Recovery of 3D Articulated Motion from 2D Correspondences

by

David Edward DiFranco

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2000

©Copyright 2000 David Edward DiFranco. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and distribute publicly paper and electronic copies of this thesis and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

March 2, 2000

Certified by...

W. Eric L. Grimson
Professor of Electrical Engineering and Computer Science

Thesis Supervisor

Accepted by...

Arthur C. Smith
Chairman, Department Committee on Graduate Theses
Recovery of 3D Articulated Motion from 2D Correspondences

by

David Edward DiFranco

Submitted to the Department of Electrical Engineering and Computer Science on March 2, 2000, in partial fulfillment of the requirements for the degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recovering the 3D motion of the human body is an important problem in computer vision. Applications that would benefit from 3D motion include physical therapy, computer user interfaces, and 3D animation. Unfortunately, recovering 3D position from one 2D camera is an inherently ill-posed problem. This thesis focuses on recovery of 3D motion of an articulated model using 2D correspondences from an existing 2D tracker. A number of constraints are used to aid in reconstruction: (i) kinematic constraints from a 3D kinematic model, (ii) joint angle limits, (iii) dynamic smoothing, and (iv) key frames. These methods are used successfully to recover 3D motion from video sequences. Also presented is a method for recovering 3D motion from motion capture data, as well as a method for recovering kinematic model connectivity from 2D tracks.

Thesis Supervisor: W. Eric L. Grimson
Title: Professor of Electrical Engineering and Computer Science
Acknowledgments

I’d like to thank Tat Jen Cham, my supervisor and Compaq Cambridge Research Lab (CRL), for all of his help and advice. Special thanks for the extensive help in obtaining the 3D tracking results presented in Chapter 3.

I’d like to thank Jim Rehg, head of the Vision Based Human Sensing project at CRL, for lots of advice and extensive proofreading help. Thanks also for acting as the subject of the motion capture experiments.

I’d like to thank Eric Grimson, my thesis advisor at the MIT AI Laboratory, for finding time for me in an already busy schedule. Thank you for great ideas and feedback.

Thanks to Professor Jessica Hodgins, Victor Zordan, and Dr. Bobby Bodenheimer of the Graphics, Visualization and Usability Center at Georgia Tech, for their help in acquiring the motion capture data presented in Chapter 3.

Thank you to Compaq Cambridge Research Laboratory for funding this research.

Finally, I’d like to thank my parents, for believing in me. I know that it wasn’t easy for you to put me through MIT, but it was worth it.
Contents

1 Introduction .................................................. 13
   1.1 Document Road Map ...................................... 15
   1.2 Contributions ........................................... 16

2 3D Articulated Motion Recovery from 2D Correspondences .... 17
   2.1 Tracking in 2D ........................................... 19
       2.1.1 Scaled Prismatic Model .............................. 20
       2.1.2 Output of 2D Tracker ............................... 21
   2.2 3D Kalman Filter Solution ............................... 21
       2.2.1 Kalman Filter Prediction Step ..................... 22
       2.2.2 Kalman Filter Measurements ...................... 23
       2.2.3 Kalman Filter Estimation Step .................... 24
   2.3 3D Batch Solution ....................................... 24
       2.3.1 Formulation of Batch Least-Squares Solution ... 25
       2.3.2 Dynamic Terms in the Batch Formulation ....... 26
   2.4 3D Jacobian .............................................. 27
   2.5 Joint Constraints ....................................... 28
   2.6 Key Frames ............................................... 30
   2.7 Synthetic Results ...................................... 31
       2.7.1 Dynamics .......................................... 31
       2.7.2 Joint Angle Inequality Constraints ............... 34
2.8 Results With Real Images ................................. 35
  2.8.1 Human Arm ................................................. 35
  2.8.2 Whole body reconstruction .......................... 37
2.9 Previous Work in Figure Tracking .......................... 39

3 3D Motion Recovery from Motion Capture Data ................. 45
  3.1 Recovering Joint Positions ................................. 46
  3.2 Synthetic Tests ............................................... 49
  3.3 Using Motion Capture to Evaluate 3D Reconstruction ........ 51

4 3D Kinematic Model Recovery from 2D Correspondences .......... 55
  4.1 Recovering Model Connectivity ............................. 56
    4.1.1 The RANSAC Algorithm ................................. 57
    4.1.2 The Geometric Information Criterion .................. 58
    4.1.3 Kinematic Segmentation .............................. 59
  4.2 Ball Joint Motion Model ..................................... 60
  4.3 Model Recovery Results .................................... 62
  4.4 Previous Work in Motion Segmentation ..................... 64

5 Conclusions ......................................................... 65

A 2D Kalman Filter Solution ....................................... 67
  A.1 2D Jacobian .................................................. 67

B Derivation of Gauss-Newton Update Equation ..................... 69
List of Figures

2-1 Reflective ambiguity. From the point of view of the camera, pose A looks the same as pose B. .................................................. 17

2-2 Block diagram of inputs and outputs of the 2D and 3D tracking algorithms. .................................................. 18

2-3 Fred Astaire, with SPM model overlaid. ................................. 20

2-4 Disambiguation using filtering, with a constant-velocity predictor. Knowledge of estimated position and velocity at time $t-2$ and $t-1$ bias the solution for time $t$, selecting pose B. .................................................. 23

2-5 Disambiguation using dynamic smoothing. Knowledge of estimated position at time $t-1$ and $t+1$ bias the solution for time $t$, selecting pose B. .................................................. 26

2-6 Disambiguation using joint constraints. The shaded area represents the possible joint angles. The joint angle limit removes the possibility of pose A, leaving B as the solution. .................................................. 28

2-7 Disambiguation using a key frame. The key frame expresses the approximate location of the joint (dashed line), biasing the solution towards pose B in this frame. .................................................. 30

2-8 Kinematic model used in first experiment. The two degrees of freedom are the two rotational joints, which can only rotate in the image plane. 31
2-9 First Experiment. Row 1: 2D input (side view). Row 2: Ground-truth output (top view). Row 3: Output for $\alpha = 0$ (top view). Row 4: Output for $\alpha = 1$ (top view).

2-10 Graph of the recovered angle of the second joint vs. iteration in first experiment. Notice that with $\alpha = 0.0$, the solution gets stuck and oscillates back and forth. With $\alpha = 1.0$, the solution is almost equal to ground truth.

2-11 Ambiguous case: watch the left joint. Row 1: 2D input (front view). Row 2: Ground-truth output (top view). Row 3: Erroneous output with $\alpha = 1.0$ (top view). Row 4: Output using joint-constraints (top view). Row 5: Output with key-frame four specified (top view).

2-12 Results of experiment with images of a human arm (selected frames). Left side: output of 2D tracker (front view). Right side: output of 3D reconstruction (top view).


2-14 Row 1: Manually specified 2D SPM measurements. Row 2: 3D motion reconstruction, shown from the original camera viewpoint. Row 3: 3D reconstruction shown from a different camera viewpoint.

3-1 Block diagram of inputs and outputs of the algorithm for recovering 3D motion from motion capture data.

3-2 Marker ambiguity. Translation from the joint to the marker has $x$, $y$, and $z$ components. The $z$-component is not visible because it is perpendicular to the image plane. Both the displayed rotations theta are possible with an arbitrary translation and rotation from the marker.
3-3 Results of synthetic experiment for recovery of joint angle from motion capture data: recovered joint angle positions.

3-4 Motion capture as an estimate of ground truth. Row 1: 2D input. Row 2: Motion reconstruction from 3D motion capture data. Row 3: Motion reconstruction from video using 3D tracker.

3-5 Plot of recovered joint angle over the image sequence, for three selected body joints. Top: left shoulder (side-to-side rotation). Middle: left hip (front-to-back rotation). Bottom: left knee (front-to-back rotation).

3-6 Plot of RMS error for all 16 limb joint angles, over the image sequence.

4-1 Block diagram of inputs and outputs of the 3D model recovery algorithms.

4-2 Ball joint. The shaded interior of the circle is the range of possible motion of the endpoint. \( r \) is the radius of the segment, \( \theta \) is the rotation within the image plane, and \( \phi \) is the rotation out of the image plane.

4-3 Example of results of model recovery algorithm. Recovered model and model motion over 3 image frames, for model recovery experiment 1 (rotation in the \( xy \) plane).
### List of Tables

2.1 Results of first experiment for Kalman filter (left joint) .................................. 33
2.2 Summary of results of ambiguity experiment, $\alpha = 1.0$ .................................. 35
2.3 Categorization of previous work in figure tracking by number of input and output dimensions .................................................. 43
2.4 Categorization of previous work in 3D figure tracking into gradient-based and non-gradient-based methods ................................ 43
3.1 Results of synthetic experiment for recovery of joint angles from motion capture data: recovered marker translation. Actual marker translation is $(10, 0, 0)$ .......................................................... 50
4.1 Summary of results of synthetic model recovery experiments ......................... 63
Chapter 1

Introduction

Figure tracking is an application of computer vision that has received considerable attention. Figure tracking is the recovery of the motion of a human figure from an image sequence. For example, given a kinematic model of the human body and its limbs, a figure tracker may calculate the pose of the model from a video of a person walking. Other methods model the human body differently, for example as an deformable outline, or a blob of pixels.

Applications of figure tracking include object and motion classification for surveillance and image-retrieval; motion analysis for sports training and physical therapy; gesture-based interfaces for computer systems; and motion capture for 3D animation. Figure tracking programs can be roughly divided into two categories: 2D trackers and 3D trackers. 2D trackers follow motion in the image plane, whereas 3D trackers determine motion in 3D space. 2D trackers are more robust, especially in the case of tracking with one camera, but only recover part of the actual motion. Ambiguities in the one-camera case make tracking in 3D difficult. However, for many applications, 3D information is very valuable. Much work has been done in 3D tracking with multiple cameras, using the multiple views to overcome ambiguity. However, the recovery of 3D motion from one camera has not been well studied to date.

There are many scenarios in which 3D reconstruction with one camera would be
useful. For example, a physical therapist in one city could record an injured patient walking with a video camera, and use a tracking system to obtain a 3D reconstruction of the patient’s motion. The doctor could then send the results to a specialist in another city for analysis. Another example is motion analysis from old movies or sports videos. We could analyze the movement of Fred Astaire dancing from one of his films, or Carl Lewis long-jumping from archival sports footage. Tracking in 3D could be used for motion classification: a web-crawler program could search video footage for certain types of 3D motion, for example all instances of dancers doing the tango. In general, obtaining 3D information is often very useful, and a process that used one camera to do 3D tracking would be more accessible for real-world applications. The multiple-camera setups required by many 3D tracking systems are both expensive and difficult to use.

Unfortunately, tracking in 3D with one camera is an ill-posed problem. However, similar problems solved every day by the human perception system. The human brain is able to interpret 2D visual signals of a person moving into a meaningful representation. Researchers do not agree on what representation the human brain uses for 3D motion, but it is clear that the brain is able to integrate different forms of information to come up with an interpretation of motion. On a less speculative level, there exists a large body of research that uses prior assumptions to solve difficult optimization problems, for example the use of smoothing to regularize the solution. In this spirit, this thesis introduces several methods that can be used to exploit all of our prior knowledge about human motion, to limit the ambiguity inherent in 3D reconstruction. By incorporating dynamics, joint angle limits, and key frames, we obtain good 3D reconstruction results on long motion sequences.

One major difficulty in the analysis of existing figure tracking systems is the absence of ground truth to compare results against. It is possible to obtain ground truth on an artificially synthesized image sequence, where the 3D motion is known. Ideally, however, we would like to know ground-truth 3D motion when tracking from
real images of a person moving. In Chapter 3, we estimate ground truth using motion capture data from magnetic markers placed on a subject's body. However, the process of obtaining 3D joint positions from motion capture data is a non-trivial optimization problem; our approach to the problem is presented in Chapter 3.

An extension of articulated motion capture is articulated model recovery — rather than assuming that we know the 3D model of the body, we can attempt to recover it. We could recover the lengths of the limbs, or the connectivity of the kinematic body of the body. Little research has been done in this area. We present theoretical results on the information necessary to recover model structure. We develop an algorithm to recover the structure of a model from 2D correspondences, and present some synthetic results.

1.1 Document Road Map

Chapters 2, 3, and 4 present the three major research areas of this thesis. Chapter 2 details our method for recovering 3D articulated motion from 2D image correspondences. It explains our algorithm, presents synthetic and real-world results, and gives a summary of previous work in the field of figure tracking. Chapter 3 presents our method for obtaining the solution for 3D model position from 3D motion capture data. We use this method to compare results from our 3D tracker (presented in Chapter 2) to 3D motion capture data. Chapter 4 presents our algorithm for recovering model structure from 2D image correspondences. It presents some preliminary results, and gives a summary of related research. In Chapter 5 we present conclusions and suggestions for future work. Finally, Appendices A and B give the mathematical details of our 2D figure tracker and our non-linear least-squares solver, respectively.
1.2 Contributions

This thesis makes the following contributions:

1. Introducing a robust framework for 3D articulated motion recovery from 2D correspondences. This is the first such framework to integrate all of the following tools together: kinematic constraints, dynamics, joint angle limits, and key frames. (Chapter 2)

2. Introducing a method for using the estimate of error covariance from the 2D tracker to weight the confidence in the 2D correspondences used in 3D reconstruction. (Section 2.2.3)

3. Using the above framework to achieve the first 3D motion reconstruction from real images in the case of complete body rotation. (Section 2.8)

4. Introducing a method for automatically calculating 3D articulated motion from magnetic motion capture data, using a variation of the 3D motion framework mentioned above. (Chapter 3)

5. Using the estimate of ground truth, obtained from motion capture data, to evaluate the results of the articulated motion recovery algorithm in Chapter 2. This is the first quantitative evaluation of the performance of a 3D motion reconstruction algorithm in a real-world situation. (Section 3.3)

6. Analyzing mathematically the problem of recovering kinematic model connectivity from 2D tracks. (Chapter 4)
Chapter 2

3D Articulated Motion Recovery from 2D Correspondences

The major difficulty in reconstructing 3D motion from 2D images is the ambiguity inherent in the problem. When a joint rotates through the image plane, it is impossible to tell if the joint is subsequently moving towards the camera or away. This ambiguity is known as the affine, or reflective, ambiguity (see Figure 2-1.) To obtain a stable solution, we use a method suggested by Morris and Rehg [19], in which we first track the figure in 2D, using 2D kinematic model that captures the configuration of a person in the image plane. We then use this data to calculate full 3D motion from the estimated 2D joint positions. We exploit the estimated errors from the 2D solution to tell us how much to trust each 2D measurement in the 3D solution.

Figure 2-1: Reflective ambiguity. From the point of view of the camera, pose A looks the same as pose B.
Separating the two processes allows us to separate the problem of registration from the problem of motion reconstruction, as is currently done in other computer vision algorithms, such as Structure from Motion. The 2D tracker deals with problems such as image noise and clutter. This is easier to do without the additional reflective ambiguity that occurs in 3D reconstruction. If we track in 2D first, we can formulate motion reconstruction as a batch estimation problem. This would be infeasible if we were working directly from images in the motion reconstruction step. Advantages to the batch formulation are given in Section 2.3.

In order to calculate the 3D position of a human figure from a 2D image stream, we use kinematic constraints, imposed by a human body model with constant link lengths. To further improve our solution and overcome the reflective ambiguity, we take advantage of several types of user-specified constraints. We use joint angle limits to force a solution that is consistent with the possible range of motion of the human body. We use dynamic prediction of joint motion from frame to frame, thereby exploiting smoothness of angular velocity for each joint. Finally, to allow the user to improve the solution, we allow the specification of key frames, in which the approximate 3D body position is specified by the user. This helps bias the state space trajectory in all frames. Joint angle constraints and key frames are often necessary to obtain a good reconstruction of motion, as we will show in Section 2.8. Figure 2-2 shows a block diagram of the input and outputs of the 2D and 3D tracking algorithms.

Figure 2-2: Block diagram of inputs and outputs of the 2D and 3D tracking algorithms.
We formulate two different methods of solving for 3D motion: a Kalman filter solution that estimates the position at each frame based on the previous frame, and a batch solution that estimates all frames simultaneously. The filtering solution is faster because it only considers a frame at a time. The batch solution, however, has the advantage of working with the data from all frames simultaneously. This allows the specification of key frames and the smoothing of the solution between the key frames.

We test our 3D motion recovery system on simple synthetic examples, and then move on to 3D tracking in real image sequences. We achieve results for a 23-degree-of-freedom full body model, for ranges of motion not previously achieved by other trackers. We believe that these are the first results of 3D reconstruction of complex human motion from monocular video.

Section 2.1 describes the 2D tracking system used to obtain the input to our algorithm. Section 2.2 and Section 2.3 describe our Kalman filter and batch optimization solutions, respectively. Section 2.4 gives the mathematical derivations for the Jacobian terms used in the optimization. Section 2.5 and Section 2.6 explain how joint constraints and key frames are included in the solution. Section 2.7 and Section 2.8 present synthetic and real-world results, respectively. Finally, Section 2.9 lists previous research in the area of figure tracking.

2.1 Tracking in 2D

For 2D tracking, we use a template based-approach. We initialize the model in the first image frame, and follow regions of pixels as they move between image frames. The 2D kinematic model governs the range of motion possible between frames. The 2D tracker solves for the 2D joint positions of the 2D model, minimizing the pixel error between the calculated templates and the actual images.
2.1.1 Scaled Prismatic Model

We use Morris and Rehg's Scaled Prismatic Model [19] to describe the motion of the figure in the image plane. An SPM model can contain two types of joints: revolute joints that allow segments to rotate within the camera plane, and prismatic joints that allow segments to stretch and shrink. Prismatic joints allow the model to handle the foreshortening effect, which occurs when a segment rotates towards or away from the camera, as shown in Figure 2-1. Figure 2-3 shows a frame of a Fred Astaire film, with the SPM overlaid on the figure.

![SPM Model](image)

Figure 2-3: Fred Astaire, with SPM model overlaid.

As Morris and Rehg show, the SPM model avoids the singularity that occurs when a 3D link rotates through a plane parallel to the image plane. Because foreshortening is the only clue that a link has rotated out of the plane of view, it is impossible to calculate whether a 3D joint is in front of or behind the plane parallel to the image plane. (Refer back to Figure 2-1 to see this). The Scaled Prismatic Model does not calculate rotation out of the plane, so this ambiguity never arises. Morris and Rehg show that in fact singularity in the SPM Jacobian occurs if and only if one of the prismatic joint lengths is zero.

We solve for the model parameters using template registration, in a Kalman filter framework. The kinematic structure of the model is expressed in terms of Denavit-Hartenberg (DH) parameters [4], a standard motion parameterization used
in the robotics community. Details of the Kalman filter solution are presented in Appendix A.

2.1.2 Output of 2D Tracker

The 2D tracker outputs the parameters of the kinematic model—a set of joint angles and prismatic joint lengths. From these parameters it is easy to compute the 2D positions of the joints of the model. These 2D joint positions are the input for the recovery of 3D motion. The 2D tracker also outputs an estimate of the state error covariance. This information can be used to determine how much we trust each calculated 2D joint location as input to our 3D solution. We prevent a novel method for doing this in Section 2.2.3).

2.2 3D Kalman Filter Solution

We estimate our 3D model states from the 2D model states. We use the same Denavit-Hartenberg parameterization to express our 3D model as we used for the 2D model. However, the 3D model has fixed link lengths, so there are no prismatic joints. The model now has only revolute joints. In contrast to the 2D model, however, links may now rotate out of the image plane.

We can solve for the parameters of the 3D model using a Kalman filter framework. For each frame in the sequence, we first make a prediction of the solution based on the previous state. Then we estimate the solution, using this prediction as a starting point.

The Kalman filter is used to recursively estimate time-varying states from observations and initial conditions [18], [29]. The general framework we use for prediction is given by the following two equations for the model states $q_t$ at time $t$ given observations $z_t$:

$$
q_{t+1} = Aq_t + w_t \tag{2.1}
$$
\[ z_t = J_t(q_t) + v_t \] (2.2)

where \( w \) and \( v \) are the process noise and measurement noise, respectively, and are assumed to be normally distributed with covariance matrices \( Q \) and \( R \), respectively. \( A \) is the dynamics matrix, mapping previous states to the current state. In our model, we set \( A \) to predict smoothness in position and velocity. This is described in Section 2.2.1. \( J_t() \) is the measurement function, mapping the current state to the measurements. We estimate \( J_t() \), as the Jacobian matrix \( J_t \), by linearizing the measurement model given the state at time \( t \), as described in Section 2.4. (From now on \( J_t \) will refer to the Jacobian matrix rather than the non-linear function.) We must recompute \( J_t \) at each time step because the state varies with time. We set the values of \( Q \) and \( R \) to trade-off between the degree that we trust 1) our measurements and 2) our predictions for each parameter.

The Kalman filter procedure consists of two stages for each time step \( t \). First, we predict the current state \( q_t \) using equation 2.1. We then calculate \( J_t \), and calculate \( q_t \) using the a combination of the prediction equation (2.1) and the inverse of the measurement equation (2.2).

### 2.2.1 Kalman Filter Prediction Step

The prediction step of Kalman filtering allows our solution to incorporate expectations of how current angle states depend on states in previous time steps. Figure 2-4 shows how the prediction of constant velocity can bias the solution towards the correct position.

In our prediction step, we predict the current state as a linear function of previous states:

\[ q_{t+1} = Aq_t. \] (2.3)

In practice, we use a combination between a constant-position predictor and a constant velocity predictor. If we assume that the position of each joint is constant over
Figure 2-4: Disambiguation using filtering, with a constant-velocity predictor. Knowledge of estimated position and velocity at time $t - 2$ and $t - 1$ bias the solution for time $t$, selecting pose B.

time, we have

$$\dot{q}_{t+1} = q_t.$$  \hfill (2.4)

Similarly, if we assume constant velocity, we have

$$\dot{q}_{t+1} - q_t = q_t - q_{t-1},$$  \hfill (2.5)

or

$$\dot{q}_{t+1} = 2q_t - q_{t-1}.$$  \hfill (2.6)

We can take a linear combination of the two to obtain the equation

$$\dot{q}_{t+1} = (\alpha + 1)q_t - \alpha q_{t-1},$$  \hfill (2.7)

where $\alpha$ in $[0...1]$ is the proportion that we weight constant-velocity prediction, and $1 - \alpha$ is the proportion that we weight constant-position prediction.

2.2.2 Kalman Filter Measurements

The measurements we use in our solution are the 2D joint positions output by the 2D tracker. The next section describes a novel method for using the estimated covariance of the 2D states to improve our 3D solution.
2.2.3 Kalman Filter Estimation Step

The estimation step uses the following Kalman filter estimation equations from Welch and Bishop [29]:

\begin{align*}
K &= P_t J_t^T (J_t P_t J_t^T + R)^{-1} \\
q_t &= \hat{q}_t + K(z_t - J_t q_t) \\
Pt &= (I - K_t J_t) \hat{P}_t. 
\end{align*}

(2.8)  
(2.9)  
(2.10)

$P_t$ is the estimated state covariance matrix, and is predicted as

\[
\hat{P}_{t+1} = A_t P_t A_t^T + Q. 
\]

(2.11)

This gives us a solution for $q_t$ at each step that is correct in a least-squares sense given the data and its uncertainties.

Because the 2D tracker is also a least-squares solver that outputs an error for each 2D joint position, we can incorporate the resulting covariance matrix into the measurement covariance $R$ of the 3D solution. The predicted state error covariance, $P$, from the 2D Kalman filter computation is equivalent to $ee^T$, where $e$ is the predicted state error vector. We can relate this to the measurement error covariance to calculate $R$ from $P$:

\[
R = rr^T = Je(Je)^T = Je e^T J^T = JP J^T. 
\]

(2.12)

This allows us to weight each 2D measurement proportionately to our confidence in the measurement.

2.3 3D Batch Solution

Rather than predicting the states at one time step based on previous time steps, added power can be gained by solving for the states at all steps simultaneously. The Kalman filter may get stuck in an incorrect solution path, because it does not exploit future
frames to aid in predicting the solution for a given frame. With a batch solution, both past and future frames are used to produce a smooth solution. In addition, a batch solution allows us to specify multiple key frames and interpolate between them.

The Kalman filter solution does have the potential to be much faster than the batch solution, however. The Kalman filter solution only solves for one frame at a time. Therefore the Jacobian matrix, which is inverted in the solution, is much smaller. Consider the case in which we have a sequence with $t$ frames and a model with $n$ states. The Kalman filter solution must do $t$ inversions of $n \times n$ matrices, each taking $\Theta(n^3)$ time. On the other hand, the batch solution must invert an $nt \times nt$ block-diagonal matrix, taking $\Theta(n^3 t^2)$ time (see Section 2.3.1). Thus, the Kalman filter solution saves us a factor of $\Theta(t)$ time.

2.3.1 Formulation of Batch Least-Squares Solution

The batch solution simply incorporates the state and measurement parameters of each time frame into one big state vector and one big measurement vector. The Jacobian, $J$, relating them is a block-diagonal matrix made up of the Jacobians for each time step. Thus, we have a state vector $q$ of size $nt$, a measurement vector $z$ of size $mt$, and a Jacobian of size $m t x n t$, where $n$ is the number of model states, $m$ is the number of measurements at each time step, and $t$ is the number of time steps. To stabilize the solution, we also add a term $kI$ to the Jacobian, where $k$ is a constant.

We solve for $q$ using the Gauss-Newton iterative least-squares method. At every iteration, we solve the state update equation

$$(J^TR^{-1}J)\Delta q = J^TR^{-1}r$$

for $\Delta q$, where $R$ is the measurement covariance matrix, and $r$ is the measurement residual error, $z - z_{estimated}$. This is the correct solution for $\Delta q$ in a least-squares sense. Equation 2.13 is derived in Appendix B.

Because the Jacobian is block-diagonal, $(J^TR^{-1}J)^{-1}J^TR^{-1}r$ is block diagonal if
we assume $R$ is also block-diagonal (i.e., if measurement errors in one frame are independent of errors in other frames.) Therefore, we can use a solver that takes advantage of this fact to run in less time. For example, Press et al. [21] describes such a solver for general sparse matrices. If $n$ is the number of model states and $t$ is the number of image frames, we can use this method to solve in $\Theta(t^2n^3)$ time. This saves a factor of $\Theta(t)$ time over standard matrix inversion. Because the method allows us to store only the non-zero elements of the matrix, we also save a factor of $\Theta(t)$ space.

### 2.3.2 Dynamic Terms in the Batch Formulation

The batch solution can be augmented to include terms that exploit dynamic constraints between time steps. Kalman filtering predicts the current states from states at previous time steps. In batch estimation, however, we can exploit smoothing prediction, which uses the estimates of states at all time steps to predict a state at a particular time step. Figure 2-5 shows how smoothing can bias the solution towards the correct position in the face of ambiguity.

![Figure 2-5: Disambiguation using dynamic smoothing. Knowledge of estimated position at time $t-1$ and $t+1$ bias the solution for time $t$, selecting pose B.](image)

The batch solution can incorporate any filtering predictor in the form of Equation 2.3. We add terms to the Jacobian and residual that enforce the constant-position constraint. Thus, to Equation 2.13 we add

\[
I \Delta q = [q_t - Aq_{t-1}].
\]
(The brackets in the above equation, and those following, denote a vector of terms for each time $t$). This update equation adds a $\Delta \mathbf{q}$ to each state $\mathbf{q}_t$ that causes $\mathbf{q}_t$ to equal $A \mathbf{q}_{t-1}$.

We can also express smoothing predictors, in terms of both past and future states. In general we have

$$I \mathbf{q} = [\mathbf{q}_t - A_- \mathbf{q}_{t-1} - A_+ \mathbf{q}_{t+1}],$$

(2.15)

The constant position and constant velocity predictors of Section 2.2.1 can be expressed in this form. The equation

$$I \mathbf{q} = [\mathbf{q}_t - \frac{1}{2} \mathbf{q}_{t-1} - \frac{1}{2} \mathbf{q}_{t+1}]$$

(2.16)

encapsulates the constant velocity assumption in terms of past and future states. If $\alpha$ is the proportion we weight constant-velocity prediction with respect to constant-position prediction, we have

$$I \mathbf{q} = [\mathbf{q}_t - \beta_1 \mathbf{q}_{t+1} - \beta_2 \mathbf{q}_{t-1}],$$

(2.17)

where

$$\beta_1 = \frac{\alpha}{1+\alpha}, \beta_2 = \frac{1}{1+\alpha}.$$  

(2.18)

As in the Kalman filter solution, the use of dynamic prediction allows us to overcome ambiguity in the 3D solution resulting when a joint rotates through the image plane. By biasing the solution towards one with constant angle velocities, we assume that a joint rotating through the plane of view will continue to the other side of the plane.

### 2.4 3D Jacobian

Our 3D model contains only one type of joint – a revolute joint that describes the 3D rotation between two links. We take limb lengths in the 3D model to be fixed. The 3D Jacobian relating joint positions to state parameters is similar to that for revolute
joints in the 2D tracker. As derived by Rehg, the velocity in the image plane of a point \( p \) is

\[ \mathbf{v}_p = P(\mathbf{a} \times \mathbf{p})\Delta q, \]  

(2.19)

where \( \mathbf{a} \) is the axis of rotation of the joint, \( \Delta q \) is the change in angular state, and \( P \) is the projection equation mapping 3D points to the image plane [22]. Therefore the Jacobian term due to a joint \( i \) at joint position \( j \) is:

\[ J_{ij} = \begin{cases} 
0 & \text{links } k, \text{where } k < i \\
\mathbf{P}(\mathbf{a} \times \mathbf{p}) & \text{links } k, \text{where } k \geq i 
\end{cases} \]  

(2.20)

2.5 Joint Constraints

One way to limit our solution for 3D motion to a physically valid result is to incorporate limits on the range of joint motion. For example, a human elbow can only rotate through approximately 135 degrees; it makes sense to use this knowledge to obtain a plausible solution for 3D motion. Figure 2-6 illustrates how joint constraints can help remove ambiguity in the solution for motion.

Figure 2-6: Disambiguation using joint constraints. The shaded area represents the possible joint angles. The joint angle limit removes the possibility of pose A, leaving B as the solution.

To incorporate limits on the range of joint angles, we add inequality constraints to the Kalman filter solution, using a scheme introduced by Grimson [8], and incorporated in the Kalman filter formulation by Hel-Or and Werman [10]. In general, we can incorporate the constraint

\[ g(x) \geq 0 \]  

(2.21)
as
\[ g(x) - \lambda^2 = 0, \] 
(2.22)

where \( \lambda \) is a new parameter that we add to the Kalman filter solution. Therefore, an inequality such as
\[ q \geq \theta \] 
(2.23)
becomes
\[ q - \theta - \lambda^2 = 0. \] 
(2.24)

Our measurement residual \( r \) for \( \lambda \) is:
\[ r = -(q - \theta - \lambda^2), \] 
(2.25)

and we augment our Jacobian with the following terms:
\[ \frac{\partial z}{\partial q} = 1, \] 
(2.26)
\[ \frac{\partial z}{\partial \lambda} = -2\lambda. \] 
(2.27)

The results are similar for constraints in the form of \( q \leq \theta \). For the prediction step, we simply predict that each state \( \lambda \) stays constant from frame to frame. To weight the constraints differently, we can give the states \( \lambda \) different errors in the measurement covariance matrix \( R \).

We can also add joint constraints to the batch solution in a similar manner as the dynamic terms. We add terms to the Jacobian and residual, to correct for each parameter that exceeds its upper limit \( \ell_U \) or lower limit \( \ell_L \). Thus we have the equation
\[ I \Delta q = \min(\ell_U - q, 0), \] 
(2.28)
or
\[ I \Delta q = \max(\ell_L - q, 0), \] 
(2.29)
which we add to Equation 2.13.
In either case, the resulting solution for $\Delta q$ pushes each out-of-range state back towards the limit. The advantage of the latter formulation is that it eliminates the need to add states to the least-squares solution. Adding states to the solution increases the size of the Jacobian matrix that is inverted, thus increasing storage requirements and run time.

2.6 Key Frames

Just as in the Kalman filter solution, in the batch solution we must initialize at least the first frame, to remove reflective ambiguity. Because of the ambiguities in projecting from 3D to 2D, it is desirable to have a framework in which we can specify the model position in multiple key frames. In the batch solution, this arises naturally — we simply initialize the desired key frames, with an estimate of uncertainty for each initialized state. Figure 2-7 shows how key frames can be used to eliminate ambiguity.

Figure 2-7: Disambiguation using a key frame. The key frame expresses the approximate location of the joint (dashed line), biasing the solution towards pose B in this frame.

For added flexibility, we can initialize only specific joints in a particular frame. For example we can specify the position of one leg in a human figure model if this is the only ambiguous limb.

As a heuristic, frames that we do not manually initialize are linearly interpolated between the key frames. The batch solver starts from there to find the optimal solution.
2.7 Synthetic Results

As a first test of the effectiveness of our 3D solution in the face of ambiguity, we ran experiments using simple articulated models. The 2D joint positions were created synthetically, with the 3D positions known. Therefore, we had ground truth to compare the results against.

2.7.1 Dynamics

To test the ability of dynamics to overcome the affine ambiguity, we ran the tracker on the simple example of a 2 degree-of-freedom (DOF), two-arm articulated model, shown in Figure 2-8. Each joint rotated at a constant velocity through several complete revolutions about the $y$-axis. The joints rotated 30 degrees per frame, over 22 frames. We tested three solutions: 1) Kalman filtering, 2) batch filtering, and 3) batch smoothing. We tested different values of $\alpha$ in the prediction model

$$q_{t+1} = (\alpha + 1)q_t - \alpha q_{t-1}. \quad (2.30)$$

As described in Section 2.2, a value of $\alpha = 0.0$ corresponds to a constant-position predictor, and a value of $\alpha = 1.0$ corresponds to a constant-velocity predictor. Intermediate values of $\alpha$ are a mixture of the two models. In the filtering solutions, we initialized the first two frames of the 3D solution. In the smoothing solution we initialized the first and last frames. The batch solver was run until it converged on a solution — in other words, until the change in state $\dot{q}$ calculated by the solver was below a threshold of 0.001 for every state.

Figure 2-8: Kinematic model used in first experiment. The two degrees of freedom are the two rotational joints, which can only rotate in the image plane.
Figure 2-9 shows the first eight frames of the input and some of the results using the Kalman filter solution. Figure 2-10 shows the results of the same experiment as a graph, for the joint on the right side. As can be seen, constant-position prediction led to a case where the calculated joint positions never rotated beyond the image-plane ambiguity. Constant velocity prediction, however, brought the solution to the correct side of the ambiguity. This illustrates the importance of velocity prediction in 3D motion reconstruction from 2D.

![Figure 2-9: First Experiment. Row 1: 2D input (side view). Row 2: Ground-truth output (top view). Row 3: Output for $\alpha = 0$ (top view). Row 4: Output for $\alpha = 1$ (top view).](image)

Finally, Table 2.1 summarizes the results for different values of $\alpha$ in all three solutions. The RMS angle error over all frames is shown for each method, for each value of $\alpha$. As is shown, all three methods performed well with $\alpha$ close to 1.0, but quickly deteriorated as $\alpha$ decreased. In a real-world example, however, there would probably be less angular rotation between each frame, and the solution would therefore be more stable. In general, the batch smoothing method produced better results than the Kalman filter. This makes intuitive sense, because the batch solution has more information to work with. It is unknown why the Kalman filter performed better at $\alpha = 0.75$. 

32
Figure 2-10: Graph of the recovered angle of the second joint vs. iteration in first experiment. Notice that with $\alpha = 0.0$, the solution gets stuck and oscillates back and forth. With $\alpha = 1.0$, the solution is almost equal to ground truth.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Kalman filter</th>
<th>batch filter</th>
<th>batch smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0342</td>
<td>0.0283</td>
<td>0.0138</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0451</td>
<td>0.5931</td>
<td>0.517</td>
</tr>
<tr>
<td>0.75</td>
<td>0.513</td>
<td>4.31</td>
<td>2.52</td>
</tr>
<tr>
<td>0.5</td>
<td>5.22</td>
<td>6.29</td>
<td>4.88</td>
</tr>
<tr>
<td>0.0</td>
<td>6.18</td>
<td>6.34</td>
<td>4.91</td>
</tr>
</tbody>
</table>

Table 2.1: Results of first experiment for Kalman filter (left joint).
2.7.2 Joint Angle Inequality Constraints

To test joint angle inequality constraints, we modified the synthetic test case so that the joint reversed direction at one of the image-plane ambiguities. Figure 2-11 shows the 2D input and some of the 3D results. The 2D joint-position input is identical to that of the previous experiment. In the ground-truth results, however, note the reversal of direction of the left joint in the fourth frame. In this case, a constant-velocity predictor causes the solution to incorrectly continue in the same direction at the ambiguity. The left joint keeps a constant velocity as it rotates through the ambiguity.

Figure 2-11: Ambiguous case: watch the left joint. Row 1: 2D input (front view). Row 2: Ground-truth output (top view). Row 3: Erroneous output with $\alpha = 1.0$ (top view). Row 4: Output using joint-constraints (top view). Row 5: Output with key-frame four specified (top view).

The only way to solve this problem is through added user input, such as a joint angle constraint. Row 4 of Figure 2-11 shows the results of using joint angle limits in the Kalman filter solution. The left joint was given a limit at the angle of ambiguity. As shown, it correctly reverses directions in the fourth frame.

Another method for coping with this ambiguity is specifying a key frame at the ambiguity. Row 5 of Figure 2-11 shows the results of using this method — which
Table 2.2: Summary of results of ambiguity experiment, $\alpha = 1.0$.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMS error (radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalman filter</td>
<td>3.24</td>
</tr>
<tr>
<td>Kalman filter with joint limits</td>
<td>0.0123</td>
</tr>
<tr>
<td>Batch smoothing</td>
<td>2.90</td>
</tr>
<tr>
<td>Batch smoothing with joint limits</td>
<td>0.00562</td>
</tr>
<tr>
<td>Batch smoothing with key frame</td>
<td>0.0118</td>
</tr>
</tbody>
</table>

works as well in this case as setting a joint angle limit.

Table 2.2 summarizes results of this experiment in both the batch and Kalman filter formulations, with joint angle limits and key frames. It shows the RMS angle error over all frames.

### 2.8 Results With Real Images

#### 2.8.1 Human Arm

For a first real-world test of our 3D reconstruction framework, we used a 30-frame image sequence of a human arm rotating through the image plane. We tracked in 2D, using template-matching to fit to a 4-DOF SPM model. The model had two degrees of translational freedom for the torso (depth is impossible to recover), and an arm with a rotational and a prismatic joint. The 2D tracker performed well. The results of 2D tracking are overlaid on the image sequence on the left side of Figure 2-12.

The right side of Figure 2-12 shows the 3D motion calculated from the 2D correspondences, using Kalman filtering. We used $\alpha = 0.75$ for dynamic prediction. As the image shows, the arm position correctly moves from behind the body to the front of the body.
Figure 2-12: Results of experiment with images of a human arm (selected frames). Left side: output of 2D tracker (front view). Right side: output of 3D reconstruction (top view).
2.8.2 Whole body reconstruction

One of the main goals of this work is to recover full human motion from a video sequence. This section presents results obtained from two movie clips of Fred Astaire dancing. We manually specified the joint positions of the 2D SPM model, in order to test just the 3D reconstruction phase of our algorithm. At this point we assume completely correct 2D correspondences. Obtaining good 3D estimates from noisy 2D correspondences is a subject for future research.

The first video sequence shows Fred Astaire moving with no rotation of his torso out of the plane of view. This is typical of the 3D tracking results presented in previous research. Figure 2-13 shows the input and results of the experiment. Row 1 shows 6 example frames of the 24-frame video sequence, with the manually specified 2D model position overlaid. Rows 2 shows results of running the Kalman filter 3D reconstruction algorithm, animated using 3D Studio Max, from a side view. Note the impossible backward bending of the left elbow in the fourth frame, and the incorrect forward position of the left leg in the final frame.

Row 3 of Figure 2-13 shows the results using joint constraints, to only allow joint rotations that are possible for an actual human body. Now the elbow position is recovered correctly, because the impossible backward bending is ruled out by the joint constraint. However the left leg is still incorrectly placed well in front of the body, in a position that would cause the dancer to fall. (We currently place no constraints the figure being balanced, however this may be possible in the future). Finally, Row 4 shows the results using a batch formulation with the first and last frames specified as key frames. Now, the arm and leg positions appear to be correctly recovered in all frames.

The second video sequence shows Fred Astaire spinning around — a situation not currently dealt with by existing methods for 3D motion reconstruction. Previous results don’t deal with the difficult case of complete body rotation. In this case we allow the shoulder and hip joints of our 2D SPM model to translate horizontally.
Figure 2-13: Row 1: Manually specified 2D SPM measurements. Row 2: 3D reconstruction using Kalman filter, side view. Row 3: 3D reconstruction using Kalman filter with joint angle constraints, side view. Row 4: 3D reconstruction using batch solution with two key frames, side view.
with respect to the body. However this method has a major limitation — we are now absolutely forced to manually specify the 2D joint positions. We must do this because our 2D tracker is currently not stable enough to handle large amounts of body rotation. Tracking templates does not work when the body rotates 180 degrees, because the template for the chest is not the same as that for the back. To accurately track body rotation, we would need to introduce a scheme for reinitializing templates when necessary, but that is beyond the scope of this work. Figure 2-14 shows our 2D input and the 3D reconstruction results. Row 1 shows four frames of the 14-frame input sequence, overlaid with the manually specified 2D model position. We used the batch formulation for 3D reconstruction, with joint angle constraints and the first and last key frames specified. The algorithm took 27 seconds to run. Row 2 shows the 3D reconstruction from the camera viewpoint, rendered in 3D Studio Max. Row 3 shows the output from a different viewpoint. The results appear very credible. One artifact is the penetration of one leg into the other, because we do not currently restrict against that in our reconstruction. Another artifact is the incorrect recovery of face direction, because currently we do not track the face in our 3D model. Of course, a quantitative measure of the quality of our results is impossible without ground truth. In this case we have none, because we are working with archival footage. In Chapter 3, however, we estimate 3D ground truth, and use our estimate to evaluate the performance of our 3D tracker on another video sequence.

2.9 Previous Work in Figure Tracking

While there is a large body of work on figure tracking in 2D and 3D, only a few researchers have considered the difficulties faced in attempting to recover 3D motion from 2D images. O’Rourke and Badler [20] and Hogg [12] were the first to attempt the problem of articulated motion recovery of the human figure from a single video sequence, using AI-based search techniques. Hogg’s work is a working implementa-
Figure 2-14: Row 1: Manually specified 2D SPM measurements. Row 2: 3D motion reconstruction, shown from the original camera viewpoint. Row 3: 3D reconstruction shown from a different camera viewpoint.
tion of the ideas set down by O'Rourke and Badler. Rehg [22] uses a gradient-based method for articulated motion recovery. He implements a system for tracking the human hand with one camera. We use a gradient-based approach in this thesis as well. Goncalves et al. [7] use non-gradient based methods to track a human arm in a very constrained environment with minimal reflective ambiguity. Shimada et al. [25] propose a gradient-based system for articulated motion recovery of the hand, including a method for incorporating joint angle limits to avoid the depth ambiguity. They are among the first to recognize the depth ambiguity problem, and propose a solution. Previous work ignores the depth ambiguity problem, and achieves good results only in cases of constrained motion where the ambiguity does not arise. However, Shimada et al. only demonstrate their system in synthetic experiments. Hel-Or and Werman [10] propose similar joint angle constraints in their method for articulated motion recovery, but they show results only for motion parallel to the image plane. We borrow their formulation of joint angle limits in this thesis.

Recently, researchers have attempted to use large amounts of training data, obtained from MRI data, to improve their solution for 3D motion from a single image sequence. Heap and Hogg [9] use this method to recover a deformable model of the hand, while Howe et al. [14] use this method to recover articulated motion of the human figure. This may prove to be a powerful method for limiting the solution space. However it is not currently clear if training data for one person will generalize to other people.

Recently, some researchers have sought to avoid the problems of 3D tracking from monocular video, and to instead recover only motion within the image plane — in effect to track in 2D only. Ju et al. [16] propose a “Cardboard People” kinematic model, composed of planar patches that are allowed to deform arbitrarily. Morris and Rehg [19] formulate the slightly more constrained Scaled Prismatic Model, which is the model this thesis uses to parameterize 2D and 3D motion. Morris and Rehg motivate their model with analysis of the singularities inherent in 3D motion recovery.
from monocular video. Along with Shimada et al. [25] they are the first to use such analysis as motivation. Other researchers, such as Wren et al. [30], forgo a kinematic model, instead tracking blobs of pixels in 2D. This has the advantage of simplifying the problem of locating a moving person within a scene, but limits the interpretation of motion. In general, tracking in 2D has only limited usefulness. For example Hienz et al. [11] analyze the 2D motion of the hand (tracked using colored markers) to identify sign-language type gestures. For applications such as motion analysis for medicine and motion capture for 3D animation, however, 3D tracking is necessary.

The problem of motion recovery from multiple cameras is much better studied than the monocular case, because it is a mathematically better-posed problem. Recently, researchers have tackled the problem using gradient-based techniques, borrowed from the field of robotics, to limit the solution space of the problem. This has made figure tracking much more feasible. Yamamoto [31] was the first to propose such a solution, and Rehg [22] implemented a real-time hand tracking system using this approach. Subsequently, other researchers have followed the gradient-based approach to track the human body using multiple cameras, for example Rohr [23], Kakadiaris and Metaxis [17], Bregler and Malik [2], Wren et al. [30], and Wachter and Nagel [28].

Table 2.3 summarizes previous work in figure tracking according to the dimensionality of the input and output. Table 2.4 categorizes previous 3D tracking methods as gradient-based or non-gradient-based. This is by no means a complete list of the work in figure tracking, but it contains the important research that influenced this thesis.
### Table 2.3: Categorization of previous work in figure tracking by number of input and output dimensions

<table>
<thead>
<tr>
<th>3D Monocular Reconstruction</th>
<th>2D Monocular Reconstruction</th>
<th>3D Multiple Camera Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>O'Rourke and Badler 80</td>
<td>Ju et al. 96</td>
<td>Yamamoto 91</td>
</tr>
<tr>
<td>Hogg 83</td>
<td>Hel-Or and Werman 96</td>
<td>Rohr 94</td>
</tr>
<tr>
<td>Goncalves et al. 95</td>
<td>Wren 97</td>
<td>Kakadiaris and Metaxis 95</td>
</tr>
<tr>
<td>Rehg 95</td>
<td>Hienz et al. 96</td>
<td>Gavrila and Davis 96</td>
</tr>
<tr>
<td>Heap and Hogg 96</td>
<td>Morris and Rehg 98</td>
<td>Bregler and Malik 97</td>
</tr>
<tr>
<td>Shimada et al. 98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wachter and Nagel 99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Howe et al. 99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2.4: Categorization of previous work in 3D figure tracking into gradient-based and non-gradient-based methods

<table>
<thead>
<tr>
<th>Gradient-based methods</th>
<th>Non-gradient-based methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yamamoto 91</td>
<td>O'Rourke and Badler 80</td>
</tr>
<tr>
<td>Rohr 94</td>
<td>Hogg 83</td>
</tr>
<tr>
<td>Rehg 95</td>
<td>Goncalves et al. 95</td>
</tr>
<tr>
<td>Kakadiaris and Metaxis 95</td>
<td>Gavrila and Davis 96</td>
</tr>
<tr>
<td>Ju et al. 96</td>
<td></td>
</tr>
<tr>
<td>Bregler and Malik 97</td>
<td></td>
</tr>
<tr>
<td>Wren 97</td>
<td></td>
</tr>
<tr>
<td>Wachter and Nagel 99</td>
<td></td>
</tr>
<tr>
<td>Howe et al. 99</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: Categorization of previous work in figure tracking by number of input and output dimensions

Table 2.4: Categorization of previous work in 3D figure tracking into gradient-based and non-gradient-based methods
Chapter 3

3D Motion Recovery from Motion Capture Data

To measure the accuracy of our 3D reconstruction algorithm (as presented in Chapter 2), we desire ground truth to measure our results against. This is easy for simple synthetic examples, such as those in Section 2.7, where we can analytically calculate ground truth. However, we would like a quantitative measure of accuracy for real-world examples as well. In the case of our Fred Astaire footage, this is impossible because there is no way to go back and measure Fred Astaire’s 3D motion. We can, however, test our algorithm if we obtain 3D motion capture data while shooting video footage of a person moving. This section details our method for estimating 3D ground truth from motion capture data, and concludes with a comparison of 3D motion obtained from our 3D tracker with our ground-truth estimate.

We obtained motion-capture data paired with video footage of a subject moving, with the help of Professor Jessica Hodgins of the Graphics, Visualization and Usability Center at Georgia Institute of Technology. The motion capture data consists of 3D translation and rotation measurements of 16 magnetic markers placed on the limbs and torso of the subject’s body. Obtaining 3D joint positions from this data is not trivial, however, as described in Section 3.1. We use a similar optimization technique
as our solution for 3D motion from 2D correspondences, with joint angle limits and key frames improving the solution. Figure 3-1 is a block-diagram of the inputs and outputs of the motion recovery algorithm.

Figure 3-1: Block diagram of inputs and outputs of the algorithm for recovering 3D motion from motion capture data.

Once we obtained an estimate of the 3D joint positions, we were able to use this as ground truth, with which to compare the results of our tracking algorithm on the corresponding video footage. Of course, the estimated joint positions are not actual ground truth, but only the best estimate we could obtain from the motion capture data.

### 3.1 Recovering Joint Positions

Unfortunately, 3D motion capture data does not give us the 3D joint positions directly, because the markers cannot not placed exactly at the joint centers. That would be impossible, because joint centers are not on the surface of the skin. The displacement from the joint center to the corresponding marker is unknown, as is the relative rotation. We recover as much of this unknown information as possible, to avoid human error in simply guessing the relative translation and rotation.

However, it is impossible to recover both translation and rotation of a single marker relative to a joint. Figure 3-2 shows a 2-DOF ball joint about a fixed rotation center, with a 3D marker measurement that has some unknown translation and
rotation relative to the joint. There are 6 marker measurements (translation and rotation), and 8 unknown states: the two degrees of freedom of the joint, as well as the 3 translational and 3 rotational degrees of freedom transforming marker state to joint state. Thus, any rotation of the ball joint is possible, with some solution for relative translation and rotation of the marker. For more complicated linkages of joints, there will be a family of possible solutions that fit the measurements.

Figure 3-2: Marker ambiguity. Translation from the joint to the marker has $x$, $y$, and $z$ components. The $z$-component is not visible because it is perpendicular to the image plane. Both the displayed rotations theta are possible with an arbitrary translation and rotation from the marker.

If, however, we fix $y$ and $z$, the translational component perpendicular to the joint, we constrain the solution (see Figure 3-2). We go from 8 unknown states to 6, and now 6 measurements are enough to provide a unique solution for our unknowns.

To recover the time-varying 3D pose of our kinematic model, along with the new time-constant states for marker translation and rotation, we use a variation of the batch solution of our 3D tracking algorithm. The algorithm is very similar to the figure tracking algorithm — in both cases we find a least-squares solution for a set of parameters with respect to a set of measurements. Now, however, we have the additional marker states, and the following additional measurements: 1) $z$-translation of each marker (in other words our markers are now 3D rather than 2D measurements), and 2) the 3D rotation of each marker.

We now have three main types of states: $\theta$, the model joint angles for each time step; $t$, the translation from each joint position to its marker; and $\hat{q}$, the rotation
of the marker relative to the joint, expressed as a quaternion. We use a quaternion because it is a singularity-free formulation for rotation, making estimation more stable. We have two types of measurements: \( p \), the 3D marker translation, and \( R_M \), the rotation matrix of the marker rotation. We express this as a matrix because it makes calculation of the Jacobian simpler. Thus we have six types of Jacobian terms for a particular link:

The Jacobian terms for change in position due to a change in model angle \( \theta \) are

\[
1.) \frac{\partial \mathbf{p}}{\partial \theta} = \begin{cases} \mathbf{a} \times \mathbf{p} & \text{links down the chain from } \theta \\ 0 & \text{elsewhere,} \end{cases} \quad (3.1)
\]

where \( \mathbf{a} \) is the axis of rotation of the link. (See Section 2.4.)

We can express \( p \) as \( R_G R_\theta \mathbf{t} \), where \( R_G \) is the global rotation matrix up to the link and \( R_\theta \) is the rotation matrix from angle \( \theta \). Thus:

\[
2.) \frac{\partial \mathbf{p}}{\partial \mathbf{t}} = R_G R_\theta \quad (3.2)
\]

\[
3.) \frac{\partial \mathbf{p}}{\partial \mathbf{q}} = 0. \quad (3.3)
\]

Finally, we can express \( R_M \) as \( R_G R_\theta R_\mathbf{q} \), where \( R_\mathbf{q} \) is the rotation matrix from the marker rotation. We therefore obtain our final three terms:

\[
4.) \frac{\partial R_M}{\partial \theta} = R_G \frac{\partial R_\theta}{\partial \theta} R_\mathbf{q} \quad (3.4)
\]

\[
5.) \frac{\partial R_M}{\partial \mathbf{t}} = 0 \quad (3.5)
\]

\[
6.) \frac{\partial R_M}{\partial \mathbf{q}} = R_G R_\theta \frac{\partial R_\mathbf{q}}{\partial \mathbf{q}}. \quad (3.6)
\]

Because we use DH-parameterization to express all of the local model rotations as rotations about the z-axis, we have

\[
R_\theta = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad (3.7)
\]
which we can easily differentiate with respect to $\theta$. We can express $R_q$ as

$$R_q = \begin{vmatrix}
q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(-q_wq_z + q_xq_y) & 2(q_wq_y + q_xq_z) \\
2(q_wq_z + q_xq_y) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(-q_wq_x + q_yq_z) \\
2(-q_wq_y + q_xq_z) & 2(q_wq_x + q_yq_z) & q_w^2 - q_x^2 - q_y^2 + q_z^2
\end{vmatrix},$$

where $\hat{q} = (q_w, q_x, q_y, q_z)$, which can also differentiate with respect to the quaternion terms [13].

### 3.2 Synthetic Tests

Before recovering 3D motion from real motion-capture data, we tested our method in simple situations to make sure that it actually converges to a unique solution. To test this, we used a simple one-joint model with synthetic motion capture data of a marker on the joint, for two time frames. The marker lay on the segment, at a distance of 10.0 away from the joint. We tested two different types of solutions: 1) attempting to recover all three components of marker translation $x$, $y$, and $z$), and 2) holding $y$ and $z$ fixed at zero and recovering only $x$, the component of translation directed along the length of the joint. In case 1) we initialized $y$ and $z$ at zero, but allowed them, but allowed them to diverge to different values. We initialized $x$ at different values, to see whether it converged to a unique solution.

In the first case, in which we recovered all three translational components, the algorithm did not converge to a unique solution. Different initial values for $x$ converged to different estimates of the model joint angle, $\theta$. This was possible because $y$ diverged away from the correct value of 0. This confirms our analysis in Section 3.1 — the solution is under-constrained if we attempt to recover $x$, $y$, and $z$.

In the second case, in which we held $y$ and $z$ to zero, the algorithm always converged on the correct solution of $x = 10.0$. This confirms that holding $y$ and $z$ constant makes the problem solvable. Figure 3-3 shows the recovered joint positions in both cases, and Table 3.1 presents the recovered translations.
Figure 3-3: Results of synthetic experiment for recovery of joint angle from motion capture data: recovered joint angle positions.

Table 3.1: Results of synthetic experiment for recovery of joint angles from motion capture data: recovered marker translation. Actual marker translation is (10, 0, 0).
3.3 Using Motion Capture to Evaluate 3D Reconstruction

This section presents a motion reconstruction of the human body, from experimentally-obtained magnetic marker data. We compare this calculation of motion with motion obtained using our 3D tracker (as presented in Chapter 2) on the corresponding 100-frame video sequence. We manually specified the 2D joint positions; our goal is to eventually use automatically obtained 2D correspondences. In both cases we used joint angle limits and key frames to improve the solution. The reconstruction from motion capture data is not strictly ground truth, as explained above, but it is the best estimate we can obtain. Thus we now have a method with which to evaluate our 3D tracker. We could use the same method to evaluate other 3D trackers on the same image sequence.

Row 1 of Figure 3-4 shows selected frames of the 2D video sequence. The wires connecting some of the magnetic markers are visible in the images. A total of 16 markers were placed on the body: 3 on each limb, 3 on the torso and one on the head. Row 2 of Figure 3-4 shows frames of an animation of the reconstruction from motion capture data, using the method presented in this chapter. Row 3 of Figure 3-4 shows frames of the reconstruction using the 3D tracker on the video sequence shown in Row 1.

Using motion capture data allows us to make a quantitative estimate of error. Figure 3-5 shows the recovered joint angles from both the 3D motion capture data and the video sequence, for three selected body joints. As can be seen, the 3D tracker produces results that roughly follow motion capture results, with some systematic offset errors. The 3D tracker results appear to be less stable, due perhaps to perspective effects that are not modeled by the 3D tracking algorithm.

Figure 3-6 shows the RMS joint angle error for all 16 limb joint angles, over the entire image sequence. The error does not appear to increase over time, which
Figure 3-4: Motion capture as an estimate of ground truth. Row 1: 2D input. Row 2: Motion reconstruction from 3D motion capture data. Row 3: Motion reconstruction from video using 3D tracker.
Figure 3-5: Plot of recovered joint angle over the image sequence, for three selected body joints. Top: left shoulder (side-to-side rotation). Middle: left hip (front-to-back rotation). Bottom: left knee (front-to-back rotation).
indicates that the algorithm is stable. The average error is about 0.09 radians, or 5 degrees.

Figure 3-6: Plot of RMS error for all 16 limb joint angles, over the image sequence.
Chapter 4

3D Kinematic Model Recovery from 2D Correspondences

A major limitation of the 3D motion recovery algorithm described in Chapter 2 is that it assumes knowledge of a 3D kinematic model — the connectivity of the model, the joint axes, and the link lengths. In fact, most current 3D tracking systems currently take such an approach. Alternatively, some systems (blob trackers) attempt to track moving regions with no kinematic model. However, these systems give little or no insight about the 3D pose of the figure.

An intermediate approach is to assume that there is a 3D kinematic model, but with some unknown parameters. We could specify the connectivity of a model, and attempt to recover the link lengths. To be even more general, we could assume the existence of a kinematic model, but with unknown connectivity. We could then use motion segmentation techniques to recover the connectivity.

One benefit of kinematic model recovery that it can reduce the number of degrees of freedom in the articulated motion reconstruction problem (presented in Chapter 2). Model recovery would allow us to use the simplest possible model. For example if we are tracking an arm that does not bend at the elbow in a particular sequence, model recovery would give us a model without an elbow joint. This would reduce the
reconstruction error that would arise from an additional degree of freedom. Another benefit of model recovery is that it allows tracking of non-human articulated objects, such as robotic arms, where the structure is unknown.

Little or no research has been devoted to the theoretic issues involved in kinematic model recovery. There is a body of work on motion segmentation, but this work does not use the constraints that arise when we assume the existence of a kinematic structure. This section explains an approach to the problem, and gives some preliminary results.

One major limitation of our approach to model recovery is that it requires fairly accurate 2D point correspondences. For now, we specify these manually, although it may be possible to obtain good enough correspondences with an algorithm such as optic flow. For kinematic model recovery to work in practice, it must be able to work using automatically obtained correspondences.

Section 4.1 details our algorithm for recovering a kinematic chain under several possible models for motion between segments. Section 4.2 presents the motion model we use, a ball joint model. Section 4.3 presents preliminary results of our algorithm. Finally, a summary of previous work in the area of motion segmentation is given in Section 4.4.

4.1 Recovering Model Connectivity

To recover the connectivity of a 3D kinematic model, we must segment the group of 2D correspondences into differently moving regions. We can use kinematic constraints to make this problem easier, assuming that different regions rotate relative to each other about fixed joints.

One approach to the problem of general motion segmentation is Torr's motion clustering algorithm [26]. Torr's approach uses the RANSAC (Random Sample Consensus) algorithm [6] to randomly select groups of points that move together in a
consistent manner, according to some motion model. He allows several possible mo-
tion models, and introduces a criterion for comparing the resultant error, to favor the
selection of simpler motion models. In Section 4.1.3, we describe additions to Torr’s
algorithm to allow recovery of kinematic models from a set of 2D correspondences.
Figure 4-1 is a block-diagram of the inputs and outputs to the algorithm.

![Block diagram of inputs and outputs of the 3D model recovery algorithms.](image)

Figure 4-1: Block diagram of inputs and outputs of the 3D model recovery algorithms.

### 4.1.1 The RANSAC Algorithm

The RANSAC algorithm is a method for choosing elements from a set to fit a model.
If we choose elements from a set $P$, and the model requires a minimum of $n$ elements
to solve for it, a description of the algorithm is as follows:

1. Choose a random subset $S$ of $n$ elements from $P$.

2. Instantiate a model $M$ with the elements $S$.

3. Choose a subset $S^*$ of all elements in $P$ within a small enough error tolerance
   of $M$.

4. If $||S^*||$ is larger than some threshold, use it to compute a new model $M^*$, and
   finish. Otherwise, return to Step 1.

By running the RANSAC algorithm repeatedly over a set of data, we can segment
the data into subsets that each fit a motion model (possibly with some outliers).
4.1.2 The Geometric Information Criterion

Torr’s algorithm allows the selection from several possible models, by using an information criterion to compare the relative errors. His Geometric Robust Information Criterion (GRIC) weights the error so as to favor more simple models. This is a way to combat overfitting, which would cause an unweighted error comparison to always select the model with more parameters. The GRIC is calculated as the sum over points $i$ as follows:

$$\text{GRIC} = \sum_{i=0}^{n} \rho(e_i^2) + \lambda_1 dn + \lambda_2 k,$$

where

$$\rho(e^2) = \min \left( \frac{e_i^2}{\sigma_i^2}, \lambda_3(r - d) \right),$$

where $e_i$ is the error and $\sigma_i$ is the standard deviation of each element of data $i$. The variable $n$ is the number of elements, $k$ is the number of model parameters, $d$ is the dimensionality of the model, and $r - d$ is the codimension. The constants $\lambda_1$, $\lambda_2$ and $\lambda_3$ are weighting factors.

For example, two motion models that can be used for segmentation are the fundamental matrix and the affinity transformation. The fundamental matrix encapsulates the change in image coordinates resulting from a change from one perspective camera viewpoint to another. If $x_1$ and $x_2$ are coordinates in the two images, the fundamental matrix $F$ can be expressed implicitly as $x_2 F x_1 = 0$. Thus it is a very general model, that can model any motion of a 3D object between two views. The affinity transformation, $x_2 = H x_1$, is much less general, covering only the motion of points lying on a plane, or rotations about the camera center. However, in these situations the solution for the fundamental matrix is degenerate — a family of solutions for $F$ is possible. This property is captured by the GRIC: the fundamental matrix has $d = 3$ and $k = 7$, whereas the affinity matrix has $d = 2$ and $k = 6$. Thus, a comparison using the GRIC will favor a solution using the affinity transformation, unless the fundamental matrix gives a significantly smaller error.
4.1.3 Kinematic Segmentation

We add to Torr’s method by recovering the kinematic connectivity between moving regions, over all available image frames. (Torr’s original framework only considers the motion between two frames.) By considering more frames in our RANSAC solution, we can recover motion using fewer points per segment.

Our algorithm for recovering kinematic structure begins by recovering the motion (rotation $R$ and translation $t$) of one segment that moves consistently between frames. It then adds segments that rotate relative to existing segments of the chain, recovering rotation $R$ about a center $c$. It considers the distance of $c$ from the endpoint of the existing segment, using this as part of the error metric. Following is an outline of the algorithm.

RecoverKinematicChain

1. Start with set of corresponding points between several images.
2. RANSAC to recover $(R, t)$ of first segment.
3. Recover endpoints of segment.
4. Add segment to chain.
5. Remove matched points.
6. While there are unmatched points
   
   For each joint $j$ of the chain recovered so far
   
   (a) Normalize points in each frame by $T_{j}^{-1}$, the inverse of the transform from the origin to $j$.
   (b) Use RANSAC to recover $(R, c)$ over all frames.
   (c) Recover endpoints of the segment.
(d) Add segment to chain.
(e) Remove matched points.

The RANSAC algorithm in Step 6b is modified to use proximity of points to weight the random choice, because points that are close together will tend to move together. It is also modified to include a penalty in the error returned for the distance of $c$ from the center of joint $j$.

One of the big problems with motion recovery in 3D is the normalization step — transforming the image points by $T_j^{-1}$, the transformation up to a joint $j$. Because the image points have an unknown depth component, the transformation $T_j^{-1}$ will not be correct unless it is a rotation purely within the image plane.

## 4.2 Ball Joint Motion Model

For our motion model we use a ball joint model, with two rotational degrees of freedom. Because we would like to use as few points per segment as possible, we are unable to use very general models such as the fundamental matrix. We use a motion model in which all points on a segment lie in a straight line in 3D. Limbs in an articulated structure such as the human body tend to be long and thin. Therefore motion correspondences on the limb, recovered with algorithm such as optic flow, would tend to lie along a line. This is a degenerate case which does not allow a solution for the fundamental matrix. In the alternative situation in which a user specifies segment positions manually, we also desire a simple motion model, to reduce the number of tracks the user must specify. In both cases, the ball joint model works well, because it assumes correspondences lie along a line, and it can be solved with few correspondences.

Another advantage of the ball joint model is that it allows us to circumvent the
normalization problem described in Section 4.1. We can ignore the rotational component of the transformation out of the image plane. The normalized motion is not the correct motion, but it still falls into the class of possible rotations of a ball joint. In other words, by leaving out part of the rotational component, we are only changing the amount of rotation about the ball joint center, without moving the joint center itself.

Figure 4-2: Ball joint. The shaded interior of the circle is the range of possible motion of the endpoint. $r$ is the radius of the segment, $\theta$ is the rotation within the image plane, and $\phi$ is the rotation out of the image plane.

For a point $j$ on a segment in image frame $i$, we parameterize the formula for image coordinates $x$ and $y$ under the ball joint model as:

$$x_{ij} = c_0 + r_j \cos \theta_i \cos \phi_i$$ (4.3)
$$y_{ij} = c_1 + r_j \sin \theta_i \cos \phi_i$$ (4.4)

where $r_j$ is the radius along the segment of point $j$, $\theta_i$ is the rotational component about the $z$ axis in frame $i$, and $\phi_i$ is the rotational component out of the image plane in frame $i$. Figure 4-2 shows the range of possible positions of the endpoint projected in 2D, which forms the interior of a circle. There are $2f + m + 2$ DOF, and we need $2fm > 2f + m + 2$ to find a unique solution. For $m = 1$ we cannot obtain a solution. For $m = 2$ we need $f > 3$ frames. For $m \geq 3$ we need $f > 1$ frames.

The ball joint model can in fact be expressed in terms of the SPM model [19] we used in Section 2 to parameterize 3D kinematic motion. In effect, we are recovering the joint positions of an SPM model, with two revolute joint rotating each segment:
1) the joint describing the angle $\theta$ within the image plane, and 2) the joint describing the angle $\phi$ out of the image plane.

An extension of our algorithm would be to add other candidate motion models. One possibility would be a revolute model with a known axis of rotation. We were unable to solve the normalization problem for this model, however, and leave this as a topic for future research.

4.3 Model Recovery Results

We tested our model segmentation algorithm on three simple synthetic examples. The input was a set of 2D correspondences. There were two image correspondences per segment. The output of the algorithm the kinematic chain and its motion. In the first experiment, a two-segment kinematic chain rotated in the $xy$ plane, over 3 image frames. In the second experiment, the same two-segment model rotated in the $yz$ plane, again over 3 image frames. In the final experiment, a one-joint chain rotated over 4 image frames, first by 45 degrees about the $y$ axis, and then by 90 degrees about line $x = y$. Thus, it rotated through all three image planes. The three experiments were chosen for simplicity, so that ground truth could be calculated and compared with the results.

Table 4.1 shows the results for the three experiments. As is shown, the recovered link lengths of the model were very close to the actual length. The RMS joint position error was calculated for all joint positions across all image frames. The accuracy of the recovered lengths and joint positions shows that the motion recovery algorithm works well in simple situations. Figure 4-3 shows an example result, the recovered model for experiment 1.

To extend the model recovery algorithm to work in real-world cases, for example recovering the kinematic structure of a human, more sophisticated methods may be needed. The algorithm would need to deal with noise in the 2D track positions, as
Table 4.1: Summary of results of synthetic model recovery experiments.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Joint 1</th>
<th>Joint 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>actual</td>
<td>recovered</td>
<td>actual</td>
</tr>
<tr>
<td></td>
<td>length</td>
<td>length</td>
<td>length</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>20.00</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>19.92</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>20.18</td>
<td>—</td>
</tr>
</tbody>
</table>

Figure 4-3: Example of results of model recovery algorithm. Recovered model and model motion over 3 image frames, for model recovery experiment 1 (rotation in the $xy$ plane).
well as the problem of recovering a much more complicated kinematic structure. One tool that might help would be dynamic prediction, as we use in our 3D articulated motion recovery algorithm (presented in Chapter 2).

4.4 Previous Work in Motion Segmentation

Many methods have been employed successfully to segment coherently moving regions in a set of images. For example, Boult and Brown [1] use Singular Value Decomposition to break the set of image correspondences into coherently moving regions. Jepson and Black [15] use EM to segment differently moving regions. Torr [26] develops an algorithm based on RANSAC to segment independently moving regions of points, according to one of several candidate motion models. Torr also gives a review of work in the area of robust model selection in [27]. This body of work attempts to fit the best of several candidate motion models to a set of points in a pair of images. None of the above research uses kinematic constraints in segmentation, however.

Kakadiaris and Metaxis [17] recover the position and link lengths of a human body model from multiple camera views. This approach is useful if the kinematic model is known, but does not recover model connectivity.

Rowley and Rehg [24] extend Jepson and Black [15] to articulated motion, locating the joint centers of a kinematic model of specified connectivity from 2D correspondences. They consider only the case of 2D motion, however. We were unable to locate a theoretical analysis of the problem of 3D kinematic motion segmentation, or an attempt to recover 3D kinematic model connectivity in practice.
Chapter 5

Conclusions

This thesis presents methods for 1) recovering 3D articulated motion from 2D image correspondences, 2) recovering 3D articulated motion from 3D motion capture data, and 3) recovering 3D model structure from 2D correspondences. Taken as a whole, these form a valuable framework for motion recovery from all available data. In general, the more information available, the better the possible solution. We use as much information as we can: 2D and 3D motion data, joint motion models, kinematic model structure, dynamics, joint angle limits, and key frames. By integrating all such data together we achieve reconstruction quality that is otherwise impossible. Our system is designed to allow the user to intervene when necessary to constrain the possibly ambiguous solution for 3D motion.

There remain significant obstacles to real-time 3D motion recovery in general situations. One difficulty in obtaining 3D body position is the lack of a good measurement of torso rotation in the SPM model. To allow the torso to rotate out of the plane of view in all possible directions, the 2D tracker must allow the torso template to deform along two axes — both horizontally and vertically. We could extend the SPM model to deal with this, but it would probably lead to increased instability in the 2D tracking step.

Another difficulty in using automatically obtained 2D correspondences for 3D
motion recovery is the inherent error in the correspondences due to image noise, clutter, and specular effects. A more sophisticated 3D tracking system would refine the estimation of the 2D correspondences along with the estimation of 3D motion. DiFranco and Kang use such an approach to refine the track used in Structure From Motion [5]. Furthermore, a more sophisticated 3D tracker would maintain multiple possible hypotheses for motion at each time step, thus avoiding local minima in the solution and allowing more complete user intervention. Cham and Rehg follow such an approach for 2D figure tracking [3]. Both of these are possible future additions to the 3D tracker.

In the area of kinematic model recovery, much more research is necessary to fully define the scope of the problem mathematically. Because this is a less constrained problem than articulated motion recovery, it is a much harder problem to solve. Dynamics, as used in our articulated motion reconstruction algorithm, would be useful to apply to kinematic model recovery.

As processor power and storage space increase exponentially, more and more things become feasible. However, vision algorithms must also improve the way that they merge several types of information to obtain the best possible solution. An ideal tracking system will merge 2D image information with expectations of how a human moves, based on learned kinematic and dynamic models. Hopefully, the tools presented in this thesis will be used in the future to produce a tracking system that realizes the goal of truly robust 3D tracking.
Appendix A

2D Kalman Filter Solution

The 2D tracker uses a Kalman filter framework, very similar to the one used for the solution of 3D motion from 2D joint positions (see Section 2.2). We use the same equations for prediction and estimation (Equations 2.1 and 2.2). In this case, however, the estimation step uses an iterative least-squares solution to register templates to the image, to determine the parameters of the kinematic model.

The estimation step is done using Gauss-Newton (GN) least-squares template registration. For each pixel on a joint, the Scaled Prismatic Model gives us a relationship between the change in model state, $\Delta q$, and the change in pixel brightness, $r$:

$$ r = J_1 \Delta q, \quad (A.1) $$

where

$$ J_1 = \nabla I^T J, \quad (A.2) $$

$\nabla I$ is the image brightness gradient, and $J$ is the Jacobian relating model parameters to pixel positions (see [19]). $J$ is a matrix, locally estimated at the state predicted by Equation 2.1. Equation A.1 is thus a linearization of Equation 2.2. This allows us to iteratively solve for the state. The equation for the state update, $\Delta q$, is given and derived in Appendix B.

A.1 2D Jacobian

To track the figure in 2D using the SPM model, we must find the Jacobian that expresses the relationship between pixel velocity and change in SPM model parameters. In the case of a revolute joint where $q$ is the angle of rotation, it will give an angular velocity to points further along the kinematic chain $\omega = \Delta q a$, where $a$ is the
axis coming out of the plane of view. Therefore, the image velocity, \( v \), of a point at position \( p \) somewhere down the kinematic chain can be expressed as:

\[
v_p = P(\omega \times p) = P(a \times p)\Delta q = p_{2D}\Delta q,
\]

where \( P \) is the orthographic projection into the 2D plane, and \( p_{2D} \) is the 2D projection of the 3D pixel position. Therefore we have the components of the Jacobian \( J \) due to joint \( i \) for each point \( j \) on link \( k \):

\[
J_{ij} = \begin{cases} 
0 & \text{links } k, \text{ where } k < i \\
p_{2D} & \text{links } k, \text{ where } k \geq i
\end{cases}
\]

(A.4)

In the case of a prismatic joint, let \( q \) be the extension of the joint. For a pixel on the link corresponding to the joint, the velocity of the pixel will be proportional to its position along the link. The pixel velocity is therefore

\[
r = bq\Delta qn,
\]

where \( b \) is the fractional position of the pixel along the link, and \( n \) is the velocity of the endpoint of the link. For a pixel on a subsequent joint on the chain, the velocity is the same as the velocity of the endpoint of the link:

\[
r = q\Delta qn.
\]

(A.6)

Therefore the Jacobian terms due to joint \( i \) for all points \( j \) on link \( k \) are:

\[
J_{ij} = \begin{cases} 
0 & \text{links } k, \text{ where } k < i \\
bq \cdot n & \text{link } i \\
q \cdot n & \text{links } k, \text{ where } k > i
\end{cases}
\]

(A.7)
Appendix B

Derivation of Gauss-Newton Update Equation

Consider the model

\[ r = J\Delta q + \varepsilon, \]  

where \( \varepsilon \) is Gaussian noise. If

\[ R = E[\varepsilon\varepsilon^T] \]

is our expectation of state error, then we have the probability of a particular error

\[ p(\varepsilon) = ae^{\frac{1}{2}\varepsilon^TR^{-1}\varepsilon} \]

under the Gaussian model. We minimize this probability by solving

\[ \min_\varepsilon \varepsilon^TR^{-1}\varepsilon \Rightarrow \]  

\[ \min_\varepsilon (r - J\Delta q)^TR^{-1}(r - J\Delta q) \Rightarrow \]  

\[ d/d(\Delta q) = (-J)^TR^{-1}(r - J\Delta q) = 0 \Rightarrow \]  

\[ J^TR^{-1}J\Delta q = J^TR^{-1}r \Rightarrow \]  

\[ \Delta q = (J^TR^{-1}J)^{-1}J^TR^{-1}r \]

This is the correct solution for \( \Delta q \) in a least-squares sense.
Bibliography


72


