Reconstruction of 3D Points From Uncalibrated Underwater Video

by

Neil Cavan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

There is a large and growing demand across many industries for underwater inspection technologies that can reduce the cost of assessing hard to reach underwater assets. The most widely used inspection method is single-camera video, and due to limited visibility the underwater environment these videos tend to be long, close-up sequences that make it difficult for the viewer to orient themselves and determine which part of the asset is being viewed at a given time. If these videos could be converted to a photorealistic 3D model of the asset, the asset could be inspected much more quickly and intuitively without the need to review hours of video.

The topic of 3D reconstruction from video or image sequences is a well-studied problem in computer vision. Without a calibrated camera, it generally proceeds in two stages: first, a projective reconstruction is built from image features matched throughout the sequence. This has been accomplished in the literature using global methods such as factorization, or geometric methods that stitch together many two-view or three-view reconstructions into a complete model. Projective reconstruction is considered a solved problem in the literature, and this work implements existing methods to generate projective reconstructions. Because the projective reconstruction is not in a metric space, it is not suitable for measurement of visualization. The second stage of reconstruction is autocalibration, which leverages the assumption that the images were taken with physical CCD cameras to transform the projective reconstruction to a metric space. This has been accomplished in the literature by solving for the elements of the absolute dual quadric $\mathbf{\Omega}^*$, or by direct nonlinear optimization of a cost function that penalizes deviation of the camera intrinsics from possible physical values. Autocalibration is still an active research area, and there is as yet no solution that works robustly in all situations; this work implements a novel autocalibration method that produces good results for underwater video sequences.

This thesis presents a 3D reconstruction software pipeline that is capable of generating point cloud data from uncalibrated underwater video. This research project was undertaken as a partnership with 2G Robotics, and the pipeline described in this thesis will become the 3D reconstruction engine for a software product that can generate photo-realistic 3D models from underwater video. The pipeline proceeds in three stages: video tracking, projective reconstruction, and autocalibration.

Video tracking serves two functions: tracking recognizable feature points through many frames of the video to produce image correspondences, as well as selecting well-spaced keyframes with a wide enough baseline to be used in the reconstruction while rejecting the bulk of the video frames which are too closely spaced. Video tracking is accomplished

using Lucas-Kanade optical flow as implemented in the OpenCV toolkit. This simple and widely used method is well-suited to underwater video, which is taken by carefully piloted and slow-moving underwater vehicles.

Projective reconstruction is the process of simultaneously calculating the motion of the cameras and the 3D location of observed points in the scene. This is accomplished using a geometric three-view technique based on the trifocal tensor, $\mathcal{T}$. Results are presented showing that the projective reconstruction algorithm detailed here compares favourably to state-of-the-art methods.

Autocalibration is the process of transforming a projective reconstruction, which is not suitable for visualization or measurement, into a metric space where it can be used. This is the most challenging part of the 3D reconstruction pipeline, and this thesis presents a novel autocalibration algorithm. Results are shown for two existing cost function-based methods in the literature which failed when applied to underwater video, as well as the proposed hybrid method. The hybrid method combines the best parts of its two parent methods, and produces good results on underwater video.

Final results are shown for the 3D reconstruction pipeline operating on short underwater video sequences to produce visually accurate 3D point clouds of the scene, suitable for photorealistic rendering. Although further work remains to extend and improve the pipeline for operation on longer sequences, this thesis presents a proof-of-concept method for 3D reconstruction from uncalibrated underwater video.

Acknowledgements

The guidance of my supervisors, Professor Paul Fieguth and Professor David Clausi, has been much appreciated during the course of this work. I would also like to thank Bill Triggs, Riccardo Gherardi, and David Nister for their valuable insights on the autocalibration process, as well as Manmohan Chandraker for providing a copy of David Nister's projective reconstruction dataset. This research has been sponsored by 2G Robotics Inc., the Canadian Institute for Photonic Innovations (CIPI), the Natural Sciences and Engineering Research Council (NSERC) of Canada, Mitacs and the Ontario Centres of Excellence.

Dedication

I would like to dedicate this thesis to my parents, Brian and Chris Cavan, and to my uncle Dan Cavan for their constant moral and financial support during my MASc studies. Without them I would not have decided to start this degree, and could not have completed it.

Thanks go out to Jason Gillham and Stephen Orlando for keeping me excited and inspired about my research, and to Claire Mousseau, Mike Pieterse, Guy Porlier and the rest of the gang for reminding me to occasionally get excited and inspired about other things.

Table of Contents

List of Tables

List of Algorithms

List of Figures

Nomenclature

$\mathbf{P}_m$	$\in \mathbb{R}^{3\text{x}4}$	The mth camera matrix
$\{\mathbf{P}_m\}$		The set of all m camera matrices
$\mathbf{X}^n = \begin{bmatrix} X^n \\ Y^n \\ Z^n \\ W^n \end{bmatrix}$	$\in \mathbb{P}^3$	The nth structure point
$\{\mathbf{X}^n\}$		The set of all n structure points
$\mathbf{u}_m^n = \begin{bmatrix} u_m^n \\ v_m^n \\ w_m^n \end{bmatrix}$	$\in \mathbb{P}^2$	The image observation of point n in camera m
$\{\mathbf{u}_m^n\}$		The set of all image observations
$\{\{\mathbf{P}_m\}, \{\mathbf{X}^n\}\}$		A 3D reconstruction, including structure and cameras
u, u', u''		Measured Corresponding Image Features in 2 or 3 Views
$\hat{u}, \hat{u}', \hat{u}''$		Reprojected Image Features in 2 or 3 Views
c, c', c''		Camera Centers in 2 or 3 Views
e, e'		Image Epipoles in 2 Views

Notation

In general, quantities are assumed to be projective ($\in \mathbb{P}$).
Metric quantities ($\in \mathbb{R}$) are specifically denoted by a tilde ($\tilde{\mathbf{a}}$).
The superscript n refers to structure points while the subscript m refers to camera views.

Thus, $\mathbf{u}_m^n$ is the value of the projective vector $\mathbf{u}$ for the nth structure point in the mth view.

Chapter 1

Introduction

There is a large and growing demand across many industries for underwater inspection technologies [1]. Whether the assets in question are offshore oil rigs owned by energy companies, undersea power or data cables owned by utilities, or large pipe networks owned by municipalities, the problem is the same: critical assets are in need of assessment and maintenance but inspection is difficult and expensive due to their underwater location. Currently, much of the work in the industry consists of flying a remotely operated underwater vehicle equipped with a camera to take video of these assets in order to determine their state and whether maintenance is required. These videos can be many hours in length, and must be reviewed by a specialist at significant cost to assess the state of the asset.

This research project was undertaken in participation with 2G Robotics Inc., a company focused on underwater inspection. The goal of this project is to implement computer software that can convert pre-existing single-camera videos of an asset, with unknown or varying camera parameters (auto-focus or zooming is allowed), to a photo-realistic 3D model of the asset. With such a model, the client could be able to rotate, zoom, and generally inspect the asset much more quickly and intuitively without the need to review hours of video. There is demand for such software, and the resulting library could be licensed to inspection companies or offered under the Software-as-Service model.

This thesis represents the first stage in the development of a photo-realistic 3D reconstruction software product. A robust structure-from-motion pipeline that can produce 3D point cloud data from uncalibrated single-camera video is described. Further development will allow this point cloud to be meshed and overlaid with images taken from the video, to produce a photo-realistic model.

This thesis assumes that the reader is somewhat familiar with the perspective camera model including intrinsics and extrinsics, as well as the projective, affine, and metric geometries and the differences between them. For an excellent treatment of these concepts, as well as multi-view projective geometry and 3D reconstruction in general, the interested reader is directed to [2].

1.1 Problem Definition

The problem solved by the research described in this thesis is to build a 3D model from underwater video data. More precisely, the problem is to recover all camera parameters and 3D structure points using only image features extracted from the video. No assumptions are made about the scene, and only minimal assumptions are made about the camera - namely, that it represents a physical CCD camera with square pixels.

Using video data to reconstruct a rigid 3D scene (structure) and path of the video camera (motion) is a well studied problem, referred to as Structure from Motion (SfM) in the literature [2]. There are many methods in the literature that address this problem, each making use of different assumptions [3, 4, 5]. Some of the most robust and successful, such as Google's Street View, make use of both calibrated cameras and strong assumptions about the scene — in Street View's case, that it will mostly be an urban environment of parallel lines and planes. The goal of this research is to use uncalibrated video of unknown underwater assets; such strong assumptions cannot be made. Uncalibrated SfM methods approach this problem in three stages:

1. **Video Tracking** — Identify and follow features in the image that correspond to 3D points to obtain a set of image observations $\{\mathbf{u}_m^n\}$.

2. **Projective Reconstruction** — Use $\{\mathbf{u}_m^n\}$ to calculate the motion of all cameras $\{\mathbf{P}_m\}$ and the location of all structure points $\{\mathbf{X}^n\}$. This reconstruction is not yet suitable for visualization or measurement.

3. **Autocalibration** — Using basic assumptions about the physical properties of the cameras, upgrade the reconstruction to a metric space that is suitable for visualization and measurement: $\{\tilde{\mathbf{P}}_m\}$ and $\{\tilde{\mathbf{X}}^n\}$.

1.2 Thesis Contribution

This thesis describes the development of a software pipeline that can generate 3D point cloud data from uncalibrated underwater video. The video tracking and projective reconstruction aspects of the pipeline are not particularly novel: they are re-implementations based directly on published literature [6, 7, 3]. Although valuable and necessary for a software product, the first two parts of the pipeline are considered solved problems.

Autocalibration without any prior knowledge of the camera, however, is still an active research area [8, 9, 10]. Several different algorithms from the literature were implemented, but none gave satisfactory results on video data. The main contribution of this thesis is the hybrid autocalibration method described in Chapter 7.3. By adopting the best parts of two autocalibration algorithms that failed to produce metric reconstructions suitable for visualization, an improved method that produces acceptable metric 3D point cloud data from underwater video sequences has been developed.

1.3 Thesis Outline

The 3D reconstruction pipeline described in this thesis is illustrated in Figure 1.1.

Chapter 2 describes the video tracking algorithm, which produces image observations $\{\mathbf{u}_m^n\}$ from raw video footage.

Chapter 3 provides the conceptual background for, and a high-level summary of, existing projective reconstruction methods found in the literature. Chapter 4 details the algorithm implemented during this work to produce a projective reconstruction $\{\{\mathbf{P}_m\}, \{\mathbf{X}^n\}\}$. Chapter 5 presents the results of the projective reconstruction algorithm.

Chapter 6 provides the conceptual background for, and a high-level summary of, existing autocalibration methods found in the literature. Chapter 7 details two of these existing autocalibration algorithms which were implemented during this work to produce a metric reconstruction $\{\{\tilde{\mathbf{P}}_m\}, \{\tilde{\mathbf{X}}^n\}\}$, then explains the motivation for and details of the proposed hybrid method. Chapter 8 compares the performance of the three autocalibration methods, and presents results for the reconstruction pipeline on synthetic and real datasets.

Chapter 9 provides some final thoughts on the pipeline, and Chapter 10 details plans for future improvements.

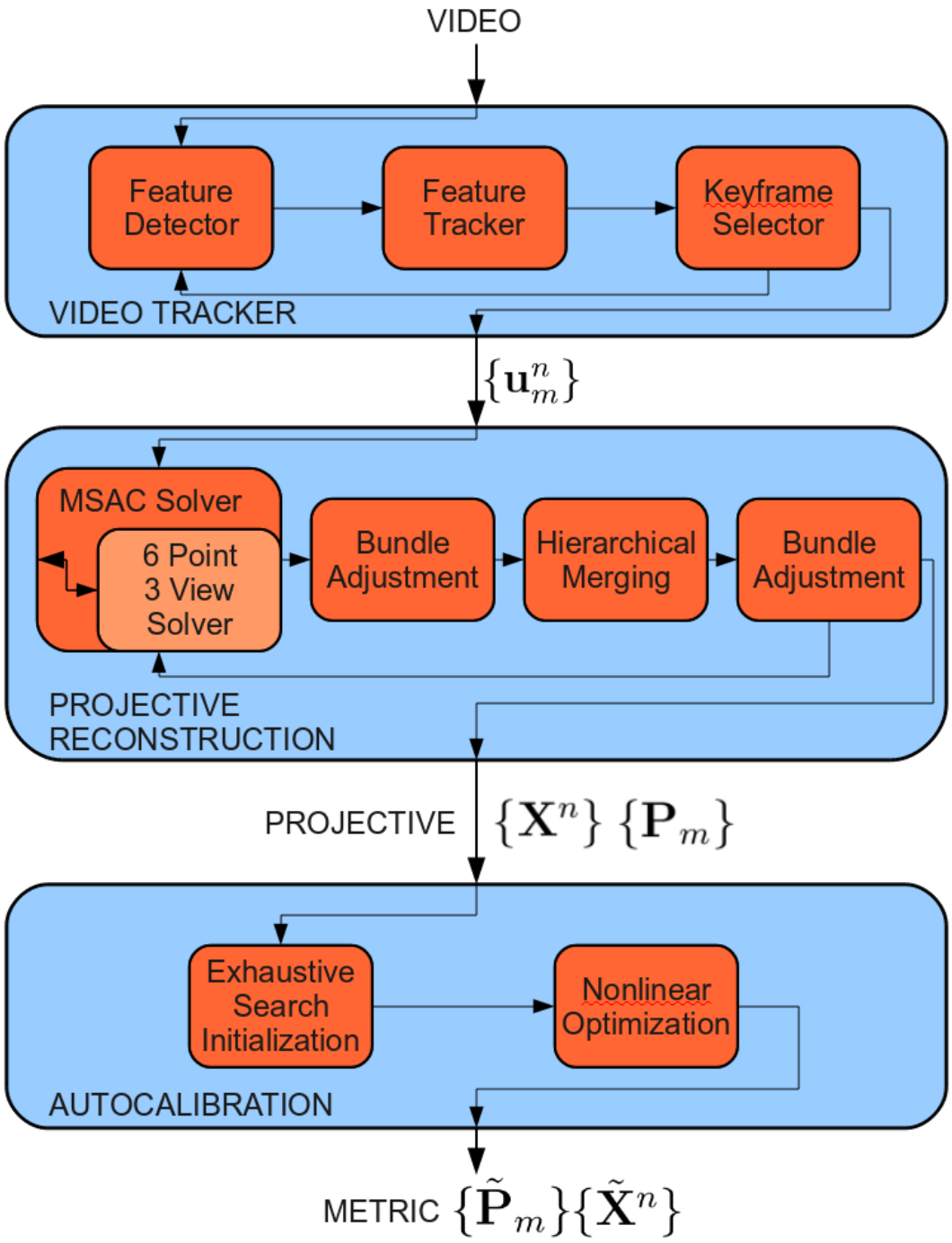


Figure 1.1: **3D Reconstruction Pipeline Flow Chart:** The pipeline described in this thesis is divided into three stages: Video tracking, described in Chapter 2, produces a set of feature correspondences from the raw video. Projective reconstruction, described in Chapter 4, uses the correspondences to solve for a 3D reconstruction consisting of camera matrices and structure points. The projective reconstruction is not suitable for visualization; Autocalibration, described in Chapter 7, transforms the projective reconstruction into a metric space and completes the reconstruction process.

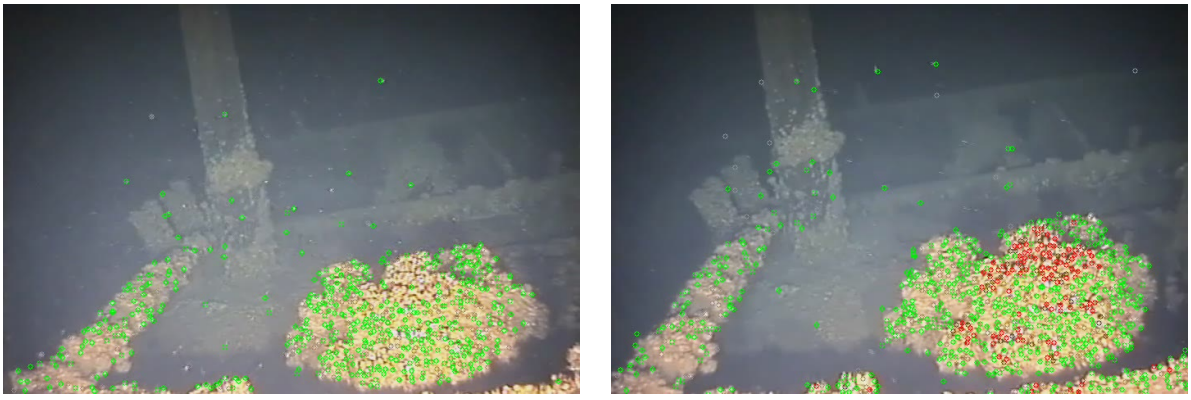
Chapter 2

Video Tracking

The first step to 3D reconstruction from video is to extract good features in the image that can be identified throughout the video — edges, corners, or other distinct points that can be identified by their location in the image, $\mathbf{u}_m^n$. These features must then be tracked through the video until they disappear, creating a set of 2D observations representing many 3D points viewed from several different angles. This dataset is referred to as feature tracks or correspondences, and many such observations are necessary to build a 3D reconstruction — typically thousands of points in dozens of views.

Underwater video is challenging to process because limited visibility and silty conditions can lead to images with few or blurred features, making feature extraction difficult. However, underwater video has one positive aspect that greatly aids feature tracking: underwater vehicles are always piloted slowly and carefully, leading to very smooth and slow-moving video sequences. This characteristic lends itself to a tracking method called Lucas-Kanade optical flow, which assumes each feature will appear in approximately the same place in the next image frame [11]. Although ideal for tracking, the slow-moving vehicle can cause problems during 3D reconstruction due to the small camera displacement between frames. If the baseline between cameras is too small, there is not enough depth information present to build an accurate reconstruction [2]. For this reason, the video tracker also identifies keyframes, a small subset of the video frames that will be used for reconstruction.

Although more sophisticated, accurate, and robust tracking methods exist, such as SIFT and SURF, Lucas-Kanade optical flow was chosen for its ease of implementation. It performed well for all videos tested, providing a reliable set of correspondences suitable for use in the robust 3D reconstruction algorithm described in Chapter 4.



Keyframe 1 Keyframe 2

Figure 2.1: **Video Keyframes During Tracking:** Two keyframes generated by the video tracker, and the points tracked. Green points were tracked successfully from Keyframe 1 to Keyframe 2, while red points were not and are plotted where tracking was lost. Keyframe 2 also includes additional green points detected and tracked into the following keyframe, evident in the bottom right corner.

2.1 Methodology and Implementation

The video tracker consists of three main components: the feature detector, the feature tracker, and the keyframe selector. A sample of the results from the video tracking process are shown in Figure 2.1.

2.1.1 Feature Detection

The feature points used are Shi and Tomasi's "good features to track", also called minimum-eigenvalue corners [12]. Similar to Harris corners [13], the feature detector identifies points where there is a strong gradient in both image directions. Minimum eigenvalue corners perform better than Harris corners in tracking applications [14]. The feature detector has an implementation in the OpenCV C++ toolkit, and is used to obtain sub-pixel feature points in each keyframe of the video. For a more detailed description of the algorithm, the interested reader is referred to [14].

2.1.2 Feature Tracking

Once features have been identified in the current image, they are located in the next video frame (the target image) using the the pyramidal Lucas-Kanade tracking algorithm [15]. The algorithm leverages three assumptions:

1. **Brightness constancy** — The lighting intensity of the image will not vary significantly between video frames.
2. **Small movements** — The camera will move only slightly between video frames, so the target image can be modeled as a small translation of the current image.
3. **Spatial consistency** — Image features will look nearly the same in the target image as they do in the current image.

Note that these assumptions relate to qualities of the video sequence, and do not pertain to the 3D scene itself. No assumptions are made about the structure of the video's subject, and any 3D shape can be reconstructed.

The algorithm tracks each feature in the current image by taking a small patch around the feature and searching a small window (15 pixels square) in the target image, centered on the feature's current location, to see how well each point in the window matches the patch. For computational efficiency and because movements are assumed to be small, the search window is kept quite small — 10–15 pixels is used in this work. The best match is chosen as long as the correlation is above a threshold, otherwise the tracking fails for that feature. The correlation threshold is a tunable parameter, and the appropriate value depends on the input sequence - videos of objects with strong texture benefit from a lower threshold to increase the number of matched points, while dim or blurry sequences must use a higher one to avoid excessive mismatches. To allow for larger motions, the algorithm also makes use of "gaussian pyramids" of both current and target images, consisting of the full-resolution image at the base and a series of downsampled, lower-resolution images rising to the tip. The search is first conducted at the coarsest resolution, and the result is progressively refined at higher resolutions. This requires much less computation than using a large search area in the full-resolution image.

The Lucas-Kanade tracker has an implementation in the OpenCV C++ toolkit [16], and is used to track feature points through the video. For a more detailed description of the algorithm, the interested reader is referred to [14].

2.1.3 Keyframe Selection

The tracker algorithm requires very small camera motion between video frames to function properly, and underwater video meets this requirement nicely. 3D Reconstruction algorithms, however, require depth information that can only be inferred from large camera motions between frames. This, coupled with the fact that underwater inspection videos can be hours long and contain hundreds of thousands of frames, makes building a 3D reconstruction using every single video frame intractable. As such, the tracker incorporates a keyframe selector that discards most of the video frames, retaining only those where the camera has moved enough for the new vantage point to provide useful information. Camera motion cannot be inferred directly from image measurements — indeed, this is the problem solved by 3D reconstruction — but the average motion of image features can be used as a rough measure of motion. It is also necessary to ensure that enough features have been tracked between frames to allow for robust reconstruction. A new keyframe is therefore selected whenever either the average feature motion rises above a threshold, or the total number of features drops below a threshold. In this work an average motion threshold of 30 pixels was used, but this value can be changed freely — a lower threshold will increase the number and density of keyframes, and a higher one will produce fewer keyframes. A minimum of 50 corresponding features is used in this work, as it was found experimentally that projective reconstruction becomes unreliable with fewer features.

Chapter 3

Projective Reconstruction — Background

Projective reconstruction is the process of calculating 3D structure and motion from image data. Using only the correspondences generated by the video tracker $\mathbf{u}_m^n$, it is possible to generate a 3D reconstruction of the imaged scene $\mathbf{X}^n$, and the camera matrices $\mathbf{P}_m$. This reconstruction is mathematically self-consistent in its projective geometry, but it is not in a metric space and so is not suitable for measurement or visualization purposes [2]. A projective reconstruction is the best that can be done without knowledge of, or strong assumptions about, the camera parameters — to deal with uncalibrated video, such assumptions must be avoided. Projective reconstruction is therefore a necessary first step for determining unknown camera parameters via autocalibration, the subject of Chapter 6.

3.1 Fundamentals

Projective reconstructions consist of a set of corresponding image features $\{\mathbf{u}_m^n\}$, camera matrices $\{\mathbf{P}_m\}$, and structure points $\{\mathbf{X}^n\}$ [17]. These quantities are related by the perspective camera equation [2]:

$$\mathbf{u}_m^n \simeq \mathbf{P}_m \mathbf{X}^n \tag{3.1}$$

This equation is illustrated in Figure 3.1. A projective reconstruction is correct if every structure point $\mathbf{X}^n$ projected through each camera $\mathbf{P}_m$ exactly matches each observation $\mathbf{u}_m^n$. In general, image noise will cause some error between an observed feature $\mathbf{u}_m^n$ and

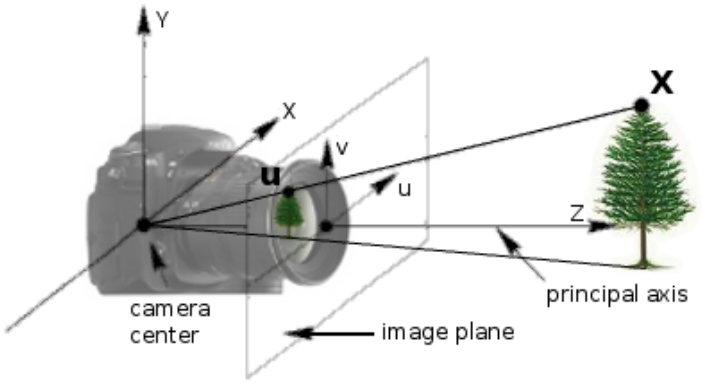


Figure 3.1: **The Perspective Camera:** This diagram illustrates (3.1), the perspective camera equation. Every 3D point, $\mathbf{X}^n$, projects through each camera's projection matrix $\mathbf{P}_m$ onto that camera's image plane to form an image observation, $\mathbf{u}_m^n$.

the projected location of its reconstructed point, $\hat{\mathbf{u}}_m^n$. This reprojection error, measured in the image plane in units of pixels, is the quality metric for a projective reconstruction: the lower the reprojection error, the better the reconstruction. Projective reconstructions are not unique, and can be altered by a 4x4 projective transformation $\mathbf{H}$ without affecting reprojection error:

$$\mathbf{u}_m^n = (\mathbf{P}_m \mathbf{H}^{-1})\ (\mathbf{H}\mathbf{x}^n) \tag{3.2}$$

A projective reconstruction, then, is really only one member of a family of projectively equivalent reconstructions. This is called projective ambiguity, and it implies that convenient projective frames can be freely chosen without impacting reprojection error. A useful choice is to map a reference camera (usually the first) to the canonical form [37]:

$$\begin{aligned} \mathbf{P}_{ref}\mathbf{H}_{can} &= [\mathbf{I}|\mathbf{0}] \\ \mathbf{H}_{can} &= \begin{bmatrix} \mathbf{P}_{ref} \\ \mathbf{c}(\mathbf{P}_{ref})^T \end{bmatrix}^{-1} \end{aligned} \tag{3.3}$$

where $\mathbf{c}()$, the projective camera center, is computed as follows:

$$\mathbf{c}(\mathbf{P}) = (c_1, c_2, c_3, c_4)^T \text{ with } c_i = (-1)^i det(\mathbf{P}^{(i)}) \tag{3.4}$$

with $\mathbf{P}^{(i)}$ defined as the 3x3 matrix obtained by removing the ith column of $\mathbf{P}$. Having a canonical reference camera is convenient because it greatly simplifies computations such as $\mathbf{F}$ and $\mathcal{T}$ estimation, covered in Chapter 3.2.2 and 3.2.3.

Another important transformation is the metric upgrade $\mathbf{H}^*$, the topic of Chapter 6. Only a small subset of valid projective reconstructions are also metric, where concepts of distance and parallelism have meaning. In a metric frame, the camera matrix $\mathbf{P}$ can be decomposed into its intrinsic and extrinsic properties:

$$\tilde{\mathbf{P}} = \tilde{\mathbf{K}}(\tilde{\mathbf{R}} \mid -\tilde{\mathbf{R}}\tilde{\mathbf{c}}) \tag{3.5}$$

where $\tilde{\mathbf{c}}$ is the camera center's position, $\tilde{\mathbf{R}}$ is a rotation matrix encoding the camera's orientation, and

$$\tilde{\mathbf{K}} = \begin{bmatrix} f_u & s & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{bmatrix}$$

is the camera intrinsics matrix. The camera's focal length, f_u and f_v, and the principal point location, $(p_u,\ p_v)$, are all measured in pixels aligned with the image plane's axes. Skew, s, is a parameter representing the angle between the axes of the image plane; for cameras with rectangular pixels, $s = 0$. For cameras with square pixels, $f_u = f_v$ These basic assumptions, true for nearly all CCD cameras, are the basis of Autocalibration, the subject of Chapter 6.

A note on normalization: image observations $\{\mathbf{u}_m^n\}$ are usually scaled and translated to a normalized, unitless co-ordinate system before beginning projective reconstruction so that, within each image, the mean of the observations is 0 and the average distance from the mean is $\sqrt{2}$ [18]. This normalization improves the numerical conditioning of the reconstruction process, but affects each camera differently and changes the pixel aspect ratio by scaling each image axis differently. In this work, a similar normalization is applied to the camera matrices, producing nearly the same improvement in numerical properties while acting identically on each camera and preserving pixel aspect ratio. The normalization is:

$$\mathbf{P}_m \leftarrow \left(\frac{1}{2} \begin{bmatrix} \sqrt{w^2+h^2} & 0 & w \\ 0 & \sqrt{w^2+h^2} & h \\ 0 & 0 & 1 \end{bmatrix} \right)^{-1} \frac{\mathbf{P}_m}{||\mathbf{P}_m||_2} \tag{3.6}$$

where w and h are the width and height of the image, respectively.

3.2 Existing Methods

The goal of projective reconstruction is to produce a consistent set of structure points and camera matrices using only image features. There are several well-tested methods in the literature for accomplishing this task, and the most promising were evaluated for their ability to deal with noisy and outlier-filled underwater video datasets.

3.2.1 Factorization

First proposed by Tomasi & Kanade [19], factorization reduces projective reconstruction to a rank-constrained matrix estimation problem. The key idea is that all image observations can be stacked into an observation matrix, and the desired structure and camera matrices should provide the nearest least-squares rank-4 approximation to the observation matrix[20].

$$\begin{bmatrix} \lambda_0^0\mathbf{u}_0^0 & \cdots & \lambda_0^n\mathbf{u}_0^n \\ \vdots & \ddots & \vdots \\ \lambda_m^0\mathbf{u}_m^0 & \cdots & \lambda_m^n\mathbf{u}_m^n \end{bmatrix} = \begin{bmatrix} \mathbf{P}_0 \\ \vdots \\ \mathbf{P}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}^0 & \cdots & \mathbf{X}^n \end{bmatrix} \tag{3.7}$$

The estimation of projective depths, λ_m^n, is essential to this process as it is equivalent to recovering the depth information lost during projection[21]. Modern factorization algorithms take an iterative linear approach to estimation: first structure points, then camera matrices, and finally projective depths are estimated in sequence while holding the other variables constant[22].

It might appear from (3.7) that factorization is inherently an off-line process, requiring all data to be known before it can begin, and this is the case for the core algorithm. However, some clever extensions have been made to allow on-line operation, adding additional image features as they become available[5].

Factorization distributes error evenly across all observations, avoiding initialization bias and error accumulation. Missing data can be dealt with easily, as the matrix-based factorization can be reduced to a least-squares minimization involving only the visible features[22]. However, because it is essentially a least-squares solution, large outliers or mis-associated data are not handled well by this algorithm. Testing has shown that even a small number of outliers can significantly degrade results. Factorization implementations in the literature that have been successfully applied to real data do not generally explain in detail how features are matched [22], and the number of matched features is usually quite low - only 14 in the case of [5]. It is likely that matching in these works has been performed manually, or using standard stereo registration techniques that simultaneously estimate $\mathbf{F}$ as described in Chapter 3.2.2 to reject outliers.

3.2.2 Two-View Epipolar Geometry: The Fundamental Matrix

As shown in Figure 3.2, epipolar geometry describes the relationship between corresponding feature points in two views. Epipolar geometry has a long history in the field of 3D

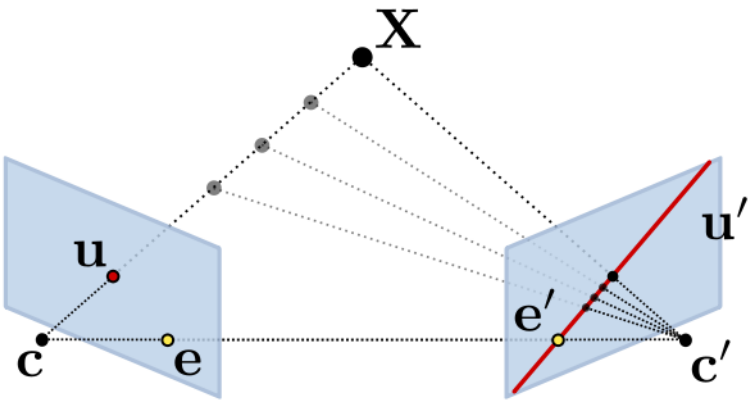


Figure 3.2: **Two-View Geometry:** Given **F** and **u**, **u′** is restricted to a line in the second image; the precise location depends on the depth of the 3D structure point in the first view. Image modified from [23].

reconstruction, and is most often employed using two-camera calibrated stereo vision systems. Epipolar geometry is by far the most common method used for 3D reconstruction due to widely available implementations in standard toolkits [24].

The fundamental matrix, $\mathbf{F} \in \mathbb{R}^{3x3}$, encapsulates the epipolar geometry in its seven degrees of freedom and a solution for **F** can be obtained from seven corresponding features[25]. Given **F** and **u**, **u′** is restricted to a line in the second image; the precise location depends on the depth of the 3D structure point in the first view. This epipolar constraint, expressed mathematically as ${u'}^T \mathbf{F} u = \mathbf{0}$, allows for detection of outlying data that does not fit the viewing geometry, and allows for guided feature matching to detect additional consistent correspondences that the video tracker did not identify.

The robust calculation of **F** from image correspondences is a well-studied problem, and modern methods use a robust statistical approach to automatically estimate an **F** that most closely fits consistent correspondences from a data set, while simultaneously identifying outliers[26]. While robust to outliers and mis-associated data, **F** cannot be calculated accurately for some degenerate camera configurations that turn out to be fairly common in practice[27]. For more detail on **F** and epipolar geometry, the interested reader is directed to [2].

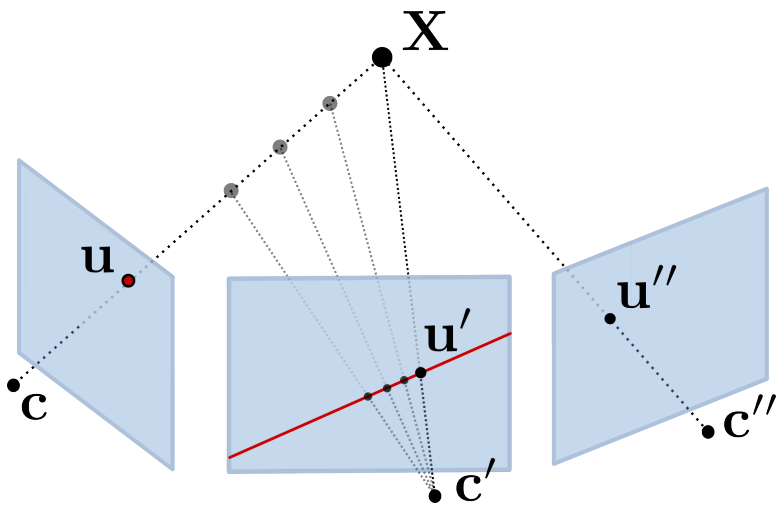


Figure 3.3: **Three-View Geometry:** Given the trifocal tensor $\mathcal{T}$, the location of a feature point in the first image is restricted to a line in the second image, and a correspondence in the first two images is restricted to a point in the third.

Because **F** can only describe two views, video sequences are processed by choosing the first pair as a projective frame and merging subsequent pairs into the reconstruction as they arrive. The merging process estimates a transformation to the base frame using shared structure or a shared camera matrix[4]. While lending itself to on-line implementation, this sequential approach is prone to error accumulation when processing long video sequences. The merging transform is an estimate, and each estimate is appended to the last one. Although the transformation error between any adjacent two-view reconstructions is small, the accumulated drift can become significant by the end of the sequence.

3.2.3 Three-View Geometry: The Trifocal Tensor

Chapter 3.2.2 and Figure 3.2 describe the projective geometry of two views; this chapter and Figure 3.3 describe the relationship between corresponding feature points and lines in three views. The trifocal tensor $\mathcal{T} \in \mathbb{R}^{3x3x3}$ encapsulates the geometry of this image triplet in its 18 degrees of freedom, and a solution for $\mathcal{T}$ can be obtained from six corresponding features[28]. Given $\mathcal{T}$ and $\mathbf{u}$, $\mathbf{u}'$ is restricted to a line in the second image, and given $\mathbf{u}$ and $\mathbf{u}'$ the location of u'' in the third image is fully constrained. These geometric constraints are stronger than those of the epipolar geometry. $\mathcal{T}$ can also be used to transfer corresponding

lines from one image to another within a triplet, allowing their use as image features in addition to points. In general $\mathcal{T}$ can be computed more efficiently and robustly than $\mathbf{F}$ [29].

Modern algorithms for estimating $\mathcal{T}$ use a minimal six-point solution as the kernel in a robust statistical framework, and use guided matching to find additional line and point features in the images that were not detected by the video tracker [30]. While remaining at least as robust to outliers and mis-associated data as algorithms based on $\mathbf{F}$, $\mathcal{T}$ is not as strongly affected by degenerate camera configurations[3]. While sequential algorithms using $\mathcal{T}$ as a building block are subject to the same baseline and drift problems, the third view helps to mitigate them: a third non-degenerate view can only improve the baseline and helps to mitigate error accumulation as well, since each reconstruction can share two camera matrices with adjacent reconstructions instead of only one.

Chapter 4

Projective Reconstruction — Methodology

Three existing methods were found in the literature review of Chapter 3.2: factorization, two-view geometry, and three-view geometry. Factorization was found to be unreliable for tracked video data, and three-view geometry was selected as the stronger of the two geometric methods.

Because projective reconstruction by factorization can be formulated as a global least-squares estimation, it shares similar drawbacks. Factorization has difficulty handling datasets containing outliers caused by incorrect or mismatched feature correspondences — that is to say, it has difficulty handling real tracking data from video sequences. This hypothesis was tested using an implementation of the factorization algorithm described in [22]. Factorization performed well in testing on synthetic datasets, including those with high levels of Gaussian noise and missing data, but failed to converge when used on tracking data from video sequences containing a small percentage of outliers. The factorization approach was therefore abandoned as untenable for robust reconstruction of underwater video.

Of the two-view and three-view geometric approaches that remain, two-view epipolar and three-view trifocal geometry, the trifocal tensor-based algorithm was chosen as the most promising due to its stronger constraints and increased flexibility during the merging step. The reconstruction algorithm used consists of two steps: robust computation of $\mathcal{T}$ for each triplet of keyframes, and merging structure points and camera matrices from all $\mathcal{T}$s together into a complete reconstruction.

4.1 Computing the Trifocal Tensor

Robust computation of the trifocal tensor was accomplished by using a minimal six-point three-view solution for structure and motion, and thus $\mathcal{T}$, as the kernel in a robust M-estimator Sampling And Consensus (MSAC) estimation framework. MSAC is a variant of the well-known RANdom Sampling And Consensus (RANSAC) family of robust statistical methods used to identify solutions in outlier-filled datasets [31]; it is described in detail in Chapter 4.1.3. Once a robust solution has been found for six points in three views, all observed points in the image triplet can be triangulated, and the reprojection error of all points is used as a metric for the triplet's accuracy. The best solution from the MSAC process is then further refined using a nonlinear bundle adjustment.

4.1.1 A Six-Point Solution for $\mathcal{T}$: Computing a Trifocal Triplet

If six distinct 3D points are observed by three camera matrices, it is possible to solve directly for all three camera matrices and six structure points. Six points are the minimum required to compute the solution, and the use of a minimal solution is important in any RANSAC-based computation. As Chapter 4.1.3 will make clear, a smaller sample size means fewer iterations are required to ensure a correct solution. The solution method used in this work was first suggested by Quan[32], includes improvements suggested by Schaffalitzky[6], and follows the implementation of Mierle[8]. Its foundation is to fix most degrees of freedom by setting the co-ordinates of the first five structure points to form a projective basis. With this simplification, (3.1) can be rewritten as:

$$\mathbf{u}_m^{1\ldots6} = \mathbf{P}_m \mathbf{X}^{1\ldots6}$$

$$\begin{bmatrix} u_1 & & u_6 \\ v_1 & \cdots & v_6 \\ w_1 & & w_6 \end{bmatrix} = \mathbf{P}_m \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & x_6 \\ 0 & 1 & 0 & 0 & 1 & y_6 \\ 0 & 0 & 1 & 0 & 1 & z_6 \\ 0 & 0 & 0 & 1 & 1 & t_6 \end{bmatrix} \tag{4.1}$$

Setting the structure for the first five observations constrains each camera matrix to a one-parameter family of solutions, or "pencil", such that $\mathbf{u}_m^i = [\,\mu_m \mathbf{A}_m + \nu_m \mathbf{B}_m]\mathbf{X}^i$ and only the ratio $\alpha = \mu_m/\nu_m$ is important. This ratio is shared between all three cameras, and the geometric constraint that rays through $\mathbf{u}_m^6$ must all converge at the unknown point $\mathbf{X}^6$ allows α to be obtained as the solution to a cubic equation. There may be one or three such solutions, but at least one will exist. Given α, the co-ordinates of $\mathbf{X}^6$ can be calculated, and given $\mathbf{X}^6$ the precise values of μ_m and ν_m can be calculated and all three camera matrices obtained.

At this point three camera matrices and six 3D structure points have been obtained which are exactly consistent with three-view geometry. $\mathcal{T}$ itself has not been obtained as a part of this procedure, but if desired it can be trivially calculated from the three camera matrices. This solution procedure is referred to as the minimal solution — minimal because it contains only the six points required to solve. An overview of the minimal solution algorithm is given in Algorithm 4.1. For further detail, the interested reader is directed to [8], which contains an excellent implementation-oriented description of this algorithm.

Algorithm 4.1 Minimal six point solution for $\mathcal{T}$: This procedure calculates structure points and camera matrices

Require: $\mathbf{u}_m^{1\ldots6}$, images of six points in three views

for $m = 1$ **to** 3 **do**
  solve for $\mathbf{A}_m$ and $\mathbf{B}_m$, where $\mathbf{u}_m^n = [\,\mu_m\mathbf{A}_m + \nu_m\mathbf{B}_m]\mathbf{X}^n$
end for
obtain α as the one or three solutions to a cubic formed from $\mathbf{A_m}$ and $\mathbf{B_m}$
numsol $\leftarrow$ number of solutions for α
for $i = 1$ **to** *numsol* **do**
  calculate $\{\mathbf{X}^{1\ldots6}\}_i$ from α_i
  for $m = 1$ **to** 3 **do**
    solve for μ_{mi} and ν_{mi} using $\{\mathbf{X}^{1\ldots6}\}_i$
    calculate $\{\mathbf{P_m}\}_i$
  end for
end for
return *numsol* solutions, each consisting of: $\{\ \mathbf{P}_{1\ldots3}, \mathbf{X}^{1\ldots6}\ \}$

4.1.2 Triangulation: Adding Additional Structure

The minimal solution described above produces an exact solution for six points in three views. However, over 100 corresponding image features are generally tracked through three video keyframes. Once the camera matrices are obtained using 6 points, any image feature observed in two or more views can have its 3D structure determined by triangulation.

If the feature were noise-free, this could be accomplished by finding the intersection of the two back-projection rays, extending from the camera's origin to the image observation, as shown by the red lines in Figure 4.1. Due to image noise, however, these rays will not

generally intersect. Before triangulating, corrected observations $\hat{\mathbf{u}}$ must be found whose back-projection rays do intersect, while remaining as close to the original observations $\mathbf{u}$ as possible — this is equivalent to selecting a 3D structure point which minimizes reprojection error.

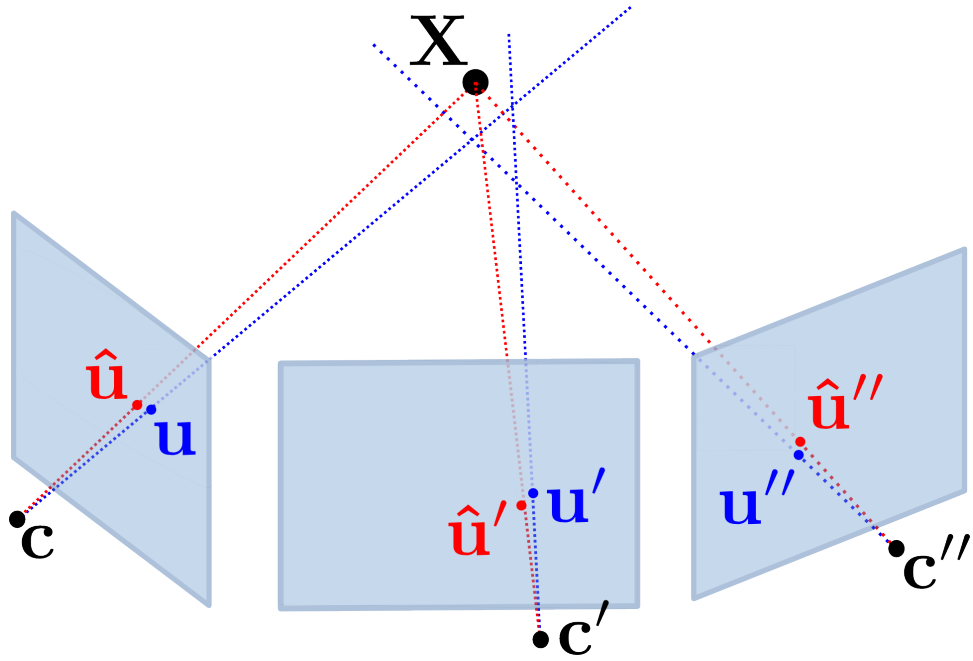


Figure 4.1: **Triangulation in three views:** Under ideal conditions, the rays from the camera origins to the image observations would intersect at the 3D structure point, as shown in red. Under real conditions with noisy measurements, the rays are skewed as shown in blue. Optimal triangulation finds observations $\hat{\mathbf{u}}$ and $\hat{\mathbf{u}}''$ which do intersect at a point, while remaining as close as possible to $\mathbf{u}$ and $\mathbf{u}''$. The procedure is optimal in the sense that the reprojection error $||\mathbf{u} - \hat{\mathbf{u}}||_2 + ||\mathbf{u}'' - \hat{\mathbf{u}}''||_2$ is minimized. The reprojection error from the middle image, $||\mathbf{u}' - \hat{\mathbf{u}}'||_2$ can be used as a sanity check: if it is not on the same order as the other two images, triangulation has not produced an acceptable result and the point must be discarded.

The triangulation method implemented in this work is optimal in terms of minimizing reprojection error[33]. Triangulation uses the observations $\mathbf{u}$ and $\mathbf{u}''$ and the fundamental matrix $\mathbf{F}_{13}$ between views 1 and 3 to solve a 6th-order polynomial, producing the optimal corrected observations $\hat{\mathbf{u}}$ and $\hat{\mathbf{u}}''$, whose intersection can be trivially found. This triangulation method is the "gold standard" algorithm for triangulation, and for further details

the interested reader is directed to [2]. $\hat{\mathbf{u}}'$ is not explicitly calculated during triangulation, but is included when the total reprojection error for the triplet is calculated using (3.1).

4.1.3 MSAC: A Robust Solution Framework

After triangulating all points, the total reprojection error is an essential metric for the accuracy of a trifocal triplet. Video tracking data often contains a significant number of outlying points, caused by tracking errors — these are not merely noisy measurements, but erroneous ones which do not represent a single 3D point imaged by three cameras. The minimal solution for $\mathcal{T}$ will only produce acceptable results if all six points are inliers. This presents a problem: without knowing which points are outliers, which six points should be selected for the minimal solution? This is a well-studied problem, and a robust statistical estimator from the RANSAC family is the right tool to solve it [31, 30, 8].

The basic idea of RANSAC is to take a random minimal sample from the dataset, find the solution, and then count the number of data points that fit the solution within some error threshold. By repeating this procedure many times it is guaranteed that a sample containing only inliers will eventually be selected, and the solution calculated from this sample will fit most other inliers. The number of iterations required to assure at least one sample has been selected containing only inliers can be calculated based on the percentage of inliers in the data set. RANSAC uses a threshold to classify points with respect to a solution: either a point falls within the error threshold and is counted as an inlier, or it does not and is an outlier. After the required number of iterations, the solution which generated the highest number of inliers is selected. The error threshold should be small relative to the image size, and is usually on the order of several pixels — a 1.0 pixel threshold was used in this work. The solution is not overly sensitive to this parameter unless it is set so low as to exclude legitimate points. True outliers to the dataset will produce reprojection errors at least an order of magnitude greater than the inliers.

It should be noted that the RANSAC approach does introduce a stochastic element to the projective reconstruction process — because the solution is chosen from a set of random samples, the solver will not produce the same solution when run multiple times on the same data set. However, it should always produce a reasonable solution that has strong support in the dataset, fitting most of the observed points while simultaneously identifying the others as outliers to be discarded.

The RANSAC process can be viewed as an optimization with a binary cost function: either a point contributes nothing to the cost function or it contributes 1. The RANSAC variant used in this work is called MSAC [8], and it modifies the RANSAC cost function:

inliers add their error to the cost function while outliers add the error threshold, and the sample with the lowest cost function is selected. The improvement is illustrated by imagining two samples that produce a solution with the same number of inliers, but one having twice the total reprojection error of the other. Using MSAC, the sample with lower reprojection error is chosen where RANSAC would not distinguish between the two. The MSAC procedure is detailed in Algorithm 4.2.

Algorithm 4.2 MSAC: a robust solution for $\mathcal{T}$

Require: $\mathbf{u}_m^n$ for a trifocal triplet
Require: ϵ, a reprojection error threshold

```
best_sol ← NULL
best_E ← MAXVAL
for it = 1 to max_it do
  take a random sample of six correspondences from u_m^n
  perform the minimal reconstruction of Algorithm 4.1
  cost ← 0
  for pt = 1 to n do
    Triangulate pt as described in Chapter 4.1.2
    E ← reprojection error of pt
    if E < ε then
      cost ← cost +E
    else
      cost ← cost +ε
    end if
  end for
  if E < best_E then
    best_sol ← this solution
  end if
end for
return best_sol: { P_1, P_2, P_3, X^n }
```

4.1.4 Projective Bundle Adjustment

The MSAC procedure robustly obtains an approximate solution for $\mathcal{T}$ that has a high level of support in the data. However, this solution was calculated using only six points; the

final step is to optimize $\mathcal{T}$ for all inliers. This is accomplished using bundle adjustment, a nonlinear gradient descent-based optimization process that slightly adjusts all parameters of the structure points and camera matrices in order to reduce the overall reprojection error [34]. The adjustments should be relatively small given the high-confidence solution produced by MSAC — in this work it was rare for any single parameter to be adjusted by more than 1-2% during bundle adjustment. When these small adjustments are made to thousands of points in dozens of views, however, the overall effect is quite dramatic: bundle adjustment almost always reduces reprojection error by a factor of two or better, and usually by an order of magnitude or more.

Bundle adjustment is applied to every triplet after the MSAC solution procedure, and again on the existing reconstruction after the addition of every new triplet, as described in Chapter 4.2.3.

Bundle adjustment is a standard and widely-used procedure in Structure from Motion, and was implemented using the SBA open source C++ toolkit[35]. The interested reader is referred to [35] for a more detailed description of the SBA algorithm, and to [34] for an excellent overview of bundle adjustment in general.

4.2 Merging Triplets into Multiview Reconstructions

The estimation procedure outlined above produces a projective reconstruction for three camera views. Many such triplets must be combined in order to obtain a reconstruction of an entire video sequence. But each reconstruction is in a different projective frame — they must be transformed into the same space in order to be combined.

Finding the merging homography requires that triplets share cameras, structure, or both. There is also the issue of selecting which triplets to use in the reconstruction. Even with the keyframe selection performed by the video tracker, some triplets will have a baseline that is too short to produce an accurate reconstruction, and attempting to include such a triplet will produce poor results. Similarly, attempting to create extremely wide-baseline triplets implies that fewer correspondences are available to calculate $\mathcal{T}$ due to occlusions and sections of the initial scene no longer being visible — this will reduce the accuracy of $\mathcal{T}$ and produce poor results.

The solution implemented in this work is to define a measure for the quality of a triplet that includes both baseline and tracked features, and proceed to hierarchically merge only the best-quality triplets to form the final projective reconstruction.

4.2.1 Finding the Merging Homography

As (3.2) suggests, there are many free parameters in a projective reconstruction. To merge two triplets, a merging homography $\mathbf{H} \in \mathbb{R}^{4x4}$ must be found that maps common structure points into the same projective co-ordinate frame. In keeping with projective geometry, the scale of this matrix is irrelevant; it has 15 degrees of freedom. To constrain these free parameters, the two triplets to be merged must share some parameters - cameras, structure, or both. With shared structure $\mathbf{X}$ and camera $\mathbf{P}$ in one triplet with corresponding $\mathbf{X}'$ and $\mathbf{P}'$ in the second, the goal is to solve for $\mathbf{H}$ such that:

$$\mathbf{P} = \mathbf{P}'\mathbf{H}^{-1}$$
$$\mathbf{X} = \mathbf{H}\mathbf{X}'$$

There are three distinct cases to consider:

Case I: Shared Structure Only
Each structure point in projective 3-space provides three constraints on $\mathbf{H}$. At least five shared points are therefore sufficient to solve for $\mathbf{H}$.

Case II: One-View Overlap
In this case the triplets to be merged share an image; usually the final image in the first triplet is the initial image in the second triplet. A projective camera matrix has 11 degrees of freedom, requiring four additional constraints to solve for $\mathbf{H}$. A common camera matrix and at least two common structure points are therefore sufficient to solve for $\mathbf{H}$.

Case III: Two-View Overlap
Two shared camera matrices provide 22 constraints, more than the 15 degrees of freedom in $\mathbf{H}$. Two shared camera matrices are therefore sufficient to solve for $\mathbf{H}$.

For further details of these merging methods, the interested reader is referred to [3]. In this work, triplets are constructed with one view overlap; that is, if a sequence contains 7 keyframes, triplets will be built from keyframes 1-2-3, 3-4-5, and 5-6-7 and then merged together using their common camera matrix and shared structure. This decision is driven mainly by the one-view overlap method's suitability for the hierarchical framework described in Chapter 4.2.3.

4.2.2 A Quality Measure for $\mathcal{T}$

The sheer number of frames in a video sequence begs the question: which ones should be used to generate triplets and form the reconstruction? Attempting to calculate $\mathcal{T}$ from

three adjacent frames in the video is bound to fail, as the camera will not have moved enough between camera views to accurately capture 3D information. This is the baseline problem, and the video tracker takes the first step in addressing it by selecting keyframes based on pixel motion: only after features have moved "enough" is a frame even considered as a candidate for reconstruction. But this image-space metric is not sufficient; pure camera rotation will produce image-space motion and cause the tracker to select a keyframe even though the baseline is zero. If the first three keyframes have low baseline, then these frames should be skipped and the first triplet built from frames 1-3-5 or 1-5-9 instead of 1-2-3. But which should it be: 1-3-5, or 1-5-9, or some other selection of frames? Clearly, a quality measure must be developed for $\mathcal{T}$ to inform this decision and avoid building low-baseline triplets.

In theory the best triplet to use as the basis for reconstruction would have the widest possible baseline, and therefore be built from the first, middle, and last frame of the sequence — but this is impossible. In most real video sequences, the final frame of the video contains an entirely different scene than the first: there are no corresponding image features with which to calculate $\mathcal{T}$. In practice, the number of tracked features drops below that needed to robustly compute $\mathcal{T}$ long before the scene from the first image is completely out of view. As the camera moves, distant features can be occluded by nearer objects in the scene, the changing viewing angle can cause a feature to appear different enough for the tracker to reject it, and illumination changes can do the same for many features at once.

There is a fundamental trade-off here: using more widely-spaced video keyframes increases the accuracy of $\mathcal{T}$ by widening the baseline, but also decreases accuracy by lowering the number of features tracked. Keyframes should be chosen that lie in a quality "sweet spot": a baseline wide enough for accurate reconstruction, but still with enough tracked features to ensure the same. The measure of $\mathcal{T}$ quality used to define this trade-off was proposed by [7]:

$$Q = b^{\alpha} p \tag{4.2}$$

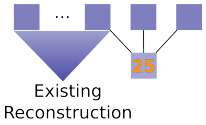
where b is the abstract baseline, simply the number of keyframes spanned by $\mathcal{T}$, and α is a tunable parameter that adjusts how "greedy" the metric is for wide tensors. Values of $0.5 < \alpha < 1.0$ are recommended [7], with larger values resulting in the selection of wider tensors. The parameter p is a measure of how much support there is in the data for the tensor in question actually having a wide enough baseline for 3D reconstruction. p is calculated as the number of features consistent with $\mathcal{T}$ divided by the number of features that can be transferred from the first keyframe to the third using a 3x3 image-space homography. If the baseline is very short, nearly all the features will be consistent with the

image-space homography; if is is long enough to generate an accurate 3D reconstruction very few will.

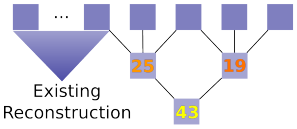
There is one further detail to the quality metric: (4.2) is used only if the number of features tracked is above a threshold — as in Chapter 2.1.3, a minimum of 50 features is used. If the triplet contains fewer tracked features, robust calculation of $\mathcal{T}$ is questionable and the quality is set to 0. Although this quality measure is somewhat ad-hoc it has proved very effective as a decision metric when comparing two triplets, in this work and as reported by [7].

4.2.3 A Hierarchical Approach to Projective Reconstruction

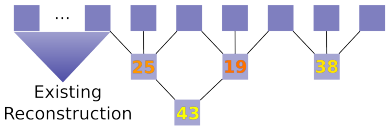
All the tools required to compute $\mathcal{T}$ for image triplets, evaluate their quality, and merge the best together to form a reconstruction have been described above. What is now required is a framework to automatically perform these operations on an arbitrary video sequence. The algorithm implemented in this work is a simplified version of the computational framework described in [7]. The key idea is that a sequence of triplets with one-view overlap can be viewed as a hierarchy of triplets, with those in the second level or higher comprised of two triplets from the level below. The algorithm begins reconstructing this hierarchy from the first triplet, always searching for the widest triplet that improves quality, as shown in Figure 4.2. If the higher-level triplet has a higher quality than both of the lower-level ones, the process continues in this vein, continuing to widen the baseline. If the higher-level triplet has a lower quality than either of its children, it is discarded and the widest triplet found is appended to the current reconstruction. In this way, only high-quality triplets are used to build the projective reconstruction.



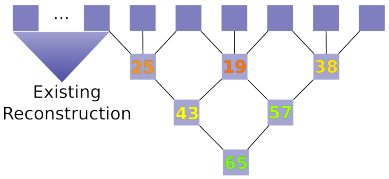
The algorithm begins by building the next level-1 triplet and calculating its quality: in this case, $Q = 25$.



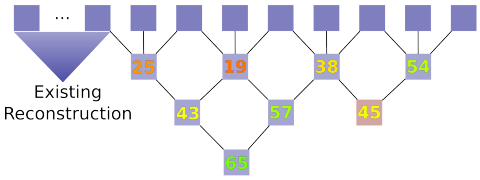
The next level-1 triplet is built, and then the parent level-2 triplet is built. With $Q = 43$ its quality is higher than either child triplet, so it is accepted.



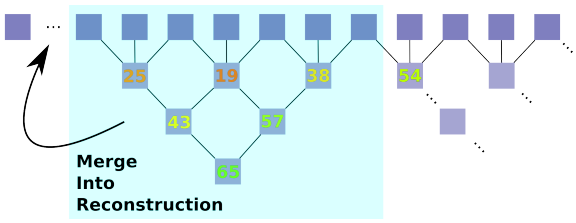
The algorithm continues searching for a wider-baseline triplet, first building the next level-1 triplet.



Two parent triplets are built, a level-2 and a level-3. In both cases their quality is higher then their children, so they are accepted.



The search continues. In this case, a level-2 triplet is calculated whose quality is lower than one of its children: this triplet is rejected.



The algorithm has identified the widest-baseline triplet that does not sacrifice quality: the level-3 triplet with $Q = 65$. It is merged into the existing reconstruction, and the algorithm repeats.

Figure 4.2: **A hierarchical reconstruction algorithm:** The algorithm proceeds through the sequence of video frames, building wider and wider tensors until the quality is no longer improved. The highest-quality tensor is then added to the existing reconstruction, and the process is repeated.

Chapter 5

Projective Reconstruction — Testing

The projective reconstruction method has been tested on a variety of synthetic and then real image sequences. For a projective reconstruction, the only relevant error metric is reprojection error; all results are judged using this metric.

5.1 Testing on Synthetic Data

Synthetic image sequences were created to test the algorithm under ideal conditions. A cubic volume of regularly spaced points and a series of virtual cameras were created. The cameras were placed randomly near the surface of a sphere around the points, and point towards their center. All points were then projected by each camera matrix to create the image observations, with zero-mean Gaussian noise added to each observation in the image space. The virtual images had size 1024x768 pixels. The number of points, number of camera views, and magnitude of the noise were all variable. A sample synthetic data set with 125 points in 11 views is shown in Figure 5.1.

The average reprojection error is plotted against the standard deviation of the added noise in Figure 5.2. The algorithm performed as expected, with higher reprojection error as noise increases. It should be noted that tracked image features usually have error in the 0.5-1.0 pixel range, where the algorithm performed very well. Ten-pixel tracking error was never seen in this work for successfully tracked image features.

The average reprojection error is plotted against the number of views in the synthetic sequence in Figure 5.3; four noise levels are considered: 0.5, 1.0, 1.5 and 2.0 pixels. It is clear that the algorithm can suppress error accumulation very well. The notable drops in error

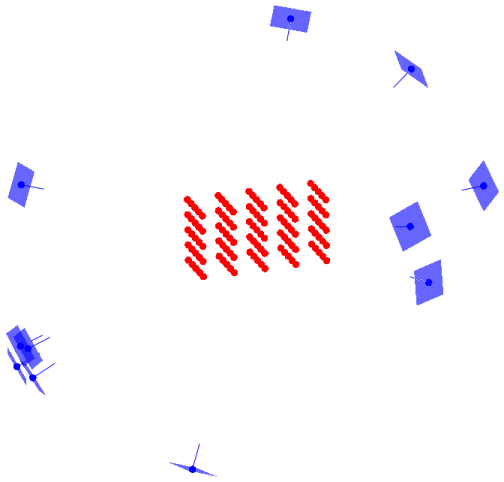


Figure 5.1: **The Synthetic Dataset:** A randomly generated synthetic grid of 125 points, shown in red, and 11 camera views shown in blue. Each camera's image plane is displayed and a line shows the direction the camera points, with its length proportional to the camera's focal length.

for all four sequences in views 9, 17, and 25 are caused by the hierarchical algorithm using only one top-level triplet for the reconstruction. As the reconstruction progresses from each of these points additional triplets must be merged together, and error is introduced in the merging step. It should be noted that these synthetic datasets do not share several important properties with image sequences taken from video: all synthetic points are visible in all views, and since the synthetic cameras are randomly positioned, keyframe number does not actually serve as a proxy for baseline. This motivates further testing on the image sequences for which this algorithm was designed.

5.2 Testing on Standard Datasets

The 3D reconstruction literature suffers from a shortage of standard datasets[8]. Most of those that exist are available only as raw images without extracted features. Because this work relies on video tracking it does not include tools to generate correspondences from widely-spaced still images, and raw image data cannot be used to generate recon-

structions. However, Oxford's Visual Geometry Group has made three datasets available, complete with tracked features [36]. The "Dinosaur", "Model House", and "Corridor" sequences have thus been widely (though not universally) used in the literature [3, 7]. One such publication[3] uses a similar method to this work, and the results are compared in Table 5.1. Three sub-sequences of the "Dinosaur" sequence are presented, as well as the whole sequence. Though it appears that this work outperforms [3] considerably, the algorithms are different enough that reprojection error alone cannot express the difference in reconstruction quality. Due to an additional guided matching step present in [3], many more points are contained in their reconstructions compared to those of this work. This results in the proposed algorithm producing lower reprojection errors for the smaller subsequences, because only the points that most closely fit the reconstruction are included. The proposed algorithm produces a higher reprojection error for the full "Dinosaur" sequence compared to [3] because fewer points are common between sub-sequences; this leads to fewer constraints and higher error during the merging step.

The projective reconstruction algorithm presented here is in the same class as state-of-

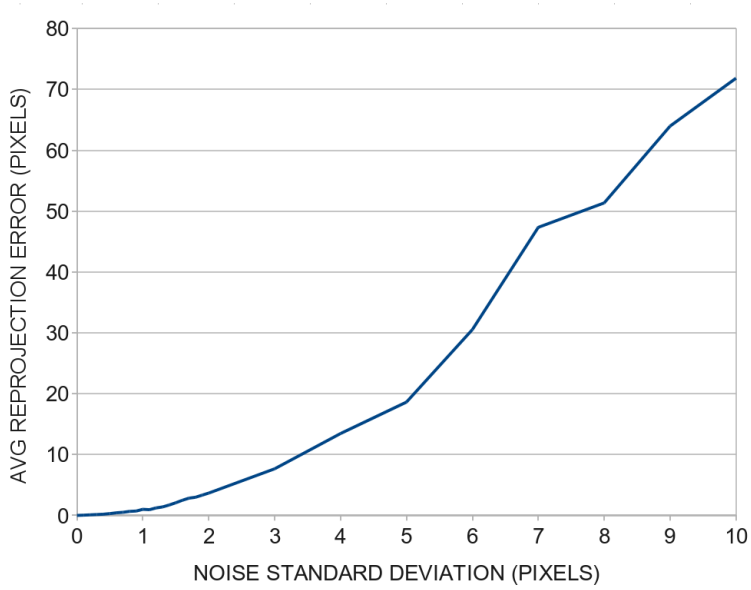


Figure 5.2: **Synthetic Data: Reprojection error vs noise:** Projective reconstruction performance under increasing noise for synthetic datasets of the type shown in Figure 5.1, containing 125 points in 3 views. The reprojection error plotted is the mean for all points, averaged over 100 trials.

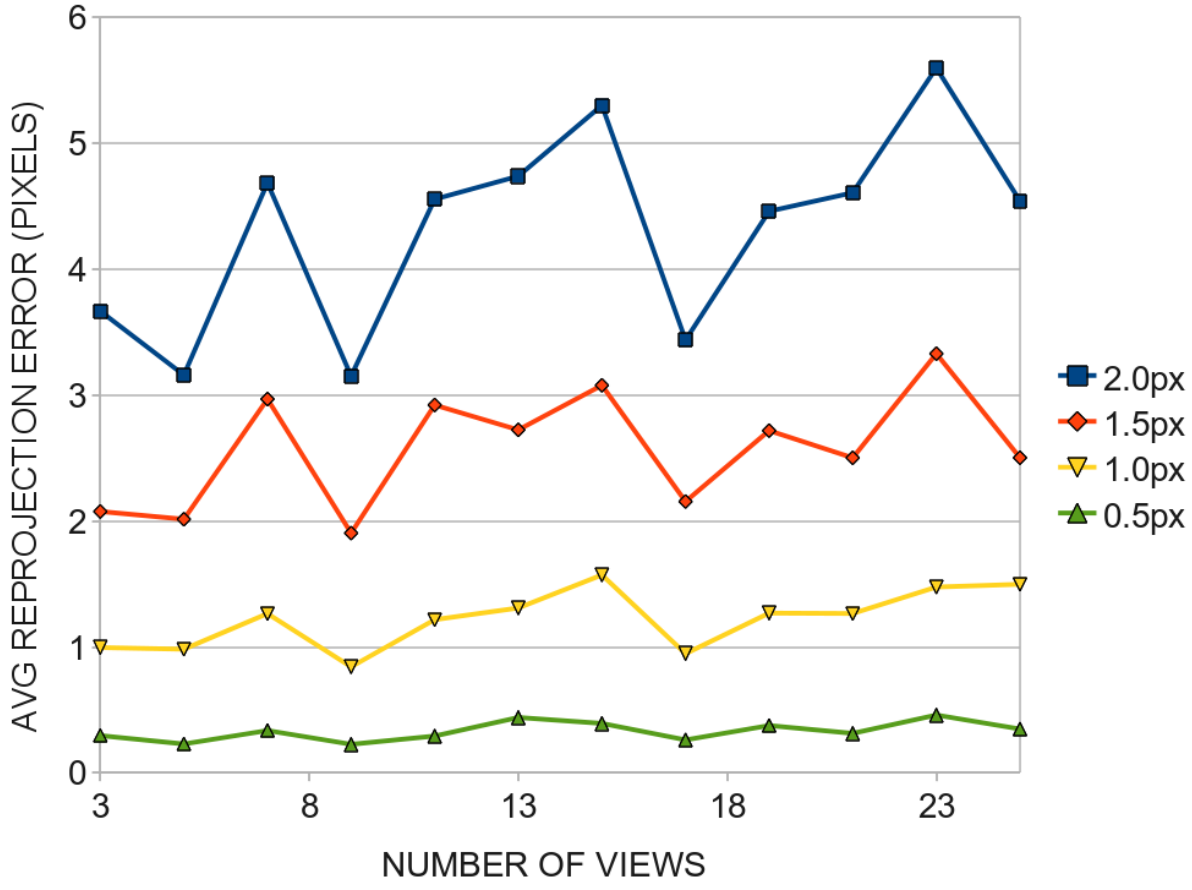


Figure 5.3: **Synthetic Data: Reprojection error vs views:** Projective reconstruction performance as the number of views increases. Four data series are plotted, each from a synthetic dataset of the type shown in Figure 5.1, and each with a different noise level: 0.5, 1.0, 1.5 and 2.0 pixels respectively. The reprojection error plotted is the mean for all points, averaged over 100 trials.

the-art algorithms. It is not perfect, and there are several features known in the literature that would improve its performance; these are discussed in Chapter 10. In its current state, the proposed algorithm produces results on underwater video sequences that are "good enough" to allow autocalibration to proceed.

Sequence	linc method of [3]	This work
Dinosaur 0-6	0.198	0.059
Dinosaur 6-12	0.197	0.099
Dinosaur 12-18	0.197	0.111
Dinosaur 0-18	0.161	0.760
Corridor 0-6	0.228	0.068

Table 5.1: Average reprojection error in pixels for image sequences published in [3], compared to the automatic hierarchical algorithm of this work. The proposed algorithm produces lower reprojection errors for short 6-frame sequences, but higher error for the long 18-frame sequence.

5.3 Underwater Video Data: The Need for Autocalibration

The projective reconstruction algorithm also runs successfully on underwater video data. Rather than presenting another table of reprojection errors, graphical results are deferred until Chapter 8. As a metric for the quality of a reconstruction, reprojection error is useful but limited. To understand why, one need only look at Figure 5.4: projective reconstructions can be distorted beyond all recognition without affecting reprojection error. In order to present graphic results from real video sequences, autocalibration is a necessary step.

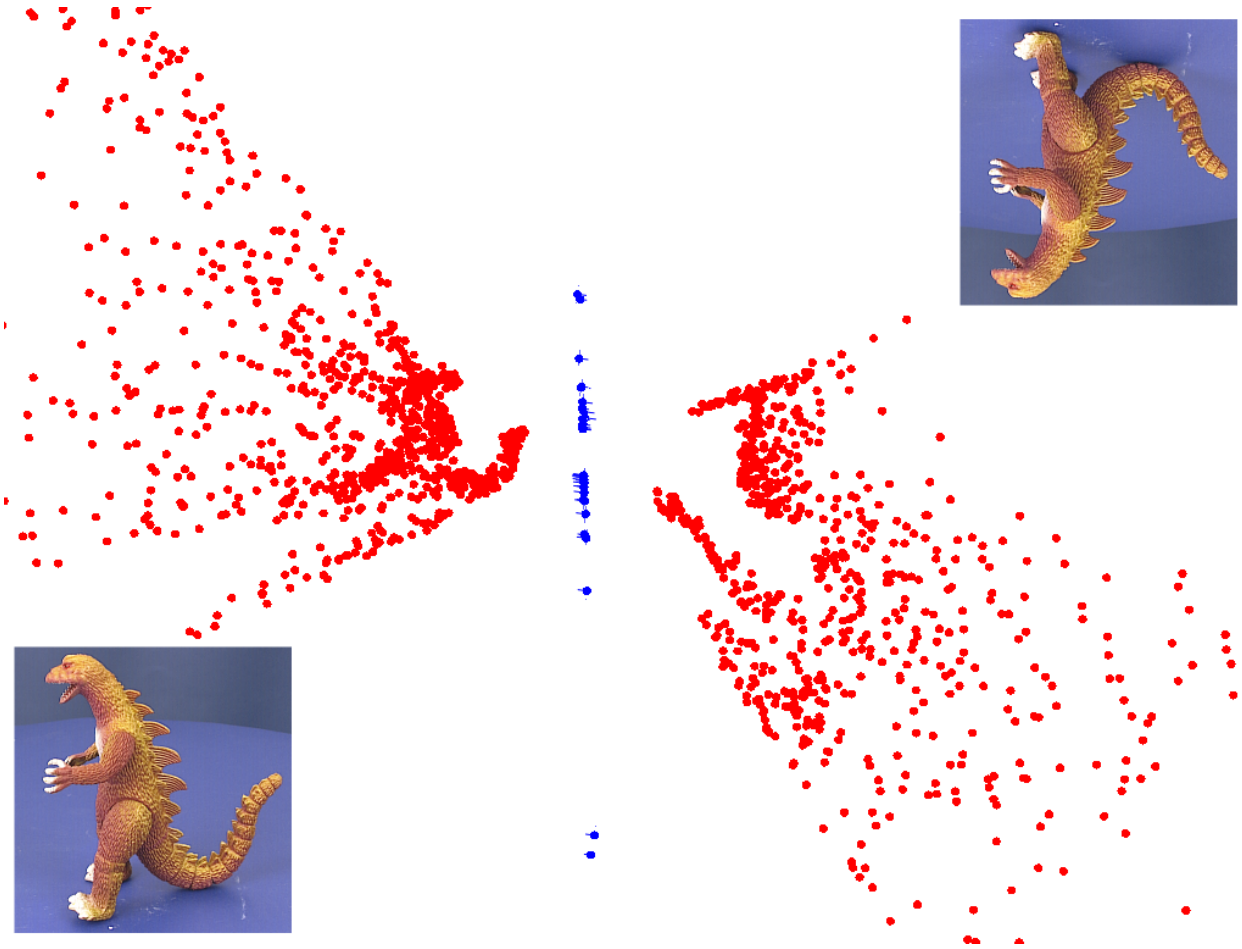


Figure 5.4: **The Limits of Reprojection Error:** Can you see the dinosaur? Despite its appearance, this reconstruction of the "Dinosaur" sequence has extremely low reprojection error. This sequence has already been processed by the first stage of an autocalibration algorithm; without this processing the dinosaur would not be recognizable at all, however the reprojection error would still be equally low. Clearly, reprojection error alone cannot measure the quality of a 3D reconstruction.

Chapter 6

Autocalibration — Background

This chapter describes the process of autocalibration, the recovery of camera parameters using only image data. Conceptually, this process is akin to flattening a fun house mirror. A projective reconstruction produces an image that is correct, but gives no clue as to the shape of the lens. As the results of Chapter 5 show, a purely projective reconstruction is not useful; camera calibration is necessary before the results of a projective reconstruction are usable for visualization or measurement. Autocalibration recovers the metric camera parameters automatically, and transforms a projective reconstruction into a usable metric reconstruction.

6.1 Fundamentals

The goal of autocalibration is to find the transformation $\mathbf{H}^*$ that maps the projective reconstruction to a metric space. This is equivalent to finding the plane at infinity $\hat{\mathbf{l}}_\infty = [l_1, l_2, l_3, 1]$ as well as the intrinsics for all cameras, $\{\tilde{\mathbf{K}}_m\}$, as in (3.5). As (3.2) implies, $\mathbf{H}^*$ is an invertible transformation matrix and it is sometimes more computationally convenient to solve for $(\mathbf{H}^*)^{-1}$. Transforming a projective reconstruction to a metric space requires both $\mathbf{H}^*$ and $(\mathbf{H}^*)^{-1}$ in any case; once either has been found the other is calculated using SVD. SVD is chosen because it provides more accurate inversion for near-singular matrices.

Constraints for solving the autocalibration problem derive from prior knowledge that $\tilde{\mathbf{K}}$ represents a physical camera. This requirement for modern CCD cameras can be sum-

marized in three prior conditions:

rectangular pixels (zero skew)	$s = 0$
square pixels (unit pixel aspect ratio)	$f_u = f_v = f$
principal point at image center	$p_u = p_v = 0$

(6.1)

Using the assumptions of (6.1), and further assuming that one of the cameras in the reconstruction has been made canonical by (3.3), the camera intrinsics of this canonical camera can therefore be written as:

$$\tilde{\mathbf{K}}_{can} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Focal length and the location of the plane at infinity have a much greater impact on projective distortion than the other parameters of $\tilde{\mathbf{K}}$ [37], so an appropriate parameterization of $(\mathbf{H}^*)^{-1}$ is [2]:

$$(\mathbf{H}^*)^{-1} = \begin{bmatrix} \tilde{\mathbf{K}}_{can}^{-1} \, \mathbf{0} \\ \hat{\mathbf{l}}_\infty \end{bmatrix} = \begin{bmatrix} 1/f & 0 & 0 & 0 \\ 0 & 1/f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l_1 & l_2 & l_3 & 1 \end{bmatrix} \tag{6.2}$$

This form is more straightforward than solving for $\mathbf{H}^*$, and some autocalibration algorithms[9, 7] optimize these four parameters directly using non-linear gradient descent.

Some autocalibration methods[38, 39] make use of a fundamental identity of projective geometry:

$$\omega_m^* = \mathbf{K}_m \mathbf{K}_m^T \propto \mathbf{P}_m \mathbf{\Omega}^* {\mathbf{P}_m}^T \tag{6.3}$$

where the dual image of the absolute conic, $\omega^* \in \mathbb{R}^{3\text{x}3}$, encodes the camera calibration parameters of $\tilde{\mathbf{K}}$ and the absolute dual quadric, $\mathbf{\Omega}^* \in \mathbb{R}^{4\text{x}4}$, encodes this information as well as the location of $\hat{\mathbf{l}}_\infty$ in the projective reconstruction. Some important properties of $\mathbf{\Omega}^*$ are that it is symmetric, positive definite, has rank 3, and as with all projective quantities only the ratio of its elements is important. Taken together, these properties mean that $\mathbf{\Omega}^*$ has only 8 free parameters in general. When the assumptions of (6.1) are made, (6.3) can be further simplified to the form:

$$\begin{bmatrix} f^2 & 0 & 0 \\ 0 & f^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \propto \mathbf{P}_m \begin{bmatrix} b_1 & 0 & 0 & b_2 \\ 0 & b_1 & 0 & b_3 \\ 0 & 0 & 1 & b_4 \\ b_2 & b_3 & b_4 & b_5 \end{bmatrix} {\mathbf{P}_m}^T \tag{6.4}$$

where all of b_i can be expressed in terms of f and the three parameters of $\hat{\mathbf{l}}_\infty$. Thus, in this form, $\mathbf{\Omega}^*$ has the same four parameters as the parametrization of (6.2).

Once $\mathbf{\Omega}^*$ is known, ($\mathbf{H}^*$) can be found via the decomposition

$$\mathbf{\Omega}^* = \mathbf{H}^* \, \tilde{\mathbf{I}} \, \mathbf{H}^{*T} \tag{6.5}$$

where $\tilde{\mathbf{I}} = diag(1, 1, 1, 0)$.

6.2 Existing Methods

A number of different methods for estimating $\mathbf{H}^*$ exist in the literature, and they fall mainly into two categories: those based on the absolute conic that solve for $\mathbf{H}^*$ using (6.3) [40, 41, 38, 39], and those that use iterative nonlinear optimization of the four parameters in (6.2) [37, 10, 9].

6.2.1 Solutions Based on ω^* and $\mathbf{\Omega}^*$

Early efforts in autocalibration focused on the manipulation of (6.3) based on the assumptions in (6.1) to produce constraints in the elements of ω^* and/or $\mathbf{\Omega}^*$.

The Kruppa Equations

The Kruppa equations are related to the dual image of the absolute conic, ω^*, and are two-view constraints. Given F between two views, the Kruppa equations provide two independent quadratic constraints in the elements of ω^* [2]. Widely viewed as the first autocalibration method [40], they require a minimum of three cameras having identical camera intrinsics, with $\mathbf{F}$ known between views 1-2, 1-3, and 2-3, to solve for the five parameters of $\mathbf{K}$ via ω^*. The Kruppa equations provide weaker constraints than newer methods, and are not recommended for use in modern autocalibration software[2]. Details of this method are given in [2].

Linear constraints on $\mathbf{\Omega}^*$

The form of (6.4) allows three linear equations in the elements of $\mathbf{\Omega}^*$ to be obtained from each camera in a projective reconstruction, only two of which are linearly independant.

Three or more views are therefore required to obtain a least-squares solution for the five elements of $\mathbf{K}$[38]. This method is straightforward and non-iterative, but is very sensitive to the accuracy of the camera matrices[2]. In addition, the positive-definiteness and rank-3 properties of $\mathbf{\Omega}^*$ are not enforced: the solution generated by this method does not necessarily result in a valid $\mathbf{\Omega}^*$[39]. Details of this method are given in [38].

Quadratic solution for $\mathbf{\Omega}^*$

To address the shortcomings of the linear method described above, it is possible to use the linear solution as the initialization to an iterative optimization. The method proposed in [39] uses Sequential Quadratic Programming (SQP), a second-order gradient descent technique, to improve the estimate of $\mathbf{\Omega}^*$. The cost function for optimization is the sum of squared deviations from (6.4). Constraints are added to the problem to enforce normalization of $\mathbf{\Omega}^*$ and ω^* (ensuring that the proportionality of (6.3) can be used as an equality), and the rank-3 property of $\mathbf{\Omega}^*$: $\det(\mathbf{\Omega}^*) = 0$. The positive-definiteness of $\mathbf{\Omega}^*$ can also be enforced at each iteration via eigendecomposition, zeroing the smallest eigenvalue, and re-composing $\mathbf{\Omega}^*$ with only three non-zero eigenvalues. Although this solution is much improved over the linear method, it is not perfect. Discussion with the author of [39] confirmed that the optimization will only converge if the linear method produces an estimate of each parameter that is accurate within approximately a factor of two. Further, enforcing the positive-definiteness of $\mathbf{\Omega}^*$ at each iteration can cause further convergence issues; the revised solution may violate the rank-3 and normalization constraints of the SQP problem. Details of this method are given in [39].

6.2.2 The Direct Approach

More recent work on autocalibration takes a much more direct approach to estimating $(\mathbf{H}^*)^{-1}$: obtain a rough initial guess of the parameters in (6.2), apply the transformation to see how close the resulting camera matrices are to the prior assumptions in (6.1), and then refine the parameters using non-linear optimization of a robust cost function. Several different initialization schemes have been tried in a Levenberg-Marquardt solver [37, 9], and a branch-and-bound solver has been developed that does not need initialization [10]. This work adopts the direct approach, and these methods will be described in more detail in Chapter 7.

Chapter 7

Autocalibration — Methodology

The linear and quadratic $\mathbf{\Omega}^*$-based autocalibration methods, discussed in Chapter 6.2.1 and 6.2.1, were implemented early in this work. Although the SQP method showed some success with synthetic data and the well-conditioned research data sets of Chapter 5.2, it failed completely when tested on underwater video sequences. This failure can be attributed to both initialization difficulties and convergence issues. The linear method used as initialization to the SQP optimization is not guaranteed to produce a valid solution, let alone one within a factor of two of the true solution as required by the SQP method. The SQP method itself displayed convergence problems when positive-definiteness of $\mathbf{\Omega}^*$ was enforced at each iteration, and otherwise would often generate a solution for $\mathbf{\Omega}^*$ that was not positive-definite, and therefore invalid. It is due to these deficiencies that $\mathbf{\Omega}^*$-based autocalibration methods are no longer considered state of the art, and results from these methods are not presented here.

This work performs autocalibration using a more recent approach: direct optimization of a cost function based on the parameterization of (6.2). Given the four parameters $[f,\ l_1,\ l_2,\ l_3]$ and a prior for $\tilde{\mathbf{K}}$, (6.1), the cost function used to evaluate potential reconstructions is outlined in Algorithm 7.1. This cost function is optimized using the Levenberg-Marquardt method[35]. Because the transform-decompose process is highly nonlinear, there are two key challenges to this local gradient descent approach: the fitness measure $\mathcal{C}(\{\mathbf{K}_m\})$ must be highly robust, and the initial guess must be sufficiently close to assure convergence.

Three methods for autocalibration are compared in this thesis in order to illustrate the importance of both initialization and a robust fitness measure:

1. **Method (I) - Nister's Method**[37] — fragile initialization, robust cost measure

Algorithm 7.1 A Direct Cost Function for Autocalibration: The procedure for evaluating a proposed metric upgrade transform $\mathbf{H}^*$ based on the intrinsics of the resulting cameras. This generalized cost function is used by methods (I), (II) and (III) with varying fitness measures $\mathcal{C}(\{\mathbf{K}_m\})$.

Require: $\{\mathbf{P}_m\}$
Require: $(\mathbf{H}^*)^{-1}$ in (6.2)

fitness $\Leftarrow 0$
for $i = 0$ to m **do**
  ${\mathbf{P}^*}_m \Leftarrow \mathbf{P}_m(\mathbf{H}^*)^{-1}$
  obtain ${\mathbf{K}^*}_m$ from (3.5)
  fitness $+= \mathcal{C}(\{\mathbf{K}_m\})$ for some fitness measure
end for
return fitness

2. **Method (II) - Gherardi's Method**[9] — robust initialization, fragile cost measure
3. **Method (III) - The Proposed Hybrid Method** — adopts the best of both

All results were produced using original implementations; code for the methods introduced by other researchers was requested, but unavailable. For all methods, the fitness measure for a good metric reconstruction should be small: the minimum fitness value for a perfect metric reconstruction is 0, and the maximum value for the fitness function is unbounded. The formulation of the fitness measure should penalize any deviation from prior expectations, and a large fitness value indicates a reconstruction in which the decomposed camera matrices do not have properties expected of physical cameras.

7.1 Method (I) — Nister's Method

Method (I)[37] attempts to solve the initialization problem by preconditioning the projective reconstruction to ensure that it is a quasi-affine reconstruction (QUARC) with respect to the camera centers. The QUARC concept was first developed by Hartley[41], and if a reconstruction is quasi-affine then it is guaranteed that the reconstruction lies entirely on one side of $\hat{\mathbf{I}}_\infty$. This is particularly useful for camera calibration because decomposition of (3.5) is applicable only to finite cameras, and should local perturbation attempt to move

$\hat{\mathbf{l}}_\infty$ across a camera center any cost function will behave erratically as the decomposition fails. The method of [37] defines a twist test for two cameras to determine whether $\hat{\mathbf{l}}_\infty$ lies between them, and uses this to define a signature for $\mathbf{l}_\infty$: $\zeta(\mathbf{l}_\infty, \mathbf{P}_{1\ldots m})$. The signature ζ is a vector of length m whose entries are either 1 or -1 depending on the results of the twist test. To obtain a QUARC, it is sufficient to map to infinity a plane $\mathbf{l}^*$ that has the same signature as $\hat{\mathbf{l}}_\infty$, and is as far as possible away from all camera centers. This can be expressed as a linear program [43]:

$$\begin{array}{lll} \max & \delta & \\ \text{s.t.} & \mathbf{l}^* \frac{\mathbf{c}(\mathbf{P})_i)}{||\mathbf{c}(\mathbf{P})_i)||_2} \zeta(\mathbf{l}_\infty, \mathbf{P}_{1\ldots m})_i - \delta \geq 0 & \forall i \\ & -1 \leq l_j^* \leq 1, & j = 1\ldots 4 \end{array}$$

where $\zeta(\mathbf{l}_\infty, \mathbf{P}_{1\ldots m})_m$ is the mth element of the signature, and the bounds on $\mathbf{l}_i^*$ guarantee a unique solution. Transforming the projective reconstruction to QUARC is then accomplished by mapping $\mathbf{l}^*$ to infinity using (3.2), with

$$\mathbf{H_{QUARC}} = \begin{bmatrix} & \mathbf{I}^{3\text{x}3} & & \mathbf{0} \\ l_1^* & l_2^* & l_3^* & l_4^* \end{bmatrix} \tag{7.1}$$

A theoretical framework has been developed in [37] showing that transforming the projective space to QUARC is equivalent to placing the naive initialization of (6.2), $[f,\ l_1,\ l_2,\ l_3] = [1,\ 0,\ 0,\ 0]$ in the correct basin of the cost function, using the fitness measure

$$\mathcal{C}_I(\{\mathbf{K}_m\}) = \sum_m \frac{(f_u - f_v)^2 + s^2 + p_u^2 + p_v^2}{(f_u + f_v)^2} \tag{7.2}$$

The penalization for deviation from (6.1) is evident. Division by the factor $(f_u + f_v)^2$, which is proportional to focal length, has a normalizing effect; a given deviation of aspect ratio or principal point is more plausible at long focal lengths than short ones. (7.2) also strongly penalizes the cost function should f approach 0, where the decomposition of $\mathbf{P}$ is undefined. Without this term f could approach 0 during optimization, where the cost function is expected to produce erratic results and optimization is likely to fail. Method (I) can be summarized as QUARC preconditioning followed by nonlinear optimization of focal length and $\mathbf{l}_\infty$ location. For further details, the reader is directed to[37].

7.2 Method (II) — Gherardi's Method

Method (II) [9] uses the fitness measure:

$$\mathcal{C}_{II}(\{\mathbf{K}_m\}) = \sum_m [\, \omega_s |s| + \omega_{ar} |f_u - f_v| + \omega_p (|p_u| + |p_v|) \,] \tag{7.3}$$

where $\omega_s = 20$, $\omega_{ar} = 2$, and $\omega_p = 1$ are appropriate weights for the terms based on prior expectations — modern CCDS have very nearly zero skew, while unit aspect ratio is less certain and principal point location much less so[44].

The initialization of $(\mathbf{H}^*)^{-1}$ is accomplished using an exhaustive search over reasonable focal lengths. A closed-form solution is given for the location of the plane at infinity given any two cameras $\mathbf{P}_\alpha = [\mathbf{I}|\mathbf{0}]$, $\mathbf{P}_\beta = [\mathbf{Q}_\beta|\mathbf{q}_\beta]$ and their intrinsics, $\tilde{\mathbf{K}}_\alpha$, $\tilde{\mathbf{K}}_\beta$. Given that the cameras are normalized by (3.6), focal length will be on the order of 1 — less for wide-angle lenses and more for telephoto lenses. For most consumer cameras, a sample space of $0.3 \leq f \leq 3$ is reasonable [9]. Making use of (6.1), all parameters of $\tilde{\mathbf{K}}$ except focal length are set to 0 during the initialization phase. Method (II) samples the space of focal lengths using Algorithm 7.2. The values of $[f,\ l_1,\ l_2,\ l_3]$ that resulted in the best fitness are used to initialize the nonlinear optimization of Algorithm 7.1.

Algorithm 7.2 Exhaustive Search Initialization for Autocalibration: The algorithm used by Method (I) and (III) to perform an exhaustive search for the plane at infinity over reasonable values of focal length.

Require: $\{\mathbf{P}_m\}$

Transform $\{\mathbf{P}_m\}$ so $\mathbf{P}_1 = [\mathbf{I}|\mathbf{0}]$ by (3.3)
$\mathbf{P}_\alpha \Leftarrow \mathbf{P}_1$
for $f = 0.3$ to 3.0 with logarithmic spacing **do**
  for $i = 2$ to m **do**
    $\mathbf{P}_\beta \Leftarrow \mathbf{P}_i$
    form $\tilde{\mathbf{K}}_\alpha$ and $\tilde{\mathbf{K}}_\beta$ from f
    solve for $\mathbf{l}_\infty^*$
    ${\mathbf{P}^*}_m \Leftarrow \mathbf{P}_m(\mathbf{H}^*)^{-1}$
    obtain ${\mathbf{K}^*}_m$ from (3.5)
    fitness $= \mathcal{C}_{II}(\{\mathbf{K}_m\})$
  end for
end for
return $[f,\ l_1,\ l_2,\ l_3]$ that resulted in the best fitness

---

7.3 Method (III) — The Proposed Hybrid Method

Method (III) is a combination of the initialization strategy of Method (II) with the fitness measure from Method (I). The hybrid fitness measure is:

$$\mathcal{C}_{III}(\{\mathbf{K}_m\}) = \sum_m \frac{\omega_{sk} s^2 + \omega_{ar}(f_u - f_v)^2 + \omega_{p_u} p_u^2 + \omega_{p_v} p_v^2}{(f_u + f_v)^2} \tag{7.4}$$

This measure clearly resembles (7.2), but has incorporated the parameter weights from (7.3) in order to increase the use of prior knowledge. The most important similarity with (7.2) is the denominator, which is proportional to focal length and penalizes very small values of f. The importance of this feature, and the motivation for developing this hybrid method, are illustrated in Chapter 8.

Chapter 8

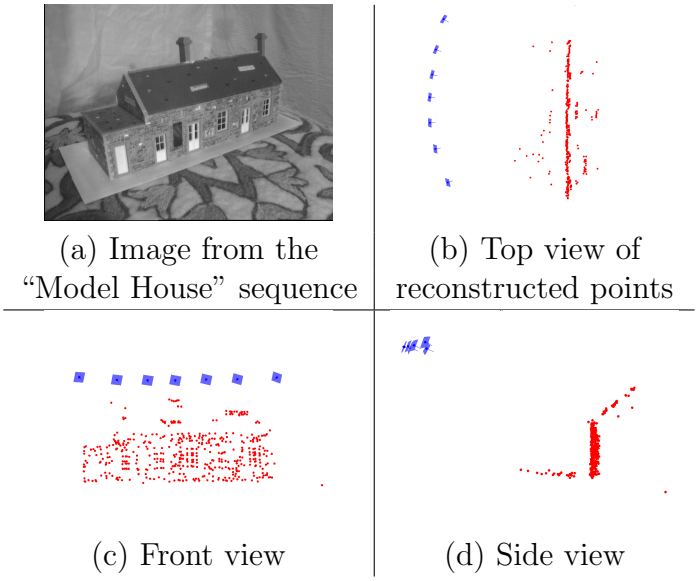
Autocalibration — Testing

The autocalibration methods were tested on a variety of datasets. The input projective reconstructions were generated using the algorithm described in Chapter 4. Because the goal of this research project is photorealistic reconstruction for intuitive inspection rather than high accuracy, the metric used to evaluate the autocalibration algorithms is purely visual: if the reconstruction looks like a reasonable rendition of the scene, then it is acceptable. A numerical 3D error for the reconstruction is unavailable for underwater video in any case, as there is no ground truth available for the underwater sequences.

8.1 Results from Standard Datasets

The three autocalibration methods have been compared for two image sequences that have appeared elsewhere in the 3D reconstruction literature[3]: the "Model House" sequence, illustrated in Figure 8.1, and the "Dinosaur" sequence, illustrated in Figure 8.2. Both sequences were taken using a stationary camera and the subject on a turntable, with approximately 10° spacing between images.

Method (I) failed due to being initialized in the wrong basin, as shown in Figure 8.5. There is another strong basin in the $-l_2$ direction, and when the optimization is initialized with $[f,\ l_1,\ l_2,\ l_3] = [1,\ 0,\ -2,\ 0]$ instead of $[1,\ 0,\ 0,\ 0]$, the result is identical to that of Method (I). Very convincing results were reported for this method using many image sequences[37], which we have been unable to reproduce. It is possible that our QUARC preconditioning is not being carried out in precisely the same way, or that some detail of normalization has been missed. Even if this is the case, the failure to implement



(a) Image from the "Model House" sequence

(b) Top view of reconstructed points

(c) Front view

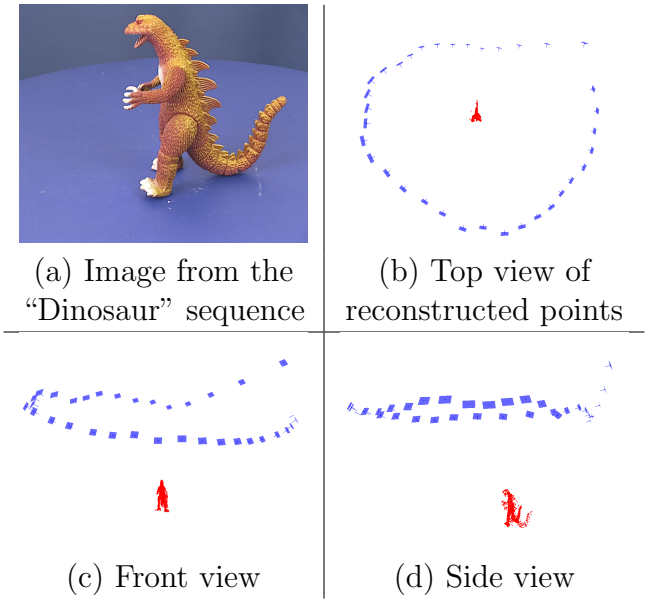
(d) Side view

Figure 8.1: **The "Model House" sequence**, reconstructed using Method (III)

the QUARC method after careful review of the literature and consultation with the author indicate that this method is sensitive to small details of the input projective reconstruction; QUARC preconditioning is neither simple, nor robust.

Method (II) failed due to focal length collapse. The exhaustive search picks $f = 0.3$, which has a cost similar to but slightly lower than those in the neighbourhood of $f = 3$. The cost function makes no distinction between these cases, though a physical camera with a short focal length is expected to have much smaller deviations from unit aspect ratio and principal point location than one with a long focal length. Because the cost function has no mechanism to penalize extremely low focal lengths, the optimization takes it into the $f \cong 0$ region, where the decomposition of $\mathbf{P}$ is ill-defined and cost function values cannot be trusted.

To understand the failure of the first two methods it is helpful to look at the behaviour of the cost function in the solution neighbourhood, which has been plotted in Figures 8.5, 8.6 and 8.7. These figures illustrate the four-dimensional space formed by the four parameters being optimized. The six graphs are of the same area in this space, and in each two of the parameters are varied while the other two are held constant. In these plots, it may sometimes appear that the optimized result is worse than the initialization. This is because for each pane the two constant parameters are held at the initialization value rather than the solution — the optimized solution is plotted in the initialization space, not



(a) Image from the "Dinosaur" sequence

(b) Top view of reconstructed points

(c) Front view

(d) Side view

Figure 8.2: **The "Dinosaur" sequence**, reconstructed using Method (III)

its own. Plots of the solution space confirm graphically that the optimization converges to a local minimum. They are omitted here because the solution space of the two methods that failed to converge to a sensible solution is not particularly informative — it is the initialization space that explains their failure.

Method (III) clearly outperforms the other two, as shown in Figures 8.3 and 8.4, producing results with very little projective distortion that would serve well as an initialization to metric bundle adjustment. As shown in 8.7, the optimization of Method (III)'s cost function is initialized within an unambiguous basin. The only other basin is in the $+l_2$ direction, and has a higher cost function value. The optimization step is fairly large in the $+f$ direction due to the true solution lying above the range $0.3 \leq f \leq 3$. Although the convergence from $f = 3$ is acceptable, an extended sample space of $0.3 \leq f \leq 10$ could easily be used.

There are clearly some errors in the reconstruction of Method (III): the cameras should form a perfect circle in Figure 8.4, and the dinosaur has a double tail as illustrated in Figure 8.8. These errors are not due to the autocalibration process, but are artifacts of the merging step of the projective reconstruction engine described in Chapter 4. As each triplet is merged into the existing reconstruction, error accumulates and the camera "drifts" from its true path. The double tail is also a symptom of this phenomenon: the tail is viewed from two sides, one near the beginning of the sequence and one near the

end. The points near the end are triangulated using the cameras in which they are first seen; these cameras have drifted and their location is incorrect, causing the points to be triangulated in the wrong location. The projective reconstruction engine is "good enough" rather than state of the art, and several improvements are discussed in Chapter 10 that will reduce error accumulation and improve reconstruction quality. Despite the imperfect projective reconstruction, the proposed Method (III) produces high-quality results with virtually none of the projective distortion evident in Methods (I) and (II).

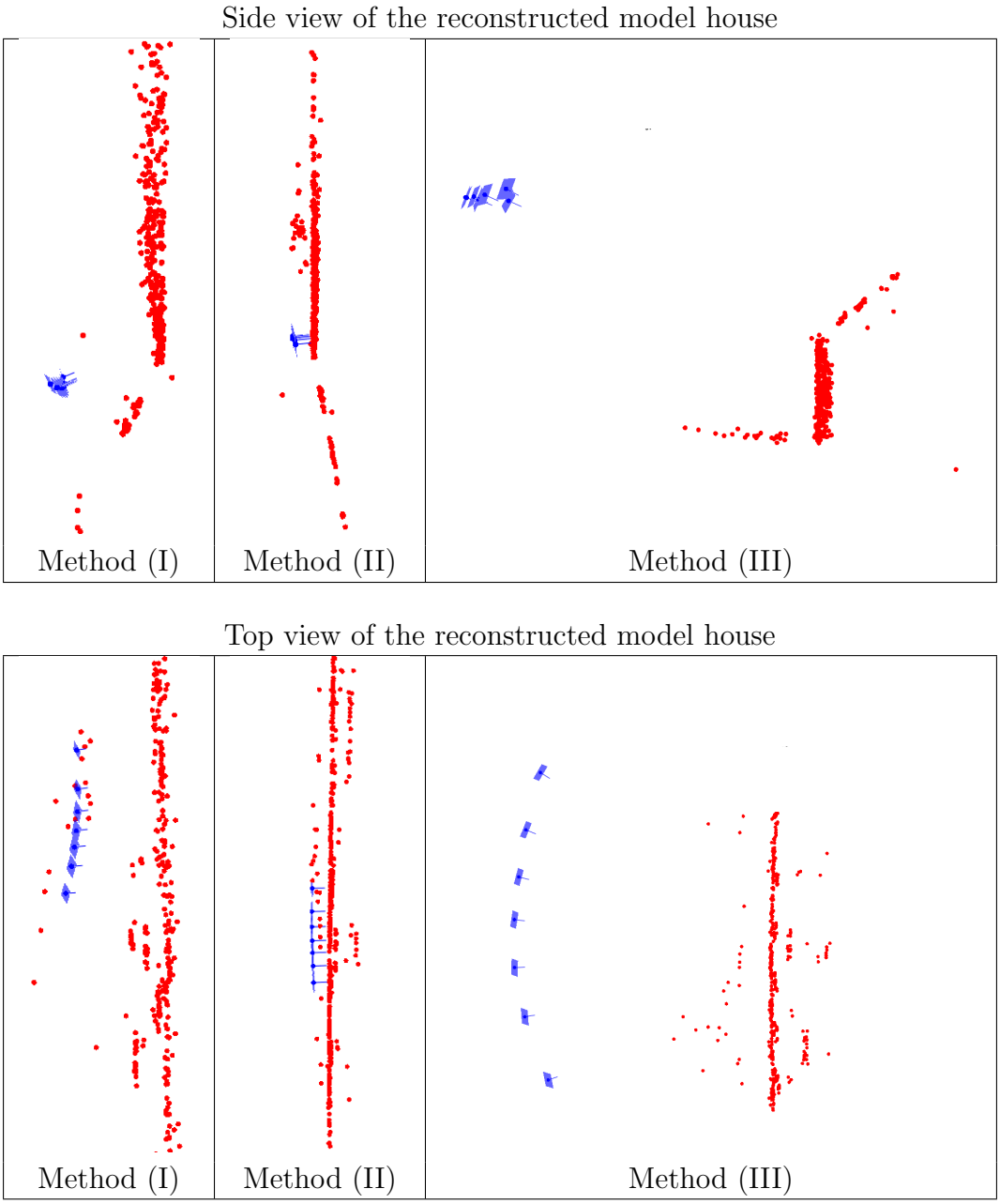


Figure 8.3: **Autocalibration results for The "Model House" sequence:** A comparison of the three metric reconstruction methods for the "Model House" sequence, shown in Figure 8.1. Methods (I) and (II) exhibit severe projective distortion, but the separate planes of the floor, wall, and roof of the house can be identified with some difficulty. By contrast, the proposed Method (III) produces a result with only slight projective distortion, illustrated by the slightly less than 90 ° angle between the floor and the wall.

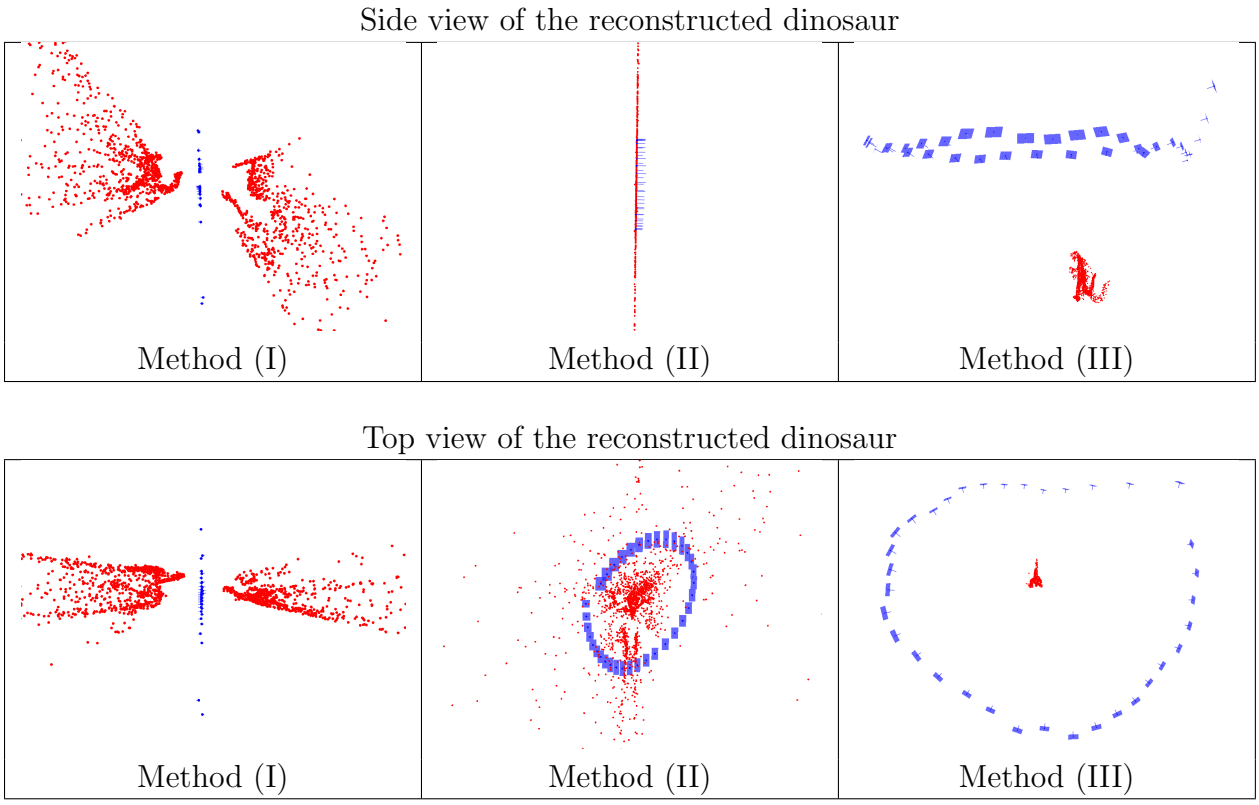


Figure 8.4: **Autocalibration results for The "Dinosaur" sequence:** A comparison of the three metric reconstruction methods for the "Dinosaur" sequence, shown in Figure 8.2. The plane at infinity still intersects the reconstruction after applying Method (I). The extreme proximity of the cameras to the scene in the side view of Method (II) indicates focal length collapse. Method (III) produces acceptable results: the Dinosaur sequence is clearly recognizable

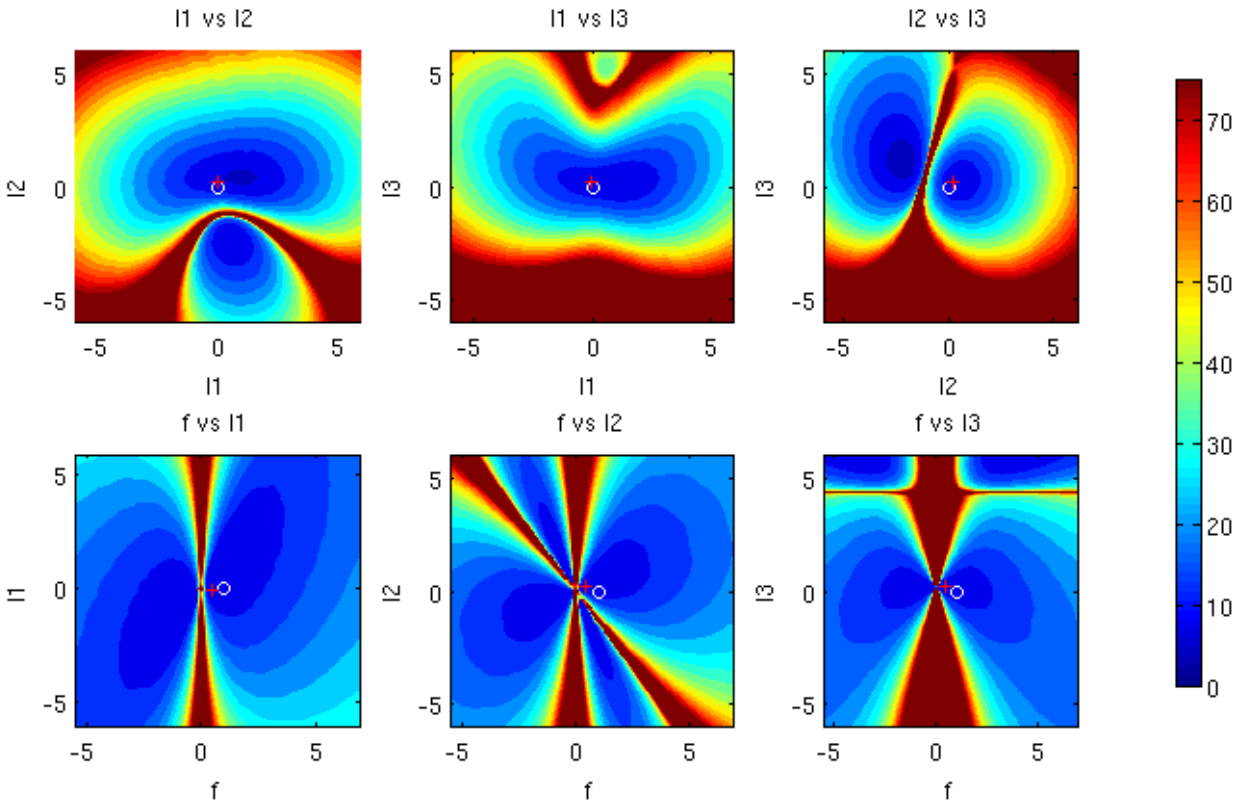


Figure 8.5: **Solution neighbourhood of $\mathcal{C}_I(\{\mathbf{K}_m\})$ plotted against its parameters:** The value of $\mathcal{C}_I(\{\mathbf{K}_m\})$ is plotted for the "Dinosaur" sequence. The white circle shows the naive initialization in the QUARC frame, and the red cross shows the optimized result. Note the other prominent basin in the cost function in the $-l_2$ direction; this basin contains the true solution for $\mathbf{l}_\infty$.

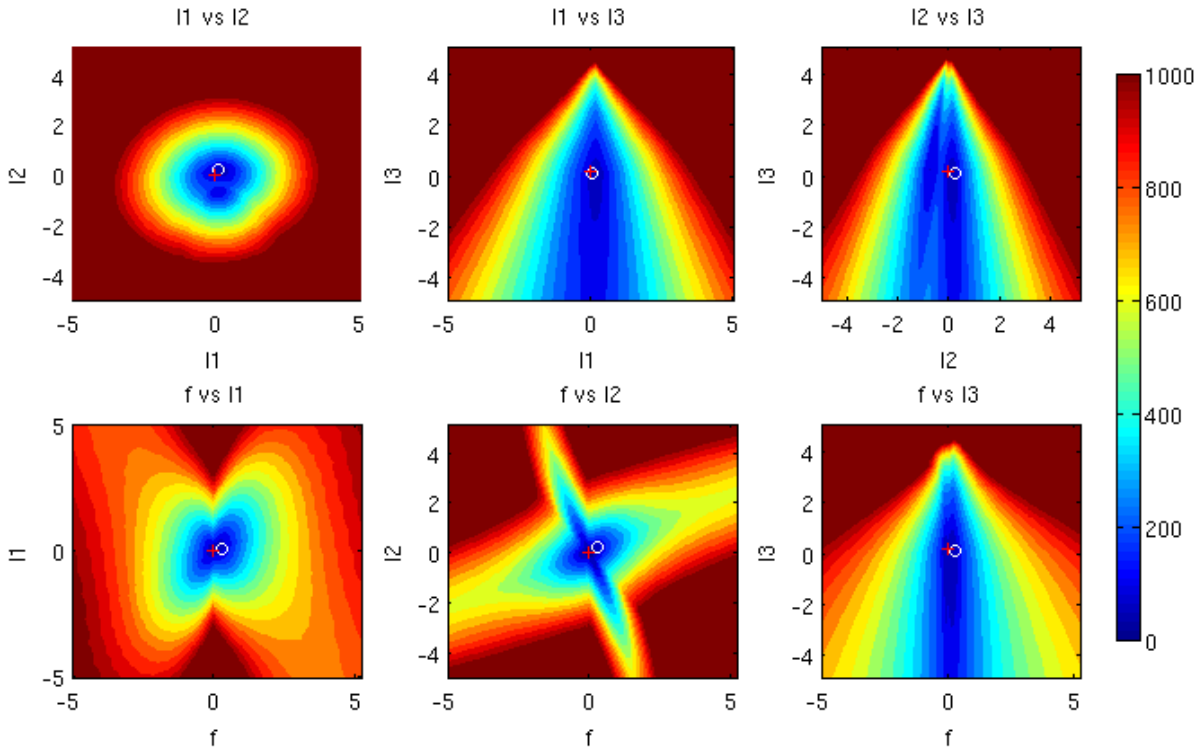


Figure 8.6: **Solution neighbourhood of $\mathcal{C}_{II}(\{\mathbf{K}_m\})$ plotted against its parameters:** The value of $\mathcal{C}_{II}(\{\mathbf{K}_m\})$ is plotted for the "Dinosaur" sequence. The white circle shows the initialization found by exhaustive search, and the red cross shows the optimized result. Note the large values compared to the other two methods. This basin is a numerical artifact generated by focal length collapse.

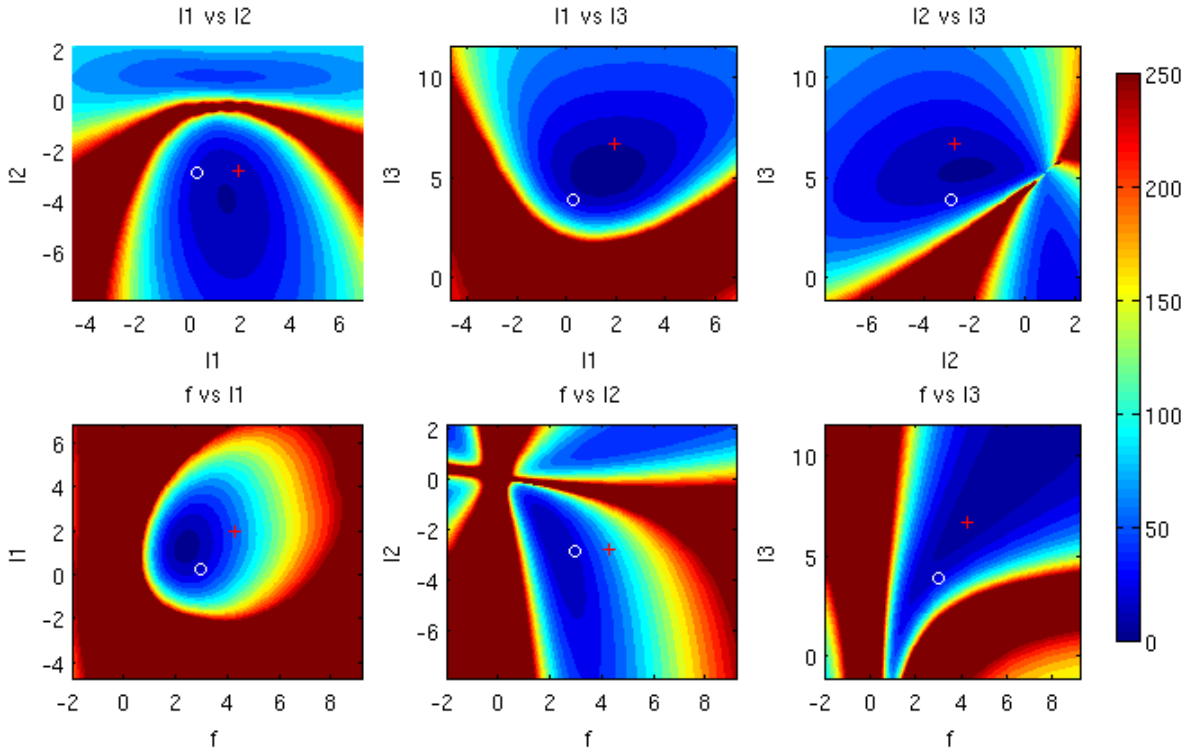
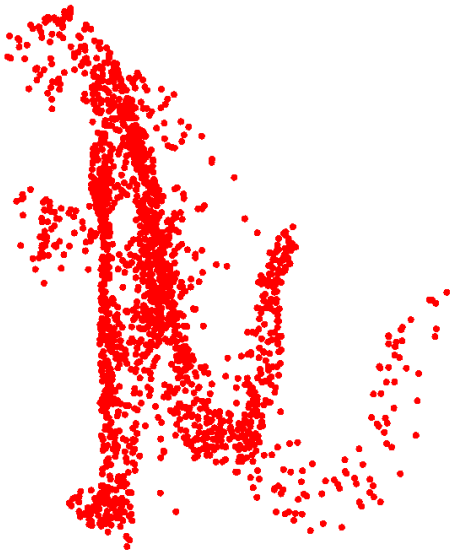


Figure 8.7: **Solution neighbourhood of $\mathcal{C}_{III}(\{\mathbf{K}_m\})$ plotted against its parameters:** The value of $\mathcal{C}_{III}(\{\mathbf{K}_m\})$ is plotted for the "Dinosaur" sequence. The white circle shows the initialization found by exhaustive search, and the red cross shows the optimized result. This plot shows Method (III) working well: there is only one strong basin in this neighbourhood, and it has a very low minimum. The optimized solution produces good results, as shown in Figure 8.4.



(a) Reconstruction of the "Dinosaur" sequence by Method (III)

(b) Image from the "Dinosaur" sequence

Figure 8.8: **Closeup of the Dinosaur:** reconstructed using Method (III). Note the doubled tail on the dinosaur.

8.2 Results from Underwater Video

Autocalibration Method (III) produces good results by combining the successful parts of its parent methods. It is the only method that performed acceptably well on video sequences, and all video results presented here were autocalibrated using Method (III). The results can be difficult to interpret on a 2D page, and Figures 8.9 and 8.10 may not adequately represent the results on their own. To augment the reader's understanding of the 3D reconstruction results, the underwater videos themselves and video of the reconstructions rotating in a 3D environment can be viewed online at the accompanying website:

http://www.eng.uwaterloo.ca/~nrcavan/MASc_UW_Recon.html

8.2.1 Silty Shipwreck

The "Silty Shipwreck" sequence consists of 480 frames, with a resolution of 704x476 pixels. The sequence is challenging due to the high level of silt in the video; moving silt particles are often strong features in the image, and are tracked reliably. However, because they are not stationary 3D points in a rigid scene, they must be discarded as outliers during projective reconstruction.

Metric reconstruction results from the "Silty Shipwreck" sequence, generated by Method (III), are presented in Figure 8.9. The video tracker identified 23 keyframes, and the reconstruction consists of three trifocal triplets built from keyframes 1-9-19, 19-20-21, and 21-22-23, containing 627 structure points. Clearly, some projective distortion remains despite the autocalibration process. This is due to the small baseline of the video itself: rather than circling around the scene, the ROV observed it from a relatively stationary viewpoint compared to the depth of the scene. The scene is thus allowed to "stretch" away from the cameras without penalty, since no side view of the scene can give additional constraints. This also explains the distribution of keyframes selected for the reconstruction: the first 19 keyframes exhibit very little motion, and only a single triplet is built from them.

8.2.2 Ship's Bow

The "Ship's Bow" sequence consists of 155 frames, with a resolution of 640x480 pixels. This sequence has no silt, but instead contains several fish that pose the same problem of non-rigid structure. This sequence is also somewhat more blurry than the "Silty Ship-

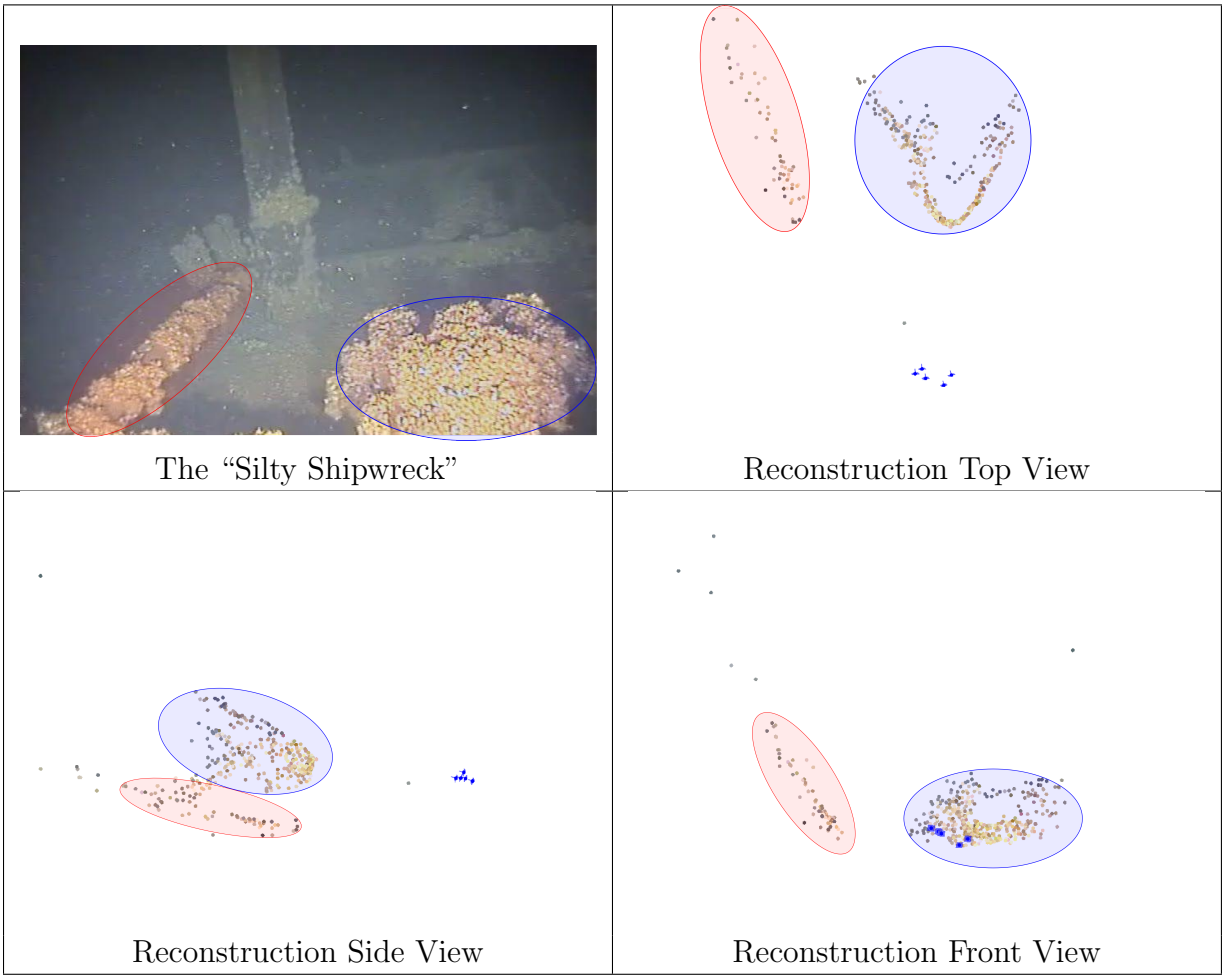
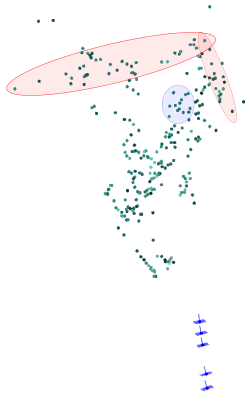


Figure 8.9: **3D Reconstruction of the "Silty Shipwreck" sequence:** The mussel-covered mass, shown in blue, and the fallen mast, shown in red, are clearly recognizable despite some remaining projective distortion. It can be difficult to interpret this 3D point cloud using only 2D views. For a better understanding of the 3D structure of the reconstruction, please visit the accompanying website to view a video of the reconstruction.

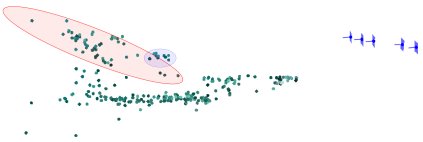
wreck" sequence and has lower contrast, making feature tracking more difficult. A higher correlation threshold was used to prevent mismatched points, as discussed in Chapter 2.1.2.

Metric reconstruction results from the "Ship's Bow" sequence, generated by Method (III), are presented in Figure 8.10. The video tracker identified 13 keyframes, and the reconstruction consists of two trifocal triplets built from keyframes 1-5-9 and 9-11-13, containing 291 structure points. There is no visible projective distortion remaining after autocalibration.

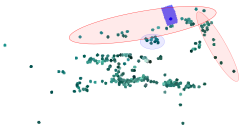
The "Ship's Bow"



Reconstruction Top View



Reconstruction Side View



Reconstruction Front View

Figure 8.10: **3D Reconstruction of the "Ship's Bow" sequence:** The railings, shown in red, and mast, shown in blue, are discernible above the deck; there is no visible projective distortion. It can be difficult to interpret this 3D point cloud using only 2D views. For a better understanding of the 3D structure of the reconstruction, please visit the accompanying website to view a video of the reconstruction.

Chapter 9

Conclusion

A software pipeline has been developed that can produce a 3D point cloud from uncalibrated video data. Results from the pipeline's three stages have been presented:

The video tracker, using the Lucas-Kanade optical flow algorithm described in Chapter 2, is able to reliably produce image feature correspondences for slow-moving underwater video.

The projective reconstruction algorithm uses the $\mathcal{T}$-based reconstruction algorithm described in Chapter 4.1.1 as the engine for the robust solver described in Chapter 4.1.3. This generally works quite well, but the solver will occasionally fail to produce acceptable results, particularly on long video sequences. The failure is intermittent: run on the same data set, the projective reconstruction algorithm will work most of the time but fail completely in approximately 5% of trials. This is possible due to the random component of the MSAC algorithm, which can produce different results given the same input. The failure occurs in a single triplet, but because each triplet is appended to the reconstruction through a shared camera, one bad link in the "chain" causes the whole reconstruction process described in Chapter 4.2.3 to fail. The failure is easy to detect: reprojection error remains low until the bad triplet is reached, and then immediately rises by several orders of magnitude. For this reason, failed reconstructions were simply omitted from the test results, and the algorithm re-run until it succeeded. The result of a successful reconstruction is a set of camera matrices and structure points whose reprojection error is comparable to state-of-the-art methods, as shown in Table 5.1.

Three autocalibration methods were compared in Chapter 7, with the proposed hybrid method adopting the best parts of two existing methods that failed to produce acceptable results. This novel autocalibration method uses an exhaustive search initialization over

reasonable camera focal lengths, described in Chapter 7.2, to obtain initial values for the four parameters that most strongly influence projective distortion: $[f,\ l_1,\ l_2,\ l_3]$, the camera's focal length and the location of the plane at infinity. The initialization is then optimized using a nonlinear gradient descent of the cost function outlined in Algorithm 7.1, using the proposed hybrid fitness measure (7.4). The quality of the results is somewhat dependant on the sequence. While the results shown in Figure 8.9 are good enough for visualization, there is still projective distortion visible — the reconstruction is not fully metric. The distortion is difficult to remove due to the relatively small motion of the video camera. The small baseline allows the scene to "stretch" away from the cameras without being strongly penalized by the autocalibration algorithm, which only enforces that each camera be plausibly physical with near-zero skew and a unit aspect ratio. The cameras in Figure 8.9 should all have the same intrinsic parameters, but instead they differ from one another while each still remains plausible. The autocalibration algorithm intentionally does not enforce these additional constraints in order to allow the reconstruction of videos that include auto-focusing cameras or zoom lenses. For this sequence, however, no focus or zooming occurred and including the additional constraints that all cameras have common intrinsics would help prevent the "stretching" phenomenon.

This 3D reconstruction pipeline is capable of producing acceptable results for short underwater video sequences, as shown in Figures 8.9 and 8.10. The current state of the pipeline is a functional proof-of-concept, and several obvious improvements have been documented in the literature and can be applied to enhance the pipeline's accuracy and robustness. These improvements will allow the handling of longer and more challenging underwater video sequences.

Chapter 10

Future Work

Several improvements have

10.1 Projective Reconstruction Improvements

The root of the projective solver's instability, described in chapter 9, is the variability of the MSAC results. Three improvements have been documented in the literature to deal with this:

1. **Implement a guided matching step [3]** — Currently, the projective reconstruction algorithm can reject image features from the tracker as outliers, but cannot add new ones. After the MSAC solution, however, the three-dimensional constraints of $\mathcal{T}$ provide a more powerful matching criteria than the tracker's two-dimensional assumptions. By detecting and including as many image correspondences as possible that fit the triplet's estimated geometry, a much denser point cloud can be obtained while simultaneously making optimization much more robust.

2. **Implement a constrained bundle adjustment for $\mathcal{T}$[30]** — The bundle adjustment procedure described in Chapter 4.1.4 adjusts all parameters of all camera matrices. This is desirable when bundle adjusting a large reconstruction, but in a single triplet the geometric constraints imposed by $\mathcal{T}$ mean we can — and should[30] — adjust only 18 parameters, corresponding to the degrees of freedom in $\mathcal{T}$, instead of the 36 parameters of all three camera matrices. This improvement will make bundle adjustment of individual triplets both faster and more accurate.

3. **Integrate guided matching with constrained bundle adjustment [3]** — By running the two steps described above iteratively until convergence, their benefits can be maximized. First, all image features consistent with the MSAC solution for $\mathcal{T}$ are added by guided matching. Then, the estimate for $\mathcal{T}$ is refined using all points, and any outliers are rejected. Subsequently, more points are added through guided matching using the new and improved $\mathcal{T}$, and this process is repeated until the total number of points stabilizes.

Taken together, these three improvements can ensure a deterministic outcome when estimating $\mathcal{T}$ for a given image triplet [3], and should solve the sporadic failures the current implementation produces.

10.2 Autocalibration Improvements

The best way to remove any remaining projective distortion after autocalibration is by post-processing the reconstruction with a metric bundle adjustment[34]. Unlike the general projective scheme described in Chapter 4.1.4, metric bundle adjustment parameterizes cameras not by the 12 elements of their camera matrix, but by the intrinsic and extrinsic parameters embedded in that matrix according to (3.5). Any subset of the cameras can be flagged as having common intrinsics, and bundle adjustment will enforce this during optimization. Post-processing the results of this 3D reconstruction using metric bundle adjustment may not completely remove the distortion from difficult sequences, but it is a well-tested technique that can only help.

References

[1] Douglas Westwood Inc. Douglas westwood predicts annual remotely operated underwater vehicle (rov) operations spend to top $3.2 billion by 2014, August 2011. http://www.dw-1.com/files/files/484-09-09%20ROV%202010%20news%20release%20FINAL.pdf.

[2] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Press, 2003.

[3] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision, Freiburg, Germany*, pages 311–326. Springer, 1998.

[4] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Automated reconstruction of 3d scenes from sequences of images. *ISPRS Journal Of Photogrammetry And Remote Sensing*, 55(4):251–267, 2000.

[5] A. Heyden, R. Berthilsson, and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17(13):981–991, 1999.

[6] F. Schaffalitzky, A. Zisserman, R. Hartley, and P. Torr. A six point solution for structure and motion. In *European Conference on Computer Vision, Dublin, Ireland*, pages 632–648, 2000.

[7] D. Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *European Conference on Computer Vision, Dublin, Ireland*, pages 649–663. Springer, 2000.

[8] K. Mierle. Open source three-dimensional reconstruction from video. Master's thesis, University of Toronto, 2010.

[9] R. Gherardi and A. Fusiello. Practical autocalibration. In *European Conference on Computer Vision, Crete, Greece*, pages 790–801, 2010.

[10] M. Chandraker, S. Agarwal, F. Kahl, D. Nistr, and D. Kriegman. Autocalibration via rank-constrained estimation of the absolute quadric. In *Conference on Computer Vision and Pattern Recognition, Minneapolis, MN*, pages 1–8, 2007.

[11] B.D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence, Vancouver, BC*, volume 3, pages 674–679. Citeseer, 1981.

[12] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, Seattle, WA*, pages 593–600. IEEE, 1993.

[13] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference, Manchester, UK*, volume 15, page 50, 1988.

[14] G.R. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[15] J. Bouget. Pyramid implementation of the lucas kanade feature tracker. Technical report, included in the OpenCV library, http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.3/OpenCV-2.3.0.tar.bz2, 2002.

[16] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[17] P. Torr and A. Zisserman. Feature based methods for structure and motion estimation. *Vision Algorithms: Theory and Practice*, pages 278–294, 2000.

[18] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[19] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[20] R. Hartley and F. Schaffalitzky. Powerfactorization: 3d reconstruction with missing or uncertain data. In *Australia-Japan Advanced Workshop on Computer Vision, Adelaide, Australia*, volume 74, pages 76–85. Citeseer, 2003.

[21] B. Triggs. Factorization methods for projective structure and motion. In *Conference on Computer Vision and Pattern Recognition, San Francisco, CA*, page 845, 1996.

[22] YS Hung and WK Tang. Projective reconstruction from multiple views with minimization of 2d reprojection error. *International Journal of Computer Vision*, 66(3):305–317, 2006.

[23] Arne Nordmann. Epipolar_geometry.svg, July 2011. http://en.wikipedia.org/wiki/File:Epipolar_geometry.svg.

[24] 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(3):207–232, 2004.

[25] Q.T. Luong and O.D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75, 1996.

[26] P.H.S. Torr and D.W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271–300, 1997.

[27] 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(1):27–44, 1999.

[28] R.I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[29] P. Beardsley, P. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *European Conference on Computer Vision, Stockholm, Sweden*, pages 683–695. Springer, 1996.

[30] P.H.S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.

[31] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, 1981.

[32] L. Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):34–46, 2002.

[33] R.I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[34] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: a modern synthesis. In *International Workshop on Vision Algorithms: Theory and Practice*, pages 153–177. Springer, 2000.

[35] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[36] Oxford. Visual geometry group multiview datasets, July 2011. http://www.robots.ox.ac.uk/ vgg/data/data-mview.html.

[37] D. Nistér. Untwisting a projective reconstruction. *International Journal of Computer Vision*, 60(2):165–183, 2004.

[38] M. Pollefeys, R. Koch, and L. V Gool. Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[39] B. Triggs. Autocalibration and the absolute quadric. In *Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, page 609, 1997.

[40] O. Faugeras, Q. Luong, and S. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 321–334. Springer, 1992.

[41] R. Hartley. Euclidean reconstruction from uncalibrated views. *Applications of invariance in computer vision*, pages 235–256, 1994.

[42] A. Heyden and K. Astrom. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, page 438. Published by the IEEE Computer Society, 1997.

[43] R. I. Hartley. Chirality. *International Journal of Computer Vision*, 26(1):41–62, 1998.

[44] M. Pollefeys, F. Verbiest, and L. Van Gool. Surviving dominant planes in uncalibrated structure and motion recovery. In *European Conference on Computer Vision, Copenhagen, Denmark*, pages 613–614. Springer, 2002.