Articulated Structure From Motion

Carl Scheffler

November 2004

Supervisor: Prof. C.W. Omlin

Acknowledgements

My sincerest thanks go out to the following people—some were involved in starting me off on this road, others in keeping me going and others still in pushing me to the destination.

To my supervisor, Professor Christian Omlin, who has seemingly boundless enthusiasm, gave me enough rope and always had faith—even when mine was waning.

To Konrad Scheffler, who acted as advisor and sounding board and whose ability is surpassed only by his willingness.

Professor Ben Herbst, who provided the support and enthusiasm to take my interest in the field beyond its formative years.

To my friends around Cape Town and Stellenbosch who supported me during the final months.

And finally, to the unknown person who forgot his printout—now this chapter is also completed.

Contents

Chapter 1

Introduction

The structure from motion (SfM) problem is that of determining 3-dimensional (3D) information of a scene from sequences of 2-dimensional (2D) images [59]. This information consists of object shape and motion and relative camera motion. In general, objects may undergo complex non-rigid motion and may be occluded by other objects or themselves. These aspects make the general SfM problem under-constrained and the solution subject to missing or incomplete data.

Humans learn to solve the problem early in life. When we move, we know the motion of our eyes (cameras) relative to the world and can discern the shape and motion of objects around us.

Figure 1.1 shows 4 frames of an artificial image sequence. To the human eye, any one of the four images does not provide enough data to determine much 3D information (depth, in this case). However, when viewing a sequence of images, the result is a vivid impression of shape and motion in all three dimensions.

1.1 Applications of SfM

Structure from motion finds application in [30, 31, 55]

- robot navigation;
- medical imaging, e.g. tomography;
- human motion capture for
  - human computer interfaces;

- analysing and adjusting athletic performance;
- diagnosing medical conditions.

Each application brings with it its own set of assumptions and considerations. Robot navigation normally takes place within a static environment. The 3D structure to be reconstructed is that of a rigid scene. In other words, the observed motion is caused by camera translation and rotation only. The same is true for medical imaging but here, we require the reconstruction of 3D volume elements. This is different from the previous example, which entails finding the depths of points on surfaces in 3 dimensions. Human motion reconstruction poses the problem that observed 2D motion is not caused by camera movement only. It could also originate from movement of the observed subject. We need to model the various types of possible motion to allow successful reconstruction.

These applications and the particular problems that they pose give rise to a natural taxonomy of structure from motion problems and their solutions.

1.2 Taxonomy of SfM

In this computer vision problem, we are actually trying to do better than the human visual system. Human beings have two eyes and thus stereo vision; stereo vision allows fast and relatively simple reconstruction of 3D data from *two simultaneous* 2D images. Here, however, we are trying to reconstruct the same 3D information by using only *one* image. To realise that this is feasible, one needs merely to walk around with one eye open and note that it is still possible to perceive depth.

Apart from assuming one camera only, we make two more assumptions. The first is on the type of camera used and the second on the type of motion that is being observed. These are discussed in detail in Chapter 3. For now, we note that the human eye and cameras are well approximated by the pinhole perspective camera model. In this model, objects that are farther away appear smaller and the projections of lines parallel to the camera view axis appear to converge at a point on the image plane.

The type of motion can often be viewed as rigid, elastic or articulated [1]. Other motion models do exist—most notably, fluid motion—but do not concern us as far as human motion capture is concerned. Rigid motion includes simple objects as well as static scenes. Elastic objects may deform in any continuous manner and the elastic model describes the motion of the human lips, face and thorax well. Articulated motion describes the movement of rigid limbs connected by joints and describes the motion of human arms, legs and hands well.

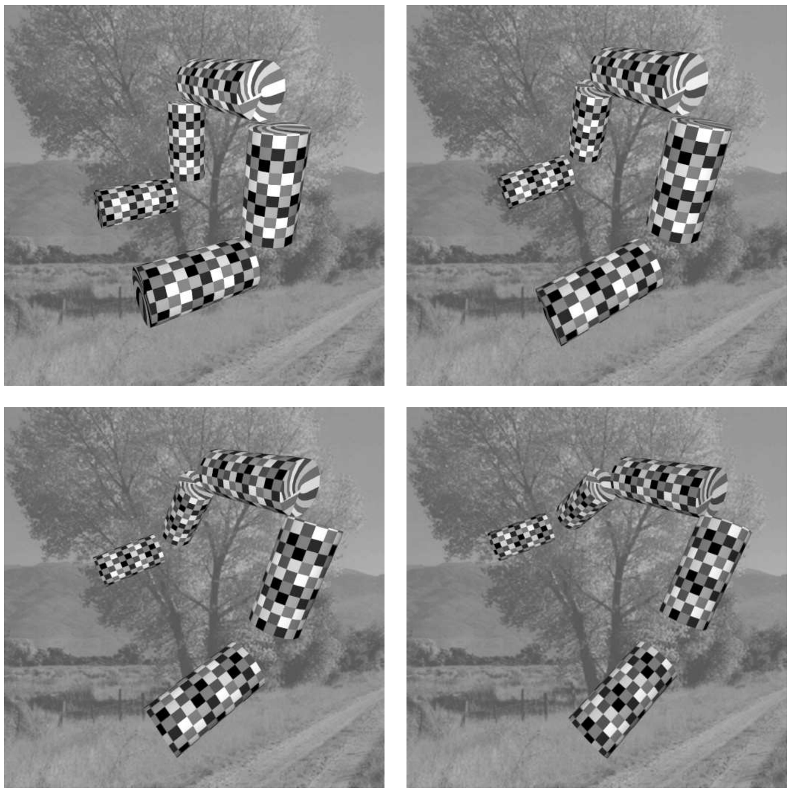


Figure 1.1: Frames 1, 21, 41 and 61 of an artificial image sequence

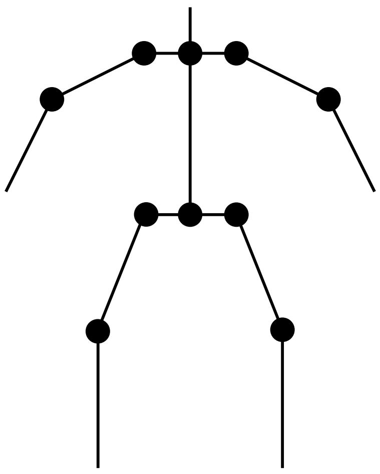


Figure 1.2: A simple model of the human skeleton

It is with the articulated subpart of the structure from motion problem that we concern ourselves in this thesis. The application of recording Sign Language[1] is mentioned in the title of this thesis and relies on the modelling of articulated motion. Articulated motion is piecewise rigid motion [1] constrained by the presence of joints connecting the rigid parts.

1.3 Objectives

It is in implementing on a computer this process of forming some internal 3D representation of objects based solely on a monocular sequence of 2D images that we are interested. Broadly speaking, we want to improve on existing motion capture systems. From this arise two questions—

- What are the existing motion capture systems?
- What improvements can be made to them?

Chapter 3 answers the first question and Chapter 4 attempts an answer to the second. We present a brief overview of the first answer here.

1.3.1 Parts of the Problem

The whole process consists of 2 parts, viz. tracking and reconstruction; we treat these parts as independent problems. There is work on integrating tracking and reconstruction for more accurate and robust solutions [62].

We want to convert raw pixel video data to a vector representation of points in 3D space. This goal is achieved by first extracting time-variant 2D vector data from the pixel data and then reconstructing the 3D vector data.

It is also possible and in some applications desirable to do some post-processing on the reconstructed model. This can include solid object reconstruction [21, 40] and object texturing. We will not discuss these topics in this study.

[1] *Sign Language* is capitalised throughout this text, as it is the proper name of a language.

Feature Tracking

The first part of the problem is known as *feature tracking* [75, 53, 6, 7]. This is essentially the problem of inter-frame similarity matching of small windows around points[2] in the video sequence. Input noise and occlusion place a limit on the accuracy that can be obtained. Noise in the video recording makes the tracking of features error prone and occlusion of image areas makes tracking of points in those areas impossible.

Occlusion is the disappearance of features due to one object obscuring another or itself from the view of the camera. When an object obscures parts of itself, we refer to it as self-occlusion. Ideally, a tracker should be able to detect occlusion and disocclusion and continue following occluded features once they reappear. This is not feasible since the tracker acts on low-level image data only. In a system where a feature tracker and vertex reconstructor is integrated, it could be possible to recover occluded features since the 3D locations of vertices are known to within some error bound [61, 62].

Vertex Reconstruction

The second step is to recover 3D information of the vertices moving in 2D. With vertex reconstruction, we use the change in the 2D vertex coordinates to make conclusions regarding the underlying 3D structure that is causing the motion.

In general this is a difficult problem. If nothing is known about the objects that caused the observed 2D motion, the number of degrees of freedom in the motion is large. In the worst case, each vertex can move independently in three dimensions, to give $3N$ degrees of freedom.

In order to make a solution feasible, limiting assumptions are made. There are assumptions on the type of camera used to record the motion and on the type of motion that is occurring. The 3D to 2D projection performed by the camera is approximated using one of the orthographic, perspective, weak perspective or paraperspective camera models. The motion is classified as either rigid, elastic, articulated or some combination of these.

Under different assumptions, different solutions to the SfM problem have been developed. Early work by Ullman [63] used an iterative approach to solve under assumptions of orthography and near-rigid motion. The work of Tomasi and Kanade [57, 58] introduced the matrix factorisation approach to solving the structure from motion problem under assump-

[2]There are other features that can be tracked. These include lines, curves [28, 29, 35] and surface patches [51, 52].

tions of orthography and rigidity. This approach was later extended to handle multiple independently moving rigid objects [15, 16] and to use the more general weak perspective [11, 32] and paraperspective [46] camera models.

The reason why matrix factorisation could be used so effectively is that the system of equations that determine the coordinates of projected 2D coordinates is linear. In the case of the perspective camera model, the projection equations are non-linear and the matrix formulation does not apply. Azarbayejani and Pentland [3] present a method for performing reconstruction in the perspective case (but still under the assumption of rigidity). They use the extended Kalman filter (EKF), a well-known technique for recursively recovering the state of a system with known state-equations that generate measured outputs [72].

The algorithm presented in [3] goes so far as to present a unifying model of the perspective and orthographic cases; the orthographic case being a special, limiting case of the general perspective case. The algorithm can also recover the camera focal length, a parameter of the perspective model that is dependent on the physical recording equipment used.

Existing work on reconstructing non-rigid objects under the perspective camera model is not based on the idea of factoring matrices or the Kalman filter for the linear and non-linear cases. Here, a very general non-rigid template model is defined and the shape of the template may be changed by adjusting certain parameters [13, 74]. Next, the shape of the template is adjusted recursively to match that of the object under consideration by varying the template parameters through solving some minimisation problem on the error in the object representation.

Model-based Versus Model-free

The intended application of this study is human motion capture in the form of recording Sign Language. Much of the existing work on human motion capture assume a known *a priori* model of human motion and treats the motion capture problem as a model parameter fitting problem [9, 10, 17, 18, 48].

This approach constrains the number of degrees of freedom to be discovered considerably. For example, a basic model of the human skeleton (see Figure 1.2, adapted from [17]) has 12 segments or limbs and 10 joints for a maximum of 39 degrees of freedom—3 per segment for rotation and 3 more for global translation of the skeleton. In reality there are even fewer degrees of freedom since not all limbs can undergo unconstrained rotation around their joints. Contrast this with 3 degrees of freedom for *each* tracked point on the video sequence when general motion is allowed.

Using an *a priori* kinematic model has 2 disadvantages—

- the physical dimensions—especially lengths—of the limbs are not known, these parameters typically have to be measured and adjusted for each motion capture subject;
- if the structure from motion problem is to be solved for any articulated object other than a human, a new kinematic model must be created by hand before reconstruction can be performed.

The latter consideration is especially limiting as the 3D kinematic model is often part of what we want to find in the first place.

1.3.2 Research Question

From the discussion above, we phrase our research question as follows: *Is accurate model-free articulated motion capture possible?*

We make no assumptions on what caused the observable motion, other than that it is articulated. Still, we will apply the methods developed to human motion capture—in particular to the capturing of Sign Language motion.

1.4 Thesis Layout

Chapter 2 provides some background information on human motion capture. This places into context the application for which this work is intended—Sign Language vocabulary acquisition. Chapter 3 reviews the current literature on the structure from motion problem and introduces the notation used in the rest of the thesis. Chapter 4 presents theoretical and implementational advances made in solving the articulated structure from motion problem. Chapter 5 presents experiments in the reconstruction of artificial and real data and discusses the results. Chapter 6 concludes the thesis and suggests areas of study for future work.

Chapter 2

Human Motion Capture

The recording and analysis of human motion has been of interest for many years before the advent of digital video equipment and computers. A noteworthy example is the work of Leonardo da Vinci (1452–1519), which records the dynamics of human motion with painstaking accuracy in the form of technical sketches and drawings [36]. With photographic equipment came what is regarded as the first movie of human movement. In 1884, Eadweard Muybridge used a battery of still cameras to record people walking under various conditions, e.g. carrying a boulder [9, 10]. With the introduction of computers of sufficient computational power, the automatic and real-time analysis of recorded human motion became a reality. Now, such recordings are used to construct virtual people in movies, computer games and interactive environments, to diagnose sports injuries and certain diseases, and to record and interpret human gestures and Sign Language.

2.1 Definition

A broad definition of motion capture is the recording and analysis of motion. To make this more specific to the subject of interest here, viz. humans, and the equipment commonly used, we consider human motion capture to be the recording of human motion using a digital video camera and subsequent analysis of the motion on a digital computer.

2.2 Applications

Human motion capture finds many applications in industries ranging from medicine and sport to entertainment. The generalised, abnormal or characteristic movement of a person or persons is recorded and, where relevant, used to make diagnoses or recommendations.

2.2.1 Dynamics of Human Motion

The nature of competitive sports has led to the need for various sophisticated means of motion analysis. Early systems based on video simply recorded an athlete in motion; analysis was then performed by people—including the athletes and trainers—by using video playback facilities.

3D motion capture allows the viewing of sports footage from any angle [68] as well as the overlaying of an individual's motion over model motion for the sport under consideration. Thus, individual performance may be enhanced by making adjustments to movement through training. A trained coach with full understanding of human motion for the sport is still needed, but he is furnished with more accurate information upon which to base his recommendations.

In a similar fashion, motion capture is also used to diagnose and correct injury. This takes two forms: preventative training and recovery therapy. By analysing the walking motion (or gait) of an athlete, under or over-developed muscle groups can be identified and retrained before injury occurs. This provides the advantages of improving an athlete's performance while lowering the risk of injury.

Motion analysis of recovering patients brings us to the field of physiotherapy rather than sport. Gait analysis is used to locate injured muscle groups in sports injury cases but is also used to diagnose cerebral palsy and osteoarthrosis [67]. Motion capture is used to diagnose patient condition, to prescribe treatment and to track patient recovery in cases of injury or illness.

2.2.2 Human Computer Interface

In the field of human computer interfaces (HCI), the transfer of information from the computer to the user has progressed by leaps and bounds. High-quality surround sound audio and rendered 3D visuals through VR goggles provide an immersive environment.

The transfer of information from the user to the computer needs to progress equally fast for properly user-oriented systems. Real-time motion capture allows a user to manipulate objects in a virtual environment directly rather than through a traditional keyboard or pointer-based input device. Existing systems can already perform gaze tracking and data-glove hand tracking to provide a more intuitive and immersive environment [48, 49].

Training, design and design testing often require people to interact with devices that require great cost to produce and maintain. In many cases these devices can be simulated in a virtual environment—with reproduction of the virtual devices at little or zero marginal cost. In the case of design testing, a design can even be changed in real-time as users interact with a prototype and design flaws are discovered.

Capturing Sign Language—one of the concerns of this project—also opens doors for non-traditional interfaces with deaf users. Such a system may integrate manipulating gestures with semantic gestures. Here, the recognition of semantic gestures will take the place of voice commands in speech recognition systems.

2.2.3 Computer Graphics

In the making of film and television media, applications of motion capture abound. Amongst these applications are

- the introduction of virtual or animated actors;
- human-like motion for characters in animated films;
- simulated stunts where there is too great a risk to stunt artists; and
- crowd scenes created by duplicating the motion of a small number of actors.

Some of these applications are simply cost and effort saving techniques while others make viable the previously impossible.

Also in the entertainment industry, computer game creators have employed many of the techniques used in film and other industries to improve their products. In sports simulation games, the technology used to improve the performance of athletes is used to generate the realistic motion of virtual athletes. 3D motion capture of anything from a golf or tennis swing to the throw of a fishing line has been used to create interactive environments for desktop athletes. In first person action games, the visual rendering of characters in games has grown in detail and complexity to the point where real-time facial animation is visible—captured facial motion is used to create realistic animations.

2.2.4 Sign Language Understanding

The motion capture applications mentioned in Section 2.2 focus mostly on what Moeslund [43] terms *body analysis*. That is, studying human body movement for the purpose of HCI, computer graphics, etc. Two other subclasses of human motion analysis are *face analysis* and *gesture analysis*. These are more interesting to us for application in creating automated Sign Language understanding and translation systems. The grammar of Sign Language is communicated in facial expressions, the shape and motion of the hands and arms, and the orientation of the torso.

In [1], motion is separated into a number of classes, each representing a type of motion. Three of the most important classes are rigid, elastic and articulated motion. Two of these, namely elastic and articulated motion, find application in Sign Language motion capture.

For facial analysis, elastic motion capture provides an accurate model. The deformation of the mouth, cheeks and eyebrows are all elastic and the work of [11, 62] has shown that facial expressions can be recovered from low-quality video footage.

Gesture analysis is best modelled as articulated motion, i.e. the movement of rigid limbs around joints connecting the limbs. The movements of arms, hands and fingers fit this description.

2.3 State of the Art

Moeslund [43, 42] presents an overview and taxonomy of the methods of motion capture described in the literature. The taxonomy is based on the elements of

- initialisation;
- tracking;
- pose estimation; and
- recognition.

The first three of these are of importance to us in solving the structure from motion problem and the last deals with analysing the meaning behind a particular motion or sequence of motions.

The initialisation category differentiates methods based on how much of the process is automatic and how much requires manual user input. Initialisation is that which needs to be done by the user before the computer software takes over. Many methods [9, 10, 17, 18, 48, 49] require a kinematic model of human motion to be defined manually. The kinematic model also needs to be customised to the subject under consideration since parameters such as limb lengths vary. The system of Bregler and Malik [9, 10] requires the user to click on joint locations in the first frame of video. From here, the initial parameters (pose angles and limb dimensions) are estimated. DiFranco *et al.* [17] require the specification of joint locations in a number of key frames.

Some kinematic models are more restrictive and take constraints on joint rotation into account [17]. Not all limbs of the human body can undergo free 3-dimensional rotation. For example, the femur and tibia can undergo rotation in one plane only around the knee joint. By encoding such information into the kinematic model—and thus constraining it—more robust motion recovery can be performed.

Methods using kinematic models can estimate motion from low-quality video data and with significant occlusion [9, 10, 17, 48, 49]. The robustness and accuracy of these methods stem from the large number of constraints placed on articulated motion by the specific articulated model. The major disadvantage of these models is that, in practice, they have to be initialised manually. There is work to either estimate the initial pose of a kinematic model automatically or generate a model from scratch. These methods require restrictive assumptions such as

- the availability of multiple camera views [18, 31];
- that motion is in the plane perpendicular to the camera view axis only [37, 38, 39].

Another aspect of initialisation is whether the camera or cameras need to be calibrated. Calibration is the procedure by which the parameters of the cameras—e.g., relative positions and orientations, focal lengths and lens characteristics—are determined. In [18, 48, 49] calibration is performed as part of the initialisation of the system. The major disadvantage of a setup that requires calibration is that it must be reinitialised whenever the parameters of the cameras change. In practice, this precludes the possibility of moving cameras.

The tracking category covers the different methods and assumptions under which image features are located and tracked. An image feature is a pixel or group of pixels that corresponds to some point or region of interest on the image being observed. In the literature the movement has been towards markerless motion capture. A marker is typically a white circle or sphere placed on a dark suit worn by the motion capture subject. The high contrast between the markers and the suit allows better tracking. None of the papers referenced in this section rely on markers; in contrast, most commercial motion capture systems still use markers or special gloves or suits. Markers have the advantages that

- they can be placed specifically on important points on the body, e.g. joints, eyebrows and lips;

- they allow fast and very high-accuracy tracking.

In contrast, markerless tracking is still more susceptible to noise and the occlusion of tracked features is difficult to handle, especially in the general case—when no information about the physical cause of the observed motion is known.

Markerless motion capture does, however, have applications that are not open to the use of markers:

- surveillance;
- low-cost capturing in out-of-the-way places where unwieldy equipment is not feasible;
- unencumbered HCI [48, 49].

The focus in this section has, so far, been on reconstructing articulated motion. Another aspect is the reconstruction of elastic motion, such as that of the face and the torso. Bregler *et al.* [11] use a matrix factorisation technique to perform reconstruction in these cases—*basis shapes* and *deformation parameters* that describe the motion are recovered. This will be described in more detail in Chapter 3, along with a discussion on the reconstruction techniques used under other assumptions.

2.4 Motion Capture and Structure From Motion

From the definitions of human motion capture and structure from motion, we see that the first is a sub-part of the second. We are attempting to find a description in 3 dimensions of the motion of a human subject and do so by using structure from motion methods tailored to human-type (elastic and articulated) motion.

Because we know the type of motion being reconstructed and can often control the environment in which motion capture is done, we can make assumptions that allow more accurate and robust solutions to the structure from motion problem.

Despite this, there are reasons for discarding these assumptions that have proven so useful. More general motion capture solutions can build kinematic, elastic and other models rather than requiring hand-crafted ones; and estimate and refine model parameters rather than requiring them as input.

2.5 Summary

Human motion capture is a field with many industry applications. We have discussed a number of these applications and the role that developments in computer vision played in making them feasible. It was shown that the state of the art still requires manual initialisation or tuning of system parameters to achieve good measurements. The use of model-based methods—where the models are based on human kinematics—are prevalent and requires the careful construction of new models for new types of objects or motion.

In the rest of this thesis, we explore and develop model-free solutions to motion capture. Such solutions would automate much of the manual work currently required and constitute a step towards computer vision systems for generic motion and structure.

Chapter 3

Structure From Motion

Here, we present an overview of the theory and methods used for structure from motion analysis found in the literature. The methods are categorised based on the type of structure that is recovered and on the mathematical techniques used in the reconstruction process.

Section 3.1 introduces the notation used in this work and some basic theory on coordinate systems and camera models. Section 3.2 provides an overview of the structure models of 3-dimensional objects found in the literature. Section 3.3 discusses the reconstruction techniques used to solve the structure from motion problem under various assumptions.

3.1 Background

When objects are observed by a camera, there are a number of coordinate systems to take into account. There is the *world coordinate system* in which all objects reside. This system has some arbitrary origin and orientation of its axes. In practice, we are interested in relative distances and orientations only and do not refer explicitly to the world coordinate system.

Each object has an *object coordinate system.* Here, the coordinate origin and the orientation of the axes are defined relative to the object and in some intuitive manner. For example, the origin is at the centre of mass of the object, the y-axis points up, the x-axis left and the z-axis forward. This gives rise to a right-handed coordinate system.

Finally, there is the *camera coordinate system.* The camera is assumed to be located at a point in 3D space and we define the direction in which the camera is facing as the z-axis, up as the y-axis and left as the x-axis.

We relate one coordinate system to another with a linear transformation that describes the translation of the origin and rotation of the axes. Such a transformation is used to determine the positions of objects—with known coordinate systems—in the camera coordinate system. This is discussed in further detail in Section 3.1.2 below.

The purpose of the camera is to project objects in 3D space onto a 2D plane known as the *image plane*. The information on the image plane is what is seen by the viewer of the camera output. The camera records this projected image at regular discrete time intervals. The images thus created are known as a sequence of *frames*. The information recorded on the frames are 2D projections of points in 3D space.

The 3D location of a point has the form

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.1}$$

This is a standard notation when manipulating points in three dimensions. The location of the projection of the point on the image plane is written

$$\begin{bmatrix} u \\ v \end{bmatrix}. \tag{3.2}$$

The image plane is parallel to the xy-plane of the camera, the u- and v-axes are parallel to the x- and y-axes, respectively, and the z-axis intersects the image plane at its origin.[1]

3.1.1 Camera Models

Here, the term *camera model* refers to the way in which points in the camera coordinate system are projected onto the image plane. The *perspective* camera model is the most intuitive since it conforms to human vision. In this model, straight lines converge in the distance from the observer and objects appear smaller as they become more distant.

The mathematical model for the perspective camera is [3]

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.3}$$

where $[x_c\ y_c\ z_c]^T$ is a point in the camera coordinate system, f is the focal length of the camera and $[u\ v]^T$ is the projection onto the image plane. In this formulation, the origin

[1] There are camera models that allow deviations from this. E.g., the image plane can be at an angle to the xy-plane. For a discussion of these variations see [71].

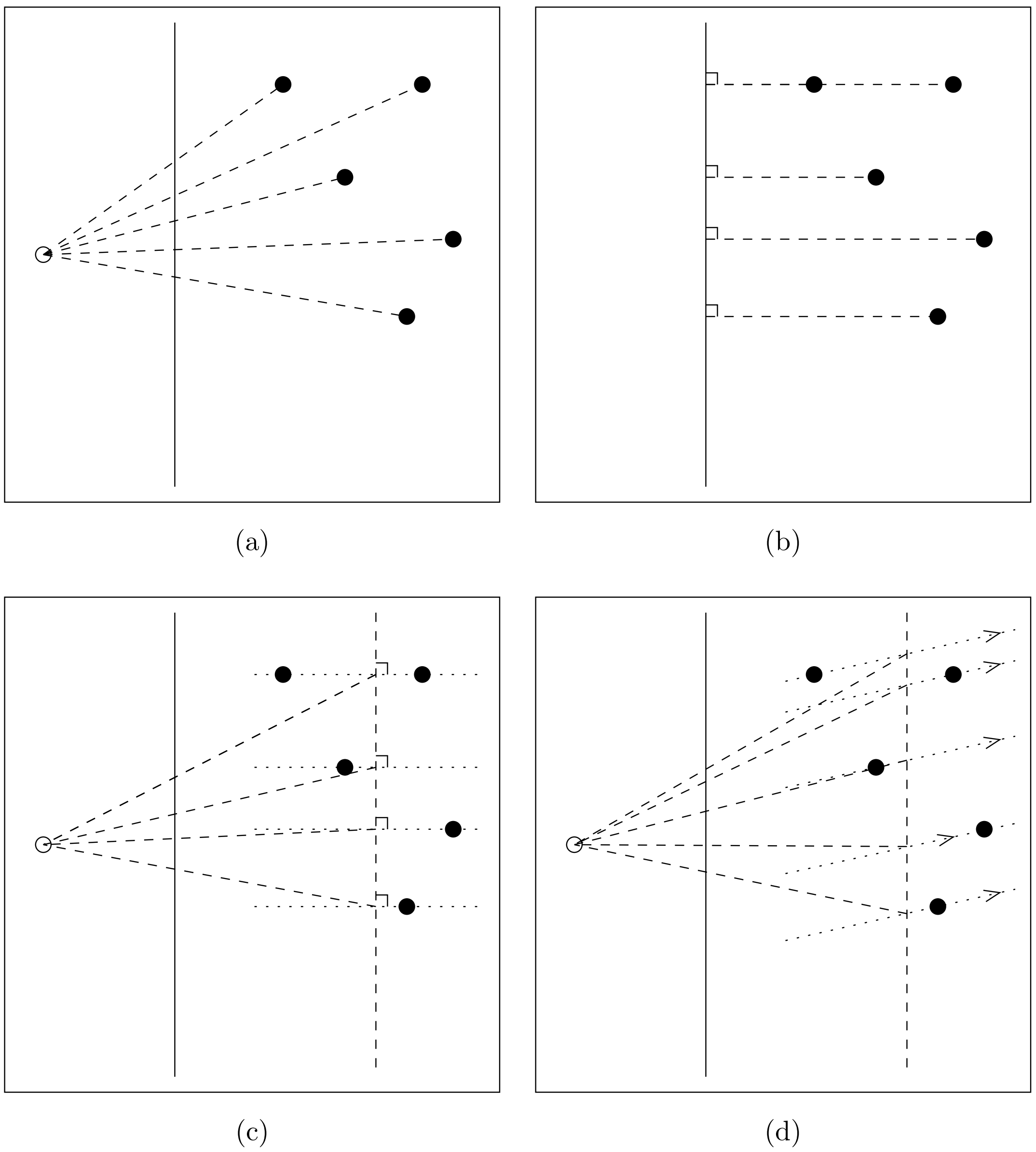


Figure 3.1: The four camera models: (a) perspective, (b) orthographic, (c) weak perspective, and (d) paraperspective. The solid line in each figure represents the image plane and the intersections of the dotted projection lines with the solid line represent the image coordinates of the observed points.

of the camera coordinate system is at the centre of projection (or location of the camera lens) and the distance from the centre of projection (COP) to the image plane is f. The coordinates $[u\ v]^T$ are simply the intersection of the line connecting the COP and the point coordinate $[x_c\ y_c\ z_c]^T$ with the image plane at $z = f$. See Figure 3.1a.

Even though the perspective camera model is the most intuitive, it is non-linear. The non-linearity is caused by the division by z_c in (3.3). A class of camera models know as the *affine models* attempts to approximate the perspective model as closely as possible while maintaining linearity. This is desirable since it allows the formulation and solution of camera projection problems using linear methods.

The orthographic camera model [15, 32, 58] is the simplest of the affine models. In the orthographic model, points in the camera coordinate system are projected onto the image plane using lines parallel to the plane rather than lines converging at the COP (Figure 3.1b). Here, the projection equations are

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix}. \tag{3.4}$$

Because the lines of projection are parallel to the image plane, the z-component of any point is simply discarded. The orthographic model is accurate only if the objects in the view of the camera have little depth, i.e. the difference between the smallest and largest z_c-values is small. Also, translation along the z-axis of either an object or the camera is not observable in the projected image.

The *weak perspective* (also known as *scaled orthographic*) camera model [11, 32] addresses some of the weaknesses of the orthographic model. Here, the projection equations are

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{\overline{z}_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix}, \tag{3.5}$$

where $\overline{z}_c$ is the average z-coordinate of all points being projected onto the image plane. This formulation is equivalent to

1. performing an orthographic projection onto the plane through the centroid of the projected object and parallel to the image plane; and then

2. performing a perspective projection of the resulting coordinates onto the image plane (Figure 3.1c).

The weak perspective camera does still not model the depth difference between points belonging to the projected object but does model motion of the entire object (or the camera) along the z-axis.

The *paraperspective* camera model [32, 46] is related to the weak perspective model and can also represent the scaling effect of objects moving along the z-axis. Furthermore, the model can represent the rotational effect when an object is displaced from the centre of the image plane. With a perspective camera, objects further from the centre of the image plane are viewed at a different angle from the COP than those at the centre.

As with the weak perspective projection, the 3D points are first projected onto a plane parallel to the image plane and through the centroid of the object and then projected onto the image plane. Here, however, the projection onto the first plane is not orthographic but along lines parallel to the line connecting the COP and the object centroid (see Figure 3.1d). Thus, the rotational effect of an object far from the centre of the image plane is approximated.

In [3], the orthographic and perspective camera models are unified. This is accomplished by making two representational changes to (3.3):

- the camera coordinate system has its origin on the image plane rather than at the COP; and
- a new parameter, the *inverse focal length* β, is defined as $\beta = 1/f$.

The perspective projection can then be expressed as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{1+\beta z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix}. \tag{3.6}$$

This formulation effectively represents both the perspective and orthographic models. The orthographic model can be interpreted as the limit case in which the focal length of the perspective camera goes to infinity. Here, using the inverse focal length, the orthographic model is obtained by substituting $\beta = 0$ in (3.6). This formulation exhibits better numerical stability than the standard perspective model, (3.3), when the focal length becomes large.

3.1.2 Structure and Motion in 3D

When points in 3D move—rotate around the origin and translate—the motion can be expressed using matrix algebra [58],

$$\mathbf{R}_f \mathbf{p}_i + \mathbf{t}_f, \tag{3.7}$$

where $\mathbf{R}_f$ is a 3×3 matrix that describes the rotation and $\mathbf{t}_f$ is a 3×1 column vector that describes the translation of point $\mathbf{p}_i$ in frame f. Like all rotation matrices, $\mathbf{R}_f$ is orthogonal—

$$\mathbf{R}_f\mathbf{R}_f^T = \mathbf{I}. \tag{3.8}$$

We can rewrite (3.7) more compactly using the *homogeneous representation* of 3D points [15]—

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}. \tag{3.9}$$

Note that $\mathbf{p}_i$ still has the same 3 degrees of freedom (DOFs) as before. But now, we can write (3.7) as

$$\mathbf{M}_f\mathbf{p}_i \tag{3.10}$$

where $\mathbf{M}_f$ is a 4×4 motion matrix, defined in terms of $\mathbf{R}_f$ and $\mathbf{t}_f$—

$$\mathbf{M}_f = \begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{3.11}$$

Note that (3.7) and (3.10) are equivalent.

The structure and motion representation above is used in most structure from motion algorithms based on matrix factorisation methods (Section 3.3.1). In other methods, the representation of structure and motion is more varied. We discuss some of these representations next.

The representation in (3.1) is popular because of its simplicity but provides too many DOFs—3 per point—for many structure from motion applications, especially those that assume object rigidity. In [3] objects are assumed to be rigid and the camera model of (3.6) is used. Here, 3D points are represented in terms of their image positions in the first frame

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ 0 \end{bmatrix} + \alpha_{fi} \begin{bmatrix} u_i\beta \\ v_i\beta \\ 1 \end{bmatrix}. \tag{3.12}$$

The inverse focal length β is constant in an image sequence and the image positions u_i and v_i are measurable. This formulation allows the representation of a 3D point in any frame with one additional parameter per frame, α_{fi}. This reduction of the number of parameters allows greater noise tolerance during reconstruction. One disadvantage is that the position of a point in *all* frames is influenced by its image coordinates in the first frame. Thus, noisy or inaccurate measurement of the features in the first frame will adversely influence all further computations.

When recovering the 3D motion of objects, the motion matrix, $\mathbf{M}_f$, in (3.10) has more parameters to estimate than there are DOFs. Particularly, in (3.11) the rotation sub-matrix, $\mathbf{R}_f$, has 9 entries but only 3 DOFs, which means that the entire motion matrix, $\mathbf{M}_f$, has 6 DOFs—3 for translation and 3 for rotation.

Two other methods of representing rotations are the *axis-and-angle* and *unit quaternion* representations. In the former—also known as the Euler angle notation—the axis

$$\mathbf{n}^T = [n_1 \; n_2 \; n_3] , \tag{3.13}$$

and the scalar angle θ determine the rotation. The axis is of unit length. The rotation consists of rotating all points through an angle of θ around the axis at the origin and pointing in the direction of $\mathbf{n}$.

This representation has intuitive appeal but has unwanted computational characteristics. Rotations are not uniquely defined—a rotation of θ around $\mathbf{n}$ is equivalent to a rotation of $-\theta$ around $-\mathbf{n}$. Also, when considering change in rotation across a number of frames, the rotational velocity is not easily defined using Euler angles.

The unit quaternion [3, 12] eliminates these problems. A quaternion is simply a length 4 column vector and, in the case of representing rotation, is related to Euler angles by

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} n_1 \sin\frac{\theta}{2} \\ n_2 \sin\frac{\theta}{2} \\ n_3 \sin\frac{\theta}{2} \\ \cos\frac{\theta}{2} \end{bmatrix} . \tag{3.14}$$

The vector $\mathbf{q}$ is of unit length.

Quaternions are related to the rotation sub-matrix of (3.11) by

$$\mathbf{R} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.15}$$

A quaternion is a compact representation of rotation. It encodes the 9 entries of a rotation matrix in 4 parameters (still with 3 degrees of freedom).

A motion matrix, containing rotation and translation, can be represented as a *twist* [9, 10]

$$\xi = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \text{or} \quad \hat{\xi} = \begin{bmatrix} 0 & -\omega_z & \omega_y & v_1 \\ \omega_z & 0 & -\omega_x & v_2 \\ -\omega_y & \omega_x & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.16}$$

Here the 6 DOFs of the motion matrix are encoded in the 6 entries of ξ. This representation is related to the motion matrix in (3.11) by the matrix exponent,

$$\mathbf{M} = e^{\hat{\xi}}. \tag{3.17}$$

This is known as the *exponential map* of the twist. As with quaternions, twists represent motion compactly and allow structure from motion methods that are more robust to noise. Using this formulation, it is also possible to incorporate velocity and rotational velocity by differentiating (3.17). This is approximated by taking the Taylor expansion of (3.17) and truncating the series.

3.2 Structure Models

Initially, the literature on structure from motion was concerned with recovering rigid structure from image sequences [12, 58, 63]. This was later extended to handling multiple independently moving rigid objects [15, 16]. Currently, there are still efforts on improving the reconstruction of rigid objects but there are growing efforts towards handling more complex non-rigid motion.

Within non-rigid motion, the reconstruction of fluid motion and articulated and elastic objects receive most attention [1]. The reconstruction of articulated and elastic motion finds application in human motion capture [9, 10, 11, 17, 18, 48, 62]. Biological motion can often be modelled as either elastic motion—e.g., a beating heart [41]—or fluid motion—e.g., growing plants [4].

We discuss each of the rigid, elastic and articulated models of motion.

3.2.1 Rigid Objects

There is evidence that rigid objects and the rigidity of non-rigid objects are important for discerning 3D information in the human visual system [63]. This supports their use as a starting point for structure from motion.

A rigid object maintains the distances between all points in or on the object [63]. This also implies that all angles in or on the object remain constant [1]. The rigidity assumption places constraints on the *relative* motion of points and their projections onto the image plane. The points all undergo the same rotation and translation under rigidity. The disadvantage of the rigidity assumption is that it limits the use of structure from motion solutions. It is very rare to have a single rigid object in practice. There is normally at least a background present in an image sequence.

The main practical application under rigidity is the analysis of an image sequence formed by a moving camera in a static scene [5, 22, 45, 46]. Here, the rigidity assumption holds since all elements in the scene are stationary and the information provided by translating and rotating the camera in the scene can be used to perform reconstruction.

3.2.2 Multiple Rigid Objects

To make the reconstruction of moving objects in a scene possible, we consider multiple rigid objects. Here, the entire image sequence is not rigid but is assumed to consist of rigid parts. The motions of the objects are assumed to be independent [15].

This model still has the same constraints as in the previous section, but applied to each rigid part of the scene. This allows more general motion and increases the complexity of solving the structure from motion problem. The additional complexity is contained mainly in finding the rigid segmentation of the scene [15, 33]—i.e. discovering which parts of the projected image belong to which objects.

3.2.3 Articulated Objects

Articulated objects are a degenerate case of the multiple rigid object motion described above. An articulated object consists of a number of limbs connected by joints. Each limb is still assumed to be rigid but the motions of the rigid limbs are now interdependent. A relevant example is the human body. Here, methods are needed to segment the interdependent motions [24, 34, 60, 65] and to recover the locations of joints in the articulated structure formulation [56].

A single articulated object is known as an *articulated tree* and is assumed to have tree-like connectivity between limbs. Each node in the tree has exactly one parent—except for one node, which has no parent and is known as the root of the tree. It is assumed that there are

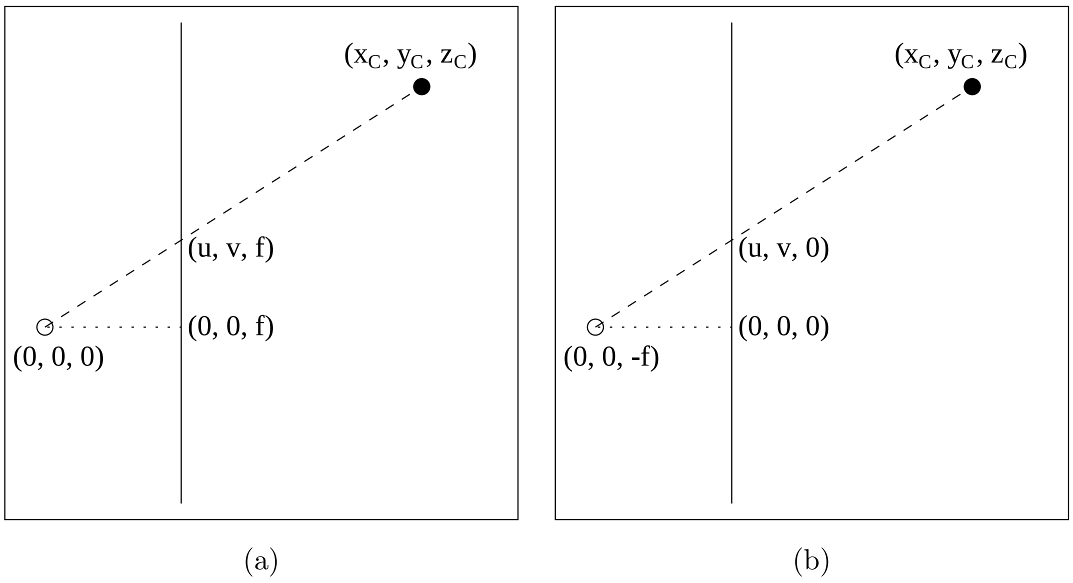


Figure 3.2: The different coordinate systems used when describing the perspective camera model with (a) focal length f and (b) inverse focal length, β, as camera parameter. A point at (x_C, y_C, z_C) in the camera coordinate system is projected onto the image place at (u, v). Note the difference in the coordinates of the COP in (a) and (b).

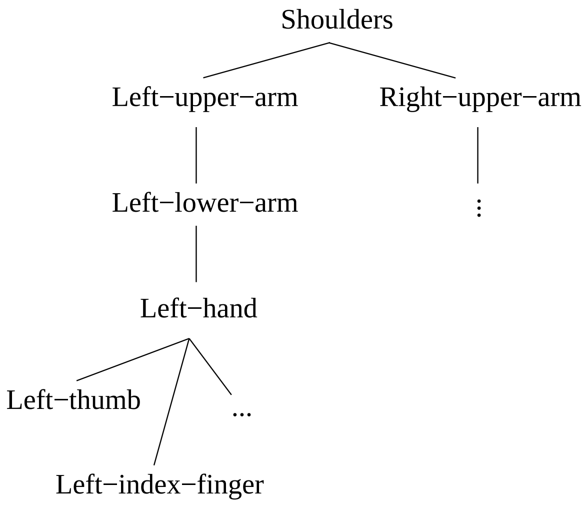


Figure 3.3: A partial representation of an articulated tree that describes the human body. The *shoulders* node forms the root of the tree.

no cycles in the graph representing the articulated connectivity. Figure 3.3 shows a partial articulated tree describing the human body.

A collection of independent articulated trees is known as an *articulated forest.*

3.2.4 Elastic Objects

Elastic objects do not necessarily contain any rigid parts. Here the only constraint is some measure of continuity [1].

In [63], the elasticity of an object is measured by the amount of non-rigidity that it displays—i.e. how much the distances between object points change. [41] describes elastic motion as a motion that is smooth spatially and temporally. An elastic object is described as consisting of a linear combination of *basis shapes* in [11, 62, 61]. The immediate structure of the object is a linear combination—weighted sum—of these basis shapes.

3.3 Reconstruction Techniques

The structure from motion problem is to recover the underlying 3D structure and motion that produced an observed 2D image sequence. We have discussed the assumptions underlying different camera models and types of objects. These models determine the constraint equations that can be used in a structure from motion algorithm.

We next classify structure from motion solutions based on the method used. We discuss batch methods, sequential methods, model-based methods and direct methods.

3.3.1 Batch Methods

Batch reconstruction methods use all frames in the image sequence simultaneously to solve the structure from motion problem. Because the number of frames may be large, such methods are not real-time but they provide greater accuracy than sequential methods (Section 3.3.2).

These methods are based on matrix factorisation as introduced in [57]. Here, it is assumed that a number of point features are tracked through an image sequence. Each feature is described by its position

$$\begin{bmatrix} u_{fi} \\ v_{fi} \end{bmatrix} \tag{3.18}$$

where we use the subscripts f and i to indicate frame f and point i, with $1 \le f \le F$ and $1 \le i \le P$. F is the number of frames in the image sequence and P the number of points being observed. These features are used to create the *observation matrix*

$$\mathbf{W} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1P} \\ v_{11} & v_{12} & \cdots & v_{1P} \\ u_{21} & u_{22} & \cdots & u_{2P} \\ v_{21} & v_{22} & \cdots & v_{2P} \\ \vdots & \vdots & & \vdots \\ u_{F1} & u_{F2} & \cdots & u_{FP} \\ v_{F1} & v_{F2} & \cdots & v_{FP} \end{bmatrix}. \tag{3.19}$$

This matrix is $2F \times P$ and will have rank at most $\min(2F, P)$. The exact rank of $\mathbf{W}$ depends on the type of structure that caused the motion under consideration (Section 3.2) and the camera model used (Section 3.1.1).

In an early formulation [58], under the assumptions of

- rigidity,
- the orthographic camera model, and
- that motion consists of rotation only,

it is shown that $\text{rank}(\mathbf{W}) = 3$. The latter assumption was later extended in [15] to include translation; in this case $\text{rank}(\mathbf{W}) = 4$.

In [26], it is shown that $\text{rank}(\mathbf{W}) \le 4$ for affine camera models (orthographic, weak perspective, paraperspective) and rigid motion. When using the perspective camera model, $\text{rank}(\mathbf{W}) \le 8$. The low rank of $\mathbf{W}$ under various assumptions is derived next.

We start by constructing a model for the observed motion, $\mathbf{W}$, under rigidity and orthography. At each frame of the image sequence, the motion of each point on the rigid object is modelled as in (3.10). This yields the 3D positions of the points in the camera coordinate system. To perform orthographic projection, we write

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.20}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.21}$$

$$\Rightarrow \begin{bmatrix} u_{fi} \\ v_{fi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{M}_f \mathbf{p}_i \tag{3.22}$$

$$= \begin{bmatrix} r_{11}^{(f)} & r_{12}^{(f)} & r_{13}^{(f)} & t_x^{(f)} \\ r_{21}^{(f)} & r_{22}^{(f)} & r_{23}^{(f)} & t_y^{(f)} \end{bmatrix} \mathbf{p}_i \tag{3.23}$$

The final equation describes the observed location of a point on the image plane in a single frame in terms of the first two rows of the motion matrix $\mathbf{M}_f$ and the 3D location of the point. Note that the orthographic projection is encoded in the motion part of the equation.

(3.23) allows us to write

$$\mathbf{W} = \mathbf{MS}, \tag{3.24}$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1' \\ \mathbf{M}_2' \\ \vdots \\ \mathbf{M}_F' \end{bmatrix} \text{ and } \mathbf{S} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_P \end{bmatrix}. \tag{3.25}$$

Here, each $\mathbf{M}_f'$ contains the first two rows of the motion matrix, $\mathbf{M}_f$. Hence, $\mathbf{M}$ is $2F \times 4$ and $\mathbf{S}$ is $4 \times P$.

This formulation shows that the observation matrix can be factored into two matrices that provide the solution to the structure from motion problem. $\mathbf{M}$ contains the 3D *motion* of the object and $\mathbf{S}$ the 3D *structure*. Note that $\mathbf{M}$ does not yield the translation component along the camera view axis, t_z. This cannot be recovered since the orthographic camera model discards information along the camera z-axis completely.

Since $\mathbf{M}$ is $2F \times 4$ and $\mathbf{S}$ is $4 \times P$, we concluded that the maximum rank of their product, $\mathbf{W}$, is 4.

The weak perspective camera model is described by (3.5), which can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} l & 0 & 0 & 0 \\ 0 & l & 0 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}, \tag{3.26}$$

where $l = f/\overline{z}_c$. Combining this with (3.10) yields $\mathbf{W} = \mathbf{MS}$, with $\mathbf{S}$ as in (3.25) and

$$\mathbf{M} = \begin{bmatrix} l_1\mathbf{M}_1' \\ l_2\mathbf{M}_2' \\ \vdots \\ l_F\mathbf{M}_F' \end{bmatrix}. \tag{3.27}$$

As in the orthographic case, rank($\mathbf{W}$) ≤ 4.

In contrast to the camera models just discussed, the perspective camera model is non-linear. This means that no matrix formulation is possible. There is, however, still a rank constraint on the observation matrix [26]—rank($\mathbf{W}$) ≤ 8. This constraint is useful to denoise the observation matrix but does not stem from a matrix formulation that describes how to perform reconstruction.

In [22], reconstruction under the true perspective camera model is performed by iteratively applying a weak perspective reconstruction technique that converges towards the perspective solution. This method can reconstruct 3D motion, structure and focal length for a single rigid 3D object.

To exploit the low-rank property of $\mathbf{W}$, the singular value decomposition (SVD) is employed. The SVD yields as output the matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ such that

$$\mathbf{A}_{m\times n} = (\mathbf{U}_{m\times r})(\mathbf{\Sigma}_{r\times r})(\mathbf{V}_{n\times r})^T, \tag{3.28}$$

where

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}, \tag{3.29}$$

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}, \tag{3.30}$$

and $\mathbf{\Sigma}$ is a diagonal matrix with the singular values of $\mathbf{A}$ arranged in descending order along its main diagonal. The subscripts indicate the dimensions of the matrices—r is the rank of matrix $\mathbf{A}$.

In general, batch reconstruction using an affine camera occurs in two stages—

1. factoring $\mathbf{W} = \hat{\mathbf{M}}\hat{\mathbf{S}}$ using the SVD; and
2. finding $\mathbf{A}$ such that $\mathbf{M} = \hat{\mathbf{M}}\mathbf{A}$ and $\mathbf{S} = \mathbf{A}^{-\mathbf{1}}\hat{\mathbf{S}}$ using the 3D motion constraints.

The first step is completed by forming $\hat{\mathbf{M}} = \mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ and $\hat{\mathbf{S}} = \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^T$. This yields the shape and motion matrices up to an affine transformation—as represented by $\mathbf{A}$ in the second step. $\mathbf{A}$ is calculated using the constraints of rotation

$$\mathbf{R}_f\mathbf{R}_f^T = \mathbf{I} \tag{3.31}$$

for each $1 \leq f \leq F$. The detailed method for this calculation is relegated to Appendix A.

The descriptions of reconstruction thus far have all been based on reconstruction of a single rigid object under different camera models. Next, this is extended to handle other structure assumptions.

In [15, 16] multiple rigid objects are discussed. It is shown that the rank of $\mathbf{W}$ is $4N$, where N is the number of objects being observed. The motion equations have the form $\mathbf{W} = \mathbf{MS}$ with the following block structure

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^{(1)} & \mathbf{M}^{(2)} & \cdots & \mathbf{M}^{(N)} \end{bmatrix} \text{ and } \mathbf{S} = \begin{bmatrix} \mathbf{S}^{(1)} & & & \mathbf{0} \\ & \mathbf{S}^{(2)} & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{S}^{(N)} \end{bmatrix}. \tag{3.32}$$

Here, the method of reconstructing a single rigid object can simply be applied to each pair of sub-matrices $\mathbf{M}^{(j)}$ and $\mathbf{S}^{(j)}$.

The greatest obstacle is that the features in the observation matrix are unordered in the sense that consecutive columns do not necessarily belong to the same rigid object. The columns of the observation matrix have to be permuted such that block structure of (3.32) holds. In [15, 16], the *shape interaction matrix* is defined—

$$\mathbf{Q} = \mathbf{V}\mathbf{V}^T, \tag{3.33}$$

where $\mathbf{V}$ is obtained from the SVD (3.28). It is shown that $\mathbf{Q}$ has the following property.

$$\begin{array}{lcll} Q_{ij} & \neq & 0 & \text{if feature } i \text{ and feature } j \text{ belong to the same object} \\ Q_{ij} & = & 0 & \text{if feature } i \text{ and feature } j \text{ belong to different objects} \end{array} \tag{3.34}$$

Since this property is invariant to motion, it can be used to determine the grouping of features into objects.

Improvements to the method in [15, 16] have been proposed. These are based on

- different criteria for determining which entries of Q are 0 under noisy measurements [19, 23];
- employing a graph matching algorithm [19]; and
- employing subspace separation on the measured feature tracks [33].

The shape interaction matrix property in (3.34) holds only in the case of *independently* moving rigid objects. These objects may be line structures, planar structures or fully 3-dimensional. An important type of motion where this formulation cannot be used is articulated motion. In this case, the motions of rigid parts are highly interdependent and the shape interaction matrix reflects this by not having any 0 entries.

Next, we consider the recovery of the structure and motion of elastic objects. The formulation in [11, 61] describes elastic motion as a linear combination of a number of *basis shapes* $\mathbf{S}^{(i)}$—

$$\sum_{i=1}^{K} l^{(i)}\mathbf{S}^{(i)}, \tag{3.35}$$

where each $\mathbf{S}^{(i)}$ is a $3 \times P$ matrix describing a rigid shape and K is the number of basis shapes. The elasticity is modelled by allowing the weights $l^{(i)}$ to vary. This allows the smooth transition from one basis shape to another.

To cast this shape model in the formulation $\mathbf{W} = \mathbf{MS}$ under the weak perspective camera model, we write

$$\mathbf{M} = \begin{bmatrix} l_1^{(1)}\mathbf{R}_1 & l_1^{(2)}\mathbf{R}_1 & \cdots & l_1^{(K)}\mathbf{R}_1 \\ l_2^{(1)}\mathbf{R}_2 & l_2^{(2)}\mathbf{R}_2 & \cdots & l_2^{(K)}\mathbf{R}_2 \\ & & \cdots & \\ l_F^{(1)}\mathbf{R}_F & l_F^{(2)}\mathbf{R}_F & \cdots & l_F^{(K)}\mathbf{R}_F \end{bmatrix} \text{ and } \mathbf{S} = \begin{bmatrix} \mathbf{S}^{(1)} \\ \mathbf{S}^{(2)} \\ \vdots \\ \mathbf{S}^{(K)} \end{bmatrix}. \tag{3.36}$$

Here, $\mathbf{R}_f$ is the camera rotation at frame f and $l_f^{(i)}$ is the weight assigned to basis shape $\mathbf{S}^{(i)}$ at frame f. In the formulation in [11, 61], camera translation is subtracted from each frame before performing the matrix factorisation. The projection coefficients of the weak perspective camera model are encoded in the weights of the basis shapes. The weights in any frame sum to the weak perspective coefficient.

Note that $\mathbf{M}$ is $2F \times 3K$ since each $\mathbf{R}_f$ is 2×3, and $\mathbf{S}$ is $3K \times P$. This places an upper bound of $3K$ on rank($\mathbf{W}$). A drawback of this formulation is that K has to be determined manually. The factorisation algorithm cannot discover it automatically and is sensitive to its choice.

No matrix formulation for or upper bound on rank($\mathbf{W}$) for articulated motion was found in the literature. We present original work on these topics in Chapter 4.

A final comment on batch methods is that they all employ the singular value decomposition and that the behaviour of the SVD under noisy measurements is therefore important. The SVD is useful in that it can be used to eliminate Gaussian noise by discarding small singular values and has a numerically stable implementation [20, 47].

In reality, the Gaussian noise assumption is violated. Most importantly, the noise is correlated between features and the noise of a feature is directional. That noise is directional can be inferred from, e.g., the feature tracking work of [53]. Here, it is shown that a feature on the corner of an object or with a high *texturedness* is more reliable than a unidirectional (or almost unidirectional) feature.

This consideration motivates the development of a SVD framework that incorporates directional uncertainty in [27]. The observation matrix $\mathbf{W}^T$ (in [27] row vectors are used to represent feature tracks, rather than column vectors) is transformed using its inverse covariance matrix. The SVD is then applied to the resulting uncorrelated observation matrix. In this matrix, all features—at corners or unidirectional—have the same reliability.

3.3.2 Sequential Methods

Sequential reconstruction methods maintain internal variables that allow estimation of the structure and motion and update these variables with every frame of feature data. Convergence of the variables to some stable values (or near-stable values with noisy measurements) leads to convergence of the structure and motion estimates to the true structure and motion.

An advantage of sequential methods is that they are more suited as real-time SfM implementations than batch methods. Estimates of the structure and motion is available at every frame and new frames are used to refine the current estimates.

Note that sequential methods frequently require that the input frames are processed in order. This is in contrast with batch methods (Section 3.3.1), where the ordering of frames in the observation matrix has no effect on the reconstructed structure and motion at each frame.

Early work [63] was based on iterative refinement. Here, the current internal model of the object is projected onto the image plane and the difference between projected and observed feature positions used to update the object structure.

There are a number of sequential reconstruction methods based on the Kalman filter or some variant thereof [2, 3, 12, 54, 66]. The Kalman filter is a set of equations that provides an iterative approach to the least-squares method [72]. The equations model an underlying process that generates observable data. The algorithm iteratively updates an internal representation of the process state variables and compares the measured data to the expected data. Based on the error between the expected and measured values, the current estimate of the process state is updated. The Kalman filter is a linear estimator that assumes Gaussian process and measurement noise. The extended Kalman filter is used to perform estimation of non-linear processes. The unscented Kalman filter [64, 69, 70] is a more recent development which handles non-linear estimation without the restrictive assumption of Gaussian process and measurement noise.

A method that does not use a Kalman filter is presented in [44]. This work builds on that of [58] and allows sequential reconstruction of rigid structure and motion using the orthographic camera model. At each frame the length P measurement vectors $\mathbf{u}_f$ and $\mathbf{v}_f$ are used to update internal variables, which allow the estimation of the shape and camera rotation at that frame. The sequential algorithm is designed such that no internal variables are maintained of which the size grows with the number of frames processed.

The internal variables of the method consist primarily of a $P \times P$ matrix, $\mathbf{Z}_f$.

$$\begin{aligned} \mathbf{Z}_f &= \mathbf{Z}_{f-1} + \mathbf{u}_f \mathbf{u}_f^T + \mathbf{v}_f \mathbf{v}_f^T \\ \mathbf{Z}_0 &= \mathbf{0}. \end{aligned} \tag{3.37}$$

It is shown that

$$\mathbf{Z}_f = \mathbf{W}_f^T \mathbf{W}_f, \tag{3.38}$$

where $\mathbf{W}_f$ contains all the rows of the observation matrix for frames 1 through f. This allows that the SVD of $\mathbf{Z}_f$ provides much of the information contained in the SVD of $\mathbf{W}_f$ without requiring the storage of a matrix that grows with the number of frames.

The 3 dominant eigenvectors of $\mathbf{Z}_f$ are computed and used to maintain a stationary basis of the shape space, i.e. vectors that span the rows of the $3 \times P$ rigid shape matrix. This basis can be used at any time-step to reconstruct the camera rotation and object shape. The method is shown to converge to the solution found by applying the batch factorisation method of [58].

3.3.3 Model-based Methods

Model-based methods use some parametric model on which the motion of features in an image sequence is based. This is often found in human motion capture. Here, the underlying kinematic model of human motion, corresponding to human skeletal structure [18], is assumed and used to constrain the possible structure from motion solutions. In [17], joint angle constraints are used in addition to the kinematic constraints imposed by the kinematic model. At any joint only a limited amount of rotation is possible by the limbs meeting at that point. This information is employed to facilitate human motion capture.

In [9, 10], twists (3.16) and exponential maps (3.17) are used to represent kinematic motion. The initial configuration of the object is defined by indicating the locations of limbs and joints on the first frame in the image sequence used for reconstruction. This manual initialisation of the kinematic model is common, see also [17]. As part of the initialisation, the length or size of the limbs needs to be defined as well as the initial position of the object or person being tracked. However, see [18] where the locations of limbs are discovered in a coarse-to-fine approach. Firstly, the major axis of the body is found using principal component analysis, then other limbs are found progressively.

3.3.4 Direct Methods

Direct methods use unprocessed image information—pixel intensities—to reconstruct depth information at *every* point in the image sequence. This is more ambitious than the previous methods, which are based on extracting features from the image and recovering the depth information for these features only [25].

Central to these methods is the *brightness constancy constraint* [53],

$$I(x,\ y,\ t+1) = I(x - u(x,\ y,\ t),\ y - v(x,\ y,\ t),\ t), \tag{3.39}$$

which states that the image intensities at the next frame (time $t+1$) can be obtained by applying image motion (u, v) to the intensities at the current frame (time t). The Taylor expansion around (x, y) of (3.39) is linearised to obtain the constraint

$$I_x u + I_y v + I_t = 0, \tag{3.40}$$

where I_x, I_y and I_t indicate image derivatives.

To reconstruct dense depth information, the entire image is assumed to undergo similar motion. The type of motion is defined by a set of equations known as the *motion model.* The equations of the motion model are substituted into (3.40) to obtain an over-constrained system. These equations are solved to determine overall image motion by estimating the parameters of the motion model. The system is over-constrained because we construct one equation per pixel of the image.

- The *translation model* [24, 53],

$$\begin{aligned} u(x,y) &= a_1 \\ v(x,y) &= a_2, \end{aligned} \tag{3.41}$$

  describes uniform motion across the image at any instant in time. This model has only 2 parameters to estimate, but cannot describe a wide range of motions.

- The *affine model* [24, 25, 53],

$$\begin{aligned} u(x,y) &= a_1 + a_2 x + a_3 y \\ v(x,y) &= a_4 + a_5 x + a_6 y, \end{aligned} \tag{3.42}$$

  is a good approximation for image sequences of distant scenes, e.g. footage of the ground from an aeroplane.

- The *moving planar surface model* [24],

$$\begin{aligned} u(x,y) &= a_1 + a_2x + a_3y + a_7x^2 + a_8xy \\ v(x,y) &= a_4 + a_5x + a_6y + a_7xy + a_8y^2, \end{aligned} \tag{3.43}$$

  is a pseudo projective transformation of a moving plane, which is also useful for approximating objects or areas of the image sequence with little depth variation.

- The *instantaneous velocity field model* [25],

$$\begin{aligned} u(x,y) &= -xy\Omega_X + (1+x^2)\Omega_Y - y\Omega_Z + (T_X - T_Zx)/Z \\ v(x,y) &= -(1+y^2)\Omega_X + xy\Omega_Y - x\Omega_Z + (T_Y - T_Zy)/Z, \end{aligned} \tag{3.44}$$

  is a 3D motion model in that the depth of each pixel is modelled in the parameter Z. Further, $(\Omega_X, \Omega_Y, \Omega_Z)$ describes the camera rotation and (T_X, T_Y, T_Z) the camera translation.

The advantages of direct methods stem from the fact that they estimate the overall image motion, which is described with a low-dimensional parameter space, using a large number of equations or constraints—one per pixel. This allows very high, even sub-pixel, precision in motion estimation [25].

There is a strong correlation between these direct methods and methods for feature tracking in images. Feature tracking is achieved by applying a direct method to a small window within the image. As long as the brightness constancy constraint (3.39) holds within this tracking window, the direct methods can be used on the pixels in that window to recover their velocity.

Chapter 4

Advances in Articulated Structure From Motion

This thesis focuses on the articulated structure from motion problem and some theoretical and practical advances were made in exploring it. In Chapter 3, it was mentioned that no matrix formulation for the articulated structure from motion (ASfM) problem was found in the literature. The theoretical part of the advances made in this research describes a matrix formulation for ASfM and derives an upper bound on the rank of the observation matrix.[1] The practical section discusses the implementation of and improvements to a sequential reconstruction algorithm for ASfM.

4.1 Matrix Factorisation

In Section 3.3.1 it was shown that the observation matrix, $\mathbf{W}$, has rank $4N$ and can be factored into motion and shape matrices with the block structure in (3.32) when the motion of N independently moving rigid objects is considered.

4.1.1 Rank Constraint on the Observation Matrix

Articulated tree motion can be modelled with

$$\text{rank}(\mathbf{W}) = 3n + 1, \tag{4.1}$$

[1]The material in Section 4.1 was first presented as [50].

where n is the number of segments in the articulated object. We can factor $\mathbf{W} = \mathbf{MS}$ such that each $2 \times (3n+1)$ row pair in $\mathbf{M}$ has the form

$$\mathbf{M}'_f = l_f \left[\begin{array}{ccccc} \mathbf{R}_f^{(1)} & \mathbf{R}_f^{(2)} & \cdots & \mathbf{R}_f^{(n)} & \mathbf{t}_f \end{array} \right], \tag{4.2}$$

where l_f is the weak perspective constant, $\mathbf{R}_f^{(j)}$ contains the first two rows of the rotation matrix for segment j and $\mathbf{t}_f$ contains the translation of the root segment—the segment with no parent—at frame f.

The shape matrix has the form

$$\mathbf{S} = \left[\begin{array}{ccccc} \mathbf{S}^{(1)} & \mathbf{O}^{(1,2)} & \mathbf{O}^{(1,3)} & \dots & \mathbf{O}^{(1,n)} \\ & \mathbf{S}^{(2)} & \mathbf{O}^{(2,3)} & \dots & \mathbf{O}^{(2,n)} \\ & & \mathbf{S}^{(3)} & \dots & \mathbf{O}^{(3,n)} \\ & & & \ddots & \vdots \\ & & & & \mathbf{S}^{(n)} \\ \mathbf{1}_{p_1}^T & \mathbf{1}_{p_2}^T & \mathbf{1}_{p_3}^T & \cdots & \mathbf{1}_{p_n}^T \end{array} \right]. \tag{4.3}$$

Here each $3 \times p_j$ matrix $\mathbf{S}^{(j)}$ contains the coordinates of the jth segment in its local coordinate system. The local coordinate system of a segment has origin at the segment point of rotation, i.e. the local motion of a segment is described as a rotation only. Each $3 \times p_j$ matrix $\mathbf{O}^{(k,j)}$ describes the relationship between segments j and k (see (4.4) below for details). The last row of the matrix consists entirely of 1s and the notation $\mathbf{1}_{p_j}^T$ represents a row vector of 1s with length p_j. Finally, p_j is the number of points in segment j such that $\sum_j p_j = p$, the total number of points in the object.

If segment j has parent segment k,

$$\begin{array}{rcll} \mathbf{O}^{(m,j)} & = & \mathbf{O}^{(m,k)} & \forall\ m = 1 \dots (k-1), \\ \mathbf{O}^{(k,j)} & = & \mathbf{o}^{(k,j)} \times \mathbf{1}_{p_j}^T & \\ \mathbf{O}^{(m,j)} & = & \mathbf{0}_{(3 \times p_j)} & \forall\ m = (k+1) \dots (j-1). \end{array} \tag{4.4}$$

Here $\mathbf{o}^{(k,j)}$ is the 3×1 column vector describing the point of rotation of segment j in the local coordinate system of parent segment k. From this definition, notice that the columns of an $\mathbf{O}^{(k,j)}$ are all equal.

The structure and motion of any articulated tree can be described using shape and motion matrices of the form described here.

4.1.2 Articulated Tree Topology

We present here a reconstruction algorithm for the articulated tree case, under the assumption that observed points have been assigned correctly to their respective segments. Because of the known point-to-segment assignment, the observation matrix can be written as

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{(1)} & \mathbf{W}^{(2)} & \dots & \mathbf{W}^{(n)} \end{bmatrix} \tag{4.5}$$

where the index of each segment is greater than that of its parent segment—this implies that $\mathbf{W}^{(1)}$ is the observation of the root segment. Each of these sub-matrices now represent the motion of a single, rigid segment. Reconstruction of the articulated tree takes place segment by segment, from left to right in the motion, see (4.2), and shape, see (4.3), matrices. We use the method for reconstruction of a single, rigid object as originally proposed in [58].

Firstly, reconstruct the root segment. This yields $\mathbf{R}^{(1)}$, $\mathbf{t}$ and $\mathbf{S}^{(1)}$ in (4.2) and (4.3). The translation $\mathbf{t}$ is reconstructed in the first step, since the root segment is the only segment that can undergo free translation. All other segments are attached to a parent segment and can undergo local rotation only.

Next, reconstruct each of the remaining segments. Despite the fact that each segment can undergo rotation only, we do find that the rigid reconstruction algorithm yields a translation component. This is due to two factors.

The first is that the rigid reconstruction algorithm chooses an arbitrary coordinate system for the shape matrix. However, for articulated motion, we want the origin of the coordinate system to be at the point of rotation of the segment. The rigid reconstruction algorithm yields $\hat{\mathbf{S}}^{(j)}$ and we want

$$\mathbf{S}^{(j)} = \hat{\mathbf{S}}^{(j)} + \mathbf{c}^{(j)} \times \mathbf{1}_{p_j}^T. \tag{4.6}$$

Here $\mathbf{c}^{(j)}$ is a 3×1 vector that adjusts the origin of the shape matrix.

The second factor is that the point of rotation of the segment moves in the global coordinate system. As the parent of a segment moves, the point of rotation of the child moves with it. This movement is described by (4.2) and (4.3). For segment j

$$\mathbf{W}^{(j)} = \sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{O}^{(i,j)} + \mathbf{R}^{(j)}\mathbf{S}^{(j)} + \mathbf{T}, \tag{4.7}$$

where $\mathbf{O}^{(i,j)}$ is as defined in (4.4) and $\mathbf{T} \equiv \mathbf{t} \times \mathbf{1}_{p_j}^T$ is a matrix with columns equal to the global translation.

From (4.7):

$$\mathbf{W}^{(j)} - \mathbf{T} = \mathbf{R}^{(j)}\mathbf{S}^{(j)} + \sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{O}^{(i,j)} \tag{4.8}$$

Next, we perform rigid reconstruction on the left hand side to get

$$\mathbf{W}^{(j)} - \mathbf{T} = \hat{\mathbf{R}}^{(j)}\hat{\mathbf{S}}^{(j)} + \hat{\mathbf{T}}^{(j)}. \tag{4.9}$$

Here $\hat{\mathbf{R}}^{(j)}$, $\hat{\mathbf{S}}^{(j)}$ and $\hat{\mathbf{T}}^{(j)}$ are the reconstructed rotation, shape and translation matrices for segment j and $\hat{\mathbf{T}}^{(j)} \equiv \hat{\mathbf{t}}^{(j)} \times \mathbf{1}_{p_j}^T$.

The origin of $\hat{\mathbf{S}}^{(j)}$ is still incorrect and we write

$$\mathbf{S}^{(j)} = \hat{\mathbf{S}}^{(j)} + \mathbf{C}^{(j)} \tag{4.10}$$

with $\mathbf{C}^{(j)} \equiv \mathbf{c}^{(j)} \times \mathbf{1}_{p_j}^T$.

From the equality of (4.8) and (4.9)

$$\hat{\mathbf{R}}^{(j)}\hat{\mathbf{S}}^{(j)} + \hat{\mathbf{T}}^{(j)} = \mathbf{R}^{(j)}\mathbf{S}^{(j)} + \left[\sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{O}^{(i,j)}\right], \tag{4.11}$$

and then, from (4.10),

$$\hat{\mathbf{R}}^{(j)}\mathbf{S}^{(j)} + \hat{\mathbf{T}}^{(j)} = \mathbf{R}^{(j)}\mathbf{S}^{(j)} + \left[\sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{O}^{(i,j)}\right] + \hat{\mathbf{R}}^{(j)}\mathbf{C}^{(j)}, \tag{4.12}$$

Next, we factor $\mathbf{O}^{(i,j)}$, $\mathbf{C}^{(j)}$ and $\hat{\mathbf{T}}^{(j)}$ based on their definitions

$$\hat{\mathbf{R}}^{(j)}\mathbf{S}^{(j)} + \hat{\mathbf{t}}^{(j)} \times \mathbf{1}_{p_j}^T = \mathbf{R}^{(j)}\mathbf{S}^{(j)} + \left[\sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{o}^{(i,j)} \times \mathbf{1}_{p_j}^T\right] + \hat{\mathbf{R}}^{(j)}\mathbf{c}^{(j)} \times \mathbf{1}_{p_j}^T. \tag{4.13}$$

It follows that

$$\hat{\mathbf{R}}^{(j)} = \mathbf{R}^{(j)}, \tag{4.14}$$

yielding the required rotation matrix and

$$\begin{aligned} \hat{\mathbf{t}}^{(j)} &= \left[\sum_{i=1}^{j-1} \mathbf{R}^{(i)}\mathbf{o}^{(i,j)}\right] + \hat{\mathbf{R}}^{(j)}\mathbf{c}^{(j)}, & (4.15) \\ &= \begin{bmatrix} \mathbf{R}^{(1)} & \mathbf{R}^{(2)} & \cdots & \mathbf{R}^{(j)} \end{bmatrix} \begin{bmatrix} \mathbf{o}^{(1,j)} \\ \mathbf{o}^{(2,j)} \\ \vdots \\ \mathbf{o}^{(j-1,j)} \\ \mathbf{c}^{(j)} \end{bmatrix}, & (4.16) \end{aligned}$$

which is an over-constrained system that allows us to solve for $\mathbf{o}^{(i,j)}$ and $\mathbf{c}^{(j)}$. From this $\mathbf{O}^{(i,j)}$ and $\mathbf{S}^{(j)}$ are constructed. With $\mathbf{R}^{(j)}$, $\mathbf{S}^{(j)}$ and $\mathbf{O}^{(i,j)}$, we have all the parameters for segment j.

To summarise, the reconstruction algorithm consists of the following steps.

- Reconstruct $\mathbf{R}^{(1)}$, $\mathbf{S}^{(1)}$ and $\mathbf{t}$.

- For each observation sub-matrix $\mathbf{W}^{(j)},\ j = 2, 3, \ldots, n$:
  - Subtract the global translation from $\mathbf{W}^{(j)}$.
  - Reconstruct $\mathbf{R}^{(j)}$ and $\hat{\mathbf{S}}^{(j)}$. $\mathbf{R}^{(j)}$ contains the rotation parameters for segment j.
  - Calculate $\mathbf{o}^{(i,j)}$ and $\mathbf{c}^{(j)}$ from the over-constrained system in (4.16). Here, $\mathbf{o}^{(i,j)}$ contains the origin of segment j in the coordinate system of its parent segment.
  - Calculate $\mathbf{S}^{(j)} = \hat{\mathbf{S}}^{(j)} + \mathbf{c}^{(j)} \times \mathbf{1}_{p_j}^T$ to find the find the shape parameters of segment j.

4.1.3 Forests

The articulated tree formulation can be extended to allow *articulated forests.* A forest is simply a collection of trees, i.e. a number of independently moving objects where each object is undergoing articulated tree motion.

The motion matrix is now of the same form as in (3.32), except that each $\mathbf{M}^{(j)}$ is an articulated tree motion matrix as in (4.2) and N is the number of trees in the forest. The shape matrix has the same form as in (3.32) but with each $\mathbf{S}^{(j)}$ an articulated tree shape matrix, as in (4.3).

Since each $\mathbf{S}^{(j)}$ has $3n_j + 1$ rows where n_j is the number of segments in tree j, the number of rows in $\mathbf{S}$ (and columns in $\mathbf{M}$) is $r = 3\sum_j n_j + N$. Consequently, this is the rank of the observation matrix $\mathbf{W}$ in the non-degenerate case.

Note that a rigid object is simply an articulated tree with one segment. If each articulated tree in the forest has one segment, the observation matrix will have rank $3N + N = 4N$. This means that the articulated forest simplifies to the case of N independent objects, as discussed in [15, 16].

Because articulated tree motion has fewer degrees of freedom than independent object motion, the observation matrix of the former has lower rank—$3n+1$ as opposed to $4N$, where n is the number of articulated segments in the one case and N the number of independent objects in the other. This shows that the articulated tree formulation is a degenerate case of N independently moving objects. In contrast, the articulated forest formulation is a generalisation thereof.

An algorithm for reconstructing articulated forests proceeds by combining aspects of the reconstruction of independently moving objects and articulated trees. Since the articulated trees within a forest are assumed to be independent, the shape interaction matrix property in Section 3.3.1 can be modified as follows to yield the correct point-to-tree assignment—

$$\begin{aligned} Q_{ij} &\neq 0 \quad \text{if feature } i \text{ and feature } j \text{ belong to the same } \textit{tree} \\ Q_{ij} &= 0 \quad \text{if feature } i \text{ and feature } j \text{ belong to different } \textit{trees} \end{aligned} \tag{4.17}$$

After the point-to-tree assignment is known, features are grouped to yield a block structure as in (3.32). Next, each articulated tree can be reconstructed individually.

4.1.4 Limitations

A limitation of this matrix factorisation method is that the assignment of points to segments—both the number of points per segment and the order of the points and segments—cannot be computed automatically. As discussed in Section 3.3.1, the assignment of points to independent objects was computed using the shape interaction matrix **Q** of [15, 16]. By reordering the rows and columns of **Q** such that it has a block-structure, the point-to-object assignment is discovered. This method fails in the case of articulated motion. The derivation of the structure of **Q** is based on the assumption that **S** has the block-structure of (3.32), which does not hold for (4.3). No analogy to the **Q** matrix was found for this case.

We circumvent this problem by annotating the first frame of an image sequence by hand. This is feasible in practice since tracked features need to be grouped in one frame only, with reconstruction being performed automatically from there.

4.2 A Sequential Reconstruction Implementation

As was mentioned in Section 3.3.2, [44] proposed a sequential factorisation approach to rigid structure from motion under orthography. The reconstruction algorithm is described below and improvements presented. The improvements allow

- the use of the weak perspective rather than the orthographic camera model;
- the occlusion and disocclusion of features; and
- the reconstruction of articulated segments and articulated topology.

There are two parts to the algorithm. The first computes and maintains a stationary basis for the $3 \times P$ shape matrix. This is done by estimating iteratively the eigenvectors of the observation matrix $\mathbf{W}$. The full observation matrix across all frames is never stored or required for computations. The first three steps of the algorithm are—

$$\begin{array}{lll} 1) & \mathbf{Z}_f := \mathbf{Z}_{f-1} + \mathbf{u}_f\mathbf{u}_f^T + \mathbf{v}_f\mathbf{v}_f^T & (\mathbf{Z}_0 := \mathbf{0}_{P\times P}) \\ 2) & \mathbf{Y} := \mathbf{Z}_f\mathbf{Q}_{f-1} & (\mathbf{Q}_0 := \mathbf{I}_{P\times 3}) \\ 3) & \mathbf{Q}_f\mathbf{R} := \mathbf{Y} & \text{(QR-factorisation)} \end{array}$$

Here $\mathbf{u}_f$ and $\mathbf{v}_f$ form the observation at frame f such that

$$\mathbf{W} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{v}_1^T \\ \vdots \\ \mathbf{u}_F^T \\ \mathbf{u}_F^T \end{bmatrix}. \tag{4.18}$$

Since $\mathbf{u}_f$ and $\mathbf{v}_f$ are length P vectors, $\mathbf{Z}_f$ is $P \times P$.

It is shown in [44] that span($\mathbf{Q}_f$) converges to span($\mathbf{S}^T$), i.e. the shape space. Even though span($\mathbf{Q}_f$) is stationary, the columns of $\mathbf{Q}_f$ are not. To find a stationary basis for the shape space, we compute the eigenvectors of $\mathbf{Q}_f$—

$$\begin{array}{lll} 4) & \mathbf{H}_f := \mathbf{Q}_f\mathbf{Q}_f^T & \\ 5) & \mathbf{Y} := \mathbf{H}_f\overline{\mathbf{Q}}_{f-1} & (\overline{\mathbf{Q}}_0 := \mathbf{I}_{P\times 3}) \\ 6) & \overline{\mathbf{Q}}_f\mathbf{R} := \mathbf{Y} & \text{(QR-factorisation)} \end{array}$$

Thus the columns of $\overline{\mathbf{Q}}_f$ converge to a stationary basis of the shape space.

The second part of the reconstruction maintains variables that allow the reconstruction of $\mathbf{M}_f$, the 2×3 rotation matrix at frame f, and $\mathbf{S}$ from $\overline{\mathbf{Q}}_f$. For the orthographic camera model, the following motion constraints hold

$$\hat{\mathbf{i}}_f^T\mathbf{A}\mathbf{A}^T\hat{\mathbf{i}}_f = 1 \quad \forall\, f = 1, 2, \ldots, F \tag{4.19}$$
$$\hat{\mathbf{j}}_f^T\mathbf{A}\mathbf{A}^T\hat{\mathbf{j}}_f = 1 \quad \forall\, f = 1, 2, \ldots, F \tag{4.20}$$
$$\hat{\mathbf{i}}_f^T\mathbf{A}\mathbf{A}^T\hat{\mathbf{j}}_f = 0 \quad \forall\, f = 1, 2, \ldots, F \tag{4.21}$$

such that

$$\mathbf{M}_f = \begin{bmatrix} \hat{\mathbf{i}}_f^T \\ \hat{\mathbf{j}}_f^T \end{bmatrix}\mathbf{A} \tag{4.22}$$

for some invertible 3×3 matrix $\mathbf{A}$. If we let

$$\mathbf{B} = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_4 & b_5 \\ b_3 & b_5 & b_6 \end{bmatrix} = \mathbf{A}\mathbf{A}^T, \tag{4.23}$$

we can rewrite (4.19)–(4.21) as

$$\mathbf{G}\mathbf{b} = \mathbf{c} \tag{4.24}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_1) \\ \mathbf{g}^T(\hat{\mathbf{j}}_1, \hat{\mathbf{j}}_1) \\ \vdots \\ \mathbf{g}^T(\hat{\mathbf{i}}_F, \hat{\mathbf{i}}_F) \\ \mathbf{g}^T(\hat{\mathbf{j}}_F, \hat{\mathbf{j}}_F) \\ \mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{j}}_1) \\ \vdots \\ \mathbf{g}^T(\hat{\mathbf{i}}_F, \hat{\mathbf{j}}_F) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} 2F \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} F \end{matrix} \tag{4.25}$$

and

$$\mathbf{g}^T(\mathbf{p}, \mathbf{q}) = [\, p_1q_1 \quad p_1q_2 + p_2q_1 \quad p_1q_3 + p_3q_1 \quad p_2q_2 \quad p_2q_3 + p_3q_2 \quad p_3q_3 \,]\,. \tag{4.26}$$

The simplest solution to this over-constrained system can by found using the pseudo-inverse method, so that

$$\mathbf{b} = (\mathbf{G}^T\mathbf{G})^{-1}(\mathbf{G}^T\mathbf{c}). \tag{4.27}$$

By setting

$$\mathbf{D} = \mathbf{G}^T\mathbf{G} \quad \text{and} \quad \mathbf{e} = \mathbf{G}^T\mathbf{c} \tag{4.28}$$

and computing the values of **D** and **e** at each iteration, we can compute **b**—and subsequently **B**, **A**, $\mathbf{M}_f$ and **S**—at any frame. These variables are maintained using the final part of the algorithm—

```
7)  $\hat{\mathbf{i}}_f^T := \mathbf{u}_f^T\overline{\mathbf{Q}}_f$
8)  $\hat{\mathbf{j}}_f^T := \mathbf{v}_f^T\overline{\mathbf{Q}}_f$
9)  $\mathbf{D}_f := \mathbf{D}_{f-1} + \mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{i}}_f)\mathbf{g}^T(\hat{\mathbf{i}}_f, \hat{\mathbf{i}}_f) +$        $(\mathbf{D}_0 := \mathbf{0}_{6\times6})$
         $\mathbf{g}(\hat{\mathbf{j}}_f, \hat{\mathbf{j}}_f)\mathbf{g}^T(\hat{\mathbf{j}}_f, \hat{\mathbf{j}}_f) +$
         $\mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{j}}_f)\mathbf{g}^T(\hat{\mathbf{i}}_f, \hat{\mathbf{j}}_f)$
10) $\mathbf{e}_f := \mathbf{e}_{f-1} + \mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{i}}_f) + \mathbf{g}(\hat{\mathbf{j}}_f, \hat{\mathbf{j}}_f)$        $(\mathbf{e}_0 := \mathbf{0}_{6\times1})$
```

4.2.1 The Weak Perspective Camera Model

For the weak perspective camera model the motion constraints of (4.19)–(4.21) change (see (A.9) and (A.10) in Appendix A) to

$$\left(\hat{\mathbf{i}}_f\mathbf{A}\mathbf{A}^T\hat{\mathbf{i}}_f^T\right) - \left(\hat{\mathbf{j}}_f\mathbf{A}\mathbf{A}^T\hat{\mathbf{j}}_f^T\right) = 0 \qquad \forall\, f = 1, 2, \ldots, F \tag{4.29}$$

$$\hat{\mathbf{i}}_f\mathbf{A}\mathbf{A}^T\hat{\mathbf{j}}_f^T = 0 \qquad \forall\, f = 1, 2, \ldots, F \tag{4.30}$$

This change affects the equations underlying the second part of the sequential reconstruction algorithm. The form in (4.24) still holds, but

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_1) \\ \mathbf{g}^T(\hat{\mathbf{j}}_1, \hat{\mathbf{j}}_1) \\ \mathbf{g}^T(\hat{\mathbf{i}}_2, \hat{\mathbf{i}}_2) - \mathbf{g}^T(\hat{\mathbf{j}}_2, \hat{\mathbf{j}}_2) \\ \vdots \\ \mathbf{g}^T(\hat{\mathbf{i}}_F, \hat{\mathbf{i}}_F) - \mathbf{g}^T(\hat{\mathbf{j}}_F, \hat{\mathbf{j}}_F) \\ \mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{j}}_1) \\ \vdots \\ \mathbf{g}^T(\hat{\mathbf{i}}_F, \hat{\mathbf{j}}_F) \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix}\right\} 2F-1 \quad . \tag{4.31}$$

Consequently the second part of the algorithm changes to

$$\begin{aligned}
&7) \quad \hat{\mathbf{i}}_f^T := \mathbf{u}_f^T \overline{\mathbf{Q}}_f \\
&8) \quad \hat{\mathbf{j}}_f^T := \mathbf{v}_f^T \overline{\mathbf{Q}}_f \\
&9) \quad \mathbf{D}_f := \mathbf{D}_{f-1} + (\mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{i}}_f) - \mathbf{g}(\hat{\mathbf{j}}_f, \hat{\mathbf{j}}_f))(\mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{i}}_f) - \mathbf{g}(\hat{\mathbf{j}}_f, \hat{\mathbf{j}}_f))^T + \\
&\qquad\qquad \mathbf{g}(\hat{\mathbf{i}}_f, \hat{\mathbf{j}}_f)\mathbf{g}^T(\hat{\mathbf{i}}_f, \hat{\mathbf{j}}_f) \\
&\qquad \mathbf{D}_1 := \mathbf{g}(\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_1)\mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_1) + \\
&\qquad\qquad \mathbf{g}(\hat{\mathbf{j}}_1, \hat{\mathbf{j}}_1)\mathbf{g}^T(\hat{\mathbf{j}}_1, \hat{\mathbf{j}}_1) + \\
&\qquad\qquad \mathbf{g}(\hat{\mathbf{i}}_1, \hat{\mathbf{j}}_1)\mathbf{g}^T(\hat{\mathbf{i}}_1, \hat{\mathbf{j}}_1) \\
&10) \quad \mathbf{e}_f := \mathbf{e}_{f-1} \\
&\qquad \mathbf{e}_1 := \mathbf{g}(\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_1) + \mathbf{g}(\hat{\mathbf{j}}_1, \hat{\mathbf{j}}_1)
\end{aligned}$$

This alteration does not influence the shape space basis vectors, only the matrices used when reconstructing $\mathbf{M}_f$ and $\mathbf{S}$ at any step. Because the weak perspective system of equations is nearly homogeneous, not all frames are used to form $\mathbf{D}$ and $\mathbf{e}$. This avoids numerical problems that occur because the pseudo-inverse yields the trivial solution in the case of a homogeneous system of equations.

The frames used to construct $\mathbf{D}$ and $\mathbf{e}$ are the most recent F'. This requires that a $2F' \times P$ observation history matrix is stored. This does increase the size of the internal variables of the algorithm but does still not require the storage of any variables that increase with the number of frames processed. Typically, F' is between 15 and 50—this corresponds to approximately 1 or 2 seconds of video in practice.

4.2.2 Occlusion and Disocclusion

The feature coordinates used to perform reconstruction are supplied by a feature tracker, which may lose features—for example when they become occluded. These occlusions leave *holes* in the observation vectors $\mathbf{u}_f$ and $\mathbf{v}_f$. To continue reconstruction, the missing features have to be discarded in reconstruction steps beyond the last frame in which they were visible.

Discarding features changes the previously stationary basis of the shape space, $\overline{\mathbf{Q}}$. Because of this, two things need to be kept in mind. Firstly, the algorithm must converge to the new shape basis so that reconstruction can continue with the remaining features. Secondly, it must also maintain enough information about the lost features to allow reconstruction of their 3D positions.

We achieve the first goal by truncating all vectors and matrices that form part of the algorithm, in order to remove the dropped features. From here, the algorithm proceeds normally. Note that truncating the algorithm variables will corrupt $\mathbf{Q}$ and $\overline{\mathbf{Q}}$ in the sense that they are no longer orthogonal matrices. However, as long as the number of features dropped is small, the algorithm will recover quickly and converge to the new stationary shape space basis.

We achieve the second goal by storing the matrix $\overline{\mathbf{Q}}_{f_d}$, of which the columns span shape space just before the features are dropped. In order to reconstruct the full shape matrix from the truncated $\overline{\mathbf{Q}}$ and the dropped features (stored in $\overline{\mathbf{Q}}_{f_d}$), we determine the transformation matrix between the old and new stationary bases. Let f_d be the frame just before features were dropped and f the frame at which reconstruction occurs. Then, we find the 4×4 transformation matrix $\mathbf{T}$, such that

$$\begin{bmatrix} \overline{\mathbf{Q}}_{f_d} \\ \mathbf{1}_P^T \end{bmatrix} \mathbf{T} = \begin{bmatrix} \overline{\mathbf{Q}}_f \\ \mathbf{1}_P^T \end{bmatrix}. \tag{4.32}$$

This coordinate transformation has the form of the motion matrix in (3.11) and describes the rotation and translation required to move form the shape space spanned by $\overline{\mathbf{Q}}_{f_d}$ to that spanned by $\overline{\mathbf{Q}}_f$. Furthermore, $\mathbf{T}$ allows the description of the dropped feature points in the new shape space and the subsequent reconstruction of their 3D positions along with the remaining features.

As features may be dropped by the feature tracker, so new features may also be gained—or previously dropped features regained. As with dropped features, adding new features will change the shape space basis and the algorithm will converge to this new basis. In order to add the new feature variables to the reconstruction algorithm, the algorithm is reinitialised using information from the newly acquired feature tracks. Up to now, algorithm initialisation merely set all internal variables to 0. In order to integrate the current internal variables at any frame, with new values from the newly added features, a history of frames processed is kept and initialisation changed to bootstrapping.

4.2.3 Bootstrapping

In order to avoid the influence of newly acquired features on the existing shape space, they are considered only after being visible for a required minimum number of frames. This allows that more information on these features is available before they are first added to the algorithm variables. Thus, their influence on the other variables can be computed more accurately.

This argument leads to the implementation of history and bootstrapping within the algorithm. We maintain a small history of feature positions to allow the initialisation of newly acquired features. Note that one of the main advantages of the algorithm as presented in [44] is that the algorithm maintains tracking and reconstruction information for a single frame only. At each step of the algorithm the previous values of the internal variables are replaced. This allows the size of the variables to be independent of the number of frames processed.

By including the history of feature tracks, we introduce variables that are dependent on the number of frames in the history—but not the total number of frames processed by the algorithm. In practice, we have found that the history is typically between 15 and 50 frames in size. These values were found empirically and depend on the video data under consideration.

The history is used to bootstrap the system. This is done by performing a small batch reconstruction (see Section 3.3.1) using the frames in the history. This batch step allows accurate initial values for the sequential algorithm and fast convergence.

4.3 Discovery of Articulated Structure

The sequential algorithm described in the previous section is designed to track rigid objects. Since articulated structure consists of a number of rigid segments, these segments can be tracked individually using the sequential method. As with the articulated reconstruction of Section 4.1, the initial assignment of points to segments needs to be known. After the reconstruction of the rigid segments, it remains to discover the articulated connectivity of the rigid segments and the 3D locations of joints between segments.

Here, we design and implement an algorithm inspired by [56], where it is assumed that rigid reconstruction of articulated segments has been performed and that the *scaled prismatic* camera model is used. This camera model is essentially 2-dimensional and too restrictive for our purposes.

In [56] a measure of segment connectivity based on mutual information between segment motions is described. This measure is calculated for the motions between each pair of segments and a minimum spanning tree algorithm [14] is used to find the articulated tree that best spans the graph with the mutual information values as edge weights.

We describe an algorithm that uses a different segment connectivity measure with the minimum spanning tree algorithm to determine articulated topology. The joint locations are discovered during the calculation of the segment connectivity measure.

Note that the motion of a child segment relative to its parent segment is always a pure rotation. This follows from the definition of articulated motion. A child segment undergoes rotation around its joint location. This joint location moves with the parent segment—but when considering motion relative to the parent segment, the joint location is stationary.

This means that translation of a segment can be described entirely in terms of translation of the parent segment and rotation of the joint location. That is,

$$\mathbf{t}_{\text{child}} = \mathbf{t}_{\text{parent}} + \mathbf{R}_{\text{parent}} \times \mathbf{O}. \tag{4.33}$$

Here, $\mathbf{t}$ is the translation component and $\mathbf{R}$ the rotation component of motion. $\mathbf{O}$ is the origin, or joint location, of the child segment in the coordinate system of its parent segment. The value of $\mathbf{O}$ is frame-independent, while the motion parameters are not. By rewriting (4.33) as

$$\mathbf{R}_{\text{parent}} \times \mathbf{O} = \mathbf{t}_{\text{child}} - \mathbf{t}_{\text{parent}} \tag{4.34}$$

and considering all frames of motion, we have an over-constrained set of equations in the 3 unknowns of $\mathbf{O}$. By solving for these unknowns, we find the joint location of a segment in its parent's coordinate system.

Note that we have not yet determined whether the two motions under consideration are actually that of parent and child segments. If they were not, we would get an inaccurate solution to (4.34) in the sense that

$$\left\| \mathbf{R}_{\text{parent}} \times \mathbf{O} - \mathbf{t}_{\text{child}} + \mathbf{t}_{\text{parent}} \right\|_2 \tag{4.35}$$

is large. This provides us with a method to find the articulated topology of the motion under consideration while computing the joint locations. We compute the joint location between each pair of motions using (4.34) and store the norm in (4.35) in a symmetric matrix. Let this matrix describe the edge weights of an undirected graph that indicates the error in the assumption that the relative motions between two segments is a rotation only. Small values indicate that two segments are closely correlated and large values that they are not.

We apply a minimum spanning tree algorithm to the graph thus defined to find the articulated topology of the motion under consideration.

4.4 Summary

In this chapter, we described some work on articulated structure from motion recovery that was not found in the literature. This original work included

- a derivation of an upper bound on the rank of the observation matrix for the cases of articulated trees and articulated forests;
- modifications to the sequential reconstruction of rigid structure and motion, as described by [44]; and
- the discovery of articulated topology once the motions of underlying rigid segments are known.

The derivation of the rank upper bound led to a matrix formulation for articulated structure from motion. This formulation describes how joint locations and observed motion are related and led to the development of the articulated topology recovery algorithm. By integrating a sequential reconstruction algorithm for rigid objects under weak perspectivity and articulated topology recovery, a complete articulated structure from motion solution was described.

Chapter 5

Experiments

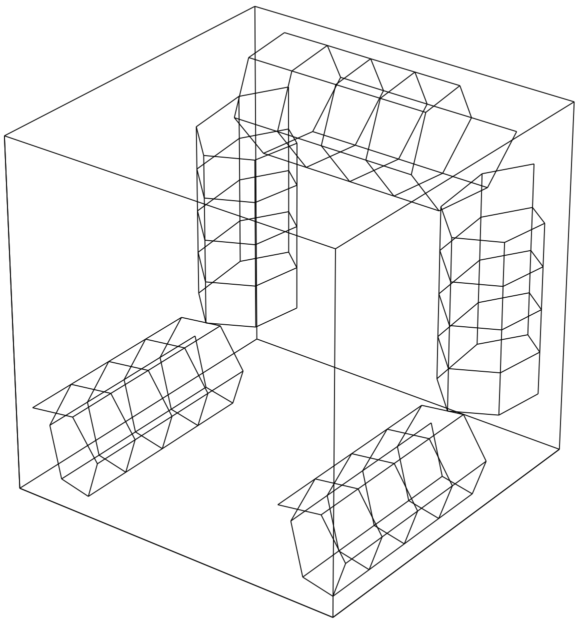
In this chapter, we present experiments that demonstrate the performance and weaknesses of the various algorithms discussed in previous chapters. The experiments include the following—

- batch reconstruction of numerically generated articulated objects, with the focus on topology reconstruction;
- sequential reconstruction of numerical rigid objects to test the accuracy of reconstruction under occlusion and disocclusion;

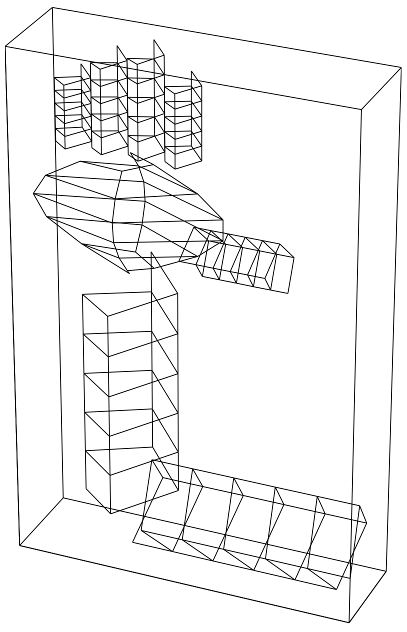
In the numerical experiments, controlled amounts of noise are added to the data and the accuracy of the reconstructed objects recorded. These experiments test the feasibility of using the reconstruction methods on real data.

5.1 Articulated Topology of Numerical Objects

In [58] it was demonstrated how batch reconstruction of rigid objects can be performed by matrix factorisation. In this first experiment, we reconstruct rigid segments using the method of [58] adapted for the weak perspective camera model. We then test the articulated topology reconstruction of Section 4.1.2 based on these reconstructions. We test the accuracy of the topology reconstruction under noisy conditions by adding Gaussian white noise up to typically high levels—with a standard deviation of 1 [15, 16] to 2 [44, 46] pixels on a 512×512 image.



(a)



(b)

Figure 5.1: The numerical datasets approximating (a) shoulders and arms, and (b) an arm and a hand.

5.1.1 Data Sets

Two numerical datasets were generated—

- the first loosely approximates shoulders and arms (Figure 5.1a); and
- the second approximates an arm and a hand (Figure 5.1b).

Note that the first dataset is a linear tree, i.e. each segment has exactly two segments connected to it, excepts for the segments at the ends, which have one segment connected to them. The second dataset contains a branch point at the hand, which has 6 segments connected to it.

The first dataset consists of 5 segments with 30 vertices per segment and the second of 8 segments with 24 vertices per segment. Both datasets span 150 frames. Normally distributed noise with zero mean and standard deviation ranging from 0.1% to 10% of the image size is added to each vertex position. To smooth out the random effects of noise, the reconstruction experiment on each of these datasets is performed 100 times for each noise level and the results averaged.

5.1.2 Comparison Criteria

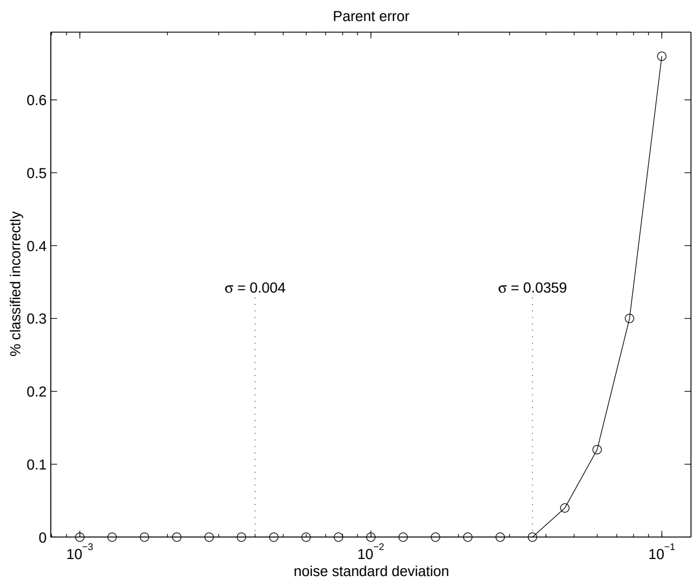
Since these datasets are wholly artificial, the structure of each limb and the topologies of the articulated objects are known explicitly. The percentage of the runs in which topology reconstruction is performed correctly is recorded.

5.1.3 Results

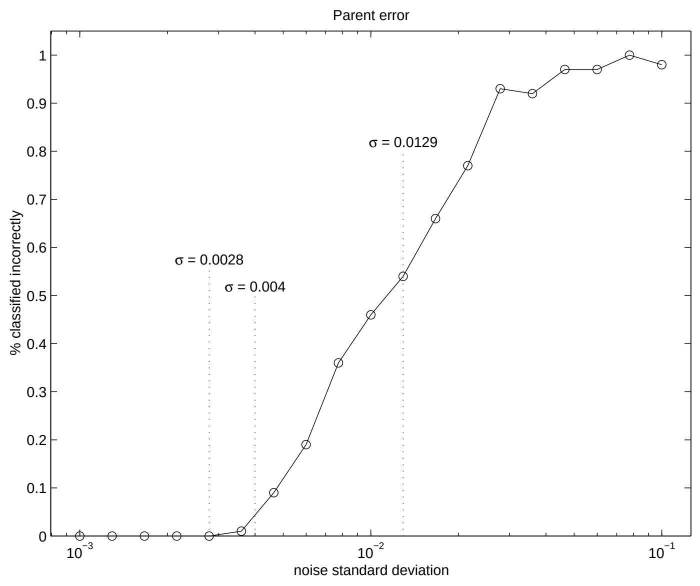
Figures 5.2a and 5.2b contain plots of the percentage of runs that are classified incorrectly for each level of noise in the shoulders-and-arms and arm-and-hand experiments respectively.

5.1.4 Discussion

The error graph for the shoulders-and-arms experiment shows zero error up to $\sigma = 3.59\%$, indicated on Figure 5.2a. The maximum noise level of $\sigma = 0.4\%$, discussed in [44, 46], is also indicated on the plot. The topology reconstruction succeeds well beyond this noise level. Figures 5.3a–d explain this using the segment interaction matrices (see Section 4.1.2) for various noise levels. Note the sharp contrast between segments that interact and those

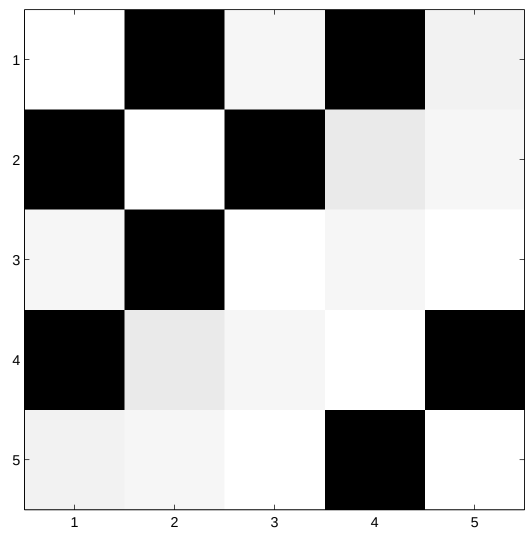


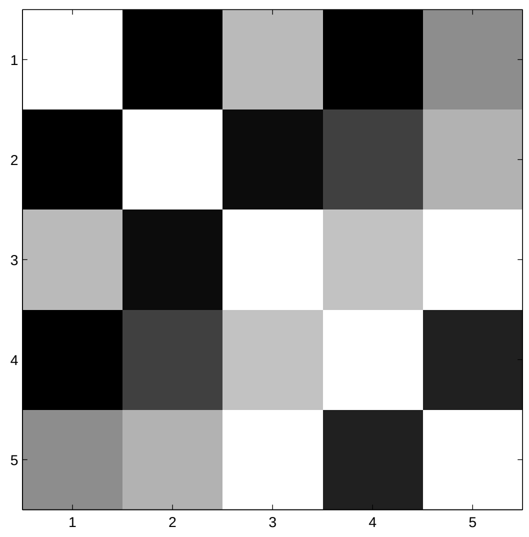
(a)

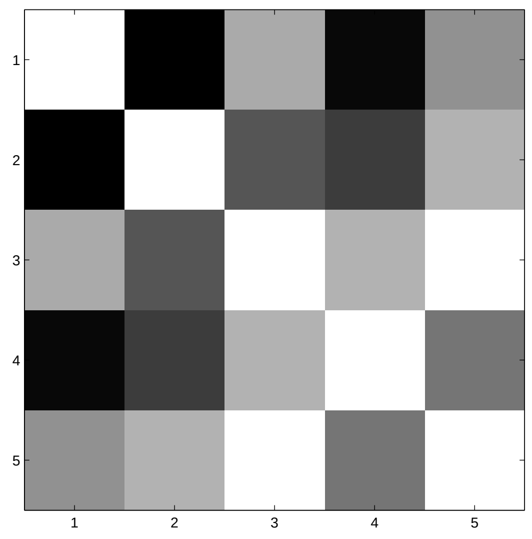


(b)

Figure 5.2: The percentage of reconstruction runs for which the articulated tree topology is reconstructed incorrectly as a function of the white noise variance. Figure (a) results from the shoulders-and-arms dataset and (b) from the arm-and-hand dataset.


(a)


(b)


(c)

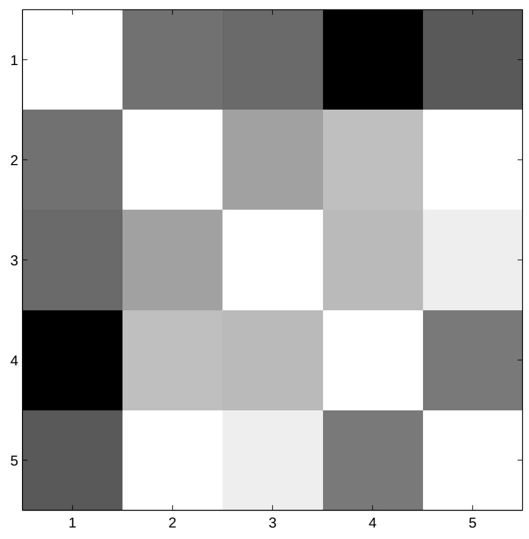
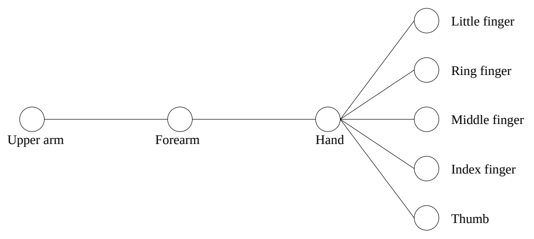
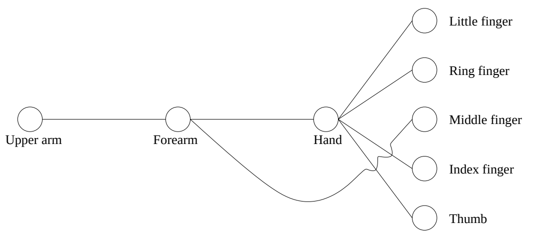

(d)

Figure 5.3: Segment interaction matrices for the shoulders-and-arms dataset. The segments—shoulders, left upper arm, left lower arm, right upper arm, right lower arm—are labelled in order from 1 to 5. Light cells indicate a weak interaction and dark cells a strong interaction. The interaction matrices are for σ (a) 0%, (b) 0.36%, (c) 3.59%, and (d) 10% of the input image size.

that do not for $\sigma = 0\%$. At $\sigma = 3.59\%$ the contrast is less obvious to the human eye but topology reconstruction is still performed without error. At $\sigma = 10\%$, it is impossible to distinguish interacting from non-interacting segments.

At first glance, the performance of the arm-and-hand experiment is much worse than that of the shoulders-and-arms experiment. Analysis of which segments are classified incorrectly shows that all of the error up to $\sigma = 1.29\%$ (indicated on the figure) is caused by a single misclassification. Rather than reconstructing the tree correctly as in the figure to the left below, it is reconstructed as in the figure to the right.





This indicates that the topology reconstruction algorithm finds a strong correlation between the motions of the *middle finger* segment and the *forearm* segment.

In conclusion, the articulated topology reconstruction algorithm performs well under noisy conditions. In particular, based on noise tolerance, the algorithm compares well with the work in [15, 16, 44, 46].

5.2 Sequential Reconstruction of Rigid Numerical Objects

We adapted (Section 4.2) and implemented the sequential rigid reconstruction method of [44] to allow for

- the weak perspective camera model; and
- occlusion and disocclusion of features.

We test the accuracy of this implementation under Gaussian white noise with the same standard deviations as in the previous experiment.

The implementation has one user-defined parameter, the number of bootstrap frames, as introduced in Section 4.2.3. This experiment adds one more parameter—the amount of occlusion in the data. Occlusion and disocclusion are simulated by discarding the observation matrix values for those vertices that are more than a certain depth from the camera centre of projection. This cutoff depth is between the maximum and mean depths of the object vertices and is determined by

$$c \operatorname{avg} z_i + (1 - c) \max z_i \tag{5.1}$$

with $0 \leq c \leq 1$ as the parameter of the experiment. Experiments are performed for various values of these two parameters in order to measure their effects.

5.2.1 Data Sets

Numerical datasets based on scanned data of a mannequin face[1] are used. The scanned data contains the 3D coordinates of 6304 points on the face but the number of points is reduced for the purposes of the experiments—see Figure 5.4. Smooth rigid 3D motion is generated randomly and the weak perspective camera model used to form the unoccluded observation matrix.

All datasets span 200 frames and normally distributed noise with zero mean and standard deviation ranging from 0.1% to 10% of the image size is added to each vertex position. To smooth out the random effects of noise, the reconstruction experiment on each of these datasets is performed 50 times for each noise level and the results averaged.

5.2.2 Comparison Criteria

The 3D structure of the face is known explicitly and the reconstructed object is compared to this using the method described below. Furthermore, the errors in the occluded and unoccluded parts of the reconstructed object are recorded separately to determine the effect of occlusion.

Calculating the error The following algorithm is used to compute the distance between two objects and aims to compensate for differences in origin, rotation and scale between the objects.

1. Place the origin of each object at its centre of mass.
2. Project objects onto the unit sphere centred at the origin. This prevents points far from the origin (i.e. position vectors with large Euclidean norm) from dominating subsequent calculations.

[1]This data was digitised at the Department of Mechanical Engineering, University of Stellenbosch.

3. Rotate and reflect one object to minimise the sum of the distances between the vertices of the two objects.
4. The error is calculated as the mean of the Euclidean distances between the resulting points on the two objects.

5.2.3 Results

6 experiments are performed and the bootstrap and occlusion parameters varied. The experiments are as follows:

A. Occlusion and disocclusion, $c = 0.5$ (493 vertices), 50 bootstrap frames;

B. Occlusion only, $c = 0.5$ (419 vertices), 50 bootstrap frames;

C. Occlusion and disocclusion, $c = 0.25$ (491 vertices), 50 bootstrap frames;

D. Occlusion and disocclusion, $c = 0.75$ (491 vertices), 50 bootstrap frames;

E. Occlusion and disocclusion, $c = 0.5$ (493 vertices), 25 bootstrap frames;

F. Occlusion and disocclusion, $c = 0.5$ (493 vertices), 75 bootstrap frames;

Note that the number of vertices changes slightly with c, as some vertices are occluded in the whole image sequence and, hence, discarded.

Figure 5.5 shows the occlusion matrices for the 6 experiments. 3 of these images are identical as they have the same occlusion parameter, $c = 0.5$, and model both occlusion and disocclusion. Note that experiment B models occlusion only, so once a vertex has been occluded, it is never recovered (see Figure 5.5b).

Figure 5.6 shows the output from the reconstruction algorithm when run on experiment A and with no noise. This demonstrates that sequential reconstruction under occlusion and disocclusion can be performed in principle.

Figures 5.7–5.9 show the reconstruction errors for the 6 experiments. The values used in the plots are calculated using the method described in Section 5.2.2. Three error graphs are plotted on each set of axes—these are the reconstruction errors in

- the vertices that are never occluded;
- the vertices that were occluded at least once; and
- the whole shape.

5.2.4 Discussion

We compare the error plots of Figures 5.7–5.9 and discuss the behaviour of the reconstruction algorithm for various parameters. In short, the plots show that the reconstruction algorithm handles occlusion well but that disocclusion of features degrades accuracy. Reconstruction accuracy also degrades when more vertices of the object are occluded but improves when more bootstrap frames are used.

Occlusion only (Figure 5.7b) shows the expected error behaviour, similar to what was seen in the previous experiment (Section 5.1). When disocclusion is introduced, the reconstruction of vertices that were occluded at least once is worse than expected. In Figure 5.7a, we see that the error for occluded vertices increases much faster than the error for visible vertices.

To analyse this in more detail, we plot the errors from each of 50 the iterations per noise level of the experiments in Figures 5.10–5.11. Each horizontal line represents one iteration and the median of these values are used to plot the error graph. For experiment A (occlusion and disocclusion), note that the errors in the reconstruction of the occluded part (Figure 5.10a) start varying much more and much earlier than those of the visible part (Figure 5.10b). For experiment B (no disocclusion), large errors in the visible and occluded parts appear at approximately the same noise level (Figure 5.11).

To explore further, we analyse the convergence of $\overline{\mathbf{Q}}$. Since $\overline{\mathbf{Q}}$ is supposed to converge to a basis of the shape space, we define the error in $\overline{\mathbf{Q}}$ as the distance between the space spanned by it and the shape space,

$$\left\|\overline{\mathbf{Q}}\,\overline{\mathbf{Q}}^T - \mathbf{S}_{\text{basis}}\mathbf{S}_{\text{basis}}^T\right\|_2 .$$

Figure 5.12 shows the error in $\overline{\mathbf{Q}}$ for the experiments with and without disocclusion. The jumps in the error are caused by updates to the shape space, i.e. whenever vertices are removed from (occlusion) or added to (disocclusion) the reconstruction. In both cases, the error fluctuates around the same level after recovering from jumps caused by occlusion or disocclusion. This excludes the possibility that the degraded performance under disocclusion is caused by failure to converge to the correct basis for the shape space. It is still uncertain what causes this behaviour.

The effects of the occlusion and bootstrap parameters are as expected. Comparing Figures 5.8a and 5.8b shows that more occlusion (larger c) causes larger errors; while comparing Figures 5.9a and 5.9b shows that using more bootstrap frames reduces the error—particularly the error in the reconstruction of the occluded part of the object.

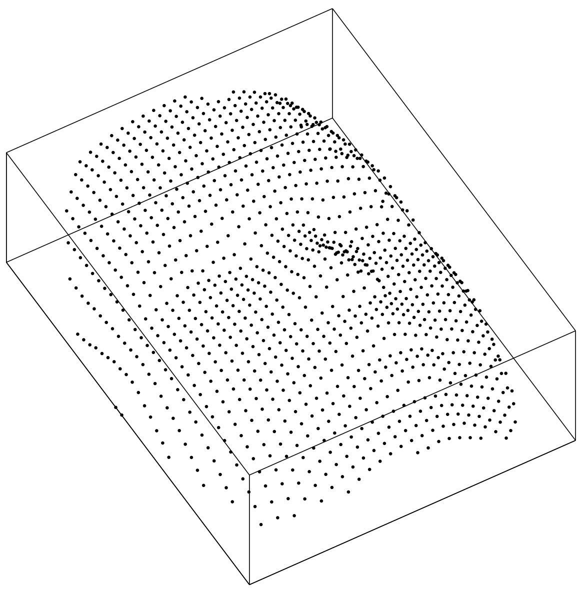
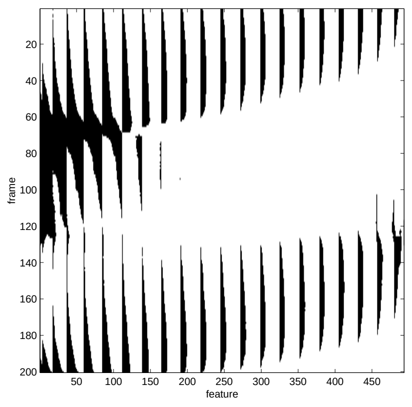
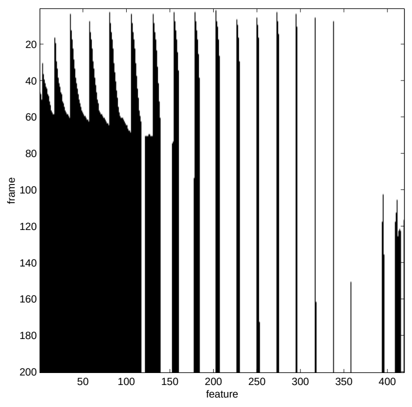


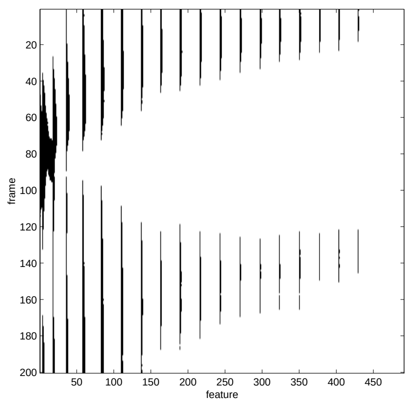
Figure 5.4: The mannequin face with 1099 vertices.



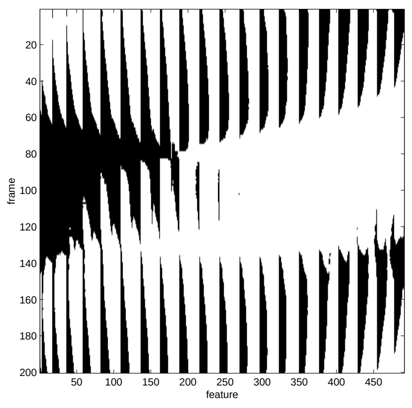
(a)



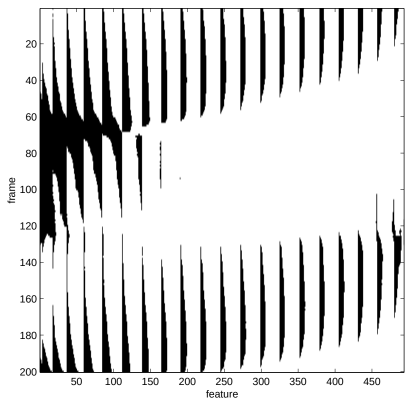
(b)



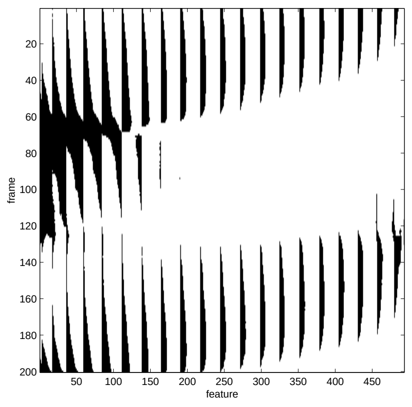
(c)



(d)



(e)



(f)

Figure 5.5: The occlusion matrices for the 6 sequential rigid reconstruction experiments. The black areas show where vertices are occluded, with the vertical axes representing frame and the horizontal representing vertex number.

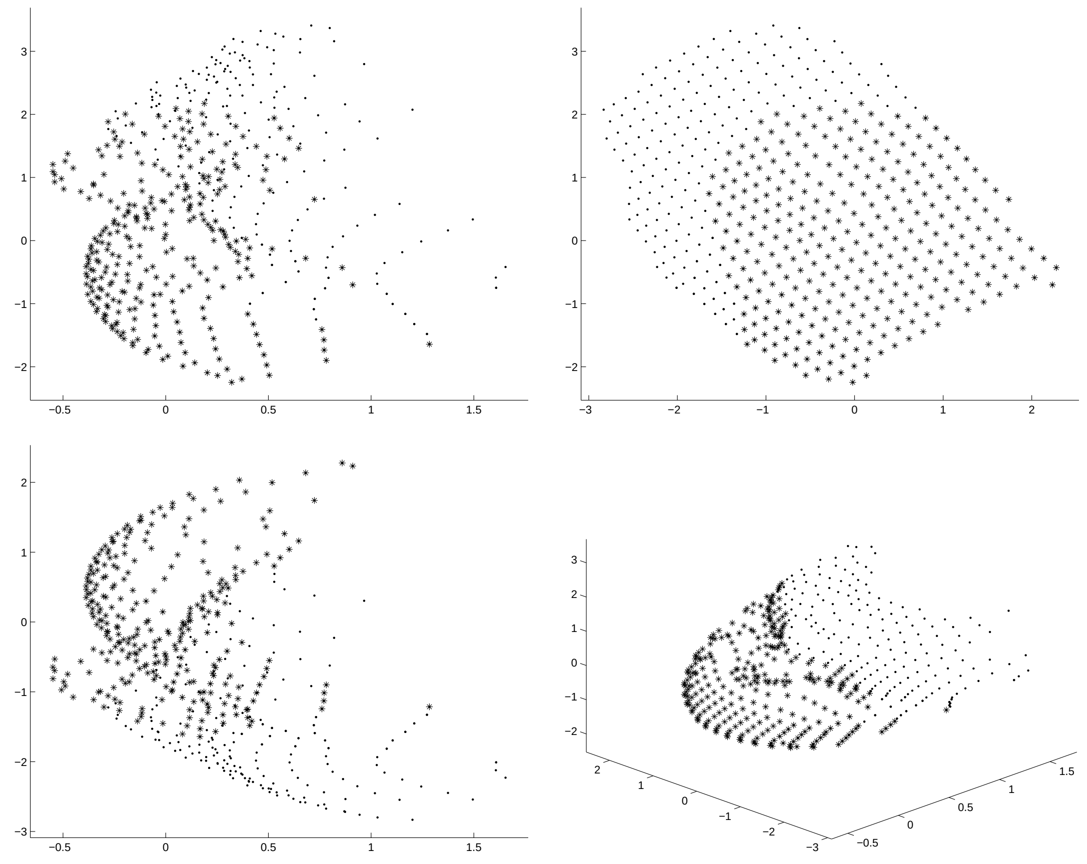
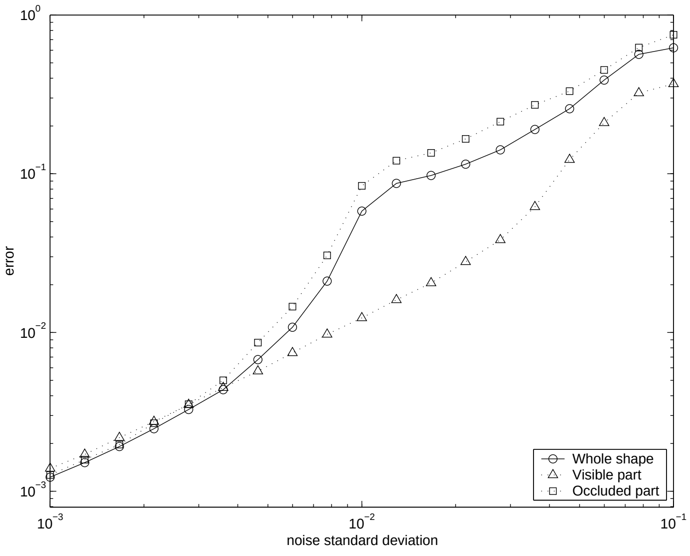
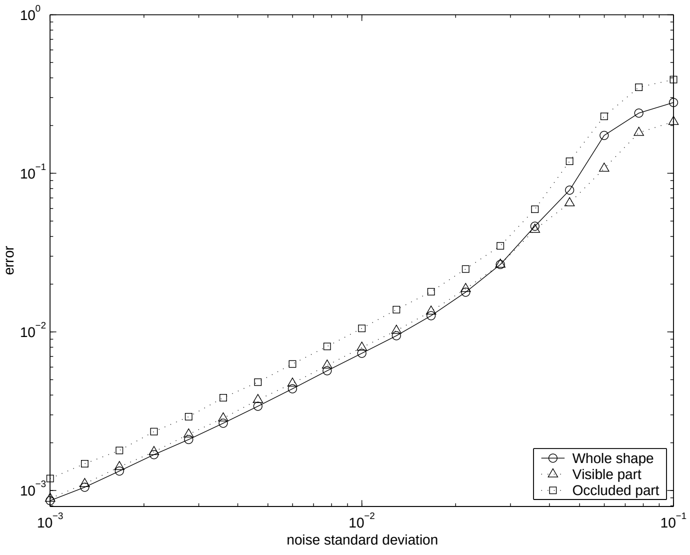


Figure 5.6: The output of the reconstruction algorithm for experiment A with no noise. The bottom right image shows a 3D plot of the reconstructed object and the remaining images, an orthographic projection thereof. The stars ($*$) represent vertices that were never occluded in the input image sequence. The dots ($\cdot$) represent points that were occluded at least once.

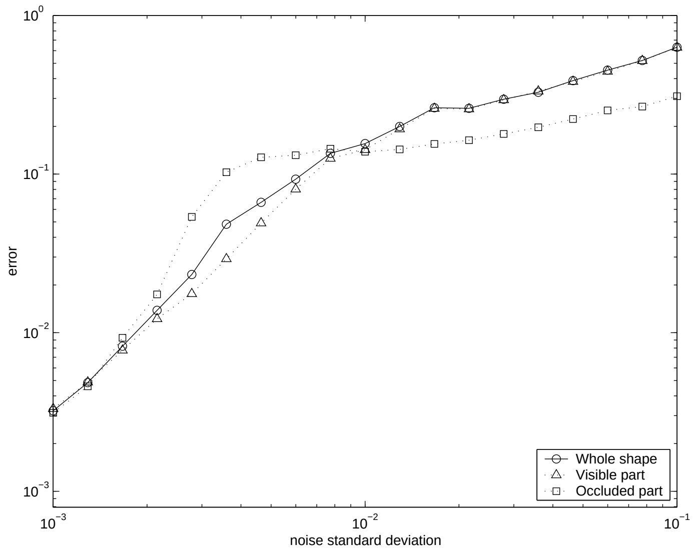


(a)

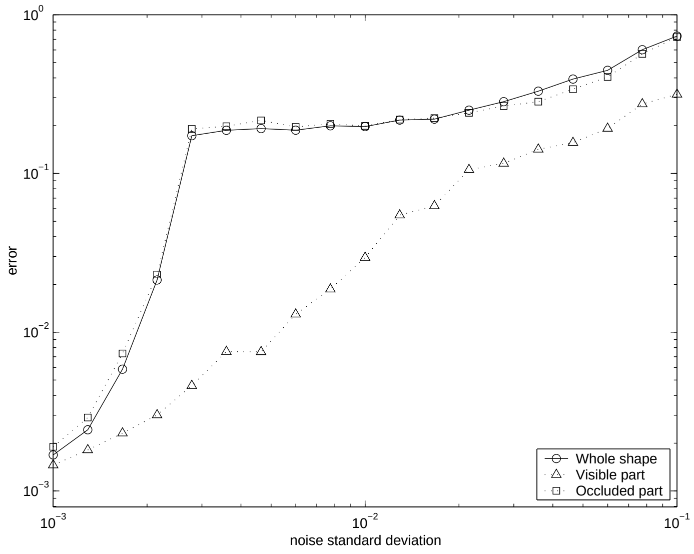


(b)

Figure 5.7: The reconstruction errors for (a) experiment A with occlusion and disocclusion, $c = 0.5$ and 50 bootstrap frames; and (b) experiment B with occlusion only, $c = 0.5$ and 50 bootstrap frames.

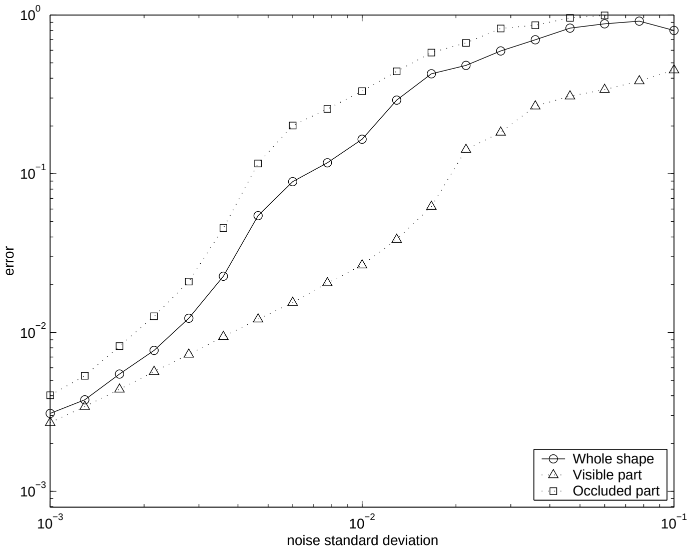


(a)

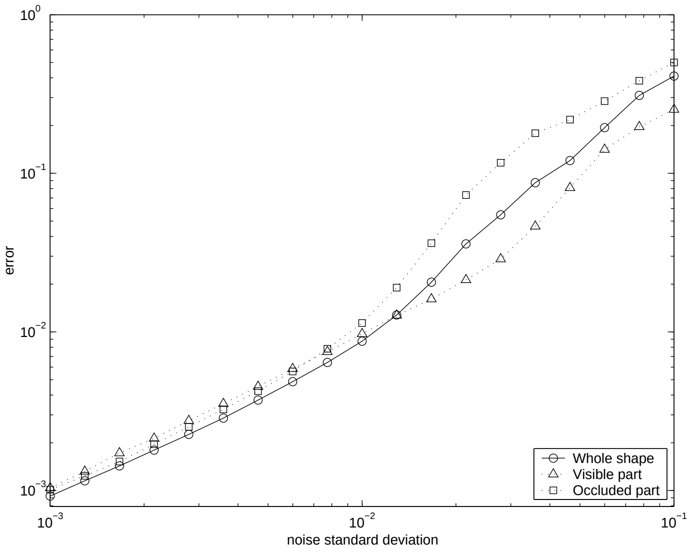


(b)

Figure 5.8: The reconstruction errors for (a) experiment C with occlusion and disocclusion, $c = 0.25$ and 50 bootstrap frames; and (b) experiment D with occlusion and disocclusion, $c = 0.75$ and 50 bootstrap frames.

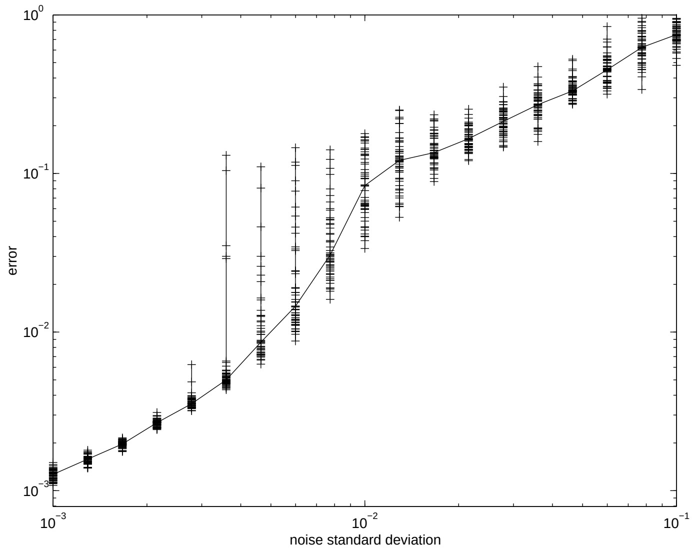


(a)

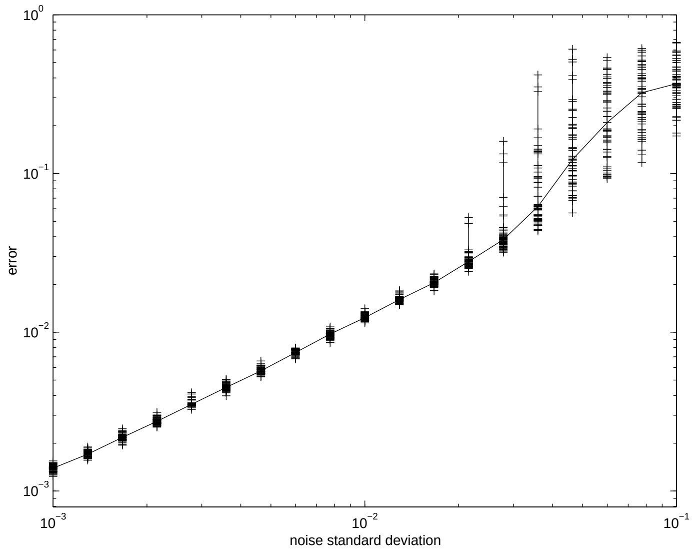


(b)

Figure 5.9: The reconstruction errors for (a) experiment E with occlusion and disocclusion, $c = 0.5$ and 25 bootstrap frames; and (b) experiment F with occlusion and disocclusion, $c = 0.5$ and 75 bootstrap frames.

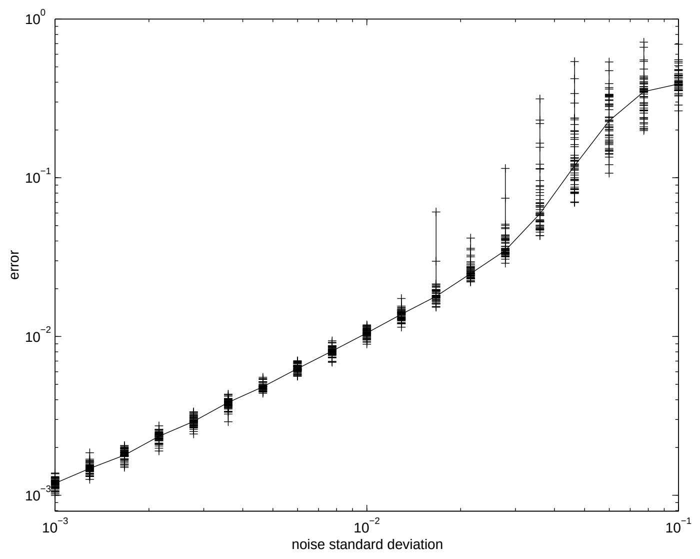


(a)

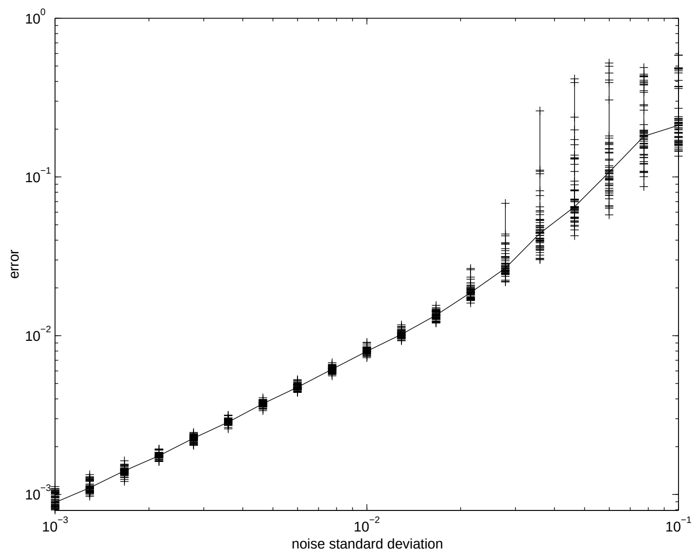


(b)

Figure 5.10: The reconstruction errors for each of the 50 runs of experiment A (occlusion and disocclusion): (a) shows the errors for vertices that were occluded at least once; and (b) shows the errors for vertices that were visible during the entire image sequence.

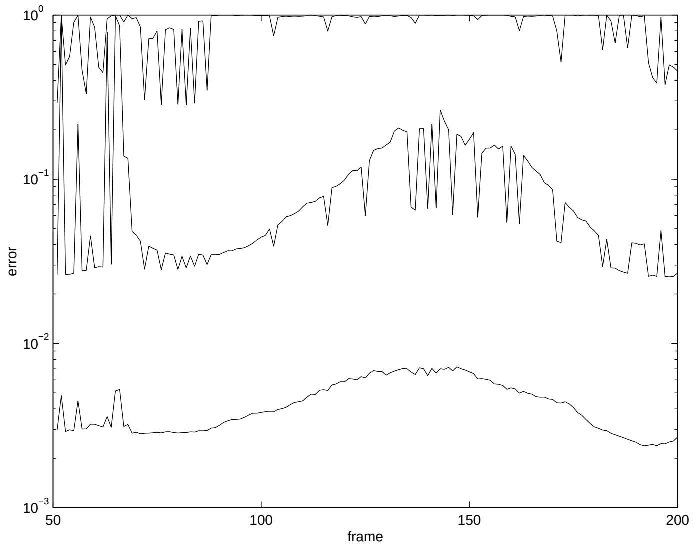


(a)

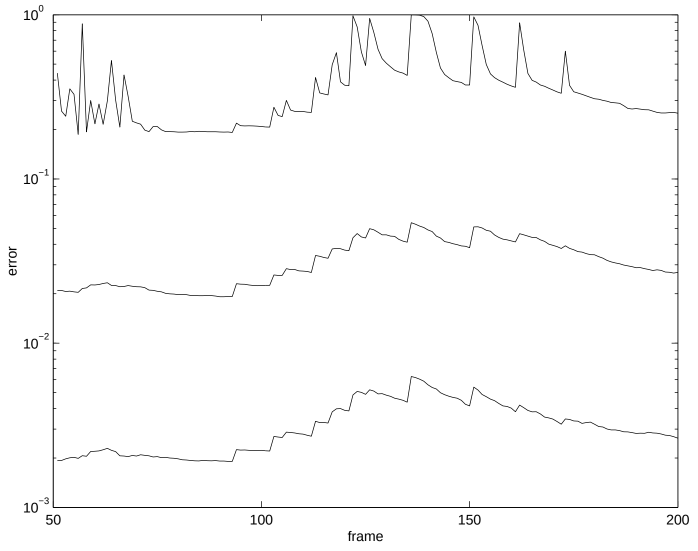


(b)

Figure 5.11: The reconstruction errors for each of the 50 runs of experiment B (occlusion only): (a) shows the errors for vertices that were occluded at least once; and (b) shows the errors for vertices that were visible during the entire image sequence.



(a)



(b)

Figure 5.12: The distance between span($\overline{\mathbf{Q}}$) and the shape space for (a) experiment A and (b) experiment B. The 3 lines on each graph are for the noise levels $\sigma = 10^{-3}, 10^{-2}, 10^{-1}$.

Chapter 6

Conclusion

In this thesis, we considered the question *Is model-free articulated structure from motion possible?* We provided a short review of existing methods for rigid and non-rigid structure from motion and it was shown that articulated structure from motion has focused mostly on model-based methods.

We explored model-free approaches to articulated structure from motion and discovered a matrix formulation under the assumption that the feature to segment assignments are known. It was shown that the observation matrix has a rank constraint analogous to those found for rigid, multiple independent and elastic objects. It was demonstrated that this formulation can be used to reconstruct articulated topology once reconstruction of individual segments has been performed.

We also integrated this approach with the sequential factorisation method of [44] for a sequential articulated reconstruction algorithm. The adapted method was extended to handle the occlusion and disocclusion of features using an observation matrix history and bootstrapping. Experiments showed that the matrix formulation is theoretically sound since all reconstruction was performed correctly under noise-free conditions. Error comparisons showed that noise affects the reconstruction accuracy as expected, except for worse than expected degradation when disocclusion of features occurs.

6.1 Future Work

The major disadvantage of the matrix approach to articulated structure from motion is that it requires known vertex to segment assignments. Even though it is analogous to the vertex to object assignment used in reconstruction of multiple independent rigid objects [15, 16], it cannot be computed in the same way since articulated segments do *not* move independently. One idea to be explored is whether image segmentation techniques [8, 23, 24, 34, 60, 65] can by used to perform this assignment automatically. Image segmentation attempts to separate parts of a moving image directly from the observed 2D pixel data.

In the sequential reconstruction of rigid segments, better analysis of how disocclusion affects reconstruction error is still needed. It was observed that the error increases faster when disocclusion of features is modelled. It was shown that the accuracy of the shape space basis is *not* greatly affected by disocclusion. It is uncertain what causes the large errors.

Finally, serious problems with tracking errors in real data make reconstruction difficult. It has been shown that the reconstruction algorithms work in principle and on numerical data with simulated noise. Since we are considering model-free solutions, no assumptions are made on the type of articulated motion observed—particularly, we do not assume that it is human motion.

Model-based tracking methods would improve the accuracy of tracked features and make reconstruction more successful. One argument against this is that using a predefined model of human motion to perform tracking defeats the purpose of exploring model-free reconstruction. There are however methods that use models to perform tracking but learn these models rather than requiring them as part of the algorithm input. The methods in [28, 29] are particularly relevant.

6.1.1 Application

Human motion capture was considered as the main application of articulated structure from motion. It was mentioned in Chapter 2 that motion capture of human gestures—in particular, the articulated motion of hands and arms—plays an important role in automatic Sign Language understanding.

One of the possible future applications of this project is the capturing for later analysis of these human gestures without equipment such as data gloves or markers that are commonly used in motion capture. This makes motion capture non-intrusive, inexpensive and portable, i.e. it can be done anywhere using a digital video camera. Realising a fully automatic reconstruction algorithm will make vocabulary acquistion for Sign Language possible.

Appendix A

Rigid Reconstruction Under Weak Perspectivity

Given the $2F \times P$ observation matrix, $\mathbf{W}$, with rank $r = 4$, we decompose using the SVD—

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{A.1}$$

with $\mathbf{U}_{2F\times 4}$, $\boldsymbol{\Sigma}_{4\times 4}$ and $\mathbf{V}_{P\times 4}$. Note that, because of measurement noise and numerical imprecision, the rank of $\mathbf{W}$ will in general be greater than 4. However, a property of the SVD is that it maximises the amount of variance in the data described by the first singular values and vectors of $\mathbf{W}$. This property allows us to denoise and factorise $\mathbf{W}$ simultaneously by discarding any singular values and vectors after the first 4.

Next, from

$$\hat{\mathbf{M}} = \mathbf{U}\boldsymbol{\Sigma}^{\frac{1}{2}} \quad \text{and} \quad \hat{\mathbf{S}} = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{V}^T. \tag{A.2}$$

Note that

$$\mathbf{W} = (\hat{\mathbf{M}}\mathbf{A})(\mathbf{A}^{-1}\hat{\mathbf{S}}) \tag{A.3}$$

for any invertible 4×4 matrix $\mathbf{A}$. Hence, $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ are the required motion and shape matrices up to an affine transformation, so that

$$\mathbf{M} = \hat{\mathbf{M}}\mathbf{A} \quad \text{and} \quad \mathbf{S} = \mathbf{A}^{-1}\hat{\mathbf{S}}. \tag{A.4}$$

We can solve for $\mathbf{A}$ by using the rotation and translation constraints on $\mathbf{M}$—[15, 16, 32].

For the weak perspective camera model,

$$\mathbf{M} = \begin{bmatrix} l_1\mathbf{M}_1 \\ l_2\mathbf{M}_2 \\ \vdots \\ l_F\mathbf{M}_F \end{bmatrix} \quad \text{with} \quad l_f\mathbf{M}_f = l_f\left[\,\mathbf{R}_f \mid \mathbf{t}_f\,\right], \tag{A.5}$$

where each $\mathbf{R}_f$ contains the first two rows of the camera rotation matrix and $\mathbf{t}_f$ the camera translation at frame f. From this, we note that $\mathbf{M}$ consists of two parts: the first 3 columns, which contain rotation components only and the final column, which contains the translation components. By writing

$$\mathbf{A} = [\,\mathbf{A}_R \mid \mathbf{a}_t\,] \tag{A.6}$$

where $\mathbf{A}_R$ is the first 4×3 sub-matrix and $\mathbf{a}_t$ the final column of $\mathbf{A}$, we obtain from (A.4)

$$\mathbf{M} = \hat{\mathbf{M}}\mathbf{A} = \left[\hat{\mathbf{M}}\mathbf{A}_R \mid \hat{\mathbf{M}}\mathbf{a}_t\right]. \tag{A.7}$$

From the orthogonality of rotation matrices, we have

$$\mathbf{R}_f\mathbf{R}_f^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{A.8}$$

Combining (A.5), (A.7) and (A.8) yields

$$\begin{array}{rcll} \hat{\mathbf{m}}_{2f-1}^T\mathbf{A}_R\mathbf{A}_R^T\hat{\mathbf{m}}_{2f-1} & = & l_f^2 & \forall\, f = 1, 2, \ldots, F \\ \hat{\mathbf{m}}_{2f}^T\mathbf{A}_R\mathbf{A}_R^T\hat{\mathbf{m}}_{2f} & = & l_f^2 & \forall\, f = 1, 2, \ldots, F \\ \hat{\mathbf{m}}_{2f-1}^T\mathbf{A}_R\mathbf{A}_R^T\hat{\mathbf{m}}_{2f} & = & 0 & \forall\, f = 1, 2, \ldots, F \end{array} \tag{A.9}$$

where $\hat{\mathbf{m}}_f^T$ is row f from matrix $\hat{\mathbf{M}}$. Note that the l_f are not yet known. To remove these unknowns, we subtract the second equation in (A.9) from the first to form

$$\hat{\mathbf{m}}_{2f-1}^T\mathbf{A}_R\mathbf{A}_R^T\hat{\mathbf{m}}_{2f-1} - \hat{\mathbf{m}}_{2f}^T\mathbf{A}_R\mathbf{A}_R^T\hat{\mathbf{m}}_{2f} = 0 \qquad \forall\, f = 1, 2, \ldots, F. \tag{A.10}$$

The over-constrained system of (A.9) and (A.10) is used to solve for the entries of the 4×4 matrix $\mathbf{A}_R\mathbf{A}_R^T$. The system is homogeneous, which implies that there is either an infinite number of solutions or the trivial solution only. To make the system non-homogeneous, we choose $l_1 = 1$,[1] which yields one additional equation since one of the equations in (A.10) is discarded, but two are added—one from each of the first two equations in (A.9).

[1] Since the ratio between the focal length and the z-coordinate of the centroid determines the constant factor of the weak perspective projection, these two parameters are undetermined up to a constant factor. We make the arbitrary selection $l_1 = 1$, which is equivalent to placing the centroid of the object on the image plane in frame 1.

To find $\mathbf{A}_R$ from $\mathbf{A}_R\mathbf{A}_R^T$, we again employ the SVD. Note that $\mathbf{A}_R\mathbf{A}_R^T$ has rank 3 since $\mathbf{A}_R$ is 4×3. Thus

$$\mathbf{A}_R\mathbf{A}_R^T = \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^T, \tag{A.11}$$

where $\mathbf{U}_A$ is 4×3, $\mathbf{V}_A = \mathbf{U}_A$ since $\mathbf{A}_R\mathbf{A}_R^T$ is symmetrical and $\mathbf{\Sigma}_A$ is 3×3. This yields

$$\mathbf{A}_R = \mathbf{U}_A\mathbf{\Sigma}_A^{\frac{1}{2}}.\ ^2 \tag{A.12}$$

Next, we find $\mathbf{a}_t$. Under the weak perspective camera model, the projection of the object centroid equals the centroid of the projected object, i.e. with $\mathbf{S}$ as in (3.25)

$$\begin{aligned} \mathbf{M}\left(\frac{1}{P}\sum_{i=1}^{P}\mathbf{p}_i\right) &= \frac{1}{P}\sum_{i=1}^{P}\mathbf{w}_i && \text{(A.13)} \\ \Leftrightarrow \qquad \mathbf{M}\overline{\mathbf{p}} &= \overline{\mathbf{w}} && \text{(A.14)} \end{aligned}$$

where the $\mathbf{w}_i$ are the columns of $\mathbf{W}$. Since $\overline{\mathbf{p}}$ is a homogeneous vector, it has 3 free parameters and since the camera coordinate system has arbitrary origin, we can choose

$$\overline{\mathbf{p}} = [0 \ \ 0 \ \ 0 \ \ 1]^T, \tag{A.15}$$

which corresponds to centering the object around the camera origin. From (A.7), (A.14) and (A.15) it follows that

$$\hat{\mathbf{M}}\mathbf{a}_t = \overline{\mathbf{w}}. \tag{A.16}$$

This is an over-constrained system from which we solve for $\mathbf{a}_t$. This completes the computation of $\mathbf{A}$ and consequently $\mathbf{M}$ and $\mathbf{S}$.

[2]As in (A.3), $\mathbf{A}_R$ is defined only up to an affine transformation. However, since $\mathbf{A}_R\mathbf{A}_R^T$ is symmetrical, the affine transformation is orthogonal, which implies that it has only 3 degrees of freedom. These are exactly the freedom in choice of the initial rotation of the camera axes. We simply choose arbitrarily that the affine transformation is given by $\mathbf{I}$.

Bibliography

[1] J.K. Aggarwal, Q. Cai, W. Liao & B. Sabata, "Articulated and elastic non-rigid motion: A review", *IEEE Computer Society Workshop on Motion of Non-Rigid and Articulated Objects*, pp. 16–22, 1994.

[2] J. Alon & S. Sclaroff, "Recursive estimation of motion and planar structure", *IEEE Conference on Computer Vision and Pattern Recognition*, 2:550–556, 2000.

[3] A. Azarbayejani & A. Pentland, "Recursive estimation of motion, structure, and focal length", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1994.

[4] J.L. Barron, S.S. Beauchemin & D.J. Fleet, "On optical flow", *International Conference on Artificial Intelligence and Information-Control Systems of Robots*, pp. 3–14, 1994.

[5] P.A. Beardsley, A. Zisserman & D.W. Murray, "Sequential updating of projective and affine structure from motion", *International Journal of Computer Vision*, 23(3):235–259, 1997.

[6] S. Birchfield, "Derivation of Kanade-Lucas-Tomasi tracking equation", unpublished notes, 1997.

[7] S. Birchfield, *An implementation of the Kanade-Lucas-Tomasi feature tracker*, 1998. From: http://vision.stanford.edu/˜birch/klt/, [2001-07-27].

[8] D.G. Borshukov, G. Bozdagi, Y. Altunbasak & A.M. Tekalp, "Motion segmentation by multi-stage affine classification", *IEEE Transactions on Image Processing*, 6:1591–1594, 1997.

[9] C. Bregler & J. Malik, "Video motion capture", Technical Report UCB//CSD-97-973, Computer Science Division, University of California, Berkeley, 1997.

[10] C. Bregler & J. Malik, "Tracking people with twists and exponential maps", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8–15, 1998.

[11] C. Bregler, A. Hertzmann & H. Biermann, "Recovering non-rigid 3D shape from image streams", *IEEE Conference on Computer Vision and Pattern Recognition*, 2:690–696, 2000.

[12] T.J. Broida, S. Chandrashekhar & R. Chellappa, "Recursive 3-D motion estimation from a monocular image sequence", *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656 1990.

[13] I. Cohen & L.D. Cohen, "A hybrid hyperquadric model for 2-D and 3-D data fitting", *IEEE International Conference on Pattern Recognition*, pp. 403–405, 1994.

[14] T.H. Cormen, C.E. Leiserson & R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1997.

[15] J. Costeira & T. Kanade, "A multi-body factorization method for motion analysis", *IEEE International Conference on Computer Vision*, pp. 1071–1076, 1995.

[16] J. Costeira & T. Kanade, "A multi-body factorization method for independently moving objects", *International Jounal of Computer Vision*, 29(3):159–179, 1998.

[17] D.E. DiFranco, T. Cham & J.M. Rehg, "Recovery of 3D articulated motion from 2D correspondences", Technical Report CRL-99-7, HP Labs, 1999.

[18] D. Gavrila & L. Davis, "Tracking of humans in action: a 3D model-based approach", *ARPA Image Understanding Workshop*, pp. 737–746, 1996.

[19] C.W. Gear, "Multibody grouping from motion images", *Internationl Journal of Computer Vision*, 29(2):133–150, 1998.

[20] G.H. Golub & C.F. van Loan, *Matrix computations*, 3rd edition, Johns Hopkins University Press, pp. 143–144, 210–221, 236–245, 251–252, 452–456, 1996.

[21] G. Greiner & K. Hormann, "Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines", In A. le Méhauté, C. Rabut & L.L. Schumaker (eds.), *Surface Fitting and Multiresolution Methods*, pp. 163–172, Vanderbilt University Press, 1997.

[22] M. Han & T. Kanade, "Perspective factorization methods for Euclidian reconstruction", Technical Report CMU-RI-TR-99-22, Robotics Institute, Carnegie Mellon University, 1999.

[23] N. Ichimura, "Motion segmentation based on factorization method and discriminant criterion", *IEEE International Conference on Computer Vision*, pp. 600–605, 1999.

[24] M. Irani, B. Rousso & S. Peleg, "Computing occluding and transparent motions", *International Journal of Computer Vision*, 12(1):5–16, 1994.

[25] M. Irani & P. Anandan, "All about direct methods", In W. Triggs, A. Zisserman & R. Szeliski (eds.), *Vision Algorithms: Theory and Practice*, Springer-Verlag, 1999.

[26] M. Irani, "Multi-frame optical flow estimation using subspace constraints", *IEEE International Conference on Computer Vision*, pp. 626–633, 1999.

[27] M. Irani & P. Anandan, "Factorization with uncertainty", *European Conference on Computer Vision*, pp. 539–553, 2000.

[28] M. Isard & A. Blake, "Contour tracking by stochastic propagation of conditional density", *European Conference on Computer Vision*, 1:343–356, 1996.

[29] M. Isard & A. Blake, "Condensation — conditional density propagation for visual tracking", *International Journal of Computer Vision*, 1998.

[30] T. Jebara, A. Azerbayejani & A. Pentland, "3D structure from 2D motion", *IEEE Signal Processing Magazine*, 16(3), 1999.

[31] I.A. Kakadiaris & D.N. Metaxas, "3D human body model acquisition from multiple views", *IEEE International Conference on Computer Vision*, pp. 618–623, 1995.

[32] T. Kanade & D.D. Morris, "Factorization methods for structure from motion", *Philosophical Transactions of the Royal Society of London, Series A*, 356(1740):1153–1173, 1998.

[33] K. Kanatani, "Motion segmentation by subspace separation and model selection", *IEEE International Conference on Computer Vision*, 2:586–591, 2001.

[34] K. Kanatani & C. Matsunaga, "Estimating the number of independent motions for multibody motion segmentation", *5th Asian Conference on Computer Vision*, 2002.

[35] M. Kass, A. Witkin & D. Terzopoulos, "Snakes: Active contour models", *International Journal of Computer Vision*, 1(4):312–331, 1987.

[36] M. Kemp, *Leonardo da Vinci—The marvellous works of nature and man*, J.M. Dent & Sons Ltd., 1989.

[37] N. Krahnstoever, M. Yeasin & R. Sharma, "Automatic acquisition and initialization of kinematic models", *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[38] N. Krahnstoever, M. Yeasin & R. Sharma, "Towards a unified framework for tracking and analysis of human motion", *IEEE Workshop on Detection and Recognition of Events in Video*, 2001.

[39] N. Krahnstoever, M. Yeasin & R. Sharma, "Automatic acquisition and initialization of articulated models", *Machine Vision and Applications*, 14(4):218–228, 2003.

[40] C. Liao & G. Medioni, "Surface approximation of a cloud of 3D points", *Graphical Models and Image Processing*, 57(1):67–74, 1995.

[41] F.G. Meyer, R.T. Constable, A.J. Sinusas & J.S. Duncan, "Dense nonrigid motion tracking from a sequence of velocity fields", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 839–844, 1996.

[42] T.B. Moeslund, "Summaries of 107 computer vision-based human motion capture papers", Technical Report LIA 99-01, University of Aalborg, 1999.

[43] T.B. Moeslund, "Computer vision-based human motion capture—A survey", Technical Report LIA 99-02, University of Aalborg, 1999.

[44] T. Morita & T. Kanade, "A sequential factorization method for recovering shape and motion from image streams", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):858–867, 1997.

[45] J. Oliensis, "A multi-frame structure from motion algorithm under perspective projection", Technical Report, NEC Research Institute, 1997.

[46] C.J. Poelman & T. Kanade, "A paraperspective factorization method for shape and motion recovery", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997.

[47] W.H. Press, S.A. Teukolsky, W.T. Vetterling & B.P. Flannery, *Numerical recipes in C: The art of scientific computing*, 2nd edition, Cambridge University Press, 1992.

[48] J.M. Rehg & T. Kanade, "Visual tracking of high DOF articulated structures: An application to human hand tracking", *European Conference on Computer Vision*, pp. 35–46, 1994.

[49] J.M. Rehg & T. Kanade, "Visual tracking of self-occluding articulated objects", Technical Report CMU-CS-94-224, School of Computer Science, Carnegie Mellon University, 1994.

[50] C. Scheffler, K.H. Scheffler & C.W. Omlin, "Articulated tree structure from motion—A matrix factorisation approach", *14th Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 137–142, 2003.

[51] S. Sclaroff & J. Isidoro, "Active blobs", *IEEE International Conference on Computer Vision*, pp. 1146–1153, 1998.

[52] S. Sclaroff & J. Alon, "Non-rigid shape from image streams", Technical Report 99-006, Department of Computer Science, Boston University, 1999.

[53] J. Shi & C. Tomasi, "Good features to track", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[54] S. Soatto, P. Perona, R. Frezza & G. Picci, "Recursive motion and structure estimation with complete error characterization", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 428–433, 1993.

[55] Y. Song, L. Goncalves, E. D. Bernardo & P. Perona, "Monocular perception of biological motion—Detection and labeling", *IEEE International Conference on Computer Vision*, pp. 805–812, 1999.

[56] L. Taycher, J.W. Fisher & T. Darrell, "Recovering articulated model topology from observed motion", *Advances in Neural Information Processing Systems*, 2002.

[57] C. Tomasi & T. Kanade, "Shape and motion without depth", *IEEE International Conference in Computer Vision*, pp. 91–95, 1990.

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

[59] C. Tomasi & J. Zhang, "Is structure-from-motion worth pursuing?", *7th International Symposium on Robotics Research*, pp. 391–400, 1995.

[60] L. Torres, D. Garcia & A. Mates, "A robust motion estimation and segmentation approach to represent moving images with layers", *International Conference on Acoustics, Speech, and Signal Processing*, 4:2981–2984, 1997.

[61] L. Torresani, D.B. Yang, E.J. Alexander & C. Bregler, "Tracking and modeling non-rigid objects with rank constraints", *IEEE Conference on Computer Vision and Pattern Recognition*, 1:493–500, 2001.

[62] L. Torresani & C. Bregler, "Space-time tracking", *European Conference on Computer Vision*, 1:801–812, 2002.

[63] S. Ullman, “Maximizing rigidity: The incremental recovery of 3-d structure from rigid and rubbery motion”, Technical Report A.I. Memo No. 721, Massachusetts Institute of Technology, 1983.

[64] R. van der Merwe & E.A. Wan, “The square-root unscented Kalman filter for state and parameter-estimation”, *International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[65] N. Vasconcelos & A. Lippman, “Empirical Bayesian EM-based motion segmentation”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 527–532, 1997.

[66] C. Venter, “Structure from motion estimation using a nonlinear Kalman filter”, Master’s thesis, University of Stellenbosch, 2002.

[67] The Vicon Standard, http://www.viconstandard.org/, Archive and Papers sections, [2004-07-19].

[68] Virtualized Reality, Robotics Institute, Carnegie Mellon University, http://www-2.cs.cmu.edu/˜virtualized-reality/, [2004-11-08].

[69] E.A. Wan & R. van der Merwe, “The unscented Kalman filter for nonlinear estimation”, *Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, 2000.

[70] E.A. Wan, R. van der Merwe & A.T. Nelson, “Dual estimation and the unscented transformation”, *Advances in Neural Information Processing Systems*, 12:666–672, 2000.

[71] A. Watt, *3D computer graphics*, 3rd edition, Addison-Wesley, pp. 156–166, 2000.

[72] G. Welch & G. Bishop, “An introduction to the Kalman filter”, Technical Report TR 95-041, Department of Computer Science, University of North Carolina, 2003.

[73] H. Zhou, Y. Wu & T.S. Huang, “Recovering articulated motion with a hierarchical factorization method”, *5th International Workshop on Gesture and Sign Language based Human-Computer Interaction*, 2003.

[74] L. Zhou & C. Kambhamettu, “Hierarchical structure and nonrigid motion recovery from 2D monocular views”, *IEEE Conference on Computer Vision and Pattern Recognition*, 2:752–759, 2000.

[75] D. Ziou & S. Tabbone, “Edge detection techniques—An overview”, *International Journal of Pattern Recognition and Image Analysis*, 8(4):537–559, 1997.