Simultaneous Camera Calibration and Pose Estimation from Multiple Views

by

Tomáš Ižo
B.S., Computer Science
Yale University, 2001

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2003

© Massachusetts Institute of Technology 2003. All rights reserved.

Author .................................................................
Department of Electrical Engineering and Computer Science
August 29, 2003

Certified by ..................<
W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Accepted by ........... Arthur C. Smith
Chairman, Department Committee on Graduate Students
Simultaneous Camera Calibration and Pose Estimation from Multiple Views
by
Tomáš Izo

Submitted to the Department of Electrical Engineering and Computer Science on August 29, 2003, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

In this thesis, we present an algorithm to estimate the body pose of a walking person given synchronized video input from multiple uncalibrated cameras. We first construct an appearance model of human walking motion by generating examples from the space of body poses and camera locations using a synthetic model. The examples are then clustered using expectation-maximization to make the appearance model more compact. Given an input video sequence segmented using background subtraction, we find the most closely matching appearance cluster for each silhouette and use the sequence of matched clusters to extrapolate the position of the camera with respect to the person’s direction of motion. For each frame, the matching cluster also provides us with an estimate of the walking phase. We combine these estimates from all of the views and find the most likely sequence of walking poses using a cyclical, feed-forward hidden Markov model. The strength of our algorithm is in the fact that it requires no manual initialization and no prior knowledge about the locations of the cameras. We present results both on synthetically generated data as well as live recorded video sequences.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering
Acknowledgments

I would like to thank my advisor, Eric Grimson, for his support, guidance and boundless patience while I worked through my first project of this kind. Many thanks, also, to Lily Lee for numerous helpful conversations, during which some of the basic ideas underlying this thesis were developed. Finally, thanks to my friend Josh Migdal for helping to keep me sane throughout the process and, of course, for the great silhouettes!
Contents

1 Introduction ........................................ 5
  1.1 Articulated Tracking ............................. 6
  1.2 Identification Using Motion ................. 7
  1.3 General Framework ............................. 9

2 Motion Appearance Model .......................... 10
  2.1 Generating a Synthetic Data Set ............. 10
  2.2 From Examples to Clusters ................. 13
    2.2.1 The Expectation-Maximization Algorithm 14
    2.2.2 Applying EM to the Clustering Problem 15
    2.2.3 Using EM to Cluster the Appearance Model 17
    2.2.4 Life Made Simpler: K-Means Clustering 19
  2.3 Final Considerations ......................... 19

3 Modeling Motion Dynamics Using Hidden Markov Processes 24
  3.1 Discrete Markov Chains ..................... 24
  3.2 Hidden Markov Models ...................... 25
    3.2.1 The Evaluation Problem ............... 26
    3.2.2 The Decoding Problem .................. 27
    3.2.3 The Learning Problem .................. 28
  3.3 The Walking HMM ............................ 28

4 Simultaneous Camera Calibration and Pose Estimation 33
  4.1 The Algorithm .................................. 33
    4.1.1 Input .................................. 33
    4.1.2 Output ................................ 33
    4.1.3 Mapping Inputs to Outputs ............ 34
    4.1.4 Data Acquisition ....................... 36
    4.1.5 Segmentation .......................... 36
  4.2 Matching Silhouettes and Clusters .......... 36
    4.2.1 Estimating the Camera Pose .......... 38
  4.3 Experiments .................................. 39

5 Summary ............................................ 48
Chapter 1

Introduction

In light of the increasing feasibility of seamless human-computer interfaces and automatic monitoring systems, tracking and analysis of human motion have received much attention in the computer vision community over the past several years. Since environments such as intelligent rooms and aware homes require as much information about their occupants as is possible to extract, the demands on tracking and monitoring systems have significantly increased. Consider a home or office setting equipped with means of collecting visual information from multiple sensors. The possibilities for the use of such information, if it is accurately analyzed, are virtually limitless. For instance, various parameters of the environment, such as temperature, lighting or the configuration of devices displaying information could be automatically altered based on the identity of the users or the type of activity being performed. Usual patterns of activity could be analyzed so that it is possible to isolate unusual or suspicious ones. Much of the information necessary for these tasks could be obtained by accurate analysis of human motion. By analyzing a person’s movements, it may not only be possible to distinguish different types of activity, but perhaps also to identify the subject or even determine the person’s mood. Does a person’s emotional disposition affect the way he or she moves? Is it possible to detect such things as confusion or frustration? Only by gaining further insight into the characteristics of human motion can we start answering these questions.

Articulated kinematic models have proven to be a very useful tool in tracking of human motion, as they are able to describe the observed movements in a concise way using the relative positions and angles of a small number of rigid (or elastic) parts. Since the locations of major body joints, when expressed as functions of time, fully describe the performed motion, it is reasonable to believe that extracting these locations from video input would provide us with valuable information about the characteristics of the motion. According to a well-known study from the 1970s, joint positions alone may provide enough information for a human observer to correctly classify the type of motion he or she is witnessing [10].

This thesis describes a system that lays the foundation for extracting such 3D information from video streams of walking people. We present an algorithm which, given video input from several synchronized, uncalibrated cameras, is able to estimate the relative position of each camera with respect to the person, as well as the body
pose (configuration of main body parts) of the person in each incoming frame. We believe that such information will serve as a good starting point for further research in human motion analysis ranging from human recognition using gait to intelligent rooms, gesture sensing, activity monitoring and classification.

1.1 Articulated Tracking

Much of the previous research in articulated tracking has been concerned with a single camera view. Cham and Rehg [3] developed a multiple hypothesis tracking method that used a highly articulated 2D Scaled Prismatic Model to track human motion. A scaled prismatic model is a tree structure of components, each of which represents a rigid part of an articulated object's appearance in the image plane. Motion of the parts in the image plane is modeled by a set of kinematic constraints for each pair of parts connected by a joint, while the foreshortening associated with motion outside of the image plane is modeled by elasticity (stretching or shrinking) of each part. Scaled prismatic models are useful for modeling of the 2D appearance of motion because they have fewer singularity problems than fully three-dimensional models and only require knowledge about the 2D kinematics [14]. Cham and Rehg [3] encoded the probability distribution of a scaled prismatic model of a human figure by keeping track of the peaks of the distribution and modeling their local neighborhoods with Gaussian densities. This multimodal approach made it possible for the tracker to keep multiple hypotheses of the body part configurations while some of the parts occluded each other and thus continue tracking after the occlusion passed. However, a single view did not provide enough information to resolve the ambiguities in the frames in which the occlusions occurred.

Bregler and Malik [2] represented 3D motion of articulated body parts by kinematic chains, in which the projection of complex motion onto the image plain is modeled by a series of twists (rotations and translations) of the individual rigid parts around their connecting joints. They then developed the framework for tracking these kinematic chains in a sequence of images. Their approach, much like most articulated tracking systems, depends on manual initialization in the first frame.

Zhao, Nevatia and Lv [21] used 3D motion capture data to create an articulated model of a walking person. Given a calibrated camera, they used background subtraction along with an ellipsoid shape model to segment out human-like moving objects from the video input. The 3D direction of motion was estimated using the known position of the camera with respect to the ground plane. Each object was then compared to a sequence of optical flow templates generated by the articulated model as rendered from the corresponding view. Optimizing over the sequence of motion templates, it could be determined whether the foreground blob was a walking person and, if so, estimate the walking phase in each frame.

Howe, Leventon and Freeman [9] used 3D motion capture to collect a set of motion snippets: short sequences of 3D coordinates of various points on the body in successive frames of various types of human motion. They then used a clustering technique to divide the motion snippets into a number of groups and thus estimate the probability
distribution over the snippets by a mixture of Gaussians. Given a single-camera video sequence with a number of manually initialized image points, they were able to estimate the depth of each point by finding the maximum likelihood estimate for the motion snippet corresponding to a short sequence of frames from the input.

One of the most daunting difficulties that stems from using a single camera is the inherent ambiguity of monocular data. Consequently, much of the research mentioned above was only concerned with tracking in two dimensions or using learned information about human motion to infer three-dimensional data. However, this kind of inferred data, while qualitatively correct, may not be precise enough to be used for the purposes of recognition and analysis. To alleviate this problem, several research groups have employed multiple camera setups.

Mikic et al. [13] used four calibrated cameras and a voxel carving technique to extract a volumetric model of the observed person. They then located the major body parts sequentially using volumetric templates. A spherical template within a given size range was used to locate the person’s head, while the best match for a cylindrical template of the correct proportions was deemed to be the location of the torso. Limbs were found by dividing the remaining voxels into four regions and matching each region to two cylindrical templates. The extracted articulated model then served as the initialization of a Kalman filter-based tracker.

Cohen et al. [4] utilized two synchronized cameras to fit a non-articulated 3D model to a set of input images of a person. They used background subtraction to segment out silhouettes of the person and then took advantage of the fact that the medial axes of the silhouettes correspond to the same location in space, making it possible to estimate the epipolar geometry of the cameras. Finally, the 3D model was constructed by fitting to the silhouettes a generalized cylinder, described by its medial axis and a set of radii along that axis.

A common assumption in multiple view setups is that the cameras can be easily calibrated, i.e. at least their relative positions can be well established. While this is often true in experimental settings, eliminating this requirement would make the resulting algorithms much more flexible and consequently more easily applicable in real world scenarios. An intelligent room might have several cameras, the positions of which may change depending on the furniture arrangement. Furthermore, it may be the case that no two cameras are positioned so that a stereo algorithm can take advantage of their epipolar lines.

Finally, an overwhelming majority of published research dealing with the tracking of human motion assumes that some sort of manual or semi-automatic initialization of the algorithm can be performed by the user.

1.2 Identification Using Motion

Our main motivation was provided by the recent research efforts in human recognition by gait and posture. Lee and Grimson [11] showed that a significant amount of information about a person’s gait can be extracted from a single-view binary silhouette obtained using background subtraction. Their approach involved registering and
normalizing the silhouette and subsequently segmenting it into a number of fixed, rectangular regions. The top portion of the silhouette was designated the head region, while the rest of the silhouette was divided into two parts along the medial axis. Each of the two parts was then further subdivided into three sections, corresponding to the upper, middle, and lower part of the body. Each region was represented by the following statistical properties:

- Centroid of the region.
- Ratio of the length of the major axis to the length of the minor axis of the ellipse best fitting the region.
- Orientation of the major axis of the ellipse.

These properties, over all of the regions, comprised the feature vector for the silhouette. The entire walking sequence was then characterized in a concise way by:

- Mean and standard deviation of the silhouette features over the entire sequence.
- Phases of the silhouette features with respect to the period of the walking cycle.

Recognition was done by indexing into a database of known subjects and computing distances between the observed feature vector and the feature vectors in the database. Phillips et al. [15] used a different silhouette-based approach, in which—instead of computing features of the silhouettes—they computed distances between sequences by comparing the silhouettes directly frame by frame. Both of these approaches produced promising results. However, both were sensitive to change of clothing as well as change of environment. Furthermore, they required a canonical (in this case, side) view of the person. Shakhnarovich et al. [19] circumvented this problem by using multiple calibrated cameras to construct a three-dimensional volumetric model of the person, then to compute the trajectory of motion and use the information to generate a desired canonical view.

We believe that the recognition rate of gait identification algorithms could be significantly improved if the positions of major body joints were used to compute features that represent a person’s gait. Such features would paint a much more detailed picture of the dynamic and kinematic properties of the observed motion. If the joint positions could be recovered in 3D, they would account for certain characteristics of gait, such as side to side motion of shoulders and knees, which cannot be observed from a side view. Moreover, being able to fit a 3D articulated model to video sequences of a walking person would also provide us with valuable insight into the limitations of identification by gait as well as into the factors that may influence a given person’s gait characteristics.

The research presented here lays the groundwork for such detailed motion analysis. Our estimates of the body pose and camera calibration could serve as starting points in a local search of body configurations leading to a more precise estimate of 3D joint positions. The advantage of the system lies in the fact that no prior camera calibration or manual initialization is needed, which greatly simplifies the practical implementation.
1.3 General Framework

We tackle the problem of simultaneous camera calibration and walking pose estimation in two main stages. First we construct a model of both the appearance and the dynamics of a human walk. We use our own as well as commercially available software to break down the walking cycle into a number of phases, and then sample from the 3D space around the person to generate examples of the 2D appearance of each walking phase. We then cluster these examples into a number of groups in order to make the appearance model more compact. We model the periodic dynamics of the walking cycle using a first-order, feed-forward, cyclical Hidden Markov Model (HMM). Chapter 2 describes and compares the various clustering methods we used and Chapter 3 gives an overview of Hidden Markov Models and associated algorithms.

Second, we develop an algorithm to fit the walking motion model to video sequences from multiple synchronized cameras. For each camera input, we apply a background subtraction algorithm to obtain a sequence of silhouettes corresponding to that viewpoint. We then use a distance metric to find the appearance model cluster which best matches each silhouette. Given these matches, we hypothesize the position of each camera with respect to the person. Finally, we use the matches from all cameras as input into the dynamic HMM model in order to hypothesize the walking phase at each frame in the input sequences. We give the detailed description of the algorithm and present some experimental results in Chapter 4. The final chapter contains the conclusions as well as the proposed direction of further research.
Chapter 2

Motion Appearance Model

In this chapter, we introduce our model of the appearance of a walking human. Our main goal in constructing such a model is to capture the appearance of the motion in a way that is invariant to the position of the observer with respect to the subject. Given a set of observations from a camera, this property of the model enables us to simultaneously estimate the relative position of the camera (camera pose) as well as the configuration of the body (body pose).

2.1 Generating a Synthetic Data Set

To model a phenomenon—in our case, the appearance of a certain type of motion—is to arrive at a description of its essential properties. The conciseness of such a description determines both the size of the model as well as the level of detail to which the model characterizes the phenomenon. To express these ideas more formally, let us assume that we have a set of $n$ observations $X = \{x_1, ..., x_n\}$. Our goal is to model the phenomenon which led to these observations by a model, which can be described by the parameter vector $\Theta$. Given this set of parameters, we can define the likelihood of the observed data given the model as the following joint density:

$$p(X|\Theta) = \prod_{i=1}^{n} p(x_i|\Theta)$$

(2.1)

We wish to find values of the parameter vector $\Theta$ for which the above likelihood is maximized. We refer to such an estimate as the maximum-likelihood estimate of $\Theta$ and denote it by $\hat{\Theta}$:

$$\hat{\Theta} = \arg\max_{\Theta} p(X|\Theta)$$

(2.2)

If the purpose of the model is to describe certain specific properties of the phenomenon, it is important that those properties are captured in sufficient detail. Our goal is to describe the appearance of a walking human as seen from an arbitrary perspective (within a defined range). Thus we need to concentrate on the properties of the appearance which vary not only with respect to elapsed time, but also with respect to the position of the observer. In the case of a walking human, they are the
Figure 2-1: An example of a walking human in several phases of the walking cycle (top) matched against the most closely corresponding phases of a simple OpenGL stick-figure model (bottom). Here, the matching was done by a human observer. Our objective is to be able to do the same automatically using the added advantage of multiple points of view.

following:

1. **Body pose.** This is, in effect, the configuration of body parts at any given time as the motion progresses. Since the human walking motion is periodic, with the period equal to the time necessary to take two full steps, there is (within some margin of error) a one-to-one mapping between the phase of the motion and the body pose. As suggested by previous research in human identification by gait, this mapping varies from person to person and is, possibly, unique to every individual. However, the appearance of a walking person is sufficiently characteristic that a human observer can easily identify the phase of the walking cycle from even a single image (see figure 2-1).

   This leads us to believe that we can model the phase of the walking cycle with a general model, and that using the model to estimate the phase for a given observation will provide us with a good initial estimate of the body pose.

2. **Camera pose.** This is simply the position of the observer (in our case, a camera) with respect to the person. More specifically, it is the linear transformation which characterizes the mapping from the camera reference frame to the person’s reference frame. In camera calibration, the parameters which describe this mapping are often referred to as the external parameters of the camera.
3. **Internal camera parameters.** These parameters, such as the coordinates of the principal point, the focal length of the camera and the effective pixel sizes, affect the observation, and thus need to be taken into account.

Perhaps the most straightforward motion appearance model which accounts for variations in the above parameters is simply a collection of sample observations chosen systematically so that the entire range of each parameter is well represented. To generate such a data set under the most controlled conditions, we used Curious Labs' Poser 5, a commercially available program capable of modeling the human body in an anatomically correct fashion as well as simulating common types of motion such as walking and running. Using this software, we generated examples by varying two of the above sets of parameters as follows:

1. We divide the full walking cycle, i.e. the space $B$ of body poses, into 20 phases, denoted $\psi_1$ through $\psi_{20}$. Based on our observations, a typical human walking cycle takes between 1 and 2 seconds to complete. Our recording equipment operates at 15Hz, and thus it would typically produce around 20 images from a single walking cycle.

2. Let $C$ be the centroid of the person’s body and $O$ be the perpendicular projection of the centroid onto the ground plane. Then let $d$ be the vector indicating the direction of motion. The position of the camera with respect to the person is specified by the pair $(\theta, \phi)$, where $\theta$ is the angle between $d$ and the line $OB$, where $B$ is the perpendicular projection of the camera’s origin onto the ground plane. $\phi$ is the angle between the plane crossing the centroid $C$ and parallel to the ground plane and the line $CO'$, where $O'$ is the origin of the camera reference frame (see figure 2-2)

For the purposes of our data set, we set the following conditions on the space $L$ of camera poses:

$$\theta \in (0, 2\pi)$$
$$\phi \in (-\frac{\pi}{6}, \frac{\pi}{3})$$

We restrict the range of $\phi$ in order to ensure that the view of the person is informative: views from extreme camera locations above and below the person may not provide enough information about the body pose.

We sample from each of these ranges at intervals of $\frac{1}{30}$, resulting in 864 different camera positions uniformly distributed over the space of all allowed camera poses.

Sampling from the space $B \times L$ as defined above results in 17,280 different examples. Example $i$ of the data set consists of the quadruple $(s, \psi, \theta, \phi)_i$, where the latter three parameters are the body phase and camera phase, as described above, and $s$ is the silhouette of the human body rendered in the given phase as seen from the given camera position. We use silhouettes rather than images for the following reasons:
Figure 2-2: The camera position is determined by the pair $(\theta, \phi)$, where $\theta$ is relative to the direction of motion and $\phi$ is relative to the ground-parallel plane crossing the centroid of the person.

1. Silhouettes are easy to extract from the input using motion segmentation techniques.

2. Silhouettes are invariant with respect to color and pattern of clothing, color of skin, lighting of the scene, and many other parameters. While two images of different people in the same body pose as observed from the same view might look quite different, the corresponding silhouettes would look similar.

In essence, the silhouette $s$ is the observation and the triple $(\psi, \theta, \phi)$ is the model parameter vector $\Theta$ from equation 2.1 corresponding to the observation. We represent our silhouettes as binary images of approximately 100x300 pixels.

### 2.2 From Examples to Clusters

While an appearance model consisting of a collection of examples would certainly be a valid one for our purposes, it would hardly be practical. Consider the problem of fitting the model to a set of observations, i.e. solving equation 2.2 above. Given each observation, we want to find the most likely parameters which may have led to that observation. Since the probability $p(x_i|\Theta)$ is large when the distance between the observed data $x_k$ and the data generated by the model with the parameters $\Theta$ is small, the problem amounts to searching through the entire collection of examples and finding the closest match according to some distance metric. The time-complexity of such a search is linear with respect to the size of the model. Consequently, in order for the model to be more practical, we need some way of making it more compact.

A simple way to approach this task is to group the examples into a smaller number of clusters and develop a distance measure to compare clusters to observations. It is
useful to notice that this is, in fact, very similar to the modelling problem discussed above. Given a set $D = \{d_1, ..., d_n\}$ of data points—in our case, our collection of examples—and some number $N$, we need to sort the data into $N$ groups in a way which is optimal in the maximum-likelihood sense. Let us assume that the data points were drawn independently from a mixture density specified by a set of parameters $\Theta = \{\theta_1, ..., \theta_N\}$ such that

$$p(d_i|\Theta) = \sum_{j=1}^{N} p(d_i|l_i = j, \theta_j) P(l_i = j)$$

(2.3)

where $l_i = j$ denotes the event that the data point $d_i$ was drawn from the $j$-th component of the mixture.

If the data points can indeed be grouped into $N$ clusters, this is a reasonable assumption, as each of those clusters would naturally correspond to a component of the mixture density. Thus, the problem is to find the unknowns $N$ and $\Theta$. Without any further assumptions, this would be a dauntingly difficult task. Luckily, we can use insight from previous research to make our life a little easier. As we mentioned in section 1.1, Rosales and Sclaroff assumed a normal mixture density in a similar clustering problem in [17] and [18] and produced promising results. This assumption turns the above mixture equation (2.3) into the more specific form

$$p(d|\Theta) = \sum_{j=1}^{N} \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu_j)^T\Sigma_j^{-1}(x - \mu_j)\right] P(j)$$

(2.4)

where the parameter vector $\Theta$ contains the means and covariance matrices of all of the mixture components as well as the prior over the components. As before, we are again looking for a maximum-likelihood estimate $\hat{\Theta}$ such that equation 2.2 holds. A technique commonly known as Expectation-Maximization lends itself readily to this type of optimization problem.

### 2.2.1 The Expectation-Maximization Algorithm

Let us first discuss the Expectation-Maximization method in the its most general form\(^2\). Let us assume that we have a set of samples $D = \{d_1, ..., d_n\}$ taken independently from the same distribution. We want to find the parameters $\Theta$ describing this distribution. However, to complicate matters, instead of observing the full set $D$, we can only see a corrupted data set with some of the features missing. We can think of this as the data set containing some good features and some bad (missing) ones. Grouping all the good features into a separate set $D_g$ and the missing ones into the set $D_b$, we have $D = D_b \cup D_g$. We can then define the following function, often

---

\(^1\)To make our notation simpler, we will from now on write $P(j)$ instead of $P(l_i = j)$, unless it is not clear from context to which random variable we are referring.

\(^2\)For a good introduction to EM and its application to clustering, see [1] or [6]. Much of the material in this chapter is based on these two sources.
referred to as the central equation of expectation maximization [1]:

$$Q(\Theta^t, \Theta^{t-1}) = \mathcal{E}_{D_b}[\ln p(D_g, D_b|\Theta^t)|D_g, \Theta^{t-1}]$$

(2.5)

We can think of this function as a way of taking the estimated parameters $\Theta^{t-1}$ at time $t - 1$, and using them to evaluate a new, improved estimate $\Theta^t$ at time $t$. It therefore makes sense that the current estimate $\Theta^{t-1}$ be treated as a constant in the above expression. Similarly, the “good” data features $D_g$ are treated as constant as well, whereas the bad feature vector, $D_b$, is a random variable. We can then interpret the right side of equation 2.5 as evaluating the expected log-likelihood of the new parameter vector given the full data set, but marginalized with respect to the previous best estimate and thus evaluating the two against each other. This evaluation is called the E-step of the Expectation-Maximization algorithm.

Having developed a measure to compare new parameter estimates to previous ones, we now need to select a new estimate which maximizes the function $Q$ as defined above. We can express this relationship formally:

$$\Theta^t = \arg \max_{\Theta} Q(\Theta, \Theta^{t-1})$$

(2.6)

This optimization is referred to as the M-step of the EM algorithm. After an initial estimate for the parameter vector $\Theta$ is chosen, the E-step and M-step are alternated to iteratively improve the estimate. One of the key properties of the EM algorithm is the fact that it guarantees the log-likelihood of the estimated parameter vector, given the full data to monotonically increase [5].

A concrete implementation would be hopeless unless we put some assumptions on the probability density of the entire data set. As we mentioned in the previous section, we are going to assume that this density function takes on the form of a mixture of multivariate Gaussian distributions, as in equation 2.3.

### 2.2.2 Applying EM to the Clustering Problem

The EM algorithms lends itself readily to be applied to our clustering problem, if we think of each element of the full data set $D$ as the pair $(x, l)$, where $x$ is an example observation and $l$ is the label of the cluster to which the observation belongs. In effect, the observations are our good features and the labels are our missing data. We can now assume that the missing labels form a vector $l$ of independent random components governed by the probability distribution $p(l|\mathcal{X}, \Theta) = \prod_{i=1}^{n} p(l_i|x_i, \Theta)$, where $\mathcal{X} = \{x_1, ..., x_n\}$. Having made this assumption, incorporating equations 2.3 and 2.1 into equation 2.5 results in the following:

$$Q(\Theta^t, \Theta^{t-1}) = \sum_{l \in \Lambda} \ln p(\mathcal{X}, l|\Theta^t)p(l|\mathcal{X}, \Theta^{t-1})$$

$$= \sum_{l \in \Lambda} \ln \prod_{i=1}^{n} P^t(l_i)p(x_i|l_i, \Theta^t_i) \prod_{i=1}^{n} p(l_i|x_i, \Theta^{t-1})$$
In the above equation, \( P^t(l_i) \) is the prior probability of the \( i \)-th mixture component at iteration \( t \), \( p(x_i|l_i, \theta^t_i) \) is the density of the \( i \)-th component with the parameter vector \( \theta^t_i \) and \( \Lambda \) is the space of all possible label assignments.

After some algebraic manipulation, we can express equation 2.7 as a sum of the following two terms:

\[
Q(\Theta^t, \Theta^{t-1}) = \sum_{j=1}^{N} \sum_{i=1}^{n} \ln(P^t(j)p(j|x_i, \Theta^{t-1})) + \sum_{j=1}^{N} \sum_{i=1}^{n} \ln(p(x_i|j, \theta^t_i)p(j|x_i, \Theta^{t-1}))
\]  

(2.8)

Since we are assuming that the \( N \) mixture components have a multivariate normal density, the term \( p(x_i|j, \theta^t_i) \) from the above equation expands to the following:

\[
p(x_i|j, \theta^t_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_j^t|^{1/2}} \exp\left[-\frac{1}{2}(x_i - \mu_j^t)^T(\Sigma_j^t)^{-1}(x_i - \mu_j^t)\right]
\]

Having thus simplified the E-step, we are finally ready for the M-step of the algorithm. We want to minimize the function \( Q \) from equation 2.8 with respect to the new parameter estimate \( \Theta^t \). This parameter vector consists of the following components:

\[
\Theta^t = \bigcup_{j=1}^{N} \{(P^t(j), \Sigma_j^t, \mu_j^t)\}
\]

where \( P^t(j), \Sigma_j^t \) and \( \mu_j^t \) are, respectively, the new estimates of the prior, covariance matrix and mean of the \( j \)-th mixture component.

We can approach this optimization problem by minimizing each of the two terms on the right side of equation 2.8 separately. The first term yields the solution for \( P^t(j) \) after setting its partial derivative with respect to \( P^t(j) \) to 0 and adding the normalizing constraint \( \sum_{i=1}^{N} P^t(j) = 1 \). The resulting update equation for \( P^t(j) \) is as follows:

\[
P^t(j) = \frac{1}{N} \sum_{i=1}^{n} p(j|x_i, \Theta^{t-1})
\]  

(2.9)

Differentiating the second term of equation 2.8 with respect to \( \Sigma_j^t \) and \( \mu_j^t \) yields the remaining two update equations:

\[
\mu_j^t = \frac{\sum_{i=1}^{n} x_i p(j|x_i, \Theta^{t-1})}{\sum_{i=1}^{n} p(j|x_i, \Theta^{t-1})}
\]  

(2.10)

\[
\Sigma_j^t = \frac{\sum_{i=1}^{n} p(j|x_i, \Theta^{t-1})(x_i - \mu_j^t)(x_i - \mu_j^t)^T}{\sum_{i=1}^{n} p(j|x_i, \Theta^{t-1})}
\]  

(2.11)

These three update equations express simultaneously the E-step (evaluation of the \( Q \) function) and M-step (maximizing the \( Q \) function with respect to the new parameters) of the EM algorithm. They make implementation of the algorithm relatively...
straightforward: generate an initial guess for the parameters $\Theta^0$ and then repeat updating the parameter values until some threshold condition is met. Perhaps the only tricky part is evaluating the term $p(j|x_i, \Theta^{t-1})$ from equations 2.9-2.11. However, a simple application of Bayes’ rule gives us:

$$p(j|x_i, \Theta^{t-1}) = \frac{p(x_i|j, \Theta^{t-1})\rho^{t-1}(j)}{\sum_{j=1}^{N} p(x_i|j, \Theta^{t-1})\rho^{t-1}(j)}$$

where $\rho^{t-1}(j)$ is the previous prior and $p(x_i|j, \Theta^{t-1})$ the previous density for the j-th mixture component.

### 2.2.3 Using EM to Cluster the Appearance Model

Let us now return to the clustering problem as it applies to our appearance model. In order to make our model—thus far a large collection of examples—more compact, we can use the expectation-maximization method to group the examples into a number of clusters, provided that we assume those clusters have normal density. We could do this by simply taking the good feature vectors $x_i$ to be the silhouettes $s$ from our example quadruples $(s, \psi, \theta, \phi)_i).$ However, while doing this would be the most straightforward way to incorporate the true appearance of the motion into our model, it would make the maximum-likelihood estimation of the cluster parameters computationally expensive due to the large feature vectors and covariance matrices. We can instead take advantage of the fact that we have some higher order information, namely the 2D locations of the major body joints which, ultimately, led to the corresponding rendering of the silhouettes (see Figure 2-3 for illustration).

These projected 2D joint coordinates form feature vectors of merely 26 elements each: two for each of the 13 major joint positions, as described in figure 2-3, making the computation much more feasible. One of the possible objections to doing this is the fact that we would effectively be clustering based on the information which we want to use the model to extract, rather than based on pure appearance. However, it is our intuition that the positions of the major body joints are so closely tied to the appearance of the silhouette that clustering based on the joint position will, in fact, group examples with similar silhouettes together. In fact, if this were not the case, we would have a very hard time using the appearance model to extract information about body pose. Clustering based on 2D joint locations was shown to produce good results in [17]. Using the projected 2D coordinates of the main body joints, the clustering algorithm functions as follows:

1. We initialize the computation by “guessing” the first estimate of the parameter vector, $\Theta^0$. We first randomly select $N$ examples to serve as estimates for the mean vectors $\mu_j^0$. We cluster the remaining examples into $N$ groups by assigning example $x_i$ to cluster $j$ whenever $j = \text{argmin}_k(||x_i - \mu_k^0||)$, i.e. we assign each example to the cluster based on distance from the cluster mean. We then calculate the covariance matrix for each of the resulting clusters, thus obtaining our estimates for $\Sigma_j^0$. Finally, the prior estimates $P(j)$ are set to the
Figure 2-3: Left: a Poser rendering; center: the corresponding silhouette; right: a plot of the projected 2D coordinates of the following main joints: neck, shoulders, elbows, wrists, hips, knees and ankles.

ratio of the size of the corresponding cluster to the size of the entire data set.

2. We iteratively update the triple \((P(j), \mu_j, \Sigma_j)\) using equations 2.9, 2.10 and 2.11.

3. We repeat step 2 until a stopping criterion is met. In our case, we terminate the algorithm when the change in the estimates for the mean vectors \(\mu_j\) falls below some threshold:

\[
\sum_{j=1}^{N} |\mu_{j}^t - \mu_{j}^{t-1}| < \epsilon
\]

Let us examine what happens when we run the above algorithm on our collection of examples with \(N = 100\). For each of the resulting 100 clusters, figure 2-4 shows all of the members of that cluster superimposed on top of each other. We can see that the clusters seem to consist of examples with approximately the same body pose, with most variance at the locations of the hands and feet and least at the neck. The top left-most cluster is shown magnified in figure 2-5, with the mean of that cluster superimposed and marked with circles. Figure 2-6 shows that the cluster means are fairly evenly distributed in the camera pose space, which should facilitate the estimation of those parameters. The clustering is not so well distributed in the body pose space, as we can see in figure 2-7 – a histogram of the mean body poses for the 100 hundred clusters. Some of the phases seem to be better represented than others, though no phases are seriously underrepresented. The difference between the more even distribution of the camera pose compared to the body pose can probably attributed to the fact that the camera pose space is much larger. After all, there
are only 20 distinct body phases in our example set. Nevertheless, at the outset, our model will likely be better suited for extrapolation of the camera position.

2.2.4 Life Made Simpler: K-Means Clustering

Because of the computational complexity of the EM algorithm, a simpler clustering method is often preferable. One such widely used method is k-means clustering. As we can see from the update equations for the EM algorithm when applied to a normal mixture distribution, the probability of vector $x$ being assigned to cluster $j$ is high when the Mahalanobis distance $(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_j)$ is large. In k-means clustering, we instead take the squared Euclidean distance as a metric and do the following:

1. Initialize the clustering by picking $N$ random mean vectors, as in the EM clustering.

2. Sort all of the feature vectors into the $N$ clusters by picking the cluster with the closest mean in terms of Euclidean distance.

3. Recompute the mean vectors.

4. Repeat steps 1 and 2 until some stopping criterion is met, for instance when the mean vectors stop changing significantly.

Each iteration of this algorithm takes $O(Ndn)$ time, where $N$ is the number of clusters, $d$ is the dimensionality of the feature vectors and $n$ is the size of the feature set. Thus, the time-complexity of each iteration is linear in the size of the data: one of the reasons which make this algorithm an appealing choice.

2.3 Final Considerations

In our discussion thus far, we have carefully avoided the important issue of determining the best (or, at least, a good) number of clusters to use. In many clustering problems, such as character recognition, the data is structured in such a way that the proper number of clusters can be readily seen or there is a number that intuitively makes sense (for instance, the number of letters in the alphabet). In other cases, such as ours, where there is no such clear candidate, it is often a good idea to try the clustering method for various numbers of clusters and choose the one that produces the "best" partitioning. The problem, of course, is deciding what constitutes a good clustering. Among the many different criterions, some common ones can be derived from so-called scatter matrices [7]. Let us define the following two matrices:

$$S_W = \sum_{j=1}^{N} \sum_{i=1}^{n} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$S_B = \sum_{j=1}^{N} n_j(\mu_j - \mu)(\mu_j - \mu)^T$$
Figure 2-4: Rotated clockwise: Plots of all 100 clusters after 30 iterations of the EM algorithm. Each plot was created by drawing all members of the given cluster, with the following color scheme: red: neck, blue and green: right and left arm; black and cyan: right and left leg.
Figure 2-5: A plot of one of the clusters from figure 2-4. The circles show the superimposed mean of the cluster.

Figure 2-6: The distribution of mean camera poses. Each circle corresponds to the average camera position for a cluster.
where $n_j$ is the size of the $j$-th cluster and $\mu$ is the mean of the entire data set. We call $S_W$ and $S_B$ the within-scatter and between-scatter matrix, respectively. Intuitively, the within-scatter matrix is a measure of tightness of clusters and the between-scatter matrix can provide us with a measure of separation. In general, it is desirable to maximize the former and minimize the latter. A common way of evaluating the scatter matrices is simply taking the trace, i.e. the sum of the diagonal elements:

$$tr[S_W] = \sum_{j=1}^{N} \sum_{i=1}^{n} |x_i - \mu_j|^2$$  \hfill (2.14)$$

$$tr[S_B] = \sum_{j=1}^{N} n_j |\mu_j - \mu|^2$$  \hfill (2.15)$$

We can see from equation 2.14 that the trace of the within-scatter matrix is simply the sum-of-squared-errors for the clustering. Thus it makes sense that we want to minimize this quantity.

To find out a good number of clusters to use, we ran the k-means clustering algorithm on our set of generated examples for different values of $N$. Figure 2-8 shows the trace of the scatter matrix for each $N$ after the clustering was complete. The best value for $N$ in terms of the trace criterion seems to be 100, since the within-scatter is smallest for this number of clusters.

To summarize, this chapter introduced our appearance model of a walking human. We describe how the examples for the model are generated and how the model is effectively compressed by grouping the examples into clusters using the EM algorithm. In the following chapter, we describe how we model the dynamics of the motion, after which we can finally move on to the implementation of our simultaneous pose estimation and camera calibration algorithm.
Figure 2-8: The trace of the within-scatter matrix after running the k-means algorithm vs. the number of clusters
Chapter 3

Modeling Motion Dynamics Using Hidden Markov Processes

In modeling of the dynamics of discretized motion, one essentially looks for descriptions of sequences of events, in which each event is influenced by some function of the history of the motion. Hidden Markov models and associated techniques have proven to be useful tools in modeling such sequences of events. In this chapter, we lay the theoretical foundation for Hidden Markov Models and then show how they can be used to model the dynamics of a human walk.

3.1 Discrete Markov Chains

Discrete Markov models (also referred to as Markov chains or processes) are useful for describing systems which can be characterized as being at any given time in one of a finite number of distinct states. Consider a system such as the one in figure 3-1 with the set of states \( S = \{S_1, ..., S_N\} \).

Let the set \( T = \{t_1, ..., t_n\} \) denote regularly spaced instances in time. At each instance \( t_i \), the system changes from the current state to a new (but not necessarily different) state according to some probability distribution. The essence of a Markov model lies in the fact that this probability distribution is only a function of the system’s past, i.e. the sequence of states up to time \( t_i \). Let \( q_j(t) \) denote the event that the system is in state \( S_j \) at time \( t \). The probability distribution governing \( q_j(t) \) is the conditional distribution \( P(q_j(t)|q_i(t-1), q_k(t-2), ...) \), where \( i, j, k \in \{1, ..., N\} \).

![Figure 3-1: A discrete Markov chain with two states \( S_1, S_2 \) and transition probabilities \( A = (a_{ij}) \)](image)
index over the set of states. We will consider the special case of a first-order, discrete
Markov model, where the state at time \( t \) depends only on the single previous state.
The probability distribution over the state at time \( t \) then simplifies as follows:

\[
P(q_j(t)|q_i(t-1), q_k(t-2), \ldots) = P(q_j(t)|q_i(t-1))
\]  

(3.1)

Given this restriction, the parameters of the system in figure 3-1 can be described
by the matrix \( \mathbf{A} = (a_{ij}) \), where \( a_{ij} = P(q_j(t)|q_i(t-1)) \). It follows from the definition
of \( \mathbf{A} \) that \( a_{ij} \geq 0 \) for all \( i, j \) and \( \sum_{j=1}^{N} a_{ij} = 1 \). In addition, let \( \pi_i = P(q_i(1)) \) denote
the probability that the system was initially in state \( S_i \). The full set of parameters
governing the system then is \( \Theta = (\mathbf{A}, \Pi) \), where \( \Pi = \{\pi_1, \ldots, \pi_N\} \).

It is useful to think of the Markov model described above as a process which
at regular intervals in time generates an output equal to the current state at that
time. The probability that a particular sequence, \( O = \{o_1, \ldots, o_n\} \), of such outputs
(observations) occurred can then be easily calculated:

\[
P(O|\Theta) = P(q_{o_1}(1))P(q_{o_2}(2)|q_{o_1}(1)) \ldots P(q_{o_n}(n)|q_{o_{n-1}}(n-1))
\]  

\[
= \pi_{o_1} \prod_{i=1}^{n-1} a_{o_i, o_{i+1}}
\]  

(3.2)

This property makes Markov models good candidates for describing systems in
which the states are observable, i.e. it is possible to tell the current state of the
system given a single observation. Suppose, however, that we wanted to model a
system in which the states were not directly observable. In such a case, a discrete
Markov model would be of little use, as there would be no way to evaluate the model
against observations. Hidden Markov models (HMMs) provide a way to deal with
this problem by adding to the model another stochastic layer which describes how a
state generates its outputs.

### 3.2 Hidden Markov Models

Consider the Markov model from figure 3-1. Let us now suppose that at each time
step, the current state of the system generates a visible output from some alphabet
\( \mathcal{V} = \{v_1, \ldots, v_k\} \), as in figure 3-2.

Let \( b_{ij} \) denote the probability that, at time \( t \), state \( i \) generates the visible symbol
\( v_j(t) \):

\[
b_{ij} = P(v_j(t)|q_i(t))
\]

Naturally, the parameters \( \mathbf{B} = (b_{ij}) \) have to obey the stochastic constraints \( b_{ij} \geq 0 \) for all \( i, j \) and \( \sum_{j=1}^{k} b_{ij} = 1 \) for all \( i \). Since the states of
the system are no longer directly observable, we call them the hidden states, whereas
the set \( \mathcal{V} \) is the set of visible states.

The three fundamental issues concerning hidden Markov models and their appli-
cation are:

1. **The Evaluation Problem:** Given a sequence of observations \( O = o_1o_2\ldots o_T \) and
   an HMM with the parameters \( \Theta = (\mathbf{A}, \mathbf{B}, \Pi) \), find the probability that the
   model generated the observations, i.e. find \( P(O|\Theta) \).
2. *The Decoding Problem*: Given a sequence of observations and an HMM, as above, find the sequence of hidden states $\hat{Q}$ which was mostly likely to lead to the sequence of observations.

3. *The Learning Problem*: Given the structure of the model, i.e. the number of hidden and visible states, and a set of observation sequences, find the model parameters $\Theta$ which best describe the data.

We will briefly describe each problem in turn. Lawrence Rabiner's tutorial on hidden Markov Models [16] is an excellent source of more detailed information about methods associated with hidden Markov models as well as implementation issues.

### 3.2.1 The Evaluation Problem

The greedy solution to the evaluation problem would be to simply evaluate the probability of an observation sequence given the model directly. We can write the joint probability of an observation and a sequence of hidden states $Q_s = s_1s_2...s_T$, given the model as

$$ P(O, Q_s | \Theta) = P(O|Q_s, \Theta)P(Q_s | \Theta) \quad (3.3) $$

The first term is the probability of a sequence of observations given a specific sequence of hidden states, which is

$$ P(O|Q_s, \Theta) = \prod_{t=1}^T b_{s_t o_t} \quad (3.4) $$

where $b_{s_t o_t}$ denotes the probability of hidden state $s_t$ selecting the visible symbol $o_t$.

The second term is the probability of a sequence of hidden states given the model, which is

$$ P(Q_s | \Theta) = \pi_{s_1} \prod_{t=1}^{T-1} a_{s_t,s_{t+1}} \quad (3.5) $$

Finally, the probability of a set of observations given the model is simply the sum over all possible hidden state sequences of the joint probability from equation 3.3. Thus:

$$ P(O | \Theta) = \sum_s \pi_{s_1} \prod_{t=1}^T b_{s_t o_t} \prod_{t=1}^{T-1} a_{s_t,s_{t+1}} \quad (3.6) $$
Each term in the above sum is a product of $O(2T)$ terms. In a general HMM with $N$ states, there could be $N^T$ different sequences of hidden states, which makes the time-complexity of the above calculation exponential, and thus impractical. A more efficient, recursive method to do the same calculation is the so-called forward algorithm. This algorithm defines a special variable $\alpha_i(t)$ to be the probability of a partial sequence of observations up until time $t$ and the state at time $t$ equal to $S_i$, given the model parameters:

$$\alpha_i(t) = P(O, q_i(t)|\Theta)$$

We can then calculate the probability of an observation sequence by summing $\alpha_i(T)$ over all $i$:

$$P(O|\Theta) = \sum_{i=1}^{N} \alpha_i(T) \tag{3.7}$$

The time-saving trick is the recursive calculation of $\alpha_i(T)$, which we can derive by induction. At time 1, we have

$$\alpha_i(1) = \pi_i b_{i o_1}$$

which is simply the joint probability of starting in state $i$ and that state generating the observation $o_1$. The inductive step then is

$$\alpha_j(t) = \left[ \sum_{i=1}^{N} \alpha_j(t-1)a_{ij} \right] b_{jo_{t-1}}$$

Consider the system at time $t$. The current state could have been reached from any of the $N$ states in the system. Each term in the above sum corresponds to the probability that the system generated the partial observations until time $t-1$ with the last state being $S_i$ and then transitioned to the current state $S_j$. Finally, we have to multiply this sum by the probability that the current state generated the observation $o_t$.

It is easy to see that the forward algorithm is polynomial in time, as equation 3.7 takes $O(TN^2)$ calculations.

### 3.2.2 The Decoding Problem

The decoding problem is less straightforward than the evaluation problem because, unless the HMM is degenerate, there is no absolute answer to it. Given a sequence of observations, there could be many sequences of hidden states which may have caused it. We need to select one which is in some sense optimal. Since this optimality criterion often depends on the specifics of the application, we have to digress from the theory and consider again our model of the human walking motion. Suppose that we have a sequence of rough estimates of the body pose (phase of the walking cycle). Some of these estimates may be approximately correct and others may be wrong. These are our observations. We know that the person progressed through a sequence of true body poses, and we would like to find that sequence. Thus, if we model the dynamics of the motion using an HMM, as we will describe later, we want to look for
the best path through the model, given our observations. Therefore, we should try to find the path $Q$ such that

$$
\hat{Q} = \arg\max_Q P(Q|O, \Theta) = \arg\max_Q P(Q, O|\Theta)
$$

A dynamic programming technique called the Viterbi algorithm ([20] and [8]) can be used to solve this optimization problem.

### 3.2.3 The Learning Problem

Suppose we have a collection of observation sequences and a rough structure of the model, i.e. the number of visible and hidden states, and our task is to find the parameters of the HMM that maximize the likelihood of the parameters given the data. It is no coincidence that the problem sounds much like the goal of the general EM algorithm mentioned in Chapter 2. It is, in fact, the same problem, if we think of the observations as our good feature set and the sequence of states which led to the observations as the missing data. This approach to the problem leads to the special case of the EM algorithm also known as the Baum-Welch algorithm for learning HMM parameters. We will not go into further detail here, as we do not require a solution to the learning problem for our modeling task. However, a good overview of the Baum-Welch algorithm can be found in [1].

### 3.3 The Walking HMM

Let us now turn to the task of creating a hidden Markov model to describe the dynamics of the walking motion. Let us assume that we have some system in place (a black box for now), which uses some number of different cameras to observe a person walking in a straight line through a room. The system records $T$ consecutive frames from all the cameras in a synchronized fashion, and then produces an estimate of the walking phase for each of those frames. This sequence of estimates, $\Psi = \psi_1, ..., \psi_T$, constitutes our sequence of observations, and it has the following properties

1. Each observation $\psi_i = [\psi_{i1}...\psi_{ic}]$ is a vector consisting of a walking phase estimate corresponding to each of the $c$ cameras.

2. The walking phase estimates $\psi_{ij}$ are either close to the true walking phase for that frame, or close to its opposite phase. As a reminder, by true walking phase we mean the phase of the computer generated model described in chapter 2 which most closely matches the appearance of the person’s current body pose. If a full walking cycle were divided into $p$ parts, the opposite of the $i$-th phase would be phase $j$, where $j = (i + p/2)$ mod $p$. Opposite phases sometimes have a very similar appearance, as we can see in figure 3-3.

Using the above information and our knowledge about the dynamics of the motion, we have the following hidden Markov model (figure 3-4):
Figure 3-3: The top row shows walking phases 1 through 10 for a given camera position, while the bottom row shows phases 11 through 20. Thus, each column contains a pair of opposite phases. Notice that some of the pairs are quite similar in appearance.

Figure 3-4: A cyclical feed-forward hidden Markov model describing the progression of estimated walking phases from multiple views
1. There are $p$ hidden states, each state corresponding to a walking phase. We use $p = 20$, which is the same number as the number of walking phases in the model we used to generate our examples.

2. The model has a cyclical structure corresponding to the periodic nature of the motion. Moreover, the model can transition from state to state only in the forward direction, since we are only concerned with forward motion. To accommodate various speeds of motion, it is also possible for the model to remain in the same state or skip up to two states. These restrictions lead to the following properties of the transition matrix:

$$
\begin{align*}
a_{ij} &= \begin{cases} 
p_r & \text{if } j = i \\
p_n & \text{if } j = (i + 1) \mod p \\
p_j & \text{if } j = (i + 2) \mod p \\
p_l & \text{if } j = (i + 3) \mod p \\
0 & \text{otherwise}
\end{cases}
\end{align*}
$$

We set the values for $p_r, p_n, p_j$ and $p_l$ based on experimentation with various data sequences. The best results were usually produced with $p_r < p_n$, which pushes the model towards moving to the next state rather than staying in the same one. We also set $p_j$ to be only slightly smaller than $p_n$, since it is quite common to skip states. Similarly, the probability of skipping two states is slightly smaller than the probability of skipping just one. In certain cases, when the person moves significantly slower or faster than the synthetic model, it helps to adjust $p_r$ accordingly. The speed of motion can be estimated by calculating the period of the walking motion in the input silhouette (e.g. by locating silhouettes with minimal width). The probability $p_r$ can then be adjusted automatically to account for this difference.

3. Since all of the phases of the walking cycle are equally likely to be the first in the incoming video sequence, we chose a uniform prior distribution on the initial state. Thus $\pi_i = 1/p$ for all $i$.

4. The final remaining set of parameters is the probability distribution which describes how the visible states are generated. In our case, we need each state to generate a vector of $c$ symbols – one for each camera. To make this possible, we need to extend our basic hidden Markov model with discrete observation probabilities to one with a continuous observation density. Each state in such a model produces its output by drawing a sample from the corresponding observation density. We need to model the fact that opposite phases of the walking cycle have similar appearance and, consequently, that a hidden state of the HMM can generate an observation, in which some of the observed phases are, in fact, opposite to the true phase. Thus, we use a mixture density with $2^c$ components, where $c$ is the number of cameras. For state $i$, the $k$-th component of the mixture is a Normal distribution $N_{ik}(\mu_{ik}, \Sigma_{ik})$ with the following properties:

$$
\mu_{ik} = [f(i, e_1), \ldots, f(i, e_c)], \quad (e_1 e_2 \ldots e_c) = (k)_2
$$
\[
f(i, e_j) = \begin{cases} 
i & \text{if } e_j = 0 \\
(i + p/2) \mod p & \text{if } e_j = 1 
\end{cases}
\Sigma_{ik} \text{ is diagonal}
\]

The first two properties may require further explanation. Take \((e_1 e_2 ... e_c)\) to be the binary representation of \(k\) (with any necessary leading zeros). The mean \(\mu_{ik}\) then consists of the corresponding arrangement of either phase \(i\) or its opposite phase for the observation in each camera. We have \(2^c\) mixture components because there are \(2^c\) such possible arrangements. We choose a diagonal covariance matrix because the elements of the mean vector, corresponding to individual cameras, should be independent and have equal variance. With these modifications, hidden states generate visible states according to the following relationship:

\[
P(v(t) = \psi_j | q_i(t)) = \sum_{k=1}^{2^c} \frac{1}{2^c} N_{ik}(\mu_{ik}, \Sigma_{ik})(\psi_j).
\]

where \(v(t) = \psi_j\) denotes the event that the visible state generated at time \(t\) was \(\psi_j\).

Having thus defined our model, we can now use the Viterbi algorithm to generate the most likely sequence of states given our observations from multiple cameras. Our input will be the sequence of walking pose estimate vectors: \(\Psi = \psi_1, ..., \psi_T\), where \(\psi_i\) consists of a walking pose estimate for each camera in the \(i\)-th frame. We want our output to be the sequence of hidden states \(w = w_1 w_2 ... w_T\) which is most likely, given the observations. Let \(P(w_1 ... w_{t-1}, w_t = i, \psi_1 ... \psi_t | \Theta)\) denote the probability of a sequence of hidden states ending in state \(i\) at time \(t\) and the observations up until time \(t\), given the parameters of our HMM. Further, let us define the following quantity:

\[
\delta_i(t) = \max_{w_1 ... w_{t-1}} P(w_1 ... w_{t-1}, w_t = i, \psi_1 ... \psi_t | \Theta).
\]

In other words, \(\delta_i(t)\) is the highest probability of any sequence of hidden states ending in state \(i\) at time \(t\) and the observations up until time \(t\), given the parameters of our HMM. Given these definitions, we are looking for a sequence of hidden states \(w = w_1 w_2 ... w_T\) such that

\[
w_t = \arg\max_{1 \leq j \leq N} (\delta_j(t)).
\]

The hidden state at time \(t\) has to maximize the probability of the sequence and observations up to that time. We can compute all of the values \(\delta_j(t)\) for \(1 \leq t \leq T\) and \(1 \leq j \leq N\) by using the inductive property

\[
\delta_j(t) = (\max_i \delta_i(t-1)a_{ij})b_{j\psi}.
\]
Notice that since we extended our HMM to one with a continuous observation density, by equation 3.8, we have

\[ b_{j|\psi_t} = \sum_{k=1}^{2c} \frac{1}{2^c} N_{jk}(\mu_{jk}, \Sigma_{jk})(\psi_t). \]

We can efficiently find the sequence \( \mathbf{w} \) as defined above by recursively calculating the values \( \delta_j(t) \) for \( 1 \leq j \leq N \) and times \( t = T, T - 1, ..., 1 \), while keeping track of the argument \( j = w_t \) which maximized \( \delta_j(t) \) at time \( t \). The resulting sequence \( \mathbf{w} \) is the most likely sequence of hidden states, i.e. walking phases, given the observed phase estimate vectors.
Chapter 4

Simultaneous Camera Calibration and Pose Estimation

In this chapter, we describe how we can apply our models of the appearance and dynamics of human walking motion to the problem of simultaneously estimating the camera position as well as the body pose in a video sequence of a walking person. We first give a high-level overview of our algorithm, and subsequently address each step in terms of performance and implementation issues. Finally, we present results on both synthetic as well as recorded data.

4.1 The Algorithm

Perhaps the most straightforward way to characterize an algorithm is to describe the mapping between its inputs and outputs. As we mentioned in Chapter 1, the goal of our work is to observe a person walking through a room from multiple points of view, the locations of which are not known, and use our knowledge about the appearance and dynamics of the motion to extract information about the positions of the observers (cameras) and the body pose as a function of time.

4.1.1 Input

Consider \( n \) synchronized cameras, all of which have overlapping fields of view, and a person walking through the scene. The input to our algorithm consists of \( n \) sequences of images capturing this motion. If we write the image sequence from the \( i \)-th camera as \( S_i = I_{i1}\ldots I_{iT} \), then the images \( I_{ik} \) and \( I_{jk} \) show two different views of the person in the same position whenever \( i = j \). All of the input sequences are of equal, finite length.

4.1.2 Output

Given the input sequences as specified above, our algorithm outputs the following:
1. A pair of coordinates \((\theta_i, \phi_i)\) for each camera, determining the position of that camera with respect to the direction of the person's motion (see figure 2-2).

2. A sequence of walking phase estimates, \(\Psi = \psi_1...\psi_T\).

3. At each time index, the triple \((\theta_i, \phi_i, \psi_i)\) can be used to render the 3D model which we used to construct our model as it would be seen from the given camera location and in the given walking phase. This 3D model can then be used as a good initial estimate for the true 3D configuration of the person's body parts at time \(i\). The algorithm thus extracts three-dimensional information from multiple two-dimensional inputs.

### 4.1.3 Mapping Inputs to Outputs

Figure 4-1 shows a schematic diagram of our entire pose estimation and camera calibration system. In chapters 2 and 3 we described the right column of this diagram: the model of the motion. We use the Poser model of a human body to sample from the space of walking phases and camera positions at regular intervals. These examples are then clustered using an expectation-maximization or k-means technique. The hidden Markov model of the dynamics is constructed based on empirical evidence and our knowledge of the characteristics of human gait. While some of these modeling tasks— in particular, the rendering and clustering—are computationally expensive, they are only performed once and thus do not affect the complexity of the algorithm, which is outlined in the left column of the diagram. We will first give a coarse description of the algorithm and then discuss each step in turn:

1. The cameras collect \(n\) sequences of images, \(S_1\) through \(S_n\), where \(S_i = I_{i1}...I_{iT}\).

2. Each image is segmented using a background subtraction method. This produces \(c\) sequences of binary silhouettes: \(S'_1\) through \(S'_n\), where \(S'_i = s_{i1}...s_{iT}\).

3. Each silhouette \(s_{ij}\) is compared to the mean of each cluster in the appearance model using a distance metric. The cluster with the smallest distance is selected, thus producing \(n\) sequences of cluster matches: \(C'_1\) through \(C'_c\), where \(C'_i = c_{i1}...c_{iT}\).

4. The camera positions of all of the examples belonging to the clusters in sequence \(C'_i\) form a cluster in the camera pose space. We fit an ellipse to this cluster and then take the center of the ellipse to be the camera pose estimate for the \(i\)-th camera. Estimating the camera pose in this fashion reduces the effect any outliers have on the result.

5. For each cluster in the sequence \(C'_i\), we compute the average walking phase over all of the examples in that cluster. In this fashion, we obtain the sequence \(\Psi'_i = \psi'_{i1}...\psi'_{iT}\) of phase estimates for each camera. The sequences \(\Psi'_1\) through \(\Psi'_c\) then serve as inputs to our HMM, which outputs the sequence of the final walking pose estimates.
Figure 4-1: A schematic diagram outlining our pose estimation and camera calibration system.
4.1.4 Data Acquisition

To collect synchronized sequences of images, we used a set of Point Grey Dragonfly cameras. Each of the two cameras we used for our experiments ran at 15Hz and produced images of 640x480 pixels. The cameras were automatically synchronized so that image acquisition in the two cameras took place within at most 20μs.

4.1.5 Segmentation

We segmented each incoming image using the adaptive background subtraction technique from [12]. We first used 100 frames of the scene without the person or any other moving objects to construct a probabilistic model of the background. We then used the method from [12] to assign to each pixel the binary value 1 whenever that pixel was classified as foreground, i.e. belonging to the person and the value 0 otherwise, thus converting each image sequence into a sequence of silhouettes. The size of the cropped silhouettes was roughly 120x250 pixels. Figure 4-2 shows a sample input image and the corresponding segmentation.

4.2 Matching Silhouettes and Clusters

For the purpose of comparing silhouettes to clusters, we represent each cluster as the mean silhouette over all of the examples belonging to that cluster. To compute the mean silhouette for cluster $c_i$, we first align all of the examples in that cluster by translating them so that the head regions overlap. The value of a pixel in the mean silhouette then corresponds to the ratio of the number of the registered examples which were non-zero at that particular pixel to the size of the cluster. In effect, this is the probability that the pixel is on in a silhouette picked randomly from the cluster. Figure 4-3 shows the plot of a cluster of examples and the corresponding mean silhouette. Note that the mean silhouette is no longer binary, which we will
have to address when defining a similarity measure between silhouettes and cluster means.

In order to compare a binary silhouette to the mean silhouette of a cluster, we need to develop a distance metric, as well as a method to register the two images. We do the latter in the following fashion:

1. We crop any rows or columns from the borders of the silhouette as well as the cluster mean in which no pixels are on.

2. We resize the larger of the pair (silhouette, cluster mean) so that both are of identical size. Resizing the smaller one would introduce more noise into our system, which would not be desirable. The scale factor is determined independently for each pair (silhouette, cluster mean).

3. We register the resized images by aligning the centroids of the head region, which we define to be the top 1/10-th of the image. Aligning the images by their centroids intuitively makes more sense, but it causes problems with recorded data, which often includes a shadow. The shadow may offset the centroid of the silhouette enough that the registration is no longer meaningful. We found that using the centroid of the head region produces much more stable results.

With the silhouette and cluster mean registered, we need a distance measure to compare them. We use a variant of the Hamming distance for this purpose. Given two bit-vectors, \( u \) and \( v \), the Hamming distance \( d_h(u, v) \) is equal to the number of bits which need to be flipped to change \( u \) into \( v \). We can also write this as \( d_h(u, v) = u \oplus v \). For two binary silhouettes \( s_i \) and \( s_j \), we define their Hamming distance to be equal to the number of pixels which need to be flipped to turn \( s_i \) into \( s_j \). Figure 4-4 shows a plot of the Hamming distance between an example from our appearance model and its neighbors in the camera pose space.
In order to use this distance measure for the comparison of silhouettes and cluster means, which are not binary, we adapt the distance measure as follows:

- We define $s'_c$ to be a binary silhouette in which a pixel is on if the corresponding pixel in the cluster mean $s_c$ is greater than 0.5, i.e. at least one half of the examples in the cluster had an on-pixel in that location. This thresholding step helps minimize the effect any outliers in the cluster might have on the distance measure. Let us call this silhouette the majority mean.

- We define $s'$ to be the complement of the intersection of $s$ and $s'_c$ multiplied by the majority mean $s'_c$: $s' = s_c(\bar{s} \oplus \bar{s}'_c)$. In effect, this amounts to setting to 0 the pixels in the majority mean $s'_c$ at which the silhouette $s$ is non-zero. If we then denote an element of $s'$ by $(s')_{ij}$, we have

$$d_h(s, s_c) = \sum_i \sum_j (s')_{ij}$$

Defining the distance measure in this fashion assures that the pixels where fewer of the examples in the cluster agree carry less weight.

### 4.2.1 Estimating the Camera Pose

As we saw in the previous chapter, opposite phases of the walking cycle may have a similar appearance. Similarly, the appearance of a person from a particular position $(\theta, \phi)$ may be similar to the appearance of the same person from the opposite camera position. If we imagine the camera position to be a location on a dome, the center of which is the person’s centroid, then opposite camera locations would be two points on the dome for which the connecting line segment passes through the dome’s center. Because the appearance of the person may be similar when seen from opposite locations, we can expect the sequence of cluster matches produced by the previous step
of our algorithm to contain a large number of examples corresponding to either of these two locations. Which of these locations should we pick? The answer becomes obvious when we remember that the camera location is relative to the direction of the person's motion. We can pick the camera position estimate which is consistent with the change in the person's position in the input sequence.

Thus we compute the mean camera pose over the sequence of cluster matches by first dividing all of the examples into two sets (for instance using the k-means method) and then setting the camera pose estimate to the mean camera position in the set which corresponds to the person's direction of motion in the input sequence.

4.3 Experiments

To test our algorithm, we first used Poser to generate a synthetic video sequence of a walking person. However, we altered the characteristics of the model's gait by changing the speed of motion, stride length and arm swing in order to make the model different from the training examples. We rendered the walking sequence from two points of view to simulate the data from a real experiment. Figures 4-5 and 4-6 show the results of the camera pose estimation and the first stage of the walking phase estimation. The top plot in each figure shows the distribution of camera poses over all of the examples in the matching cluster sequence corresponding to the input. Brighter dots indicate more examples with the corresponding camera position. The bright square indicates the correct answer. Although the centers of the cluster of camera positions differ slightly from the correct answers, in reality the difference in appearance due to this discrepancy is small. The bottom plot shows the distribution of the walking phases among the examples from the best matching cluster at each frame. Brighter squares again indicate more examples with the corresponding phase. In this case the correct answer would be a diagonal line from top left to bottom right. As we can see, some of the frames show a phase estimate equal to the opposite of the true phase. Figures 4-7 and 4-8 show the results after the phase estimates have been passed through the HMM. The top row shows the input silhouettes, the middle row shows the best matching cluster and the bottom row shows the example from our model which corresponds to the camera pose and walking phase estimated by our algorithm. The results from both views closely match the input.

We further tested the algorithm on live data collected using our set of cameras. Figures 4-9 through 4-12 are equivalent to the four figures described in the previous experiment. Despite the much noisier input, the algorithm still performed quite well, with the results matching the input closely. We can see from the phase estimate plots that the phase was harder to determine in this case, which was probably due to the fact that the silhouettes were particular noisy around the legs. The second camera view (figure 4-10) generated two clusters of camera positions, so we selected the one which corresponded to the direction of motion.

Our algorithm will only perform well if the multiple camera views provide enough information about the body pose throughout the video sequence. While the camera pose can be estimated from a single video sequence, the walking phase is often am-
Estimate of Relative Camera Position

Cam Pan (0: directly behind person, 180: directly ahead)

Estimate of Walking Phase

Frame number

Figure 4-5: Results of the camera pose estimation and first stage of the walking pose estimation for the first camera in the synthetically generated test sequence.

ambiguous and thus requires a second, informative point of view. The algorithm may not do well if the multiple cameras are placed near each other or in an arrangement which does not provide enough difference in perspective.

The most computationally complex part of our algorithm, after the appearance model is created, is the matching of silhouettes to all of the clusters in the model. On a workstation with a 1GHz processor, this takes approximately 3 seconds per frame when the algorithm is running in Matlab. The performance of the system would be much improved if it were ported to a compiled rather than interpreted environment, such as C. The performance could further be improved by lowering the resolution of both the observations and the model.
Figure 4-6: Results of the camera pose estimation and first stage of the walking pose estimation for the second camera in the synthetically generated test sequence.
Figure 4-7: Top row: input silhouettes from the first camera angle of the synthetic test sequence. Middle row: best matching clusters. Bottom row: examples generated using the estimated camera position and body pose.
Figure 4-8: Top row: input silhouettes from the second camera angle of the synthetic test sequence. Middle row: best matching clusters. Bottom row: examples generated using the estimated camera position and body pose.
Figure 4-9: Results of the camera pose estimation and first stage of the walking pose estimation for camera 1 in the recorded test sequence.
Figure 4-10: Results of the camera pose estimation and first stage of the walking pose estimation for camera 2 in the recorded test sequence.
Figure 4-11: Top row: input silhouettes from the first camera angle of the recorded test sequence. Middle row: best matching clusters. Bottom row: examples generated using the estimated camera position and body pose.
Figure 4-12: Top row: input silhouettes from the second camera angle of the recorded test sequence. Middle row: best matching clusters. Bottom row: examples generated using the estimated camera position and body pose.
Chapter 5

Summary

This thesis described a system which can be used to extract the three-dimensional body pose of a walking person as a function of time using a set of uncalibrated, synchronized video cameras.

One of the most important advantages of such a system stems from the fact that it puts almost no restrictions on the input devices. It would thus be very well suited to scenarios in which the number of cameras and their positions cannot be known in advance or cannot stay fixed. Examples of such scenarios include intelligent office and home environments or areas with ubiquitous computing devices which may carry cameras. At any given point in time, the system would be capable of recalibrating the cameras based on the observation of a walking person – an event which is quite common in most environments that involve human activity.

Furthermore, the system lays a foundation for further study of the possible biometric properties of human gait. To date, there has not been a gait analysis system which utilizes fully three-dimensional information. The output of our pose estimation algorithm can be used as a good starting point in determining the precise 3D locations of major body parts. This information could then be used to gain further insight into the properties of gait, such as the extend to which a person’s gait is unique or the relationship between gait and other observable factors such as time of day or the presence of other people.

The immediate future holds possibilities for a number of extensions to this work:

1. Although our 3D body pose estimates do not themselves carry biometric information, we could investigate whether any biometric information can be extracted from the way these estimates compare to the input. For instance, we could define a distance metric between the input video sequence and the matched sequence of our model rendered using the estimated camera calibration and walking phase. We would effectively be comparing the appearance of the observed person to the appearance of our model, thus using the model as a fixed reference point. This would enable us to directly compare any two input sequences, even those which could otherwise not be compared because they are from different views.

2. The appearance model which we used to describe a walking person could be
made more general by including different body types and various types of clothing. While the aim of this thesis was mostly the proof of the concept, including other parameters in the model would improve its performance in a general office setting.

3. So far we don't take advantage of the fact that the system estimates the camera calibration in addition to the 3D body pose. This calibration information, along with some reference point such as the height of a known person, could be used to find the relative positions of each pair of cameras. We could subsequently use a voxel carving technique to find a volumetric representation of the observed person. Combined with our body pose estimate, we could then use the volumetric model to further hone in on the configuration of the person's body parts.

4. We could relax the assumption that the observed person is walking in a straight line. We could easily detect the change of direction of motion by continuously re-estimating the locations of the cameras. To make the computational complexity less prohibitive, the system could operate at a lower resolution and frame rate.
Bibliography


