lines correspond to the intersection of the epipolar plane with each respective image plane. Also, note that the back-projection of x from C to X is mapped as the line l' in the second view. In the same way, the back-projection of x' from C' to X is mapped as l in the first view. The family of possible epipolar planes, depending on the position of X, is known as the 'epipolar pencil', which all contain the baseline. Given these definitions, Section B.1 will formally define the fundamental matrix F, which mathematically encapsulates the epipolar geometry information. Section B.2 describes a variety of methods used in the literature to compute

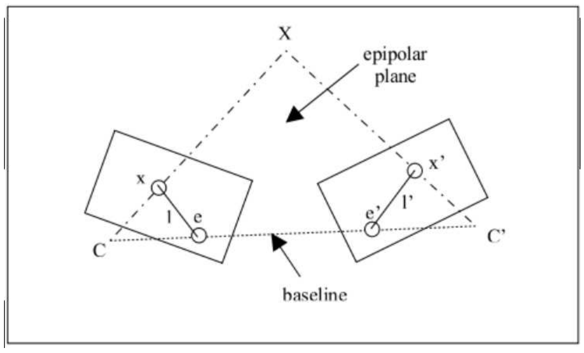


Figure B.1: Epipolar geometry, where C and C' are the camera centers, x and x' the image correspondences, e and e' the epipoles, l and l' the epipolar lines and X is the point in 3D space seen by x and x'.

F. Finally, Section B.3 discusses the definition of epipolar geometry when more than two views are present.

B.1 Properties of the Fundamental Matrix

The fundamental matrix F corresponds to the algebraic representation of the epipolar geometry. To each position x in one image, there exists a corresponding epipolar line l' in the other, such that there is a projective mapping from points to lines. Also, x is mapped to x' lying on l', and l' is obtained as the line joining x' to e'. Other important properties of F are the following. The epipolar equation is given by Eq. B.1. The matrix is of rank-two, with 7 degrees of freedom, not 9, since one is lost due to common scaling, and another one is lost since $det(F) = 0$. There are two DOF corresponding to e, two for e', and three for the epipolar line homography mapping a line through e to a line through e'. Full rank is not achieved since any x on l maps to l', so there's no inverse mapping. The epipolar lines for x and x' are given respectively by Eqs. B.2 and B.3. The epipoles e and e' are given respectively by Eqs. B.4 and B.5, such that e is the right null-space of F, while e' is its left null-space.

$$x'^T F x = 0 \tag{B.1}$$

$$l' = Fx \tag{B.2}$$

$$l = F^T x' \tag{B.3}$$

$$Fe = 0 \tag{B.4}$$

$$F^T e' = 0 \tag{B.5}$$

B.2 Computation of the Fundamental Matrix F

The F matrix can be computed either directly, using *5*, *6*, *7* or *8-point* methods as will be discussed, or through non-linear optimization methods such as ‘algebraic minimization’, ‘minimization of the epipolar distance’ or the ‘gold standard method’. An excellent overview of these methods and epipolar geometry in general is Hartley and Zisserman’s *Multiple View Geometry* book [30].

Our initial implementation uses Hartley’s ‘Normalized 8-point’ algorithm [27], or N8P. Being a direct method, it yields a unique solution and usually works as a good starting point for a posterior non-linear minimization technique. However, based on results from the literature, the *5-point* algorithm yields the best results overall. Even though 10 possible solutions are obtained for the entries of F, it doesn’t suffer from degeneracies like the *8-point* method, where in certain cases F cannot be uniquely determined. However, to better understand the methods for computing F in the literature, the N8P algorithm will be explained in detail, since it is representative in general of the methods used for this purpose. The *5-point* method is much more involved, and the reader is referred to [84] for further details. After describing N8P, other direct and non-linear methods from the literature will be briefly described.

B.2.1 8-point Algorithm

With direct methods such as the *8-point* method, the F matrix can be recovered from feature matches alone, without needing any additional information such as camera intrinsics. Assume a pair of matches (x, x'). The epipolar equation for the pair is given by Eq. B.1. If the left side of the equation is expanded, the result in Eq. B.6 is obtained. For ‘n’ pairs of matches, a system of the form $Af = 0$ can be solved to obtain the 1×9 vector

f, whose entries make up the 3×3 F matrix, per Eq. B.7.

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} \tag{B.6}$$

$$\begin{bmatrix} x_1x_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_n & x_ny'_n & x_n & y_nx'_n & y_ny'_n & y_n & x'_n & y'_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{B.7}$$

With this mathematical definition in mind, the *8-point* method has the following properties. First, the solution is determined up to scale only. For exactly 8 points, a unique solution is obtained. For more than 8 points, a least-squares solution is required on the over-constrained system. To do this, first the singular value decomposition $A = UDV^T$ of the data matrix A is obtained, and the solution for f is the last column of V, which corresponds to the smallest singular value. This effectively minimizes $\|Af\|$ subject to the constraint $\|f\| = 1$.

Normalized 8-point algorithm

Without normalization of the image coordinates, the standard *8-point* algorithm becomes sensitive to the origin of the image coordinates and the scale. The condition number of the data matrix A becomes large, and this can result in numerical instabilities. Hartley [27] proposed a method, the *Normalized 8-point* algorithm or N8P, to perform this normalization, such that much more robust results can be obtained when using the *8-point* algorithm.

The following steps are performed. First, all image coordinates are translated such that their centroid lies at the origin $(0, 0)$. This implies first finding the centroid of all x_i positions and independently the centroid of all x'_i positions, and multiplying the original coordinates by respective transformation matrices that perform the translation to the centroid, as seen in Eqs. B.8 and B.9.

$$\hat{x}_i = T x_i \tag{B.8}$$

$$\hat{x}'_i = T' x'_i \tag{B.9}$$

After translation, all coordinates are scaled so that the RMS distance from the origin is $\sqrt{2}$. Using the normalized coordinates, the F matrix is obtained using the *8-point* method, such that $\hat{x}_i'^T F \hat{x}_i = 0$. To ensure that F is of rank-two, a singularity constraint must be applied, which finds the closest singular F' to F such that the Frobenius norm $\|F - F'\|$ is minimized. To do this, given the singular value decomposition $F = UDV^T$, where $D = diag(r, s, t)$, with $r \geq s \geq t$, with U and V as orthogonal matrices, the closest singular F' can be obtained per Eq. B.10. The final step in the N8P algorithm is to denormalize by the original transformation matrices, as shown in Eq. B.11.

$$F' = U diag(r, s, 0) V^T \tag{B.10}$$

$$F'' = T'^T F' T \tag{B.11}$$

B.2.2 Other Linear Methods Used to Compute F

The following methods have also been used in the literature for the computation of the fundamental matrix.

7-point method

Details on this method can be found in Hartley and Zisserman's book [30], but a summary of the main algorithm is provided. The F matrix has 9 entries but is defined only up to scale. The singularity condition $det(F) = 0$ gives a further constraint, which is cubic since F has three rows. Since F has only 7 degrees of freedom, it is possible to solve for it from just 7 point correspondences. By forming the 7×9 set of equations $Af = 0$, which yields a two-dimensional solution set, the general solution using SVD has the form $f = \lambda f_0 + \mu f_1$, or in matrix terms, $F = \lambda F_0 + \mu F_1$. Thus, the condition $det(F) = 0$ gives a cubic equation in λ and μ, and either one or three real solutions for the ratio $\lambda : \mu$, which must be individually tested for the correct solution.

6-point methods

Two different 6-point methods have been developed: one by Pizarro et. al [63] and a previous one by Philip [61], who himself proved in [62] that the original method does not work for planar scenes. They are both based on the essential matrix and involve its 'trace constraint', defined in Eq. B.12.

$$2EE^TEtrace(EE^T)E = 0 \tag{B.12}$$

- Method of Pizarro et al.: The 9 equations from Eq. B.12 are composed into a 9×10 matrix from which the four rows corresponding to the four largest singular values are selected. From the four resulting equations, a sixth-degree polynomial is computed and then solved to obtain the entries of E.
- Linear six-point solver from Philip: The 9 equations from Eq. B.12 are composed into a 9×10 matrix and the unknown entries of E are solved for linearly, for example with SVD.

5-point method

In this method, developed by Nistér and Stewenius [84], solutions are found as roots of a 10^{th}-degree polynomial. First, it uses the linear equations from the epipolar constraint to parametrise the essential matrix with three unknowns. Then, it uses the rank constraint and the trace constraint of E to build ten third-order polynomial equations in the three unknowns. Finally, a *Gröbner basis*, which is a subset of the terms of the polynomials, is derived, which is then used to construct a 10×10 'action matrix' whose eigenvalues and eigenvectors encode the ten solutions to the polynomial.

B.2.3 Non-Linear Methods Used to Compute F

Detailed explanations on the three main nonlinear methods can be found in [30], but an overview of each will now be given.

Algebraic minimization

This method minimizes $\|Af\|$ subject to the two constraints $\|f\| = 1$ and $det(F) = 0$. The main idea is to vary the epipole e such as to minimize the algebraic error $\|Af\| = \|AEm\|$, where E is a 9×9 matrix composed of the entries of e, which is assumed known, such that $F = M[e]_x$, where the entries of the matrix M correspond to the values of the vector m. The Levenberg-Marquardt algorithm is used to iteratively minimize the error, and involves computing an SVD at each step. A non-iterative algorithm involving SVD also exists.

Minimization of the epipolar distance

The epipolar distance is defined as the distance of point x' to the epipolar line Fx. The algorithm is as follows. Let $Fx = (\lambda, \mu, \nu)$ and $x' = (x, y, 1)^T$. The distance measure used is known as the 'Sampson distance', and is defined in Eq. B.13. The total cost function over all x_i and x'_i is given by Eq. B.14, which must be minimized over the parametrization of F.

$$d(x', Fx) = x'^T Fx(\lambda^2 + \mu^2)^{-\frac{1}{2}} = \frac{x'^T Fx}{((Fx)_1^2 + (Fx)_2^2)^{\frac{1}{2}}} \tag{B.13}$$

$$\sum_i d(x'_i, Fx_i)^2 = \sum_i \frac{x_i'^T Fx_i}{((Fx_i)_1^2 + (Fx_i)_2^2)^{\frac{1}{2}}} \tag{B.14}$$

Maximum-likelihood (Gold-standard) method

This algorithm assumes Gaussian image noise, and requires an initial 3D reconstruction. It works as follows. Let $P = [I|0]$ and $P' = [M|t]$ be the projection matrices corresponding to the two cameras, as explained further in Appendix C. The 3×3 matrix M should be a good estimate of the second camera's rotation with respect to the first, and t a good estimate of its 3D translation. Let $X_i = (X_i, Y_i, Z_i, W_i)^T$ be the 3D position in homogeneous coordinates for the i_{th} 3D point. Then, the values $\hat{x}_i = PX_i = (x_i, y_i, 1)^T$ and $\hat{x}'_i = P'X_i = (x'_i, y'_i, 1)^T$ are computed. Given these pixel coordinates, the objective is to iterate over P' and X_i to minimize the cost function provided in Eq. B.15. This minimization involves a total of $3n + 12$ parameters, corresponding to 12 parameters for the projection matrix P' and three for each point X_i. Finally, once $P' = [M|t]$ is obtained, the final step is to compute $F = [t]_x M$.

$$\sum_i (d(x_i, \hat{x}_i)^2 + d(x'_i, \hat{x}'_i)^2) \tag{B.15}$$

B.2.4 Using Random Sample Consensus (RANSAC)

The 'Random Sample Consensus' or *RANSAC* algorithm is based on testing hypotheses and choosing the one for which the greatest number of inliers is achieved. Details on the method in general can be found in its original publication [18], but a brief outline of how it is applied to the specific case of estimating a fundamental matrix F from a set of correspondences is given here. The method is based on the $7-point$ algorithm as this has given the best results [30], so the first step in the algorithm is to select a random sample of 7 feature matches, from which the fundamental matrix is computed. The next step is to measure the number of inliers for this hypothesis value of F, which corresponds to the number of matches for which the Sampson distance (Eq. B.13) is lower than a given threshold. After running the algorithm a number of times, the F matrix with the largest inlier support is chosen. Finally, a final F matrix is estimated from only the inlier matches.

B.2.5 Degeneracies in the Estimation of F

A *degeneracy* in the estimation of the epipolar geometry occurs either when the epipolar geometry is undefined or when it cannot be uniquely determined. There are two cases for degeneracy, known as the 'motion degeneracy' and the 'structure degeneracy'. The motion degeneracy arises when the relative movement between frames consists of only a rotation and no translation. In such cases the epipolar geometry between the views is undefined. The structure degeneracy occurs when matches between frames correspond to scene points that lie on the same plane in space. In this case the epipolar geometry is not unique, as there is a two-parameter family of possible solutions. For example, it has been proven that the N8P algorithm fails when matches for scene points that lie on the same plane are used. However, the *5-point* method [84] has been proven to work well under these circumstances. Additionally, the work in [23] deals with cases where correspondences include a low percentage of inlier correspondences and/or a large subset of the inliers is consistent with a degenerate configuration of the epipolar geometry that is totally incorrect. More details on epipolar geometry degeneracy and its detection can be found in Chapter 5.

B.3 Epipolar Geometry Between More Than Two Views

The description of the epipolar geometry can be extended to three views, using the *trifocal tensor* [3]. This tensor plays a similar role to the fundamental matrix for two views, but unlike F, it also relates lines, not just points. Mixed combinations of lines and points can also be related through the tensor.

The estimation of the trifocal tensor is more difficult than the estimation of the two-view epipolar geometry, since several constraints must be evaluated to determine if a given tensor is valid. In [3], a threading function operates on two consecutive fundamental matrices and connects them using the trifocal tensor. The threading operation guarantees that consecutive camera matrices are consistent with a unique 3D model, without ever recovering a 3D model.

Appendix C

Pose Estimation

In Appendix B it was described how to obtain the 'fundamental matrix', which is the mathematical description of the epipolar geometry between two views. As it will be discussed in this section, its importance lies in that it encapsulates the relative pose between the two cameras, which is defined as a 3×3 rotation matrix R and a 3D translation vector t, better known as the camera's extrinsic parameters, given that the intrinsic calibration of the camera is known, as was described in Appendix A. Section C.1 will define the camera's extrinsic parameters. Section C.2 describes the calibrated version of the fundamental matrix, known as the 'essential matrix' E, and how it can be decomposed into extrinsic parameters. Finally, Section C.3 discusses other methods for pose estimation from the literature.

C.1 Extrinsic Camera Parameters and Projection Matrices

In Appendix A, the relation between image coordinates and scene positions was derived with respect to the camera's coordinate frame. In general, however, the camera's position in 3D space does not coincide with the origin of the world coordinate system. This implies that 3D coordinates with respect to the camera coordinate system must undergo a Euclidean motion, described by a 3×3 rotation matrix R and a 3D translation vector t, in order to determine their positions with respect to the world coordinate system. Eq. C.1 describes this relationship between camera coordinates $(X_{cam}, Y_{cam}, Z_{cam})$ and world coordinates (X, Y, Z).

$$\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{C.1}$$

If the conversion from image pixel coordinates to camera coordinates (as encapsulated by K and described in Appendix A) is concatenated with the conversion from camera coordinates to world coordinates, under the perspective projection model of Eq. A.5, this results in a 3×4 matrix known as a *projection matrix* P. This matrix expresses the relationship between 2D image positions measured in pixels and the corresponding 3D points with respect to the origin of the world coordinate system, and contains the information from the intrinsic parameter matrix K as well as the extrinsic parameters R and t, as displayed in Eq. C.2.

$$\tilde{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K[R|t]X = PX \tag{C.2}$$

Projection matrices have the following main characteristics:

- The camera center C is located at $(X, Y, Z)^T = -R^T t$. It can also be obtained as the null-vector of P, such that $PC = 0$.
- P has 11 degrees of freedom, and is of rank-three.

Now that the camera's extrinsic parameters have been explained, it will be discussed how these can be recovered from the previously-obtained fundamental matrix, through a matrix called the *essential matrix* E. For this discussion, it will always be assumed that the first camera is placed at the origin of the world coordinate system, and thus its projection matrix is $P = K[I|0]$. The second camera will be defined by $P' = K[R|t]$.

C.2 The Essential Matrix

Let a projection matrix $P' = K[R|t]$ and a 2D image point $x = P'X$. If the K matrix is known, let $\hat{x}$ correspond to the image point expressed in normalized coordinates, as shown in Eq. C.3. The value $\hat{x}$ can be interpreted like an image of X with the camera $[R|t]$ having the identity matrix I as its calibration matrix. In this way, $K^{-1}P' = [R|t]$ corresponds to the normalized camera matrix. Let $P = [I|0]$ and $P' = [R|t]$ be a pair of normalized camera matrices. Then E is the fundamental matrix corresponding to the normalized cameras, as shown in Eqs. C.4 and C.5.

$$\hat{x} = K^{-1}x = [R|t]X \tag{C.3}$$

$$E = [t]_x R = R[R^T t]_x \tag{C.4}$$

$$\hat{x}'^T E \hat{x} = 0 \Rightarrow x'^T K'^{-T} E K^{-1} x = 0 \Rightarrow E = K'^T F K \tag{C.5}$$

The matrix E obtained in (C.5) is a very important quantity known as the *essential matrix* E, which is basically a special case of F for image coordinates normalized by the camera matrix. Now, the properties of E will be examined, and it will be seen how the camera's extrinsic parameters can be recovered directly from it. A detailed mathematical derivation is found in [82, 100], but here only the main concepts involved will be explained.

C.2.1 Properties of the Essential Matrix E

The essential matrix has the following properties. First, it has has five degrees of freedom; R and t have three each but there's an overall scale ambiguity. Also, E has two equal singular values and the third is null. Let W and Z be respectively the orthogonal and skew-symmetric matrices defined by Eqs. C.6 and C.7. Then E can be decomposed through the singular value decomposition as expressed in Eq. C.8, with $S = kUZU^T$, where k is simply a scale factor.

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{C.6}$$

$$Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{C.7}$$

$$E = [t]_x = SR = Udiag(1,1,0)(WU^T R) \tag{C.8}$$

As seen in Eq. C.5, the essential matrix E can be computed from normalized image coordinates (Eq. C.9) or from F if the K matrix is known, as shown in Eq. C.10.

$$x'^T Ex = 0 \tag{C.9}$$

$$E = K'^T FK \tag{C.10}$$

The camera matrix P' can be retrieved from E up to scale and a four-fold ambiguity, and this is the main importance of the E matrix. If E is factorized as $E = SR$, S as defined per Eq. C.11 determines the translation t up to a sign, while there are two possible and valid values for R, which will be called $R1$ and $R2$, defined respectively in Eqs. C.12 and C.13.

$$S = UZU^T \tag{C.11}$$

$$R1 = UWV^T \tag{C.12}$$

$$R2 = UW^T V^T \tag{C.13}$$

The orthogonal matrices U and V are obtained from the singular value decomposition of E. Thus, for a given $E = Udiag(1,1,0)V^T$ and the projection matrix for the first camera $P = [I|0]$, there are four possible solutions for the second camera's projection matrix, where u_3 and v_3 correspond to the last columns of U and V, respectively:

$$\begin{gathered}
P' = [UWV^T| + u_3] = [UWV^T| - R1 * -v_3] \\
P' = [UWV^T| - u_3] = [UWV^T| - R1 * v_3] \\
P' = [UW^T V^T| + u_3] = [UW^T V^T| - R2 * -v_3] \\
P' = [UW^T V^T| - u_3] = [UW^T V^T| - R2 * v_3]
\end{gathered}$$

The 3D point X will be in front of both cameras in one solution only. This concept is known as *cheirality*, and is further discussed in [28]. How to choose the correct of the four solutions for P' based on this constraint will be explained in further detail later, after first introducing the concept of triangulation, in Appendix D.

C.3 Pose Estimation Through Other Methods

It is worth noting that a number of different approaches have also been taken to find the extrinsic camera parameters. A few examples will be given in this section.

C.3.1 Pose Estimation Using Optical Flow

In two publications by the same authors [9, 10], a procedure is described for self-calibration of a moving camera from instantaneous optical flow, which allows the extrinsics and some intrinsic parameters of the camera to be determined solely from the instantaneous positions and velocities of a set of image features. The proposed method relies on the use of a differential epipolar equation that relates optical flow to the intrinsics and extrinsics of the camera. The information about the camera's extrinsics and internal geometry enters the differential epipolar equation via two matrices. It emerges that the optical flow determines the composite ratio of some of the entries of the two matrices. It is shown that a camera with unknown focal length undergoing arbitrary motion can be self-calibrated via closed-form expressions in the composite ratio.

C.3.2 Pose Estimation Using Vanishing Points and Lines

'Vanishing points' are points to which parallel lines appear to converge, due to perspective projection. The set of all vanishing points on a projective plane constitutes a 'vanishing line'. A good example of their use appears in [11], where in the first step, the focal length and principal point are recovered from a single image of a cube. In the second step, the extrinsic parameters are recovered from an image stereo pair of a suitable planar pattern. By matching the corresponding vanishing points in the two images, the rotation matrix can first be computed, and then the translation vector is estimated by means of a simple triangulation. Another approach using vanishing lines and rectangular parallelepipeds is presented in [99].

Appendix D

Triangulation

In Appendix C, it was shown how the relative pose between two cameras can be estimated, most commonly through the decomposition of the epipolar geometry-based 'essential matrix'. This appendix will describe techniques to recover the 3D structure of a scene given a set of sparse or dense feature matches, the camera intrinsics and the relative pose (R, t) between the cameras, assuming that the first camera is placed at the origin of the world coordinate system. Optionally, intrinsic and extrinsic information can be encapsulated together in projection matrices, which can be used as inputs. Section D.1 will describe the most common method for triangulation in the literature, known as *linear triangulation*. Section D.2 describes a more-accurate yet more expensive method, known as *optimal triangulation*.

D.1 Linear Triangulation

The input to linear triangulation for the computation of scene structure is a set of matches (x_i, x'_i) between two images and the corresponding 3×4 projection matrices for each camera, respectively P and P'. Let $x = PX$ be the image plane coordinates of a scene point X, and $x' = P'X'$ the image plane coordinates of its match x' given the scene point X'. Linear triangulation makes use of the fact that X and X' should be equal since the feature matches x and x' should yield the same 3D point. However, with noise, these back-projected lines usually do not intersect. Bearing in mind that there are four possible values for P', as explained in Appendix C, the procedure described in the next steps is used to both obtain a 'test' 3D point value from a single pair or pairs of feature matches and determine which is the correct P' matrix. Once this has been determined, the same procedure used to obtain the mentioned 3D point is followed for every pair of matches to obtain all 3D points that make up the scene's structure.

Assuming that the i_{th} feature match is tested, and since X_i should equal X'_i, the relations for P and P' shown respectively in Eqs. D.1 and D.2 should hold. Using normalized image coordinates, the expressions in Eqs. D.3 and D.4 are obtained.

$$P = KP_{cam} \Rightarrow K^{-1}x_i = P_{cam}X_i \tag{D.1}$$

$$P' = KP'_{cam} \Rightarrow K^{-1}x'_i = P'_{cam}X_i \tag{D.2}$$

$$x_{cam,i} \equiv P_{cam}X_i \tag{D.3}$$

$$x'_{cam,i} \equiv P'_{cam}X_i \tag{D.4}$$

From this, 2D image positions up to a scale factor w can be obtained from the rows of P_{cam}, as shown in Eq. D.5. The same procedure can be followed for x'_{cam}, by using P'_{cam}. Expanding Eq. D.5, the result in Eq. D.6 is obtained. If the exact same equations are also set up for $x'_{cam,i}$, a 4×4 system of the form $AX = 0$ is obtained. Once A has been set up, let A_{norm} be the normalized version of A, defined per Eq. D.7.

$$w \begin{bmatrix} x_{cam,i} \\ y_{cam,i} \\ 1 \end{bmatrix} = \begin{bmatrix} P_{cam,1} \\ P_{cam,2} \\ P_{cam,3} \end{bmatrix} X_i \tag{D.5}$$

$$\begin{bmatrix} P_{cam,3}x_{cam,i} - P_{cam,1} \\ P_{cam,3}y_{cam,i} - P_{cam,2} \end{bmatrix} X_i = 0 \tag{D.6}$$

$$A_{norm} = \begin{bmatrix} \frac{A_1}{||A_1||} \\ \frac{A_2}{||A_2||} \\ \frac{A_3}{||A_3||} \\ \frac{A_4}{||A_4||} \end{bmatrix} \tag{D.7}$$

Finally, X_i is obtained as the eigenvector corresponding to the smallest eigenvalue of ${A_{norm}}^T A_{norm}$, or also the last column of V in the singular value decomposition $A_{norm} = UDV^T$. Obtaining X_i through the use of Eqs. D.1 - D.7 is known as 'linear triangulation'. The obtained 3D point is in homogeneous coordinates. With X_i, the depths in front of each camera are now computed with the expressions shown in Eqs. D.8 and D.9, where r_3 and r_3' correspond to the last rows of the left 3×3 sub-matrices of P and P', respectively. Also, w and w' correspond to scale factors, such that pixel coordinates x and x' are defined as shown respectively in Eqs. D.10 and D.11.

$$Depth(X, P_{cam}) = \frac{sign(det(I))w}{W||r_3||} = \frac{w}{W} \tag{D.8}$$

$$Depth(X, P'_{cam}) = \frac{sign(det(R))w'}{W||r'_3||} \tag{D.9}$$

$$x = P_{cam}X = w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{D.10}$$

$$x' = P'_{cam}X = w' \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{D.11}$$

$$\tilde{X} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{D.12}$$

Now, Eqs. D.1 - D.9 must be evaluated for the four possible P'_{cam} matrices. Finally, the correct P'_{cam} is chosen such that positive depths are obtained for both images. The final step consists of denormalizing the camera matrices, as shown in Eqs. D.13 and D.14. Since we now have both P and the correct P', linear triangulation is used to obtain homogeneous 3D points $\tilde{X}_i$ for every pair of matches, with X_i as the final 3D scene point per Eq. D.15.

$$P = KP_{cam} \tag{D.13}$$

$$P' = KP'_{cam} \tag{D.14}$$

$$\tilde{X}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ T_i \end{bmatrix} \Rightarrow X_i = \frac{1}{T_i} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{D.15}$$

D.1.1 Reconstruction Ambiguity

If no metric information from the scene is available, but the intrinsic parameters from the cameras are known, the scene can be reconstructed up to a *similarity transformation*, consisting of a rotation, translation and scaling factor, such that true angles and length ratios are respected. However, this leads to what is known as the 'two-view reconstruction ambiguity', and can be stated as follows. Given image correspondences x_i and x'_i, the reconstruction (P, P', X_i) results, with $x_i = PX_i$ and $x'_i = P'X_i$. However, Eqs. D.16 and D.17 show that the same pixel positions are obtained if using different projection matrices and points, as evidenced by multiplication using a 4×4 similarity transformation H and its inverse. Such an equivalent reconstruction yields the exact same image points. If the cameras are not intrinsically calibrated, the same ambiguity occurs, though H would be a 4×4 projective transformation instead of a similarity, where angles and length ratios are not respected, only image positions, and a projective reconstruction is the most that can be accomplished.

$$x_i = PX_i = PHH^{-1}X_i = \tilde{P}\tilde{X}_i \tag{D.16}$$

$$x'_i = P'X_i = P'HH^{-1}X_i = \tilde{P}\tilde{X}'_i \tag{D.17}$$

D.2 Optimal Triangulation

A method which has been proven to be more accurate than linear triangulation, known as *optimal triangulation*, was developed by Hartley and Sturm [29]. This algorithm uses the previously-computed epipolar geometry in order to move the original feature match positions x and x' over their respective image planes such that they end up at the closest positions which lie on epipolar lines. This results in minimizing a 6^{th}-order polynomial, using parameters from the epipolar lines and fundamental matrix. This is performed prior to applying linear triangulation, since the new positions for x and x' are ensured to lie on the same epipolar plane as the scene point they represent, and therefore any triangulation method would triangulate directly to that 3D point in space. This method was implemented and tried in our pipeline. However, the difference both visually and in the new position of the feature matches is usually very small, and with a much greater processing time from solving the 6^{th}-order polynomial for each point.

Lindstrom's 'fast triangulation' algorithm [45] provides a way to solve essentially the same cost function but using a quadratic equation instead, and has proven to be the fastest and most accurate algorithm so far for the triangulation of scene structure. Specifically, it is based on minimizing the L_2 reprojection error, and re-writes optimal triangulation's equations in terms of Kronecker products, which allows for terms to cancel out. Convergence occurs in exactly two iterations, so the method is non-iterative, and agrees to very high precision with the original optimal triangulation result [29], but with higher stability and 1-4 orders of magnitude greater speed. Additionally, unstable camera configurations are handled with great results, where other methods such as linear triangulation with near-parallel cameras will in general not work.

Appendix E

Bundle Adjustment

As was discussed in Sections C and D, the result of pose estimation and triangulation is respectively the projection matrices P and P' and a set of 3D points, one corresponding to each feature match. If using the 'Direct Linear Transformation' (Appendix F) or some other method to extend to multiple views, eventually there would be a reconstruction consisting of M cameras viewing N 3D points. If all estimates were perfect, there would be a set of rays starting from each camera center, going through each corresponding pixel in each image plane and then on to intersect at an exact 3D position is space. Since in general this will not occur, the objective of *bundle adjustment* is to adjust these rays in such a way that the 'total reprojection error' of the 3D points with respect to their corresponding 2D feature tracks in each camera is minimized. The end result of this minimization is a change in both the positions of the original 3D points as well as in the cameras' projection matrices $P_1 .....P_M$, where intrinsic and radial distortion parameters may be allowed to vary in the minimization along with the pose parameters, which are typically optimized.

E.1 Standard Bundle Adjustment

The cost function which is traditionally minimized can be expressed as the sum of squares of potentially a very large amount of non-linear, real-valued functions, each corresponding to the geometric reprojection error between each 3D point and its corresponding feature track, as shown in Eq. E.1.

$$min(a_i, b_j) \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij}(d(Q(a_i, b_j), x_{ij}))^2 \tag{E.1}$$

Since each 3D point has 3 parameters and each camera has 11, this minimization involves a total of $3N + 11M$ parameters. The following notation is used:

- N 3D points are seen in M cameras
- x_{ij}: projection of the i_{th} point on image j
- v_{ij}: binary variables that equal '1' if point i is visible in image j, and '0' otherwise
- a_j: vector that parametrizes each camera j
- b_i: vector that parametrizes each 3D point i
- $Q(a_j, b_i)$: predicted projection of point i on image j
- $d(x, y)$: Euclidean distance between the image points represented by vectors x and y

The minimization is achieved using non-linear least-squares algorithms, from which *Levenberg-Marquardt* has proven to be one of the most successful due mainly to its use of an effective damping strategy that lends it the ability to converge quickly from a wide range of initial guesses. By iteratively linearizing the function to be minimized in the neighborhood of the current estimate, the LM algorithm involves the solution of linear systems known as the 'normal equations'. The solution of such linear systems determines an increment to the current estimate. In the particular case of bundle adjustment, these equations have a sparse block structure due to the lack of interaction between the parameters, as shown in Fig. E.1.

This sparse block structure can be exploited to greatly speed up the algorithm, which is inherently time-consuming and computationally expensive from the minimization involving perhaps millions of parameters.

A sparse bundle adjustment $C/C++$ package known as *SBA*, written by Lourakis and Argyros [47], is widely used to implement bundle adjustment given initial structure

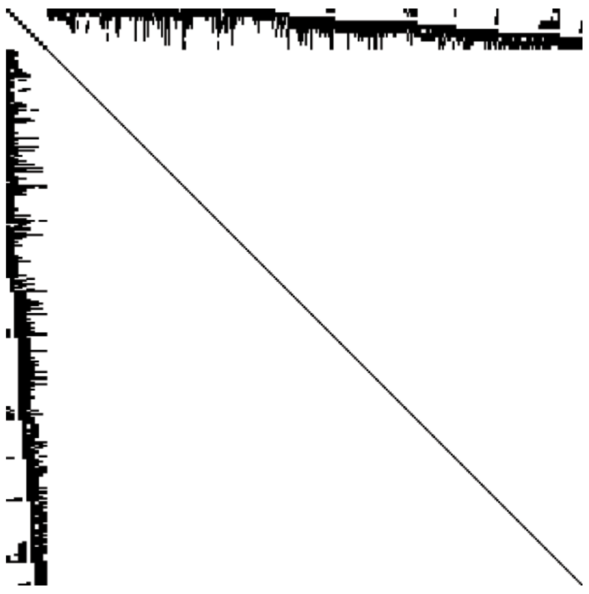


Figure E.1: Sparsity pattern of a 992 × 992 normal equations matrix.

and pose estimates. In this sparse variant of the LM algorithm, the zeros pattern is explicitly taken into account to avoid storing and operating on such elements. SBA relies on 'LAPACK' to solve the normal equations resulting in the Levenberg-Marquardt minimization process. It allows for the minimization of both structure and pose parameters, or alternatively only structure or pose parameters. Intrinsics and radial distortion may be allowed to vary in the minimization as well, though typically with an increase in the number of iterations and processing time. The SBA program requires at least the following input information, depending on the desired parameter optimization:

- Initial estimates for the camera pose parameters, with a separate line for each camera, containing 7 parameters, a 4-element quaternion for rotation and a 3-element vector for translation. For example, in the three-view case:

```
1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.999739 0.000005 -0.008636 -0.021136 0.001269 0.000208 0.000099
0.997184 0.002234 -0.051503 -0.054470 0.002923 0.000172 0.000044
```

- Initial estimates for the 3D point parameters along with their respective image feature matches, made up of lines of the form:

```
X  Y  Z   nframes   frame0  x0  y0  frame1  x1  y1  ...
```

- Camera intrinsic calibration parameters, provided as a 3×3 camera calibration matrix K in a file.

For example, the following output is obtained when using a camera with fixed intrinsics across 7 frames:

Starting BA with fixed intrinsic parameters SBA using 465 3D pts, 7 frames and 1916 image projections, 1437 variables. Method BA-MOTSTRUCT, expert driver, analytic jacobian, fixed intrinsics. SBA returned 19 in 19 iter, reason 2, error 0.675396 [initial 19.0947], 26/19 func/fjac evals, 26 lin. systems. Elapsed time: 0.33 seconds, 330.00 msecs.

The above means that SBA applied 19 iterations of the Levenberg-Marquardt algorithm on 1437 variables, lowering the total reprojection error from 19.0947 pixels to 0.675396 pixels in 0.33 seconds.

The following example shows a comparison of what happens when intrinsics are also optimized, for the *Leuven City Hall* [85] dataset. Frame pair $0-1$ was used, and only a sparse reconstruction was obtained. In general, convergence takes significantly longer and usually reaches the maximum number of allowed iterations, set by default at 150. This shows that optimizing over intrinsics should be used with care, and in our experience it has given better results to leave this step for the very final bundle adjustment. All intermediate bundle adjustments should be carried out only over pose and structure estimates, keeping both intrinsics and radial distortion parameters fixed.

Optimizing only pose and structure:

SBA using 948 3D pts, 2 frames and 1896 image projections, 2856 variables. Method BA-MOTSTRUCT, expert driver, analytic Jacobian, fixed intrinsics, without covariances. SBA returned 35 in 35 iter, reason 2, error 11.2591 [initial

170.885], 44/35 func/fjac evals, 44 lin. systems. Elapsed time: 0.28 seconds, 280.00 msecs.

Optimizing all intrinsics, radial distortion parameters, pose and structure:
SBA using 948 3D pts, 2 frames and 1896 image projections, 2876 variables. Method BA-MOTSTRUCT, expert driver, analytic Jacobian, without covariances, variable distortion (3 fixed), variable intrinsics (2 fixed). SBA returned 150 in 150 iter, reason 3, error 2.3036 [initial 172.986], 156/150 func/fjac evals, 184 lin. Systems. Elapsed time: 4.30 seconds, 4300.00 msecs.

The final example is for a dense reconstruction between frame pair $1-3$ of the *Palmdale* dataset. It goes to show how extremely time-consuming bundle adjustment can be for the dense reconstruction case, even when using only two frames, and especially if intrinsics and radial distortion are included as part of the optimization. This is one of the main reasons why an initial sparse approach for pose and structure estimation is preferred, deferring the dense reconstruction process until later on once an accurate sparse structure has been estimated along with the associated poses.

Optimizing only pose and structure:
SBA using 294665 3D pts, 2 frames and 589330 image projections, 884007 variables. Method BA-MOTSTRUCT, expert driver, analytic Jacobian, fixed intrinsics, without covariances. SBA returned 150 in 150 iter, reason 3, error 25.0539 [initial 131068], 191/150 func/fjac evals, 190 lin. Systems. Elapsed time: 162.27 seconds, 162270.00 msecs.

Optimizing all intrinsics, radial distortion parameters, pose and structure:
SBA using 294665 3D pts, 2 frames and 589330 image projections, 884027 variables. Method BA-MOTSTRUCT, expert driver, analytic Jacobian, without covariances, variable distortion (3 fixed), variable intrinsics (2 fixed). SBA returned 47 in 47 iter, reason 2, error 26148.3 [initial 131068], 52/47 func/fjac

evals, 68 lin. Systems. Elapsed time: 247.21 seconds, 247210.00 msecs.

Other good references regarding bundle adjustment are [30], [17] and [98], which is the most detailed survey on this particular subject. In [106], a basis of equations for formulating an improved bundle adjustment cost function is presented. It involves less unknowns than the ones used in standard BA by eliminating the camera orientation parameters through algebraic manipulation.

E.2 Weighted Bundle Adjustment

As mentioned, the Levenberg-Marquardt algorithm is based on solving the normal equations at each iteration. In *weighted* bundle adjustment, each input feature is weighted differently with the objective of improving convergence by giving less weight to those features that are more likely to be inaccurate. Such weights are implemented as covariances. The normal equations have the form shown in Equation E.2, but when using weighted bundle adjustment, the equations change to the form shown in Equation E.3, where Σ corresponds to a block-diagonal matrix consisting of 2×2 covariance matrices for each input feature, J is the parameter Jacobian matrix, δ_p the parameter update step, μ the damping term and ϵ the error vector.

$$(J^T J + \mu I)\delta_p = J^T \epsilon \tag{E.2}$$

$$(J^T \Sigma_x^{-1} J + \mu I)\delta_p = J^T \Sigma_x^{-1} \epsilon \tag{E.3}$$

Appendix F

Direct Linear Transformation for 3D-2D Registration

The *Direct Linear Transformation* (DLT) is a method for 3D-2D registration, such that new images can be incorporated into an existing reconstruction. The derivation can be found in the work of Sutherland [89], but the main concepts will be explained here.

To give an example, assume a reconstruction has previously been obtained between two views, and that a set of sparse or dense matches exists between the second camera and a third camera which will be registered to the initial two-view reconstruction. The output of the DLT process in this case is a 3×4 projection matrix P for the third camera such that $x_{i,3} = PX_i$, where $x_{i,3}$ corresponds to the pixel positions in the third image of the i_{th} feature track that spans all three images, while X_i is the 3D position of the scene point corresponding to that track, which had been computed from the first two images.

In general, the projection equation given by Eq. F.1 can also be written as a cross product, as shown in Eq. F.2. This can be further re-written in matrix form such that Eq. F.3 is obtained, where P^1, P^2 and P^3 correspond to the rows of the projection matrix P. The last row of this system is redundant, so a 2×9 system of the form $A_i p = 0$ remains. For the three-view case mentioned above, for N feature tracks extending into the third image and their corresponding scene points, a $2N \times 9$ system $Ap = 0$ is formed and can be solved for ‘p’ using SVD, where ‘p’ corresponds to the 12×1 vector form of the 3×4 matrix

P for the third image. The SVD solve essentially minimizes $||Ap||$ subject to $||p|| = 1$. In general, if an input reconstruction involving N images is computed from feature tracks that also span image $N + 1$, DLT allows for the recovery of the 3×4 projection matrix P_{N+1} for image $N + 1$.

$$x_i = PX_i \tag{F.1}$$

$$x_i \times PX_i = 0 \tag{F.2}$$

$$\begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i \\ -y_i X_i^T & x_i X_i^T & 0^T \end{bmatrix} \begin{bmatrix} P^1 \\ P^2 \\ P^3 \end{bmatrix} = 0 \tag{F.3}$$

Appendix G

Patch-Based Pose Estimation

The main work presented in this dissertation consisted of a sequential approach to multi-view reconstruction, using point features as the primitive for feature matching. However, the question arises: is the sequential approach using point features the best way to go about the reconstruction problem? As an alternative, using *image patches* instead of simple point features could potentially provide added robustness and completeness.

There are a number of possible ways to work with patches. One of them is to obtain poses and a sparse point-based structure first and then apply a dense patch-based method, along the lines of 'Patch-Based Multi-View Stereo' (PMVS) [21] but not necessarily PMVS itself. For example, the dense correspondence algorithm [15] used throughout this document could be used to help extract stable patches, as an alternative to PMVS, which tends to leave 'holes' where patches do not meet local photo-consistency and global visibility constraints. Starting out with a sparse point cloud, and extending it using the dense correspondence algorithm, it may be possible to achieve a strictly dense scene reconstruction. Furthermore, it is proven here that it is possible to solve for all camera positions and orientations using just a few tracked image patches, which could potentially be more robust than sequential pose, structure and bundle adjustment estimation. The theory behind this novel method for pose estimation will now be detailed, limiting the discussion only to the mathematical derivation of the method without providing concrete results here.

The proposed method works as follows. Assume that a set of image patches has been tracked for a set of images. By working with patch *differential* information, equations can be set up to solve for the poses of each camera viewing the set of patches, specifically the (X, Y, Z) positions of each camera along with the quaternion elements representing orientation.

Assuming patch information is known in some parametrization, for example such as that provided by the PMVS algorithm [21], the goal is to obtain pose for a new camera from information about patches that have been tracked in that camera. Starting from the projection equations of a given patch in the new image, differential information provides 10 equations in the 7 extrinsic parameters. These are systems of multivariate polynomial equations. Their solution has been achieved in the literature using 'algebraic geometry' techniques. In such scenarios, it is typical to obtain multiple solutions, each of which must be tested to obtain the correct one.

For the 7-parameter, 10-equation case, we've achieved a solver based on the *hidden variable resultant* [44], and that process will now be described. The projection of a 3D point X into the image plane, at position x, is given by $x = PX$. The projection matrix P is defined as shown in Eq. G.1, where K corresponds to the 3×3 matrix of intrinsic camera parameters, R is the 3×3 rotation matrix (Eq. G.2) and t is the 3D translation vector (Eq. G.3) corresponding to camera pose. In quaternion notation, where $q = (w, x, y, z)$, the elements of R can be expressed as shown in Eq. G.4.

$$P = K[R|T] \tag{G.1}$$

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{G.2}$$

$$t = \begin{bmatrix} t_X & t_Y & t_Z \end{bmatrix}^T \tag{G.3}$$

$$R = \begin{pmatrix} w^2+x^2-y^2-z^2 & 2(xy-wz) & 2(wy+xz) \\ 2(wz+xy) & w^2-x^2+y^2-z^2 & 2(yz-wx) \\ 2(xz-wy) & 2(wx+yz) & w^2-x^2-y^2+z^2 \end{pmatrix} \tag{G.4}$$

If pose parameters are fixed, such that the derivatives of the elements of both R and t are zero, we can create a set of equations based on patch differential information, by first writing out the terms in Eqs. G.1, G.2 and G.3 as shown in Eq. G.5. Taking derivatives on the parameters from Eq. G.5 with respect to the u and v patch coordinates, the relations shown in Eqs. G.8 and G.9 are obtained, respectively. Now, the parameters displayed in Eqs. G.6 and G.7, defined in terms of the u and v patch coordinates, are all assumed known from the given patch parametrization. Additionally, an equation arises from the definition of a quaternion, as in Eq. G.10.

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = K \begin{pmatrix} r_{11}X + r_{12}Y + r_{13}Z + t_X \\ r_{21}X + r_{22}Y + r_{23}Z + t_Y \\ r_{31}X + r_{32}Y + r_{33}Z + t_Z \end{pmatrix} \tag{G.5}$$

$$x_u = \frac{\partial x}{\partial u} \quad y_u = \frac{\partial y}{\partial u} \quad w_u = \frac{\partial w}{\partial u} \quad X_u = \frac{\partial X}{\partial u} \quad Y_u = \frac{\partial Y}{\partial u} \quad Z_u = \frac{\partial Z}{\partial u} \tag{G.6}$$

$$x_v = \frac{\partial x}{\partial v} \quad y_v = \frac{\partial y}{\partial v} \quad w_v = \frac{\partial w}{\partial v} \quad X_v = \frac{\partial X}{\partial v} \quad Y_v = \frac{\partial Y}{\partial v} \quad Z_v = \frac{\partial Z}{\partial v} \tag{G.7}$$

$$\begin{pmatrix} x_u \\ y_u \\ w_u \end{pmatrix} = K \begin{pmatrix} r_{11}X_u + r_{12}Y_u + r_{13}Z_u \\ r_{21}X_u + r_{22}Y_u + r_{23}Z_u \\ r_{31}X_u + r_{32}Y_u + r_{33}Z_u \end{pmatrix} \tag{G.8}$$

$$\begin{pmatrix} x_v \\ y_v \\ w_v \end{pmatrix} = K \begin{pmatrix} r_{11}X_v + r_{12}Y_v + r_{13}Z_v \\ r_{21}X_v + r_{22}Y_v + r_{23}Z_v \\ r_{31}X_v + r_{32}Y_v + r_{33}Z_v \end{pmatrix} \tag{G.9}$$

$$w^2 + x^2 + y^2 + z^2 = 1 \tag{G.10}$$

With Eqs. G.5 through G.10, a system of polynomial equations can be set up, of the form $AX = 0$. The system contains 10 equations, 7 variables (4-element quaternion and 3D position vector) and 11 *monomials* made up of combinations of the variables. Eq. G.11 displays the resulting system of equations.

Using the *hidden variable resultant* method [44], we can choose one of the variables, say z, and treat it as a parameter, such that the data matrix would be a function of this variable. A 10×10 system $A(z)\hat{X} = 0$ results, where A is a function of the hidden variable and $\hat{X}$ corresponds to the parameter vector without the z variable. This system has a solution if and only if the determinant of the data matrix is zero. The resulting 10^{th}-order polynomial in the hidden variable can be solved using QR-reduction of the companion matrix, which many software packages support. Then, each real solution is substituted into A, and x, y and w are read-off from the singular vector corresponding to the smallest singular value. Cheirality is tested for each obtained quaternion to obtain the final rotation. In tests performed so far, only one of the 10 candidate solutions meets cheirality requirements, and is thus the correct solution.

$$\begin{pmatrix} X_u & 2Y_u & 2Z_u & 0 & -X_u & 0 & 2Z_u & -X_u & -2Y_u & X_u & 0 & 0 & 0 & -x_u \\ -Y_u & 2X_u & 0 & -2Z_u & Y_u & 2Z_u & 0 & -Y_u & 2X_u & Y_u & 0 & 0 & 0 & -y_u \\ -Z_u & 0 & 2Xu & 2Y_u & -Z_u & 2Y_u & -2X_u & Z_u & 0 & Z_u & 0 & 0 & 0 & -w_u \\ X_v & 2Y_v & 2Z_v & 0 & -X_v & 0 & 2Z_v & -X_v & -2Y_v & X_v & 0 & 0 & 0 & -x_v \\ -Y_v & 2X_v & 0 & -2Z_v & Y_v & 2Z_v & 0 & -Y_v & 2X_v & Y_v & 0 & 0 & 0 & -y_v \\ -Z_v & 0 & 2Xv & 2Y_v & -Z_v & 2Y_v & -2X_v & Z_v & 0 & Z_v & 0 & 0 & 0 & -w_v \\ X & 2Y & 2Z & 0 & -X & 0 & 2Z & -X & -2Y & X & 0 & 0 & 0 & -x \\ -Y & 2X & 0 & -2Z & Y & 2Z & 0 & -Y & 2X & Y & 0 & 0 & 0 & -y \\ -Z & 0 & 2X & 2Y & -Z & 2Y & -2X & Z & 0 & Z & 0 & 0 & 0 & -w \\ -1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x^2 \\ xy \\ xz \\ xw \\ y^2 \\ yz \\ yw \\ z^2 \\ zw \\ w^2 \\ T_X \\ T_Y \\ T_Z \\ 1 \end{pmatrix} = 0 \tag{G.11}$$

The proposed method must be tested further, and several aspects of this, such as the most appropriate patch parametrization, are still to be fully defined. Some recent algorithms provide ways of doing this, such as PMVS [21], Adobe's *PatchMatch* [4] and the work by Goesele et al. [22]. Furthermore, to generalize the current method to different patch and camera configurations, all of which cannot be solved using the hidden variable resultant, other algebraic geometry methods and packages must be analyzed. In general, the solutions involve Gauss-Jordan elimination to obtain a *Gröbner basis* for the set of monomials, followed by eigen-decomposition of the action matrix for a chosen monomial. Software packages for algebraic geometry such as *Singular* (`http://www.singular.uni-kl.de/`), *Macaulay2* (`http://www.math.uiuc.edu/Macaulay2/`) and *PHCPack* (`http://www.math.uic.edu/~jan/PHCpack/phcpack.html`) all provide support for such solvers.

It is also important to note that we have also achieved a system for solving for the 3D position of a representative point on the patch along with its differential information.

This can be achieved with either one patch and three cameras (which involves 23 parameters and 23 equations) or using two patches and two cameras (25 parameters and 26 equations). Solving such systems is extremely time-consuming and that is the main reason why we chose to work instead with known patch parametrizations.