HUMAN POSE ESTIMATION FROM A SINGLE VIEW POINT

by

Matheen Siddiqui

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2009

Dedication

To my parents.

Acknowledgements

It has been mentioned to me on many occasions to first thank God and then thank those who supported you on your journey, whatever that may be. Indeed, no man transcends his time or place in an absolute sense (a virtue unique only to the Divine). One, instead, is shaped by the circumstances and the personalties that surround him and the degree to which he realizes his own accomplishments, is the degree to which he recognizes those people that shaped it.

I would like to thank my advisor, professor Gérard Medioni, for his guidance during my time at USC. I would also like to thank my committee, members professor Jonathan Gratch, and professor C.-C. Jay Kuo. I also would like to thank, professor Ramakant Nevatia, professor David Kempe and professor Wlodek Proskurowski for serving on my guidance committee

In the early part of my time at USC I had to the opportunity to work with Dr. Kwangsu Kim and Dr. Alexandre Francois on ETRI related projects. I would like to thank them for their insights on the dimensions of research and life. I similarly would also like to thank Dr. Changki Min, Adit Sahasrabudhe, Dr. Philippos Mordohai, Anustup Choudhry, Paul Hsiung, Dr. Douglas Fidaleo, and Yuping Lin.

I am glad to have counted Dr. Wei-Kai Liao amongst my close friends in the lab. We have had many conversations on life, and research that seemed to traverse lofty positions much farther then my own individual reach. Truly, we have run many miles together (figuratively and literally).

I am grateful for the interactions I have had with other current and past members of the USC vision lab: Dr. Qian Yu, Dr. Bo Wu, Dr. Pradeep Natarajan, Dr. Chi-Wei Chu, Vivek Kumar Singh, Dr. Chang Yuan, Dr. Mun Wai Lee, Dr. Xuefeng Song, Dr. Tae Eun Choe, Jan Prokaj, Dian Gong, Xumei Zhao, Vivek Pradeep, Dr. Cheng-Hao Kuo, Li Zhang, Yuan Li, Eunyoung Kim, Thang Ba Dinh, Derya Ozkan, Cheng-Hua Jeff Paim, Dr. Sung Chun Lee, and Nancy Levien. I wish them all the best.

I would like to thank my family and friends who have both encouraged and supported me, and tolerated my idiosyncrasies in completing this thesis. This includes my parents, Zakia and Mohammad Siddiqui, my siblings Aleem, Aqeel, Adeel, and Amina, my nieces, Sana, Isra, Sophi, and my nephew Hasan.

While in LA, I owe a great deal to my close friends Javeed and Anjum Mohammed and their family. They took me into their home right away and helped me navigate an unfamiliar space by their positive example.

And finally, I would like to thank my roommate Mohammed Hassanein and my friends Abdul Jabbar Sani, Jahan Hamid, Aziza Hasan, Shazia Bhombal, and of course my dear Nazia Khan for their support and their insight into the interplay between the mind and the spirit, and the lives we live and the lives we aim for.

Table of Contents

List Of Figures

Abstract

We address the estimation of human poses from a single view point in images and sequences. This is an important problem with a range of applications in human computer interaction, security and surveillance monitoring, image understanding, and motion capture. In this work we develop methods that make use of single view cameras, stereo, and range sensors.

First, we develop a 2D limb tracking scheme in color images using skin color and edge information. Multiple 2D limb models are used to enhance tracking of the underlying 3D structure. This includes models for lateral forearm views (waving) as well as for pointing gestures.

In our color image pose tracking framework, we find candidate 2D articulated model configurations by searching for locally optimal configurations under a weak but computationally manageable fitness function. By parameterizing 2D poses by their joint locations organized in a tree structure, candidates can be efficiently and exhaustively localized in a bottom-up manner. We then adapt this algorithm for use on sequences and develop methods to automatically construct a fitness function from annotated image data.

With a stereo camera, we use depth data to track the movement of a user using an articulated upper body model. We define an objective function that evaluates the

saliency of this upper body model with a stereo depth image and track the arms of a user by numerically maintaining the optimum using an annealed particle filter.

In range sensors, we use a DDMCMC approach to find an optimal pose based on a likelihood that compares synthesized and observed depth images. To speed up convergence of this search, we make use of bottom up detectors that generate candidate part locations. Our Markov chain dynamics explore solutions about these parts and thus combine bottom up and top down processing. The current performance is 10fps and we provide quantitative performance evaluation using hand annotated data. We demonstrate significant improvement over a baseline ICP approach. This algorithm is then adapted to estimate the specific shape parameters of subjects for use in tracking.

Chapter 1

Introduction

1.1 Background

Given the larger supporting role technology plays in our lives, we increasingly find ourselves interfacing with machines and computers. This interaction occurs at all levels ranging from video games, and cameras to more mundane activities such interacting with ATMs or other automated kiosks. Examples of this are illustrated in Figure 1.1.

The dominant forms of human computer interaction consist of tactile/visual interfaces such as keyboards, touch screen, mice. While these interfaces enable a user to direct and control a machine, they usually assume the user already has an understanding of its operation. Furthermore they require sensing that is tactile and feedback that is visual. This is problematic as robots and machines find uses in novel domains in which it is inappropriate for an operator to interface with a console. It then becomes critical to employ "natural" modes of communication that allow users to operate with limited prior knowledge.

A central limitation of current methods to interface with machines is the requirement that people to communicate with machines in a manner very different from the way people interact with each other. To enable natural, human like communication, we need to make use of natural communication signals such as speech, facial and body gestures. The availability of such knowledge can enable people to interact naturally with machines.

In particular, the position and orientation of a subject's limbs comprise an essential geometric description of the image of a human and can be used to enable natural communication between humans and machines. This pose information can be also be used for direct measurement based tasks such as localization, and high level image and video analysis such as behavior understanding and interpretation.

There are solutions to pose estimation in highly controlled environments with arrays of disparate cameras and sensors. While multi-camera configurations offer a rich source of data, they have limitations in deployment in natural environments. For this purpose, single view systems offer a lower level of complexity in terms the spatial positioning and alteration of the environment in which they are deployed. Furthermore, they present a communication paradigm similar to human-human interaction.

1.2 Goals

The goal of this work, as shown in Figure 1.3 is to develop methods to extract models and estimate the poses of a human in images and image sequences taken from a single view point. As the target environment for this work is an interactive system, performance in terms of speed and robustness plays an important role in these methods.

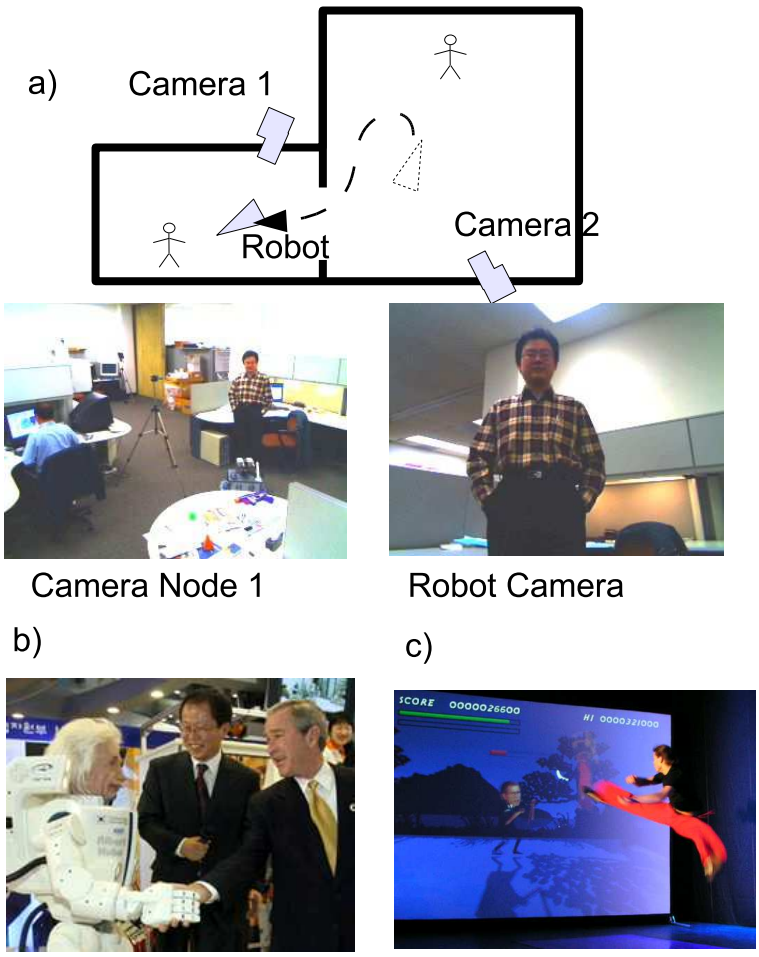


Figure 1.1: Modes of interaction between a user and a machine.

In this work, we consider both single view cameras and a range sensor. Digital cameras are are both widely available at low cost and with high resolution and low signal noise and high acquisition rate. They are thus a potentially attractive sensor. However the 2D images they produce are 2D projections of a scene and thus complicate estimation tasks.

Real-time depth-sensing cameras, as shown in Fig. 1.2, produce images where each pixel has an associated depth value. While these sensors have their own difficulties such as limited resolution in *time of flight* sensors or texture requirements in stereo, depth

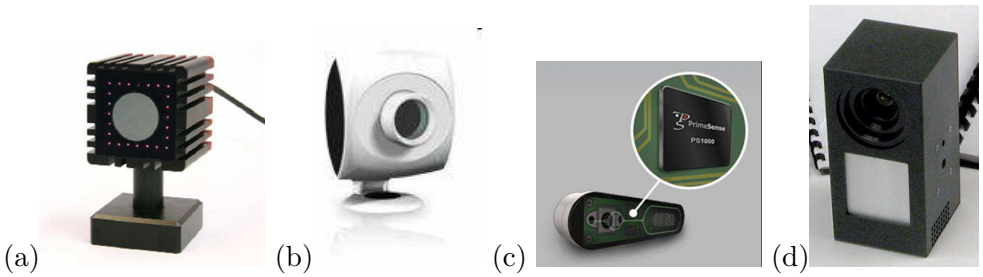


Figure 1.2: Examples of Real-Time Depth-Sensing Cameras: (a) The SR4000 from Mesa Imaging, (b) the ZCam from 3DV Systems, (c) the Prime Sensor from PrimeSense, and (d) a model from Canesta.

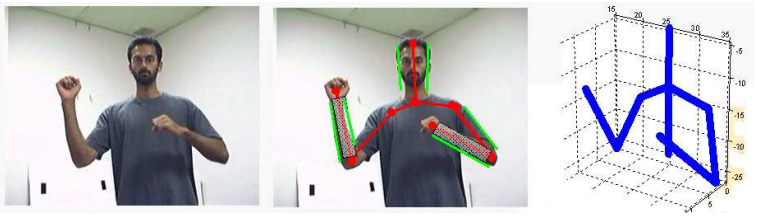


Figure 1.3: Pose Estimation

input enables the use of 3D information directly. This eliminates many of the ambiguities and inversion difficulties plaguing 2D image-based methods.

1.3 Challenges

Articulated pose estimation and tracking in images and video has been studied extensively in computer vision. The estimation of an articulated pose, however, continues to remain a difficult problem as a system must address many challenges that complicate the process of extraction.

The core difficulty in pose estimation is that the solution space is high dimensional and riddled with local minima. The dimensionality of even a very simple skeleton model can be 14 dimensions. This limits the effectiveness of an exhaustive march through parameter

Figure 1.4: Human pose estimation is complicated by the high degree of variablity in postures, shape, and appearnce.

space in search of a global optima. That the space has many local optima also limits the effectiveness of a pure descent based search, as simply following the gradient from an arbitrary starting position while almost surely yield the wrong answer. A method that extracts pose must address these issues.

Pose estimation is further complicated by high degrees of variability in the signature of people in sensor data. In general, this variability is due to changes in the pose configurations of people, the variation the shapes of bodies can take, and the viewpoint from which the signal was acquired. Clothing can also mask the observability of pose. These

difficulties, illustrated in Figure 1.4, complicate the construction of meaningful observation metrics, body part detectors, or the construction of mappings between pose and the observation space.

Intensity or color sensors are particularly challenging in this regard. This is because the variability is also contingent on changes in illumination and the appearance of clothing, and the body. Furthermore, the image itself only provides a 2D projection of scene information. The lose of depth gives rises to many ambiguous configurations. With a depth senor direct measurements of depth information is available and changes in appearance are not significant. However, we must deal with measurement artifacts, and limited resolution.

Other challenges include handling fast motion and motion blur, which can invalidate smoothness constraints. One must also disambiguate background clutter from the foreground image to prevent assigning background observation data to visible parts. Self occlusion of limbs and the torso must also be considered. These are often compensated for with strong priors (for tracking hidden parts) and robust detectors (to handle partly occluded parts).

A successful pose estimation system must contend with these issues robustly and efficiently. In what follows, we describe different approaches to solve this problem in cameras and depth sensors.

1.4 Summary of Approach

We develop methods to estimate models of humans in images from a single view point in color image cameras, as well as range sensors. For color images and image sequences we focuses on estimating 2D models. This includes models that account for just the forearms as well as 2D articulated poses that account for joint positions in a human pose.

With the availability of direct depth information in range sensors, a significant amount of the ambiguity inherent in 2D color images is reduced. Here we explicitly model the poses in 3D and make use of a robust generative model in its estimation.

1.4.1 Color/Intensity Image Sensors

We first develop a system to track the forearms of a user in a single image using an optimization framework employing skin color and edge features. By focusing our effort on the forearm we are able to reduce the dimensionality of the problem and track efficiently.

We also consider full articulated poses by modeling them as a collection of joints. To estimate the position of these joints we design a method that explores this space of joint configurations and identifies locally optimal yet sufficiently distinct configurations. This is accomplished with a bottom up technique that maintains such configurations as it advances from the leaves of the tree to the root.

We also adapt this algorithm for use on a sequence of images to make it even more efficient by considering configurations that are either near their position in the previous frame, or overlap areas where significant motion occurs in the subsequent frame. This

allows the number of partial configurations generated and evaluated to be significantly reduced, while accommodating both smooth and abrupt motions.

To accommodate generality and robustness, we propose methods to compute, from labeled training data, the saliency metrics used in the pose tracking. In particular we design a set of features that make use of generic image measurements, including edge and foreground information. Detectors are constructed from these features automatically using annotated imagery and feature selection techniques [71].

1.4.2 Depth Sensors

We consider both stereo and range sensors. In stereo sensors, we track the movement of a user by parameterizing an articulated upper body model using limb lengths and joint angles. We then define an objective function that evaluates the saliency of this upper body model with a stereo depth image. We track the arms of a user by numerically maintaining the optimal upper body configuration using an annealed particle filter [20].

We also estimate and track articulated human poses in sequences from a single view, real-time range sensor. Here, we use a data driven MCMC[45] approach to find an optimal pose based on a likelihood that compares synthesized depth images to the observed depth image. To speed up convergence of this search, we make use of bottom up detectors that generate candidate head, hand and forearm locations. Our Markov chain dynamics explore solutions about these parts and thus combine bottom up and top down processing. We also design a method to extract model parameters to make them specific to an individual.

1.5 Thesis Overview

In the rest of this thesis, we present the details of our work. In Chapter 2 we give a general review of the relevant literature. More specific work is review in subsequent chapters. In Chapters 3, 4, and 5, we provided the details of our work in single view color images. In particular, in Chapter 3 we we discuss limb detection and tracking, and in Chapter 4 we discuss the estimation of joint configurations, from images and sequences. In Chapter 5 we discuss a method to learn saliency metrics from annotated training data.

In Chapters 6, 7, and 8, we discuss our work with depth sensors. In particular, in Chapter 6 we discuss our work with with stereo sensors. In Chapter 7 we discuss our pose tracking system in range images and in Chapter 8, we present a method for learned person specific model parameters. We conclude and present possible future directions in Chapter 9.

Chapter 2

Literature Review

Human body pose estimation and tracking from images and video has been studied extensively in computer vision. There exist many surveys [30][53],[52][79] and evaluations [35][4], and efforts in creating standardized data sets[78]. These approaches differ in how the bodies are encoded, image observables or the visual saliency metrics used to align these models, and the machinery used to perform this alignment with the underlying image.

In what follows we will provide a general review the literature related to human pose estimation based these criteria. More specific discussion is detailed in subsequent chapters.

2.1 Human Body Representations

There exist many representations of the human body. In general the more complex the representation, the harder it is to estimate. Simpler representations are easier to estimate, but provide a coarser representation of the body.

Models employed tend to be either 2D or 3D skeletal structures that bear close resemblance to their true articulated nature. 3D models include skeletons with flesh attached or collections of individual limb models. The limb models can be cylinder, generalized cylinder, or superquadrics. In [20] truncated cones were used to represent body parts. In [31][82], superquadrics were used to represent human figures. In [59] a body model is constructed using 3D Gaussian blobs arranged in a skeletal structure. In [3] a low dimensional yet highly detailed triangular mesh model of a human is learned from 3D scans of humans. This model is used for detailed pose recovery in [6]. In [43][5][86] clothing is modeled.

These geometric models, when estimated from image observables, provide a direct representation of the human pose. It is something that can be use directly in higher level processing such as gesture or activity recognition. However, due to the large number of degrees of freedom these models may be difficult to estimate especially in a single view.

Examples of alternate representations are those used in whole body human detection and tracking methods. These methods model a human as a single bounding box [89]. The aim here is to find people independent of their postures. In [34] human bodies are modeled as a collection of 2D spatial and color-based Gaussian distributions corresponding to the head and hands.

An alternative to searching directly for a 3D parameterization of the human body, is to search for its projection. In [57] this idea is formalized using the Scaled Prismatic Model (SPM), which is used for tracking poses via registering frames. In [55] a 2D articulated model is aligned with an image by matching a shape context descriptor. Other 2D models include pictorial structures[24] and cardboard people[41].

Other relevant approaches model a human as a collection of separate but elastically connected limb sections, each associated with its own detector[24][63]. In [24] this collection of limbs is arranged in a tree structure. It is aligned with an image by first searching for each individual part over translations, rotations, and scales. Following this, the optimal pose is found by combining the detection results of each individual part efficiently. In [77] limbs are arranged in a graph structure and belief propagation is used to find the most likely configuration. In [27] uses several methods, each further reducing the size of the search space to find 2D human poses in TV sequences.

Modeling human poses as an collection of elastically connected rigid parts, is an effective simplification of the body pose, however it does have limitations in its expressive power. It has difficulty modeling self occlusions of limbs. In [90] multiple trees are used to extend this graphical model to deal with occlusions. In [98] a AND/OR graphical model is used to increase the representational power.

Pose Priors

An important aspect of the model is the information about the a priori likelihood of specific poses. This could be represented by exemplars, an actual distribution, or an embedding in a lower dimensional manifold. Priors play an important role in probabilistic methods as they can help shape a likelihood function or reduce its effective dimensionality.

While too strong of a reliance on a prior may cause difficulties in recognizing novel postures, priors can mitigate occlusions and keep recovered poses within an expected range. They can also prevent impossible situations. In [38] a mixture of Gaussian model learned from motion capture data is used to represent plausible 3D poses. This prior is

used to assist a 2D tracker. In [43] whole body priors were used on the global orientation of the pose, joint angles, human shape, and clothing. Part based methods such as [24][63] make use of priors between pairs of limbs. In [82] priors on the proportions of human data along with bias towards resting positions, anatomical joint angle limits, and body part interpenetration avoidances are employed.

Modeling distributions biases the solution to particular places but does not explicitly reduce the size of the problem. When working with specific motions or classes of postures, using low dimensional embeddings reduces the dimensionality of the problem and also restricts the generality of the search. In [97], models are constructed for walking motions only. An alternative to modeling priors with distributions is to learn an embedding on the space of valid poses. In [23][80][87][47] low dimensional manifolds are used to represent the space of admissible poses. In [55] a small set of exemplar poses are used. In [75] a low dimensional model is learned to represent human motion. This low dimensional model can then be used to index a database of training poses which constitute the prior.

In [12], a strong motion model specifically designed for walking is used.

2.2 Observations

The observable data used in a pose estimation method is essential to its success. There are many choices, including edge, color, and appearance. The availability of robust image observables greatly affects the ability to find correct poses. In particular, if reliable image data is available, a weak prior and simple algorithm can be used, whereas if observables are unreliable, more complex priors and sophisticated algorithms are necessary [4].

If foreground/background information is available, the silhouette can be an informative feature. This is used in [1][69][14] as input to learning based methods and in [24] with an ad hoc part detector.

Silhouette information is most often obtained from background subtraction. While reasonable silhouettes can be obtained using current methods, the extraction of reliable silhouettes in a general setting is still an area of investigation. This is because extraction is complicated in dynamic environments or when the camera moves. In addition, the silhouette feature does not provide information about the interior which may be vital for postures that involve arm positions close to the body.

The use of additional low level features, such as edge and boundary, can provide information in these cases. In [39] line segments and pairs of parallel line segments are used as key visual features. In [60] responses to boundary detectors steered to the orientation of a predicted boundary are used as the main visual feature.

For image sequences, there are additional sources of information. In particular, optical flow has been used extensively. In [11][92], optical flow is used directly to track a human figure. In [22], edge information in a window of frames is used.

While boundary based features are generic, their responses often do not provide a sufficiently unique signature. This is the case when there is background clutter inducing false edges and also when clothing with texture or folds is present. The use of the appearance of limbs or body parts an provide additional discriminative ability in dealing with such clutter. In [13] appearance based templates are used. In [25] the appearance of each limb is modeled as a fixed template.

Due to variations in clothing, appearance based features are more useful if they can be learned while tracking as in [63]. In [73] each limb is modeled with a texture map derived from its appearance in a previous frame. In [64] poses are estimated independently at each frame. From poses of high confidence, appearance information is adjusted. In [65] the appearance models of limbs are learned from the image sequence itself by clustering.

In addition to appearance and edge boundaries, color information has been used. In [56][66] images are segmented into color consistent regions with the idea that limbs would correspond to individual segments. In [45] a metric is derived based on how well a predicted model silhouette can be reconstructed with a given color segmentation. In [54] they use segmentation preprocessing to improve the efficiency. They use the segments as superpixels and constrain joint positions to be at their center. This reduces the size of search space which they optimize using an Gibbs sampler.

Higher level features and detectors can also be learned directly from training data. In [74] responses from boundary detectors are empirically derived from image data. Individual limb detector learned from training data have been used [68][51][76][49]. Learning observables in this manner provides a set of responses that are more reliable than low level features such as edge or flow but also more generic then appearance based detectors. In [48] boosting is used to construct select features that form a saliency measure to separate valid poses from non-valid poses.

2.2.1 Single vs Multi-view Methods

The various systems in pose estimation, can be categorized according to the number of views used. The number of views can have a great effect on the success of a pose

estimation system, and single and stereo processing tends to be much more difficult than wide baseline multi-view.

Single view methods must try to infer a representation of the human pose from a single image or sequence from one camera. Approaches such as [36][24][41] accomplish this by exploiting 2D representations of human poses. Other approaches such as that of [1][45] exploit 3D models.

While single view analysis is useful in many practical applications it faces many challenges. Approaches must contend with both depth ambiguities, and self occlusions. In multi-view approaches, different views of the same person can be used to mitigate these difficulties.

When background information is available one can make use of voxel or visual hull data directly as in [50]. Other approaches such as [31][20][76] fuse the multiple image sources using calibrated setups and projecting models into these images. In [42] orthogonal views are utilized.

While these approaches make use of general although calibrated configurations of cameras, other methods make use of uncalibrated configurations. In particular in [33] and [70] single view methods are used in conjunction with uncalibrated 3D data recovery methods to infer 3D poses.

From the standpoint of attaining strong and accurate measurements, multi-view methods are very useful. However, in interactive systems a single view setup can be more practical. Stereo and depth based sensors offer an alternative that provides depth information in a modular form. This is typically available in 2.5D. In [19][18], both depth from stereo and image data are used to infer a 3D pose. While in [7] only depth information is used.

The use of depth information has also been explored. In [99], a coarse labeling of depth pixels is followed by a more precise joint estimation to estimate poses. In [94], control theory is used to maintain a correspondence between model based feature points and depth points. In [19], Iterative Closest Point (ICP) is used to track a pose initialized using a hashing method. In [93], information in depth images and silhouettes is used in a learning framework to infer poses from a database.

2.3 Alignment Frameworks

There are many techniques that can be used to align a model with image data. This includes formulating optimization problems, probabilistic modeling, and learning image to pose mappings directly from data. The success of these algorithms depends on the efficiency with which it produces results, the amount of data needed to train, and their ability to generalize from this training data.

2.3.1 Direct Optimization

These methods try to align a pose with image observables by formulating it as an optimization problem. This include standard techniques such as gradient descent [11], dynamic programming techniques[24], and other optimization frameworks[66][10].

Gradient based search methods are standard numerical techniques to find local optimum of a cost function given an initial guess. As such, they are suited for tracking. In [36] a 2D model is fitted to an optical flow field using such a technique. As this kind of search method is largely a local search, gradient based methods are better suited for tracking and can lose track of the underlying object in the presence of fast motion or unreliable

visual observations. Nevertheless, they have a large degree of modeling flexibility as they can accommodate arbitrary but smooth objective functions.

Due to the high dimensionality of the search space, a purely exhaustive search in this setting is not practical without making assumptions on the form of the function to be optimized. In [24], the human body is treated as an elastically connected set of rigid limb sections. Each limb is observed independently, while elastic constraints only exist between pairs of limbs arranged in a tree structure. In this setting, a globally optimal pose can be computed in a linear in the number of parts by first searching for each individual part over translations, rotations, and scales. Following this, the optimal pose is found by combining the detection results of each individual part efficiently.

While this methodology can efficiently produce a global solution, it is limited in what can be modeled. In particular, in [24] the only pairwise terms that are considered are the elastic constraints. This limitation is necessary because it yields a structure that can be solved tractably. In [66] additionally pairwise constraints can be explored using integer quadratic programming techniques. This method works with image segments and assume limbs correspond to segments.

In localizing 2D poses we can also make use of deformable template matching. In [55] shape context descriptors are used to establish point correspondences between templates and novel images the *weighted bipartite matching*. Poses can be aligned between these two by matching these points on a 2D kinematic structure. Several templates can be constructed to cover a variety of poses. This reduces the problem to that of fitting a 2D pose to point correspondences. This simplifies the matching of poses, provided points can be matched correctly.

Works such as [66] combine over-segmented images into human poses, often assuming that individual segments correspond to limbs. In [10] image segmentation and pose estimation are integrated by framing the segmentation problem in a graph-cut framework. This method offers an advantage over other segmentation based methods in that it can combine various image features (edge, background, foreground etc) and won't be sensitive to errors in the initial segmentation.

2.3.2 Sampling Based Methods

In [24][66] a specific form of model was assumed which resulted in an algorithm that can be used to find an optimal pose efficiently. Framing pose estimation in a manner that can be solved efficiently limits modeling flexibility. While gradient based methods can model in greater generality, they have difficult dealing with local minima and the multi-modality of solution spaces.

Sampling based methods seek to align poses to image observables by maintaining a set of stochastically generated candidate configurations. This set of pose configurations is a representation of the distribution of poses with the given set of image observables. Particle filter methods track the evolution of a distribution of poses over time [40][73] [76][47].

Without special consideration, sampling based methods require a large number of particles to adequately represent high high-dimensional spaces. This greatly increases the computational demands of these algorithms. This problem can be mitigated with strong motion constraints or priors[73] or dimensionality reduction techniques[47].

Many works have been proposed to reduce the number of particles required. In [21] an annealed particle filter is used to numerically find optimum in an objective function through a randomize exploration of pose space. An annealing process is used to allow few particles to explore the search space and concentrate on a global optimal.

In [13][15] and [82][57] methods that augment particle filters or stochastic search with local search such as gradient based optimization have been proposed. Using the local search greatly reduces the number of particles needed to find optimum.

Techniques have also been proposed to improve the manner in which particles are generated. In [83] a kinematic flip process is used to cope with ambiguity of limb orientations along the line of sight in single camera tracking. A processes is introduced to flip through possible limb configurations by generating samples along depth. In [43] data driven MCMC is used to add particles to the steady state distribution using bottom-up part detectors.

The high dimensionality of the pose space can also be reduced by considering part-based representations. This effectively reduces the size of the search problem because not all parameters need to be estimated simultaneously. For example, belief propagation is also used in [84][76] to find pose configurations by considering pairwise constraints between limbs. Also in [49] individual body parts are robustly assembled into body configurations using a Ransac approach. False configurations are eliminated using a weak heuristic. The resulting configurations are then weighted with an a priori mixture model of upper body configurations and a saliency function and are used to infer a final pose.

2.3.3 Regression Methods

Sampling based methods seek to directly model the interaction between image observations and pose models in order to find potential poses. These largely generative methods, while model based and generalize well, are fundamentally computationally demanding. An alternative to this is to directly learn mappings from image observables to a pose using sets of labeled training data (discriminative algorithms). This can be accomplished using a variety of approaches which range from multi-dimensional functional approximation to hashing.

Due to the high dimensionality and multi-modality between pose and image observables, many works find clusters or groupings that form simple maps. In [69], clusters are formed between image observations (silhouette) in the input space and model parameters in the output space. After these clusters are learned, mappings between the clusters can be computed. In [23] activity manifolds are learned on the domain as well as the mapping between the manifold and image data. Estimation reduces to projecting the input onto the manifold and then mapping the input to the corresponding pose. In [2] a mixture of regressors is used to help deal with the multi-modal nature of this mapping.

In [1] a direct mapping between image descriptors (histogram of shape context descriptors of silhouette) and pose parameters is learned without an explicit body model. In this work, both regularized least squares is examined, and regression with Relevance Vector machines is proposed.

Mathematical mappings between image data and poses may generate configurations that do not corresponding to physically meaningful poses. In [72] a direct mapping

between image and a database of poses is learned using Parameter Sensitive Hashing. Here a database of poses and corresponding image observations is available. The hash function is sensitive to the similarity in parameter space, so neighboring poses can be found in sub-linear time. Given the top candidates in a novel image a linear mapping is then used to further refine the fit to image observations.

The use of appropriate features is important for these learning methods. The work of [8] makes use of boosted feature selection methods to construct a mapping between an image patch known to contain a human figure and its corresponding articulated pose. In [58] use HOG features to train a set of piecewise linear regressors that map partitioned regions of pose space.

Chapter 3

Single View Forearm/Limb Tracking

We describe an efficient and robust system to detect and track the limbs of a human from in color image sequences. By focusing our effort on the forearm, we are able to reduce the number of parameters to a manageable number, while still maintaining pose information.

Of special consideration in the design of this system are real-time and robustness issues. We thus utilize a detection/tracking scheme in which we detect the face and limbs of a user, and then track the forearms of the found limbs.

Robustness is implicit in this design, as the system automatically re-detects a limb when its corresponding forearm is lost. This design is also conducive to real-time processing: while detection of the limbs can take up to seconds, tracking is on the order of milliseconds. Thus, reasonable frame rates can be achieved with a short latency.

Detection occurs by first finding the face of a user. The location and color information from the face can then be used to find limbs. As skin color is a key visual feature in this system, we continuously search for faces and use them to update skin color information. Along with edge information, this is used in the subsequent forearm tracking.

In this system, we make use of a 2D articulated upper body model for detection, and simple forearm models for tracking. Using simple 2D models for tracking people has several advantages over full 3D systems. In particular, because they have reduced dimensionality they tend to be less computationally demanding. Also, 2D models exhibit higher stability and fewer degeneracies with a single camera [57]. Since depth is not directly observed in a single camera system, it tends to be highly unreliable when estimated.

While 2D models are better suited numerically for single view analysis, they are limited in expressive power. To address this, we make use of multiple 2D tracking models tuned for motions ranging from waving to pointing.

The rest of this chapter is organized as follows: In section 3.1 we present an overview of relevant work. In section 3.2 we present the details of our system. In section 3.3 we demonstrate the effectiveness of this approach on test sequences. In section 3.4 we conclude and provide future directions of research.

3.1 Related Work

Our strategy for limb detection is based on the work of [24]. Here human models consist of a collection of 2D part models representing the limbs, head and torso. The individual parts are organized in a tree structure, with the head at the root. Each part has an associated detector used to measure its likelihood of being at a particular location, orientation and scale within the image. Also, soft articulated constraints exist between part/child pairs in the tree structure. The human model is aligned with an image by first searching for

each individual part over translations, rotations, and scales. Following this, the optimal pose is found by combining the results of each individual part detection result efficiently.

As with detection, human body tracking has been extensively studied in the computer vision literature. In this problem, however, the task is to simply update a model's pose between successive image frames. Tracking limits the scope of the solution space, as continuity may be assumed. There are various kinds of tracking methods ranging from gradient based optimization methods [11] to sampling methods [76] and combinations of these [81] used in both single and multi-camera settings.

In this work we concentrate on tracking image regions corresponding to the hands and forearms rather than a complete articulated object. This allows for very fast processing. Our approach to tracking is more akin to the kernel based tracking methods [16]. However, we use a tracker that also accounts for the orientation of a region of interest such as in CAMShift [9]. Similar to [96], this is found via an optimization framework, however we perform this optimization directly using gradient based methods.

3.2 Approach

We now describe the details of this system. This system is designed to be robust and run in real-time. We thus make several simplifying assumptions. First, we assume only one user is present. We also do not explicitly modeling clothing. Instead, we assume the user wears short sleeve shirts and the forearms are exposed. This assumption simplifies the detection of good visual features, as skin regions and boundaries can be extracted with greater ease. The assumption is also fairly valid in warmer environments. We also assume

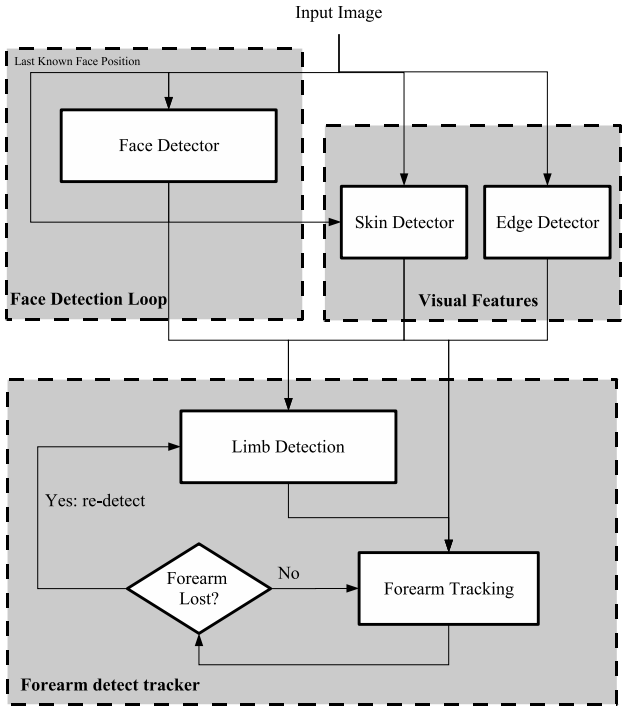


Figure 3.1: An overview of our approach

that the scale of the objects does not change dramatically. This greatly reduces the search space during detection as we only need to search over rotations and translations.

The overall approach is illustrated in Figure 3.1. The system contains a face detection module, and a visual feature extractor, in addition to the forearm detector/tracker. Knowing the face location constrains the possible locations of the arms (i.e. they need to be close to the head) as well as dynamically provides information about skin color, an important visual cue used for both detection and tracking. This module is described in section 3.2.1.

The forearm detector/tracker module contains a limb detector and forearm tracker. Limb detection is described in section 3.2.2. The detected forearm locations can then be used to initialize the tracking system described in section 3.2.3. Here, we use multiple 2D

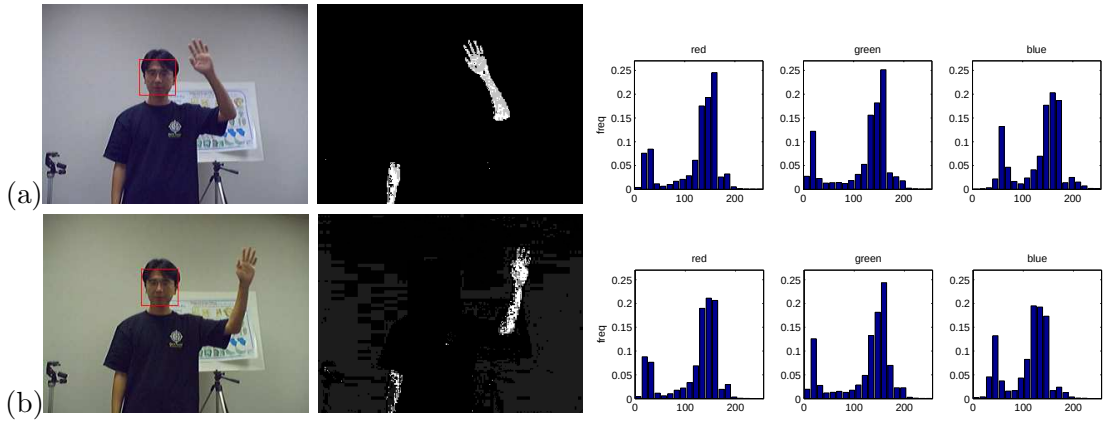


Figure 3.2: Use of the results of the face detector in skin detection as described in section 3.2.1. Between (a) and (b) the illumination and white balancing of the camera changes. This can be seen in the blue channel of the color histograms. Skin pixels are still properly detected.

limb tracking models to enhance tracking of the underlying 3D structure. This includes models for lateral views (waving) as well as for pointing gestures as shown in Figure 3.4. The switch between these two tracking models is described in section 3.2.4.

3.2.1 Face and Visual Feature Extraction

This module searches for key image features used in the limb detection and tracking modules.

In this module we make use of a Harr face detection as implemented in OpenCV [9]. To increase computational speed and, to a lesser extent, robustness, we embed this process in another detection/tracking scheme. Initially, we search for a face in the entire image. In subsequent frames, we only search in a neighborhood around the previous face result. If a face is not found in this limited image area, we search the entire image again. If the face is still not found, we use information from the last found face.

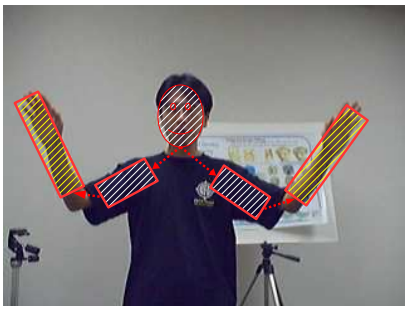


Figure 3.3: Articulated upper body model used for limb detection. The limbs are arranged in a tree structure as shown, with the head as the root. Between each parent/child pair their exists soft constraints. In our work we anchor the head at the location found from the people detection (section 3.2.2) module and only incorporate visual features of the forearm.

The color information in the image region of the detected face can then be used to initialize a hue-saturation space histogram. This histogram can then be used to assign each pixel in the entire image a likelihood of being a skin pixel. Following this we create of a histogram of skin likelihoods and zero out those pixels falling in the least likely bin. This is done to eliminate pixels that constitute the least likely 40% of skin pixels. The likelihood scores of the remaining pixels are then rescaled to be between 0 and 1.

Examples of this process are shown in Figure 3.2. From this we see that adapting the skin color histogram to the pixels in the face region increase the robustness of skin detection to changes in illumination. In addition to skin color information we make use of a Canny Edge detector for boundary information.

3.2.2 Limb Detection

Given the face, skin, and edge information we can then search for the arms. This is accomplished with the use of the upper body model used shown in Figure 3.3. This model encodes the upper arms, lower arms and head. Between each limb are soft constraints

that bias the model to a rest state shown. These constraints allow the limbs to easily spin about their joints, while making it more difficult for the limbs to move away from the shown articulated structure.

Each limb can be associated with a detector that grades its consistency with the underlying image. In this work, we only attach feature detectors to the lower arms as they are likely to be the most visible part of the arm during a gesture, and the head position is constrained to be at the position found by the face detector. The lower arm detector used in this work is based on both edge and skin color information. In particular, a given translation, $\mathbf{t}$, and orientation, θ, within the image is graded according to the following:

$$\begin{aligned} y(\mathbf{t},\theta) &= \sum_{\mathbf{x}\in BP} D(R(\theta)\mathbf{x}+\mathbf{t}) \\ &\quad +\lambda \sum_{\mathbf{x}\in SP} -logP_{skin}(R(\theta)\mathbf{x}+\mathbf{t}) \end{aligned} \tag{3.1}$$

where $R(\theta)$ is a rotation matrix, BP consists of boundary points, SP consists of skin colored regions such as the hand/forearm, $D(\mathbf{y})$ is the distance to the closest edge point in the image (computed using a distance transform [24]) and $P_{skin}(\mathbf{y})$ is the probability of the pixel $\mathbf{y}$ being skin color.

To align this model with an image, we seek to optimize the response of (3.1) subject to the soft constraints present in the model and the given location of the head. This can be formulated as an optimization problem:

$$\mathbf{\Theta}* = \arg\min y(\mathbf{\Theta}) + \gamma c(\mathbf{\Theta}) \tag{3.2}$$

where $\mathbf{\Theta}$ is the translation and rotation of each part shown in Figure 3.3, $y(\mathbf{\Theta})$ represents the image matching score shown in (3.1) , $c(\mathbf{\Theta})$ quantifies the deviation from the relaxed limb configuration shown in Figure 3.3. This term is detailed in [24].

Observing that the constraints between each limb form a tree structure, we can solve (3.2) using the method of [24]. This is accomplished by first computing (3.1) over translations and rotations (except over the face) for each of the forearms which are at the leaves of the tree. We can then *assemble* the configuration that minimizes (3.2) by considering a discretized set of possible locations of successive limbs in the tree structure. At the end of this process is an array defined over the range of poses of the root limb (i.e. the head). In each element of the array is the optimal configuration of the child limbs along with its overall score. The process is described in detail[24].

Rather than selecting the overall optimal pose in this array, we simply use the configuration attached to the head location found by the face detector. Also, instead of treating the upper body as a single entity, we treat the left and right arms separately. This allows us to detect and track them separately. The underlying model disambiguates them.

3.2.3 Limb Tracking

Detection is useful when we have no prior knowledge of the limb location. In this case we need to search the entire image to find potential limbs. After detection, however, we know the limb in the next frame must be near the limb found in the previous. Using this smoothness assumption, we can track the forearms of the user using local information. This is more computationally efficient than a full detection and is often more robust. In general, however, it is possible that one can move faster than frame rate, and thus cause

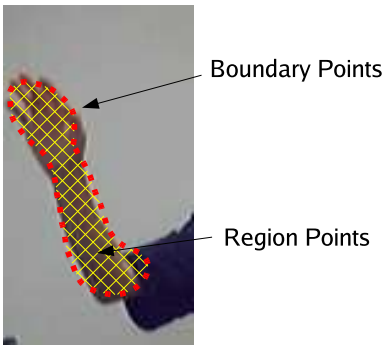
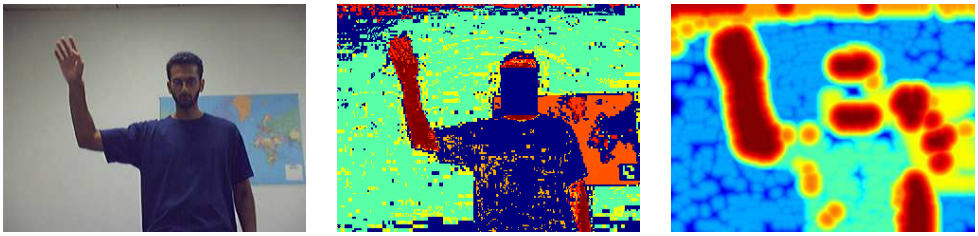


Figure 3.4: The tracking model represents an object as a collection of feature sites that correspond to either skin colored (at the intersection of the yellow lines) or a boundary pixel (red squares). The consistency of the model with the underlying image is measured as a weighted sum of the distance of the boundary pixels to the nearest edge pixel and the skin color scores under the skin feature sites.



(a) (b) (c)

Figure 3.5: The negative log of Skin color probability before (b) and after (c) applying the continuous distance transform. The figure in (c) is more suitable for the optimization of equation (3.3) as the skin regions have smoother boundaries. As reference, the original frame is shown in (a). In these figures a red color indicates a smaller value while a blue color indicates a larger value.

the tracker to lose its object. It is thus imperative that a tracker knows when it loses track so that a re-detect can be executed.

In this approach we use a new efficient tracker. We model regions of interest as a collection of *feature sites* that indicate the presence of a skin colored pixel or a boundary pixel. This is illustrated in Figures 3.4.

Tracking is achieved by maximizing the consistency of the feature sites with the underlying image. This can be posed as an optimization problem over translation and orientation as expressed in the following equation:

$$\begin{aligned} \theta*, \mathbf{t}* &= \arg\min \sum_{\mathbf{x}\in BP} D_{dist}(R(\theta)\mathbf{x}+\mathbf{t}) + \\ &\lambda \sum_{\mathbf{x}\in SP} F_{skinScore}(R(\theta)\mathbf{x}+\mathbf{t}) \end{aligned} \tag{3.3}$$

where $R(\theta)$ is a rotation matrix, BP consists of boundary points, RP consists points in what should correspond to skin, and $D_{dist}()$ yields the distance to the nearest boundary point within the region of interest. This is efficiently calculated using a distance transform of the detected edge points[24]. In addition, the term $F_{skinScore}(x, y)$ represents a function that is zero when the image has a skin colored pixel at location (x, y) and is large otherwise.

While a natural choice for $F_{skinScore}(x, y)$ is the negative logarithm of the skin pixel probability, $(-logP_{skin}(\mathbf{y}))$ we solve (3.3) using gradient based methods. Thus we need the $F_{skinScore}(x, y)$ to be *smooth*. This is achieved by using its continuous distance transform [24]:

$$F_{skinScore}(\mathbf{y}) = min_{\mathbf{x}}(\|\mathbf{x}-\mathbf{y}\| + -\alpha log(P_{skin}(\mathbf{x}))) \tag{3.4}$$

This transform tends to give smoother images that have basins around regions of high skin probability as shown in Figure 3.5. This serves to improve both the speed of convergence when solving (3.3) as well as the range of convergence.

We solve (3.3) directly by using a Levenberg-Marquardt optimizer [61] with a fixed number of iterations. We also only compute D_{dist} and $F_{skinScore}$ in a fixed size region

(a) Profile Model (b) 3/4 Profile Model (c) Pointing Model

Figure 3.6: Tracking models used for forearms moving in 3D. The tracking model used for laterally viewed forearms is shown in (a), pointing forearms is shown in (c). The model in (b) is for forearms viewed between (a) and (c).

about the previous pose of the forearm. This region is large enough to accommodate movement while keeping computational costs low. Finally, the face region is masked out to prevent detectors from being attracted to it.

3.2.4 Tracking Models

We use the tracker described in section 3.2.3 to track the forearm of the user. For this purpose an appropriate model is required. The advantage of using simple 2D models to track the 3D forearm is that they can be used efficiently and robustly so long as they *match* the underlying 3D structure. However, a single 2D model is not ideal for all views. For example a lateral, waving forearm in which the profile of the forearm is visible, is significantly different from tracking an arm where a user is pointing near the camera and only the hand is visible.

To effectively track the forearm as it moves in 3D we utilize multiple 2D tracking models as shown in Figure 3.6. The model shown in (a) is designed to track laterally viewed forearms which often occurs in waving gestures. In (c) the circular shaped model is designed to track forearms pointing towards the camera. In this case only the hand

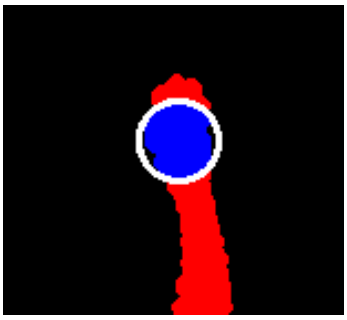


Figure 3.7: Fullness and coverage scores in the tracking system for the pointing model. The blue pixels correspond to skin colored pixels the model accounts for, while the red colored pixels correspond to skin colored pixels the model misses. The fullness score corresponds to the percent of pixels in the interior that are skin colored. This is just the ratio of the blue pixels to the total number of skin sites in the model. The coverage score correspond to the percent of skin pixels covered in the expanded region of interest. This is the ratio of the number of blue pixels to the total number of skin colored pixels (blue + red).

is visible and the circle effectively tracks this hand. The model in (b), which is just a shorter rectangle, is useful for situations between (a) and (c).

Model Switching

After a forearm is detected using the method of section 3.2.2 we must track the forearm using one of the models described in section 3.2.4. This choice is based on an intuitive understanding of how to switch between models as well as how well each model accounts for the underlying visual features. This is quantified using the percent of pixels in the model's interior that are skin colored (i.e. the *fullness* score), as well as the percent of skin pixels covered by the model in an expanded region of interest (i.e. the *coverage* score). These scores, as illustrated in Figure 3.7, correspond to the skin pixels accounted for by the model and those that are missed by model.

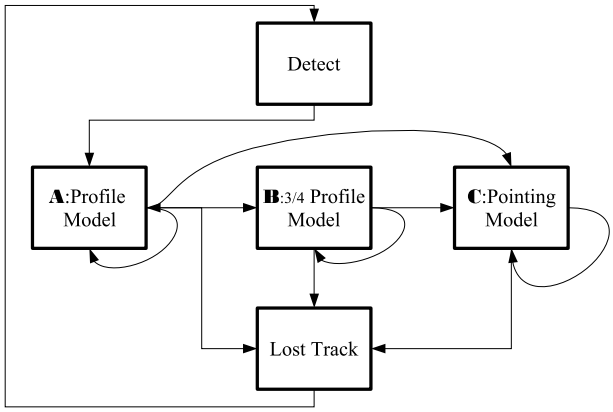


Figure 3.8: Summary of tracking model selection state transitions.

The transitions between models are summarized in Figure 3.8. This transition diagram was designed to help track the forearm as it switches from waving to pointing at the camera. Initially we consider the fully lateral forearm model. This model is selected because a gesture is frequently initiated by waving to a device. If the fullness score falls below a threshold, we can switch to 3/4 profile model or the pointing model. We switch to the 3/4 profile model if its fullness score is above 90% and it accounts for 90% of the nearby skin pixels and we can switch back to the profile model if its fullness is above 90%. From either profile model, we can switch to the point model when its fullness is above 90% and only miss 10% of the skin pixels around it.

We also note that in transitioning from pointing to a profile view, a re-detect must be issued. This is because, without any orientation information, it is easier to just re-detect.

In addition to switching between the models we must also detect when the tracker simply loses track of the underlying forearm. This can occur, for example, if the user moves too quickly for the given frame rate. We detect this by simply keeping track of the number of underlying skin pixels in the currently used tracking model. If the total number

of skin pixels falls below a threshold for any model, the system resets and re-detects using the method of 3.2.2.

3.3 Results

As illustrated in the following sequences acquired at 15fps, this system has been extensively tested with various users and indoor settings. Although these experiments were executed off-line, they illustrate the effectiveness of our approach. Real-time performance is discussed in section 3.3.1.

In Figure 3.9 we show an example in which the tracker loses track and must be re-initialized. Adjacent to each figure is its frame number. In Figure 3.9 the initial pose fore each limb is correctly detected at frame 0. From this, each tracker can be initialized and successfully track the limbs until frame 6. In frames 6, 7 and 8, the subject moved significantly faster then the acquisition rate. The trackers lose track and are re-initialized with another detection. In the remaining frames (9-14) limbs are tracked correctly again.

In Figure 3.10(a) we show the results of the limb detection and tracking module when model switching is employed. In this 32 frame sequence the user transitions from waving to pointing. In frame 0 the initial pose of each limb is correctly detected. From this, each forearm tracker can be initialized and they successfully track the limbs until frame 14. Between frames 14 and 15, the subject's right forearm transitions from waving to pointing. The corresponding tracker successfully switches to the pointing model. In frame 18 the left forearm follows suit. Over the remaining frames the user lowers his

arms. The tracking model subsequently switches via re-initialization and the limbs are tracked correctly again.

In Figure 3.10(b) we show an additional example in which the user moves his arms up and down in a 20 frame sequence. This requires the system to switch between models and re-initialize itself. From this figure we see the system is able to keep reasonable pace.

In Figure 3.11 we show the system running on a PETS sequence. Note that this environment is very different than that of the other test sequences. To run the system, the scale was manually adjusted and attention was focused on the user to the far left. The system was able to detect and track both his arms.

Numerical evaluation is shown in Figure 3.12. Here we show the errors in terms of *joint positions* as shown in Figure 4.1(a). In particular, we report the average distance between corresponding points on the arm models and labeled joint positions. In Figure 3.12(a) the average errors are shown for the overall system, the detection system, and the tracking system. Here we see that errors in the detection system were large compared to tracking. This is due to mis-detections that were either smoothed out by tracking or recovered through another detection.

In Figure 3.12(b), the errors for each joint are shown for the tracking system. As the tracker only maintains the position of the forearm, only the elbow and handTip joints carry any information. The larger error in the hand tip corresponds to the placement of the pointing circle at the *center* of the forearm skin *blob*. In Figure 3.12(c), we see the system spent most of its time tracking full profile views, and significantly less time in detection then in tracking.

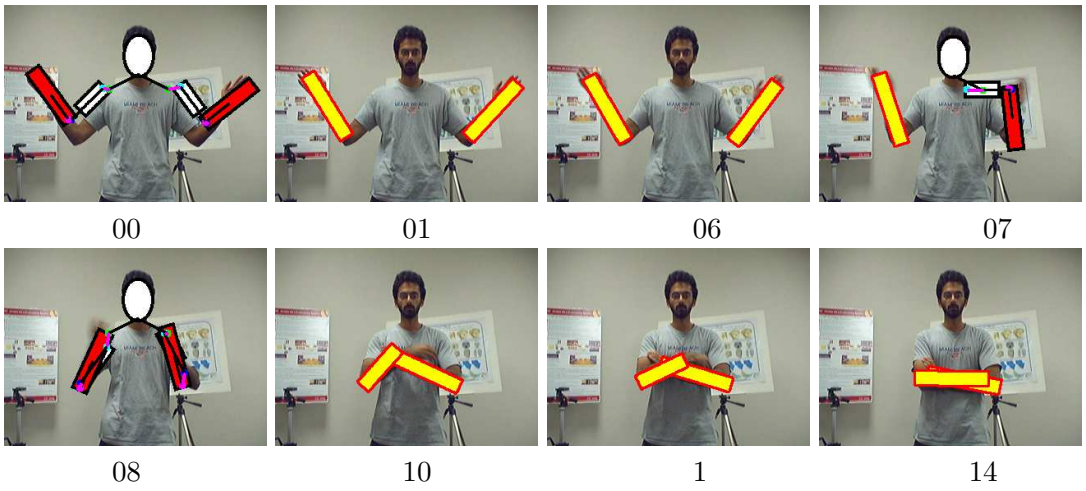


Figure 3.9: Results of the limb detector and tracker when automatic re-initialization is required. In frame 0 the initial pose of each limb is detected and successfully tracked until frame 6. Here the trackers lost tracker and the system re-initialized itself.

3.3.1 Real-Time Implementation

This system has been implemented on a Dual 3GHz Xeon with hyper-threading enabled. To make full use of this multi-processor platform, we use the multi-threaded programming models offered by the Software Architecture for Immersipresence (SAI)[28]. Using SAI we can create separate processing centers known as *cells* which we arrange in a chain. Each cell has its own static data contained in *Node* structures. Data can also be passed down this chain (via *pulses*) to facility the passing of runtime results between cells.

In our system, we have separate cells for image acquisition, face detection, and feature extraction. We also have separate cells for the detection/tracking of each individual arm. While data is passed between cells serially, each cell is allowed to process data as soon as it is available, thereby enabling pipelined processing. Buffering of data between cells is implicit in SAI via critical sections surrounding the static data.

On this platform face-detection and visual feature extraction takes about 50ms per frame, while limb detection takes 400-700ms (per limb) and forearm tracking takes about 50ms per frame. Clearly, detection is the bottleneck in this system. We reduce its impact by preventing the system from performing successive detection on a given limb until at least 5 frames have passed. This prevents excessive buffering of frames and keeps the latency low. When users are moving at normal speeds, detection is not called too frequently and the dropped frames are not a noticeable issue. The system currently runs at about 10 frames per second.

3.4 Discussion

We have described the design and implementation of a limb detection and tracking system. The system works robustly and in real-time as demonstrated by the examples. We have successfully implemented this system on a dual Xeon 3GHz machine with hyper-threading technology. The system works robustly and efficiently, and has been extensively tested qualitatively.

While this method achieves realtime performance, some restrictive assumptions were made. Firstly, we assumed that the forearm was visible and could be largely modeled by skin blobs. While this assumption works well when true, it limits the use of this system in a general setting.

We also do not provide a mechanism to deal with extensive background clutter and false alarms in the detection process. The existence of such clutter causes the likelihood

to become multi-modal and detecting a single *optimal* forearm does not necessarily yield the *correct* forearm.

We defer the automatic construction of robust detectors to Chapter 5. In the following chapter, we propose methods to improve the pose detection component of this system. This is accomplished by searching for multiple candidates that form local optima in the space of solutions.

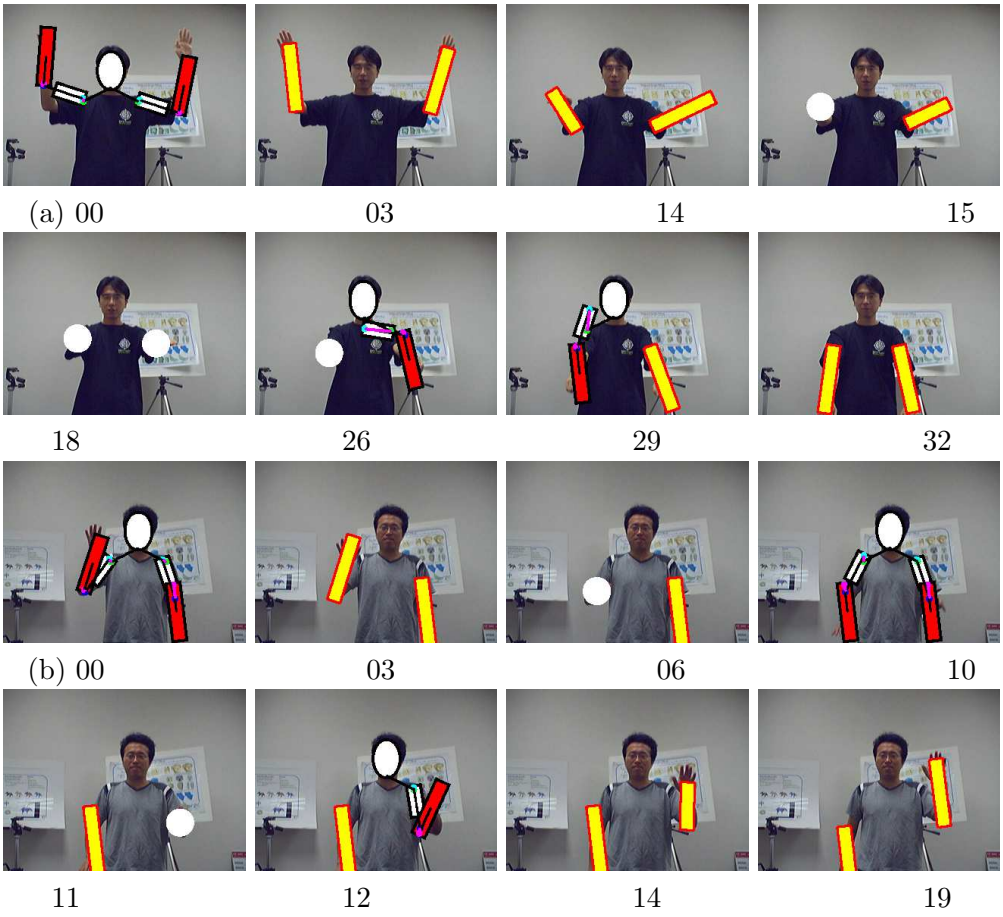


Figure 3.10: The results of the limb detection and tracking module when model switching is employed. In (a) the tracking models for each forearm switched from the profile (rectangular) model to that of the pointing (circular) model between frames 14 and 18. The tracking models switch back to profile model in the remaing part of this sequence. In (b) the the user moves his arms up and down requires the system to switch between models and re-initialize itself.

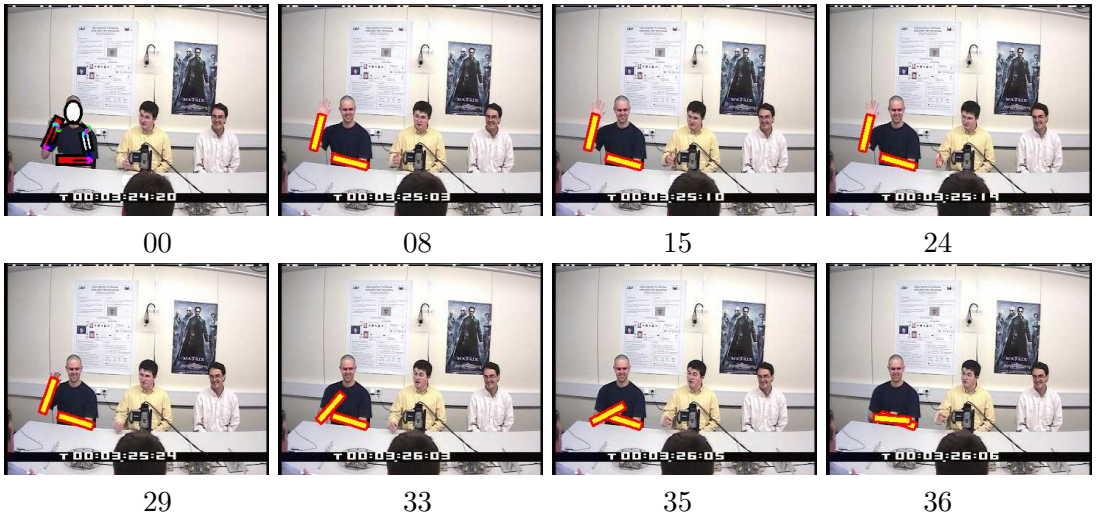


Figure 3.11: The results of the limb detector and tracker on a PETS sequence. In frame 0 the initial pose is detected and successfully tracked through frame 36.

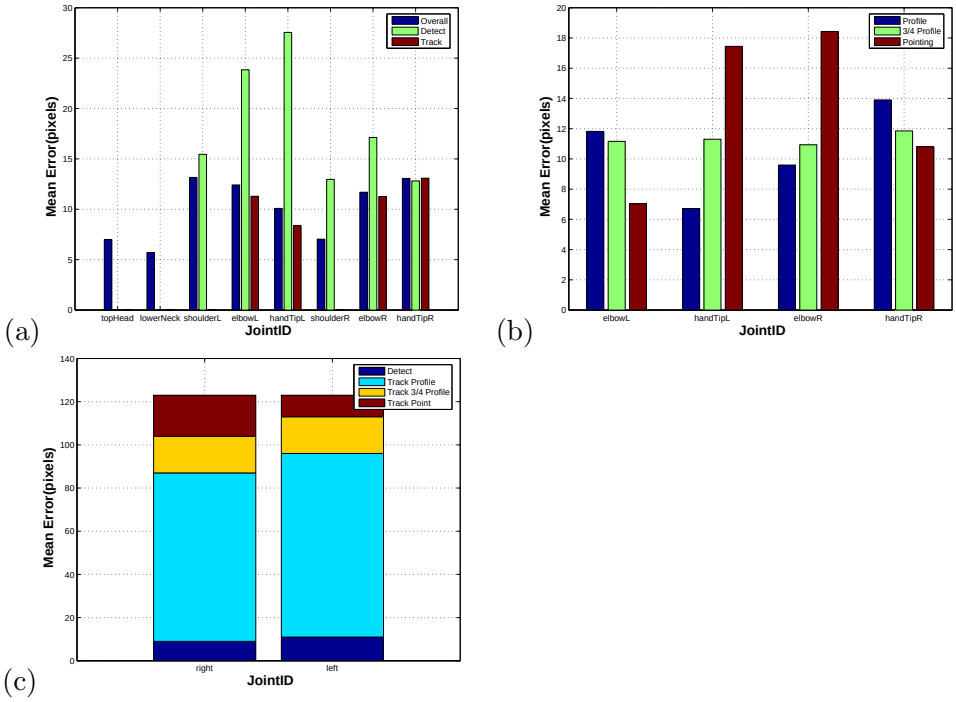


Figure 3.12: In (a) the average errors for each joint over all test sequences. In (b) the average errors in each tracking state. In (c) the frequencies in each state of the overall detection and tracking process

Chapter 4

Single View 2D Pose Search

In this chapter we present our framework for pose estimation from a single color image and sequences. In general, pose estimation can be viewed as optimizing a multi-dimensional quality of fit function. This function encodes fidelity of a model to observables and a prior distribution. The success of aligning a model in this way depends on the amount of information that can be encoded into this function as well as the ability to optimize it. The more relevant observable and prior information one can fuse into a fitness function, the more likely the error surface becomes peaked on the *right* solution. However, highly detailed models often become computationally expensive and difficult to optimize.

In many cases, the form of a fitness function can be restricted so that the global optimal or good approximations to the global optimum can be obtained efficiently. This limits what one can actually model, which may result in configurations that minimize the fitness function but do not necessarily correspond to the correct answer. Nevertheless, it is likely that the true solution has at least a local optimum under such a function.

In this chapter we model the projection of poses in the image plane as a tree of 2D joints positions. We then define a quality of fit function on this tree structure by attaching

simple part based detectors between parent child joint pairs. Defining a pose in this way allows us to efficiently find an optimal pose configuration with respect to the saliency measure.

To address the modeling limitations of this representation, we then propose a method that explores this space of joint configurations and identifies locally optimal yet sufficiently distinct configurations. This method makes use of a bottom up technique that maintains configurations as it advances from the leaves of the tree to the root. From these candidates, a solution can then be selected using information such continuity of motion or a detailed top down model based metric. Alternatively, these candidates can be used to initialize higher level processing.

We also adapt this algorithm for use on a sequence of images to make it even more efficient by considering configurations that are either near their positions in the previous frame or overlap areas of interest in the subsequent frame. This allows the number of partial configurations generated and evaluated to be significantly reduced while both smooth and abrupt motions are accommodated. These algorithms are then validated on several sets of data including the HumanEva set.

4.1 Related Work

Modeling the projection of a 3D human pose has been explored in [57]. This idea is formalized using the Scaled Prismatic Model (SPM), which is used for tracking poses via registering frames. In [55] a 2D articulated model is aligned with an image by matching a

shape context descriptor. Other 2D models include pictorial structures[24] and cardboard people[41].

Similar to a collection of joints, other approaches model a human as a collection of separate but elastically connected limb sections, each associated with its own detector[24][63]. In [24] this collection of limbs is arranged in a tree structure. It is aligned with an image by first searching for each individual part over translations, rotations, and scales. Following this, the optimal pose is found by combining the detection results of each individual part efficiently. In [77] limbs are arranged in a graph structure and belief propagation is used to find the most likely configuration.

Modeling human poses as an collection of elastically connected ridge parts is an effective simplification of the body pose, however it does have limitations in its expressive power. This is because the 2D images are actually formed by projecting 3D models. In particular self occlusions are not modeled. Also, since representations are more targeted for lateral facing limbs, changes in perspective are modeled as changes in scale.

To extend the expressive power, in [90] multiple trees are used are used to extend this graphical model to deal with occlusions. In [98] max use of and AND/oR graphical model to increase the representational power. [[layered pictorial scrutures]]

In this chapter we address these modeling limitations by finding the top N locally optimal pose configurations efficiently and exhaustively. Higher level information, which is usually much more computationally expensive, can then be used to select from or be initialized by these candidates.

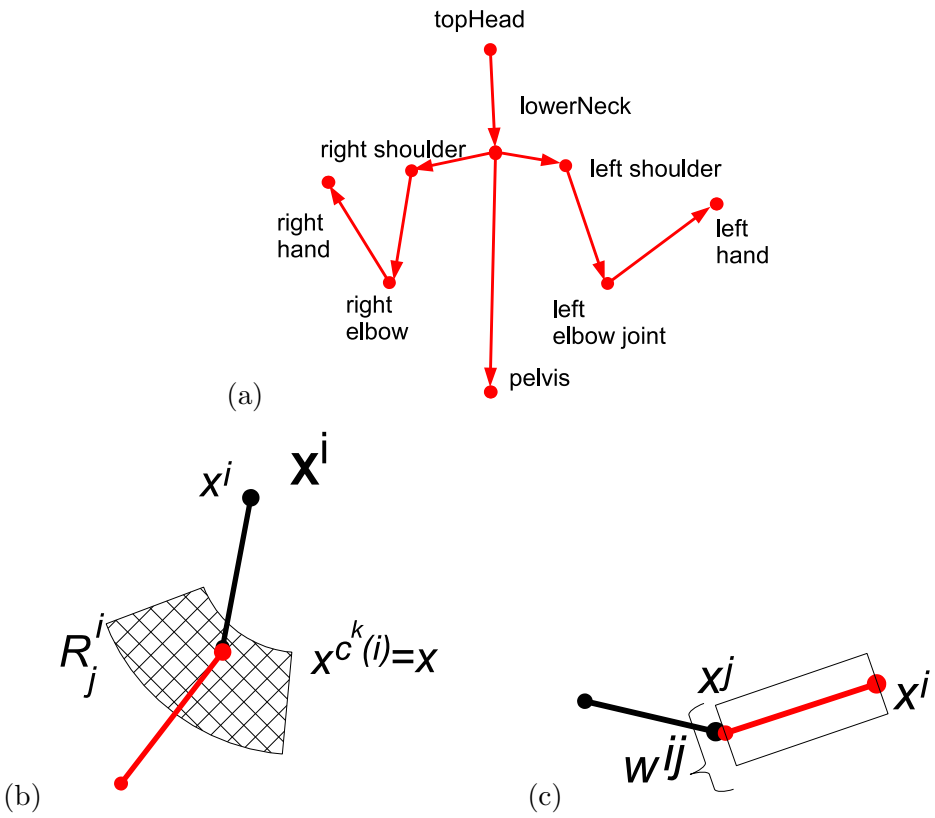


Figure 4.1: In (a) a configuration of joints (labeled) assembled in a tree structure is shown. In (b) the notation used is illustrated along with the permitted locations of child joint relative to its parent. In (c) a fixed with rectangle associated with a pair of joints is shown.

4.2 Model

We model the projection of a 3D articulated model. These are positions in the image plane as shown in Fig. 4.1(a). This is a natural representation for human image alignment in a single image. As shown in [57], modeling the projection of a articulated 3D object eliminates depth related degeneracies. Furthermore it may be possible to reconstruct a 3D joints in a post processing step using either multiple views or geometry [85][55].

We further encode this collection of joints in a tree structure (shown in Fig. 4.1(a)) and constrain the locations where a child joint can be relative to its parent joint as shown

in Fig. 4.1(b). We will refer to $\mathbf{X}$ as a tree, or configuration, of joints. Individual joints are specified as $x^i \in \mathbf{X}$. A sub-tree is specified with the super-script of its root joint, $\mathbf{X}^i$. Also note that $root(\mathbf{X}^i) = x^i$. The k^{th} child of joint of x^i is specified by $x^{c^k(i)}$. The locations a child joint can have relative to its parent are specified by R^i_j.

Similar to a collection of elastically connected rigid parts[24], this representation can be used to find configurations in a bottom up manner. A collection of joints, however, has fewer explicit degrees of freedom. In a pictorial structure, the additional degrees of freedom incurred by parameterizing each rigid limb section separately are constrained by using elastic constraints between parts. A collection of joints enforces these constraints implicitly.

For example, by modeling the upper body as a collection of fixed width limbs, we end up with 15 joints. This gives us 30 parameters. A similar limb model would give us 10 limbs each with a translation and rotation and length, for a total of 40 parameters.

4.3 Quality of Fitness Function

To align this model we construct a cost function:

$$\Psi(\mathbf{X}) = \alpha P_{image}(\mathbf{X}) + (1 - \alpha) P_{model}(\mathbf{X}) \tag{4.1}$$

where $\mathbf{X}$ denotes a tree of joint locations defined in section 4.2. The terms P_{image} and P_{model} evaluate $\mathbf{X}$'s image likelihood and prior respectively. The parameter α controls the relative weight of the two terms.

The term P_{image} is a part-based metric computed by evaluating part detectors within the fixed width rectangles between pairs of joints in $\mathbf{X}$. This is illustrated in Fig. 4.1(c). In particular

$$P_{image}(\mathbf{X}) = \prod_{(i,j)\in edges(\mathbf{X})} P_{part_{ij}}(x^i, x^j, w^{ij})\mathbf{M}^i(x^i)\mathbf{M}^j(x^j) \tag{4.2}$$

where x^i and x^j are parent-child pairs of joints in $\mathbf{X}$. This pair of joints correspond to a limb with fixed width w^{ij} . The term, $P_{part_{ij}}$ is a part based detector defined on rectangle of width w^{ij} extending from joint x^i to x^j. The term $\mathbf{M}^i(x^i)$ is a mask that can be used to explicitly force a joint to be within (or away from) certain locations.

The P_{model} term biases a solution toward a prior distribution. In this work, we do not model this term explicitly. Instead, we have constrained the locations a child joint can have relative to its parent, R^i_j to be points sampled on a rectangular or polar grid. We thus assume poses that satisfy the parent-child constraints are equally likely.

4.4 Peak Localization

To solve this problem, one could use the algorithm in Fig. 4.2 as a baseline design. Each configuration is graded according to Ψ. The least cost configuration, $\mathbf{X}*$, is repeatedly identified, and all configurations that are sufficiently similar (i.e. $diff(\mathbf{X}, \mathbf{X}*) < \sigma$) are removed. In this work $diff(\mathbf{X},\mathbf{X}^*)$ is the maximum difference between corresponding joint locations.

This procedure would produce an optimal sequence of solutions that are sufficiently different. The complexity of this procedure is

$$\begin{aligned} & O(|C_{in}|MN + F(N)|C_{in}|) \\ = \; & O(|C_{in}|(MN + F(N))) \end{aligned} \tag{4.3}$$

where the first term arises from applying $diff(\mathbf{X}, \mathbf{X}*)$, which is $O(N)$, to elements of C_{in} in order to get M configurations. The second term arises from applying Ψ, whose complexity we denote for now as $F(N)$ to elements of C_{in}.

Such an approach is computationally intractable given the size of C_{in}. We note that there are 15 joints, 14 of which have a parent. Thus, if we define R to be the maximum number of distinct locations a child joint can have relative to its parent (i.e. $\forall_{ij}|R_j^i| < R$), and denote $|I|$ to be the number of locations the root joint can have in the image, the number of candidate solutions is on the order of $R^{14}|I|$.

We can approximate this procedure, however, by assembling partial joint configuration trees in a bottom-up manner. Working from the leaves of the tree to its root, we maintain a list of locally optimal, yet sufficiently distinct configurations for each sub-tree. These lists are pruned using the algorithm shown in Fig. 4.2 to avoid exponential growth. As the configurations for sub-trees are assembled, they are reweighted with likelihood functions, $\Psi(\mathbf{X}^i)$, that depend only on the sub-tree.

This process continues until the root of the tree, and a list of optimally distinct configurations of joints is returned. The complexity of this procedure is $O(M^3N^3)$ and is described in detail below.

Function C_{out} = wprune(C_{in}, , σ, M)
/* Finds the best M configuration that are different by at least σ.
C_{in} ${}^{k}\{\mathbf{X}\}_{i=1}^{N_k}$ input candidates
C_{out} output candidates
/

grade each configuration in C_{in} according to Ψ
do
remove $\mathbf{X}*$ with lowest score from C_{in}
insert $\mathbf{X}*$ into C_{out}
remove any $\mathbf{X}$ from C_{in} s.t. $diff(\mathbf{X}, \mathbf{X}*) < \sigma$
while $|C_{out}| <= M$ and $|C_{in}| > 0$

Figure 4.2: Pseudo-code for baseline algorithm.

4.5 Candidate Search

To generate these partial configurations we maintain a list of candidate partial configurations for each sub-tree in $\mathbf{X}$, and at each possible location this tree can exist in the image. This is denoted by: $\{{}^{k}\mathbf{X}_l^i\}_{k=1}^{M}$. Here i refers to the node id of the root node in this configuration (for example, $i = shoulder$). These configurations are located at the l^{th} pixel $p_l = (x, y)$ and each candidate configuration in this list has a common root joint referred to as x_l^i. The index, k, specifies one such configuration.

This list can be constructed from the candidate configurations associated with the children of joint x_l^i, denoted by

$$\begin{aligned} \{\mathbf{X}_l^i\} = \quad & wprune(\{x_l^i \otimes {}^{k}\mathbf{X}_{l'}^{c^1(i)} \otimes \ldots \otimes {}^{k'^{(nc_i)}}\mathbf{X}_{l^{(nc_i)}}^{c^{nc_i}(i)}\} \\ & l' \in R_{c^1(i)}^i, \ldots, l^{(nc_i)} \in R_{c^{nc_i}(i)}^i \\ & k', \ldots, k^{(nc_i)} \in [1, M], M) \end{aligned} \tag{4.4}$$

The operator $\otimes$ denotes the joining of branches into trees, and $wprune()$ is shown in algorithm in Figure 4.2. As before, the variable, R_j^i, is the list of locations where the child joint j can be relative to its parent i, and nc_i is the number of children of node i.

Here, we are combining the M candidates from each sub-tree located at each point in R_j^i. If R is a bound on the size of $|R_j^i|$, the number of candidates passed to $wprune$ is bounded by $(MR)^{nc_i}$. This can be reduced if we prune candidates as we fuse branches in pairs:

$$\begin{aligned}\{\mathbf{X}_l^i\} &= wprune(wprune(\{x_l^i \otimes {}^{k'}\mathbf{X}_{l'}^{c^1(i)}\}\forall k'l', M)\otimes \\ &\quad \ldots) \otimes \{{}^{k^{nc_i}}\mathbf{X}_{l^{(nc_i)}}^{c^{nc_i}(i)}\}\forall k^{nc_i} l^{nc_i}, M)\end{aligned} \tag{4.5}$$

By processing pairs, we limit the number of candidates sent to $wprune()$ to be $M(RM)$. If we denote N^i as the number of joints in the sub-tree $\mathbf{X}^i$, the complexity for $wprune()$ is $(MN^i + F(N^i))$ times the size of the list to operate on. It will also be called nc_i times. Thus the overall complexity for an individual joint is $nc_i(MRM)(MN^i + F(N^i))$. This processing must be done for all N each joints and at every pixel, p_l that a sub-tree's root can be located. Since the number joints in each sub tree is bounded by N and the number of locations p_l is bounded by the size of the image, $|I|$ the overall complexity is bounded by:

$$O(NR|I| \max_i(nc_i)(M^3N^2 + M^2F(N)) \tag{4.6}$$

We also note that the Ψ defined in section 4.3 is computed as a sum of responses to parts of a configuration. In this framework, it can be computed in constant time, β, as

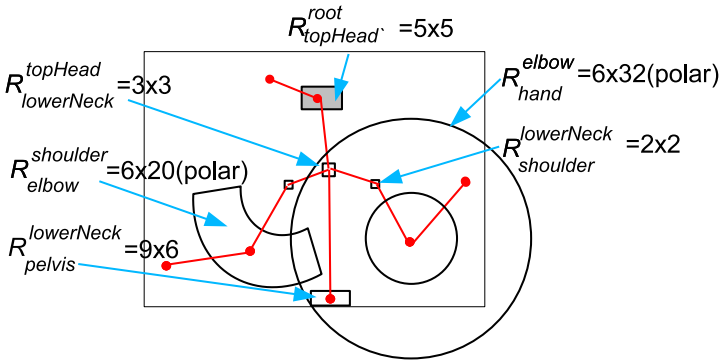


Figure 4.3: The relative positions of each child joint relative to its parent. Sizes shown are the number of discrete point locations in each region.

a sum of the scores of the partial configurations already computed and the computation of a constant number of terms. Thus the overall complexity is

$$O(R|I|\max_i(nc_i)(M^3N^3 + \beta M^2 N)) \tag{4.7}$$

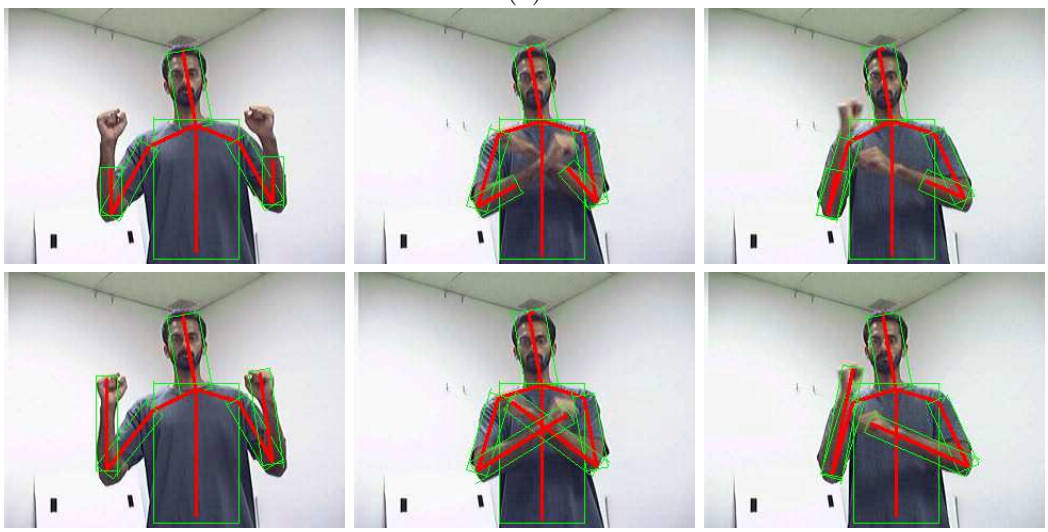
We must preserve the second term, because the constant is very large.

4.6 Results and Analysis

Examples of output of the method described in section 4.5 are shown in Figure 4.4. Here the model used is shown in Figure 4.3, with each R^i_j superimposed. In this sequence, we assumed the *topHead* joint to be within the gray rectangle shown. We further constrained the relative positions of the elbow and hand joints to be at polar grid locations within the regions shown. In particular, we considered 6 different lengths and 20 angular positions within the 90 degree angular range for the elbow joints relative to the shoulder and 6 different lengths with 32 angular positions in a 360 degree angular range for the hand

Rank	Image Error(pixels)	std
0	17.52	21.83
2	15.21	19.66
4	13.09	17.23
6	11.79	15.48
8	10.97	14.19

(a)



(b) (c) (d)

Figure 4.4: In (a) the average positional joint error for the Rank N solution taken over a 70 frame sequence along with its standard deviation. As the number of candidates returned increases, the error decreases. While optimal solution with respect to Ψ, may not correspond to the actual joint configuration, its likely a local optimal will. The top rows in (b)-(d) shows the optimal results with respect to Ψ, returned from the algorithm in 4.4. The second row shows the *Rank 5* solution.

joint relative to the elbow. The other joints are quantized at 4 pixel locations within their corresponding rectangles. The images used here are part of a 70 frame annotated sequence.

The term, $P_{part_{ij}}$ is the image likelihood of an individual part and is computed as:

$$P_{part_{ij}}(x^i, x^j, w^{ij}) = \prod_{cp \in Rect(x^i, x^j, w^{ij})} \hat{P}_{ij}(cp) \tag{4.8}$$

This likelihood is based on how well each underlying color pixel, cp, in the rectangle of width w^{ij} extending from joint x^i to x^j. (i.e. $Rect(x^i, x^j, w^{ij})$) belongs to a color distribution, . These distributions are modeled as simple color RGB and HS histograms and trained from example images. The widths of the limbs, w^{ij}, are known.

We found a set to 10 configurations under Ψ, such that no two joints are within 40 pixels (i.e $\sigma = 40$). Results with respect to the ground truth joint locations are summarized in Figure 4.4(a). We show the average joint error for the *Rank N* solution. Since our algorithm produces an ordered set of $M = 10$ configurations, the *Rank* $N < M$ solution is the configuration among the first N of M with the smallest average joint error with respect to the ground truth. From this, we see that as the number of candidates returned increases, the average distance to the correct solution decreases. This shows that while the solution that minimizes Ψ may not correspond to the actual joint configuration, it is likely a local minium will.

This is consistent with the results shown in Figure 4.4(b)-(d). In the top row the optimal solution with respect to Ψ is shown, while the Rank 5 solution is shown in the second row. In these images, the *Rank 5* solution is closer to the ground truth. On average it takes 982ms seconds to process each image. Of this time, 232ms is not dependent on the size of this problem (i.e. does not depend on N,M, R and nc) and can be thought of as a pre-processing step necessary for evaluating Ψ. Of the remaining 750ms that depend on the size of this problem, 200ms are devoted to evaluating Ψ.

4.7 Joint Localization in an Image Sequence

The algorithm in section 4.5 is polynomial, but it may still too slow for practical applications. Significant speed improvements can be gained if we exploit the smoothness of motion available in video, and limit the number of times Ψ is evaluated.

4.7.1 Motion Continuity

The complexity of our algorithm is directly proportional to the number of pixel locations, p_l, where each joint can be located. In computing the complexity in equation 4.7, this was bounded by the size of the image, $|I|$. If the motion of the joints in an image sequence is smooth, we only need to look for joints in a subsequent frame around their position in a previous frame. In this work we seek to maintain a list of M configurations. We can avoid having to commit to any one of these solutions by considering joint locations about any of the M candidate positions in the previous frame. In particular, we constrain each joint to be in a small a rectangle, W, about the corresponding joints in one of its M previous positions. This translates to a complexity of:

$$O(R|MW|\max_i(nc_i)(M^3N^2 + \beta M^2 N)) \tag{4.9}$$

Constraining the joints position in this way works well when the motion is smooth. However, there may be significant motion between frames that violate this assumption. This will likely occur on the hands and arms especially when the frame rate is 10-15fps.

We now describe an efficient way to handle the presence of such discontinuities, while enforcing smoothness.

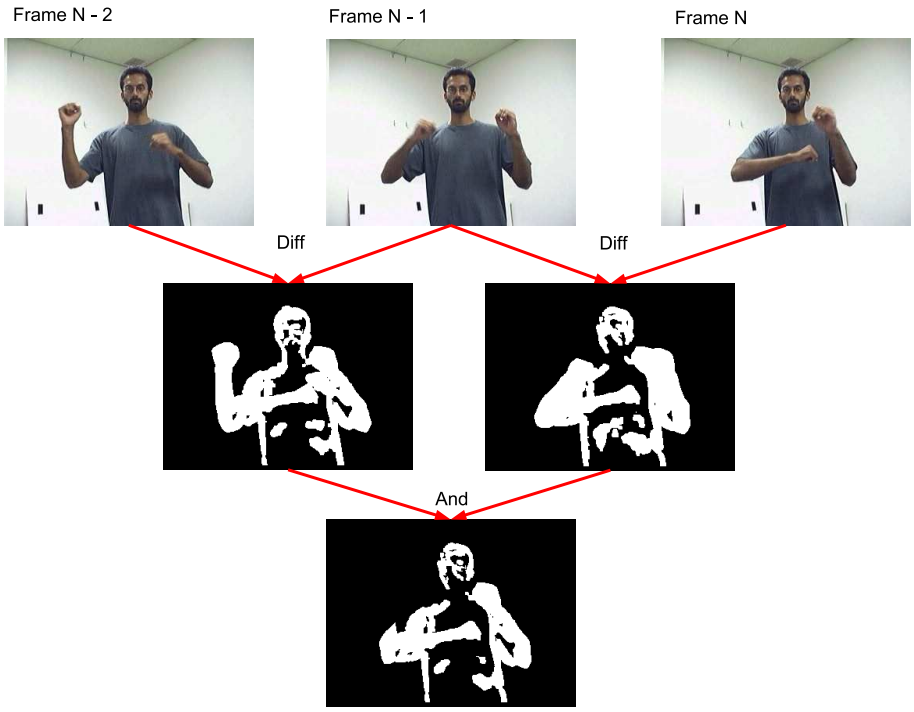


Figure 4.5: Computation of a mask that coarsely identifies regions of the image that have changed

4.7.2 Motion Discontinuities

To contend with fast motion, we first estimate moving foreground pixels by frame differencing. In particular we compute:

$$\begin{aligned} F_n(i,j) &= D(I_n, I_{n-1}, \sigma_{TH})(i,j) \bigcap \\ & \quad D(I_n, I_{n-L}, > \sigma_{TH})(i,j) \end{aligned} \tag{4.10}$$

Here $D(I_i, I_j, \sigma_{TH})$ computes a difference mask between frames I_n and I_{n-1} and then between I_n and I_{n-L}. The resulting differences mask are then fused with a Boolean *and* operation. The result of this procedure is a mask that identifies those pixels in frame n

that are different from two previous frames $n-1$ and $n-L$. As shown in Figure 4.5 this coarsely identifies regions of the image that have changed. The parameter L is the frame lag used in choosing the second frame for differencing. Typically $L = 1$.

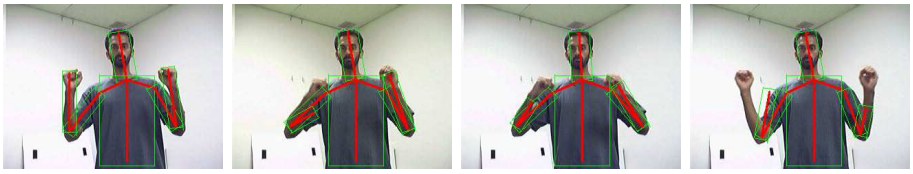
This mask can be used when generating candidates in equation 4.5. We assign to each candidate a number P_{limb} based on the fixed with rectangle associated with the joint position x_l^i and the root location of its child configuration, $\mathbf{X}_l^{c^k(i)}$. In particular P_{limb} is the percent occupancy of this rectangle with foreground pixels identified from F_n.

Instead of blindly evaluating each candidate sent to $wprune()$ with Ψ, we instead only consider those candidates that are either in the windows, W, about their previous location (for smooth motion) or have $P_{limb} > thresh$ (for discontinuous motion). Computation of P_{limb} is still $O(R|I|M^2N)$, however the computation can be computed with integral images [88] and is extremely efficient. It also significantly reduces the number of candidates generated and the number of calls to Ψ.

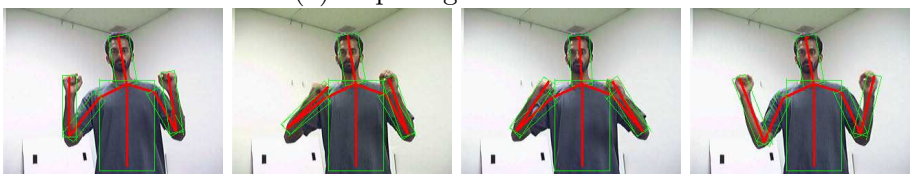
4.7.3 Partial Update

We also reduce the run time by first updating only the head and torso, while fixing the arms and then updating the arms and fixing the head and torso. This is a reasonable updating scheme as the head and torso are likely to move smoothly, while the arms may be moving more abruptly.

This is done by updating the joint locations, *topHead*, *lowerNeck*, and *pelvis* in equation 4.5 while ignoring the sets, $\{{}^k\mathbf{X}_l^{shoulderL}\}$ and $\{{}^k\mathbf{X}_l^{shoulderR}\}$. These sets are indexed



(a) Imposing smoothness.



Frame 1 Frame 44 Frame 53 Frame 70

(b) Preserving fast motion.

Figure 4.6: The top row shows the Rank 5 results with respect to Ψ, when only continuous motion is assumed using the method in section 4.7.1. The second row shows the Rank 5 solution when discontinuous motion is allowed using the method in section 4.7.2

when $\{{}^k\mathbf{X}_j^{topHead}\}$ is constructed. When updating the head and torso we assume continuity and only consider the region defined in section 4.7.1.

Once the joints *topHead*, *lowerNeck*, corresponding to *pelvis* have been computed, we can lock the topHead and lower neck positions and recompute $\{{}^k\mathbf{X}_j^{lowerNeck}\}$. Updating in this way reduces the number of candidates generated significantly and allows the *topHead* to move about the image as needed.

4.8 Results

Examples of output of methods describe in section 4.7 are shown in Figure 4.6. Here the same model shown in Figure 4.3 and the sequence in section 4.6 are used. This sequence was acquired at 30 frames per second and then down sampled to 6 frames per second. In the first row, continuous motion is assumed and the modification described in section 4.7.1 and in section 4.7.3 are used. Window sizes of 60x60 are used. In the

first frame a full search with the topHead joint positioned on the head is completed. The processing time devoted to finding joint configurations is 781ms. In subsequent frames this time is reduced to 70ms.

In the second row we also use the method described in section 4.7.2. Here we reduce the window size to $W = 30 \times 30$ and use candidate configurations when $P_{limb} > 1/2$. In these frames it takes on average 84ms to compute the foreground masks, F_n (shown in the 3rd row), and the time associated with configuration construction increases to 114ms.

From these sequences, we see that assuming continuous motion allows for significant improvements in speed. If we enforce smoothness only it is easy to drift significantly as shown in Figure 4.6(a). Adding the information from the motion mask corrects this situation as shown in Figure 4.6(b) with reasonable gains in speed.

HumanEva Data Set

We also evaluated this algorithm performance on a sequence from the HumaEva [78] data set. In particular we used the sequence S2/Gesture_1_(C1) sequence frames 615 to 775 in increments of 5. The model use here is essential the same as that shown in Figure 4.3. The main difference is that $R^{root}_{topHead}$, $R^{topHead}_{lowerNeck}$, $R^{lowerNeck}_{pelvis}$, are enlarged and elongated to better accommodate changes in scale.

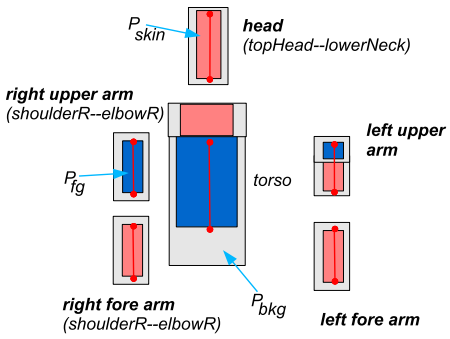


Figure 4.7: The limb detectors used on the HumanEva data set

The limb detectors used in this sequence consist of several non overlapping areas representing foreground, background, and skin colored regions. The likelihood of each patch is the product of underlying color pixel's probability of member ship in each region.

$$\begin{aligned} & P_{part_{ij}}(x^i, x^j, w^{ij}) \\ & = \prod_{p \in fg} P_{fg}(p) \prod_{p \in bkg} P_{bkg}(p) \prod_{p \in skin} P_{skin}(p) \end{aligned} \tag{4.11}$$

where P_{bkg}, is the obtained from the background model provide with the HumanEva. The termsP_{fg}, the foreground likelihood and and P_{skin}, the skin likelihood are modeled as histograms extracted from the sequence itself. The shape of each part detector is shown in Figure 4.7.

For the range of images we worked with, we established the ground truth by annotating the joints of the user. This is because in several of these frames the projected ground truth joints were off. Also, we are looking for the hand tip, not the wrist, which is what is marked in this data set.

The average error with respect to corrected projected joints is shown in Figure 4.8 and example poses are shown Figure 4.9. In this sequence we identified a point near the

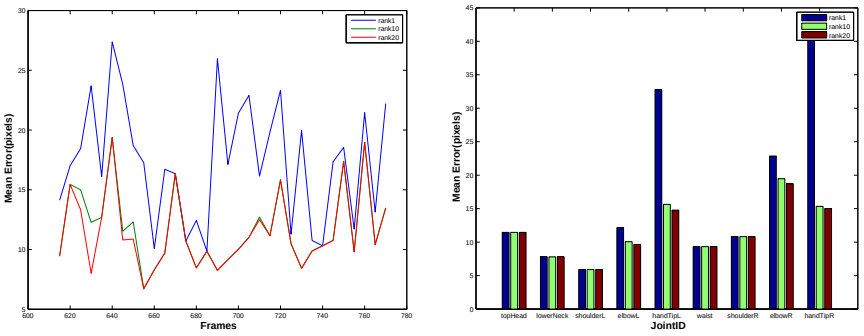
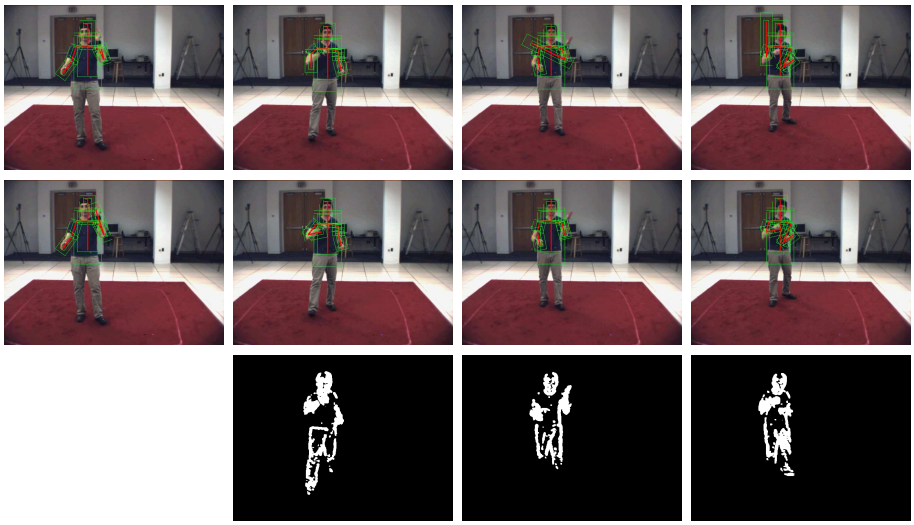


Figure 4.8: Average joint error at each frame in the sequence (a) and for each joint over the sequence (b)

top of the head in the first frame and use the method of 4 to align the pose. Following this, the pose is tracked using the methods described in 4.7.

Through the sequence we maintain 20 candidates. In the first frame we detect 10 using the method of 4 and then another 10 constraining the hand to be away from a the detected face using $\mathbf{M}^{handR}$ and $\mathbf{M}^{handL}$. Time devoted to assembling cadidates during the initial detection is 1.531s (i.e. not including the image pre-proccessing and the like) while the associated only with constructing candidates while tracking is on average 188ms.

In Figures 4.8 and 4.9 the ranked results are shown. Here we see the rank1 solutions, which minimize Ψ are not correct and performance is poor. However the rank 10 (and rank 20) coincide with pose that appear more correct. The joints for which this has the greatest effect are the hand tips. Though we are focusing on the upper body, the performance on this sequence is comparable to that of [37] on the S3/Walking_1_(C2) sequence.



(a) Frame 0 (b)Frame 10 (c) Frame 20 (d) Frame 31

Figure 4.9: The top row shows the Rank 1 results with respect to Ψ, when only continuous motion is assumed using the method in section 4.7.1. The second row shows the Rank 10 solution when discontinous motion is allowed using the method in section 4.7.2. The third row shows the moving forground pixel as computed using three consecutive frames (not shown).

4.9 Discussion

In this chapter, we developed a method to find candidate 2D articulated model configurations by searching for local optimum under a tractable fitness function. This is accomplished by first parameterizing this structure by its joints organized in a tree structure. Candidate configurations can then efficiently and exhaustively be assembled in a bottom-up manner.

In this work, we focused on the estimation of the upper body. A complete system would include a full body representation and this work can be extended to include the lower body with additional computational costs.

Our results suggest that while the configurations that globally optimize the fitness function may not correspond to the correct pose, a local optima will. After finding these

local optima, one can then make a selection or use these candidates to initialize higher level processing. This problem, however, is much smaller as one of the candidates is "near" the true solution. For this purpose, we can make use of a top-down functions such as the one described in chapter 6 or chapter 7 as well as spatial continuity.

Integral to the success of finding 2D poses are the design of meaningful limb detectors. In this chapter, we focused on hand tuned appearance based models. In Chapter 5 we discuss a method that learns these detectors using labeled training data.

Chapter 5

2D Pose Feature Selection

The work described in Chapter 4 shows how to extract a series of local optima in an objective function constructed for a given set of part detectors an on articulated structure. There we designed part detectors based on appearance and skin information.

For a system to work in a more general setting, the set of detectors should be invariant to changes in appearance due to lighting or different clothing types. To accommodate this kind of robustness, we propose in this chapter a method to construct these detectors from annotated training data.

In this Chapter, we learn an objective function from labeled training data using a classification framework. Positive samples are pose-image pairs that are close to the correct answer, while negative samples are pose-image pairs that are far from it. Real-valued AdaBoost, which has been used extensively in object detection and exhibits good generalization in practice, can then be used to construct a strong classifier as an objective function.

While the constructed saliency metric can be used in any pose estimation framework such as [21][44][81][24], our search strategy is well suited for this task. In particular,

the multiple candidates returned can be used as part of a bootstrapping algorithm in the feature selection process. Returned configurations that are wrong represent problematic poses for the current saliency metric and are used as negative samples in further refinement.

One issue in using AdaBoost is the number of training samples required. This is because the high dimensionality of 2D poses requires many samples before the constructed classifier generalizes. To reduce the number of training samples needed, we consider both part-based and branch-based training strategies.

The rest of this chapter is organized as follows: In section 5.2 the form of our objective function is given. In section 5.3, we present the features used in learning this objective function. In sections 5.4 and 5.5 we describe how they are combined using real-valued AdaBoost. In section 5.6 we present quantitative results and we conclude in section 5.7.

5.1 Related Work

Deriving observation likelihoods from data for pose estimation has been explored in works such as [68][65], while more explicit design of robust objective functions has been explored in [91] and [95]. The design of our part detectors is similar to the use of parameter sensitive boosting in [95].

Higher level features and detectors can also be learned directly from training data. In [74] responses from boundary detectors are empirically derived from image data. Individual limb detector learned from training data have been used [68][51][76][49]. Learning observables in this manner provides a set of responses that are more reliable than low

level features such as edge or flow but also more generic then appearance based detectors. In [48] boosting is used to select features that from an saliency measure that separate valid poses from non-valid poses. In this work, however, we explicitly encode the relationship between the feature positions and orientations and the configuration of joints. Also, because the search is exhaustive, we do not need the recovered objective function to be smooth, as was considered in [91].

5.2 Formulation

As detailed in Chapter 4, we model image saliency as a sum of terms dependent on parent-child joint pairs:

$$\Psi_{image}(\mathbf{X}, \mathbf{I}) = \sum_k h^k(x^i, x^j, \mathbf{I}, \phi^k) \tag{5.1}$$

Here $\mathbf{X}$ denotes a tree of joint locations, $\mathbf{I}$ represent an image, h^k corresponds to a term that depends on the parent-child pair of joints, x^i and x^j, and a fixed set of parameters, ϕ. These terms only depend on the positions of pairs of joints. We note that this effectively assumes the underlying limb widths are fixed. This is a reasonable assumption as this is the case in the projection of a cylinder (i.e. limb) under an orthographic camera.

The ability to estimate pose from an image is largely dependent on the quality of the objective function. While it is possible to construct such functions manually, we construct (5.1) using an AdaBoost framework.

Treating image-joint configure pairs, $(\mathbf{X}, \mathbf{I})$, as single objects to be classified, we define positive samples as those for which the distance to the actual configuration of joints in

an image, $\hat{\mathbf{X}}$, is below a threshold (i.e. $dist(\mathbf{X}, \hat{\mathbf{X}}) < \sigma$). A confidence rated classifier defined on this domain would yield large positive values for samples where $\mathbf{X}$ is close to $\hat{\mathbf{X}}$, and large negative values when $\mathbf{X}$ is far from $\hat{\mathbf{X}}$.

The objective function thus be formulated as a confidence rated, and expressed as a sum of weak hypotheses.

$$\Psi_{image}(\mathbf{X}, \mathbf{I}) = H(\mathbf{X}, \mathbf{I}) = \sum_k h^k(\mathbf{X}, \mathbf{I}, \phi^k) \tag{5.2}$$

where, h^k, corresponds to a term that depends on a the configuration of joints, the image, and a fixed set of parameters, ϕ.

In principle a weak hypothesis, h^k, can depend on the entire set of joints, however, to make use of algorithms that can optimize (5.1) we further constrain it to only depend on the positions of parent child joint pairs:

$$h^k(\mathbf{X}, \mathbf{I}, \phi^k) = h^k(x^{i_k}, x^{j_k}, \mathbf{I}, \phi^k) \tag{5.3}$$

Each part detector in (5.1) can be constructed by combining weak hypotheses that correspond to the same limb. This allows us to efficiently and exhaustively find candidate joint configurations (as positive samples).

To construct the detector in (5.2), we make use of a set of features, $f^k(x^{i_k}, x^{j_k}, \mathbf{I}, \phi)$, and a set of labeled training data. Positive and negative samples can be constructed from the training data and individual terms in equation (5.2) can then be learned using the AdaBoost algorithm described in section 5.4 with domain partition weak learners[71].

These features depend on various sources of information including canny edges, sobel edges, foreground estimation and skin color saliency. While it is possible to use background subtraction, in this work we estimate foreground pixels by thresholding a stereo disparity map. Skin saliency is estimated using a hue-saturation histogram derived from face pixels found using a face-detector as described in Chapter 3.

5.3 Model Based Features

The features we use are parameterized by the configuration of joints, $\mathbf{X}$. Image measurements are made by first transforming these *model based features* into the image and then making image measurements.

While an arbitrary parametrization with respect to pairs of joints is possible, we further specify the form of our features. In particular, key points and angles of the features defined below are embedded in an affine coordinate system between joint pairs. This coordinate system scales linearly with the distance between joint pairs as illustrated in Figure 5.3(d).

In particular a model point, $\mathbf{p}_m = (x, y)$, is affixed between a pair of joints, x^i, x^j. This point is transformed into a position in an image using:

$$\mathbf{p}_{im} = T(\mathbf{p}_m, x^i, x^j) = R(\angle(x^i, x^j))D([|x^i - x^j|, 1])\mathbf{p}_m + (x^i + x^j)/2 \qquad (5.4)$$

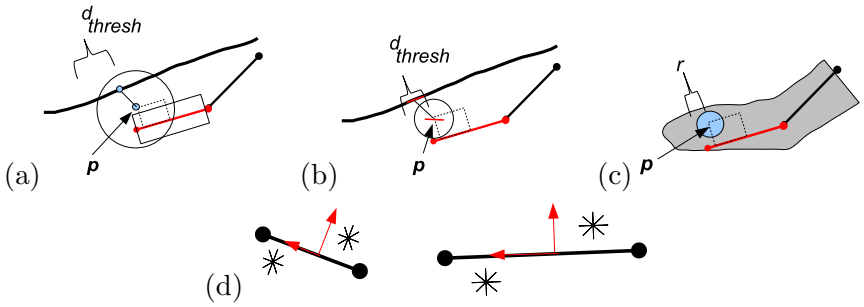


Figure 5.1: In (a)-(c) Model based Features. In (d) Feature positions are defined in an affine coordinate system between pairs of joints.

where $R(\theta)$ is a rotation matrix, and $D([a,b])$ is a diagonal matrix. Similarly model angles, θ_m are transformed into the image using:

$$\theta_{image} = T(\theta_m, x^i, x^j) = \theta_m + \angle(x^i, x^j) \tag{5.5}$$

5.3.1 Distance to Nearest Edge

As shown in Figure 5.3(a), this feature computes the distances to the closest Canny edge within a threshold. This distance can be computed efficiently using the distance transform of the canny edge image [26]. This feature is thus parameterized by its position between a pair of joints and the maximum allowable distance, d_{thresh}, to the closest canny edge. In particular:

$$\begin{aligned} f_{dist}(x^i, x^j, \mathbf{I}, \phi_{dist}) &= \min(D_{dist}(T(\mathbf{p}, x^i, x^j), d_{thresh}) \\ \phi_{dist} &= \{\mathbf{p}, d_{thresh}\} \end{aligned} \tag{5.6}$$

5.3.2 Steered Edge Response

This feature computes the steered edge response of a model edge against the Sobel response under the closest Canny edge. This is illustrated in Figure 5.3(b). If the closest edge is further then d_{thresh} a constant value is returned. This provides local orientation information that can be used to discriminate against clutter and better limb alignment. In particular

$$f_{edge}(x^i, x^j, \mathbf{I}, \phi_{edge}) = \begin{cases} s_x cos(\theta_{image}) + s_y sin(\theta_{image}), if D_{dist}(\mathbf{p}_{image}) < d_{thresh} \\ 0 \end{cases} \tag{5.7}$$

where,

$$\begin{aligned} s_x &= S_x(P_{dist}(T(\mathbf{p}, x^i, x^j)) \\ s_y &= S_y(P_{dist}(T(\mathbf{p}, x^i, x^j)) \\ \theta_{image} = \hat{T}(\theta), \mathbf{p}_{image} &= T(\mathbf{p}, x^i, x^j), \phi_{edge} = \{\mathbf{p}, \theta, d_{thresh}\} \end{aligned}$$

Here $\mathbf{S}_x$ and $\mathbf{S}_y$ are the Sobel responses in the x and y directions respectively. P_{dist} is computed along with the distance transform and holds the coordinate of the closest canny edge at each point in the image. This feature is thus defined by both a position, $\mathbf{p}$, and orientation, θ, as well as a distance threshold, d_{thresh}.

5.3.3 Foreground/Skin Features

Foreground information is a relatively strong feature when it can be computed. This feature computes the occupancy of foreground pixels within a circle of fixed radius. This is illustrated in Figure 5.3(c).

$$\begin{aligned} f_{fgdot}(x^i, x^j, \mathbf{I}, \phi_{fgdot}) &= \sum_{p \in |p - T(\mathbf{p}, x^i, x^j)| < r} Fg(p) \\ \phi_{fgdot} &= \{\mathbf{p}, r\} \end{aligned} \tag{5.8}$$

This feature is specified by its position and the radius of the circle. The summation can be computed efficiently using an integral image[88].

In a similar manner, a skin feature, f_{skin}, can be defined using the skin saliency map instead of a foreground mask.

We can also measure contrast information by considering the average difference between the occupancy of a pair of foreground dots within the same pair of joints. In particular:

$$\begin{aligned} f_{fgpair}(x^i, x^j, \mathbf{I}, \phi_{fgpair}) &= |f_{fgdot}(x^i, x^j, \mathbf{I}, \phi^1_{fgdot}) - \\ &\quad f_{fgdot}(x^i, x^j, \mathbf{I}, \phi^2_{fgdot})| \\ \phi_{fgpair} &= \{\phi^1_{fgdot}, \phi^2_{fgdot}\} \end{aligned} \tag{5.9}$$

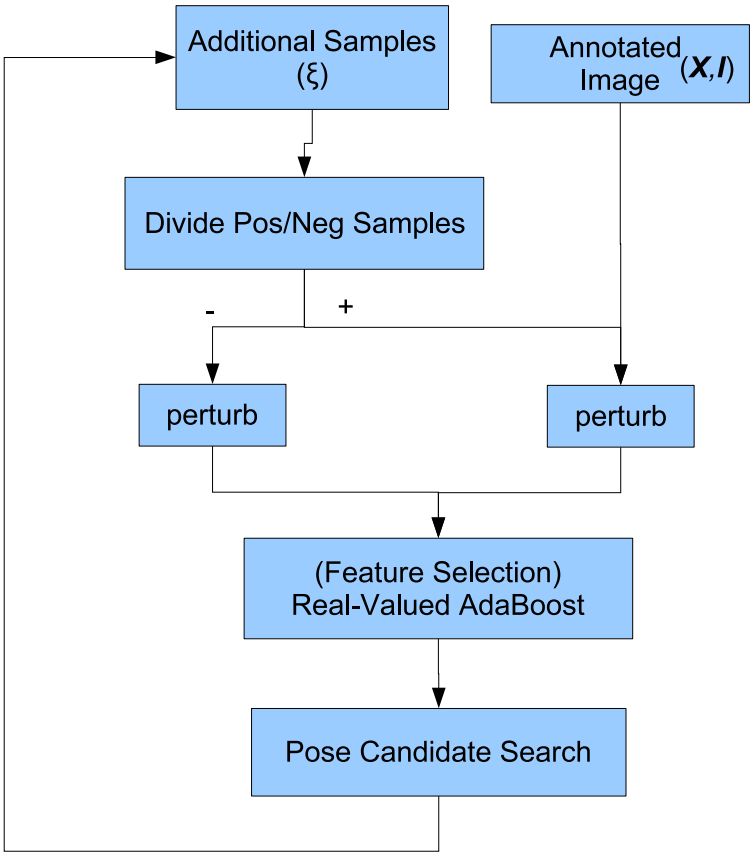


Figure 5.2: Feature Selection Overview

5.4 Feature Selection

The overall feature selection method is shown in Figure 5.2. In this approach, we construct a set of positive and negative training samples consisting of configurations of joints and an input image from annotated data and the results of the search described in section 4. This set can then can be used to construct our objective function using real-valued AdaBoost.

5.4.1 Training Samples Construction

The set of training samples is constructed by first combining the ground truth (annotated) images, and the set ξ. The set ξ contains samples generated using the previous estimate of the objective function and the the algorithm of section 4. It is initialized by searching for candidates using a joint tree with no detectors. In our experiments, we find 20 candidates that are at least 30 pixels apart for each training image.

The set of samples, ξ, is then divided into positive and negative samples based on how close the associated configuration is to the ground truth. If all joints in a configuration are within 10 pixels of their corresponding ground truth locations, the sample is considered positive, otherwise it is negative.

From these sets of positive and negative samples, additional samples are generated by perturbing each configuration with a set of random displacements:

$$(\mathbf{X}', \mathbf{I}) = (\mathbf{X} + \delta\mathbf{X}, \mathbf{I}) \tag{5.10}$$

$$\forall i \; x^{i\prime} = x^i + \delta x^i \tag{5.11}$$

where δx^i is displacement uniformly drawn from the range $(-10, -10) \times (10, 10)$.

These perturbations are necessary to compensate for the discretization of the relative positions between joint pairs in R^i_j used during the search in section 4. In practice, there are many more negative samples then positive samples. We thus generate additional positive samples by jittering the current set of positive samples with the above until the sizes of the two sets are comparable.

This set of training data can then be used in real-valued AdaBoost to assemble our objective function (i.e. the strong detector) from the features defined in section 5.3.

From the trained detector, additional samples can be found by searching for configurations on the training images and appending them to the set ξ. This bootstrapping process adds to the training data configurations that are likely to create local optima that do not correspond to correct poses. New samples are accumulated into ξ between iterations. In practice, this algorithm only needs a few iterations before meaningful detectors are constructed.

5.5 Real-Valued AdaBoost

In principle a direct application of real-valued AdaBoost could be used on the full set of joints to construct the strong detector in equation 5.1. This, however, is not practical, because the number of samples needed to adequately represent the distribution of $(\mathbf{X}, \mathbf{I})$ is very large. This makes it difficult to learn a detector that generalizes well, even though the training error is low. We thus consider methods for feature selection based on training each pair of joints (i.e. part) separately and each branch of the tree of joints separately. The resulting detectors found for these partial configurations can then be assembled into a single configuration.

5.5.1 Part Based Training

In this method of feature selection we train each part separately. This is similar to several methods that have been proposed to learn part detectors such as[49]. The difference, however, is the manner in which negative training samples are generated in the bootstrapping process described in section 5.4.1.

To train a given part we constrain the feature pool to only contain features between the corresponding parent-child joint pair. We also prune irrelevant branches from the tree. For example, when we are training the lower right arm we remove the branches containing the left shoulder joint and its children, and the head and torso. Similarly if we are training the upper-right arm we remove the left arm, head, torso, and the right-hand-tip joints, as these joints do not affect the upper right arm.

After each part is trained we can combine the recovered detectors into a single tree.

5.5.2 Branch Based Training

In this approach, rather than train each part separately, we consider each arm and the branch consisting of the head,neck, and pelvis joints separately. Here the feature pool is constructed so that only features on the branch we are training are considered. We also prune irrelevant branches from the tree. For example, when we are training the left arm we prune the branches containing the right shoulder, and the head and torso. After each branch is trained, we can combine the recovered detectors by assembling them all into a single joint tree.

Training this way has the advantage over part-based feature selection in that features are not forced to be uniformly distributed across parts. Furthermore the negative

samples found during the bootstrapping procedure represent false positives that arise by considering combinations of joints.

5.6 Experiments

We evaluated our method using a training set consisting of 30 annotated images of three different subjects and a testing set consisting of 47 annotated image from another set of videos of the same three subjects. In one case the clothing of the user changed, as he wears a long sleeve shirt. Skin colored pixels in each image were identified individually, using a color histogram derived from pixels in a detected face. These images are part of a stereo sequence. We make use of only one image in the pair (i.e. the right image) in our tests, however, foreground masks were estimated by thresholding disparity.

5.6.1 Saliency Metric Training

From the features defined in section 5.3, a pool of features is constructed by assigning instances of each feature type to 200 random positions on joints pairs corresponding to the upper arms, lower arms, head, and torso. For the edge correlation features we construct edges oriented at 0,45, 90, and 135 degrees at each location. For the dot based features we use radii of 2, 4, and 6 pixels at each selected location. For distance thresholds we consider values from 20 to 80.

We used two iterations in the training algorithm shown in Figure 5.2. The features selected are illustrated in Figure 5.3. During AdaBoost feature selection, we limited each part and branch to select at most 40 features.

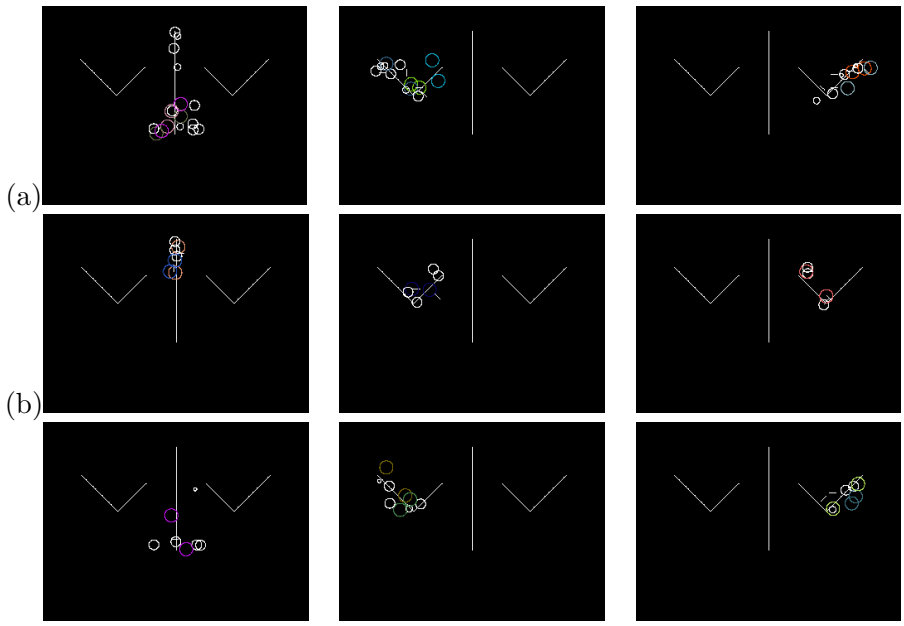


Figure 5.3: Feature selection. In (a) branch based selection , In (b) part based.

To compare the constructed saliency metrics we consider the case where the total number of features in a combined joint tree is 42. In Figure 5.3(a) the first 14 features are shown on each branch. In Figure 5.3(b)the first 7 are shown on each part. From this figure we see that with arm-based learning more features are placed on the forearms and about the waist. This is natural as there is higher contrast in these regions. In part-based learning, more features were placed in those locations for the individual limbs. However, features were also placed on upper arms and the head locations where branch based learning deemed to be of less importance.

5.6.2 Single Frame Detection

The recovered joint trees were tested on 47 annotated images from another set of videos of the same three subjects. In one case the clothing of the user changed, as he wears a long sleeve shirt.

In these single image detection tests, we reduced the size of the search space by making use of a face detector. We anchor the root joint to the top center point of the detected face. This does not need to be precise, as there is some slack allowed between the root joint and the topHead joint. The location of the face is only needed in the first frame in our tracking results of the following section.

In the quantitative results that follow, we show the average joint error for the $RankN$ solution. Since our algorithm produces an order set of $M = 20$ configurations, the $RankN$ $< M$ solution is the configuration among the first N of M with the smallest average joint error with respect to the ground truth.

Examples of single frame detections are shown in Figure 5.4(a) for the branch-based joint tree and Figure 5.4(b) for the part-based joint tree. In both cases we show both the optimal result with respect to the learnt objective function (i.e the $Rank1$ solution) and the $Rank20$ solution. From this figure we see the $Rank1$ result does not correspond to the best solution for the user on the left, while the $Rank20$ solution does. In this case, this user wears clothing not found in the training set (i.e. the long sleaves). This shows that while the global optimum did not correspond to the correct solution a local optimum did.

Aggregate statistic are shown in Figure 5.5(a) and (b) for both the part-based and the branch-based detectors. In Figure 5.5(a) we traced the $Rank5$ and $Rank15$ error rates as we increased the number of features on the overall joint tree. This was accomplished by adding one feature per limb or branch and using the resulting joint tree over the test set. From this plot we see that increasing the number of features decreases the test error rates. With few features, the part-based joint tree generalizes better, but as the number of features increases the branch-based detector out performs. In fact as the number of features increases the joint tree constructed from part based learning starts to get worse, suggesting that parts begin to overfit the training set. This is not the case with the branch-based learning. This effect is less pronounced in the Rank 15 statistics. In Figure 5.5(b) we see the statistics for the individual joints as computed with 42 features total. From this we see that localization of the hand tips was most difficult.

5.6.3 Pose Tracking

In this test, we used the detector constructed from branch-based learning to track a subject in a sequence of images limited to 14 features per branch and the optimizations from section 4.7. Here, we initialized the root joint with the result of a face detector only in the first frame. In subsequent frames the candidate poses were adapted as described in section 4.7. Example results for the Rank 15 solutions are shown in Figure 5.4(c). In Figure 5.5(c) aggregate statistics over this 70 frame sequence are shown. Also shown in this graph are the average joint errors using per frame detection with a face detector. While performance is similar, tracking had larger errors for the topHead joint since this

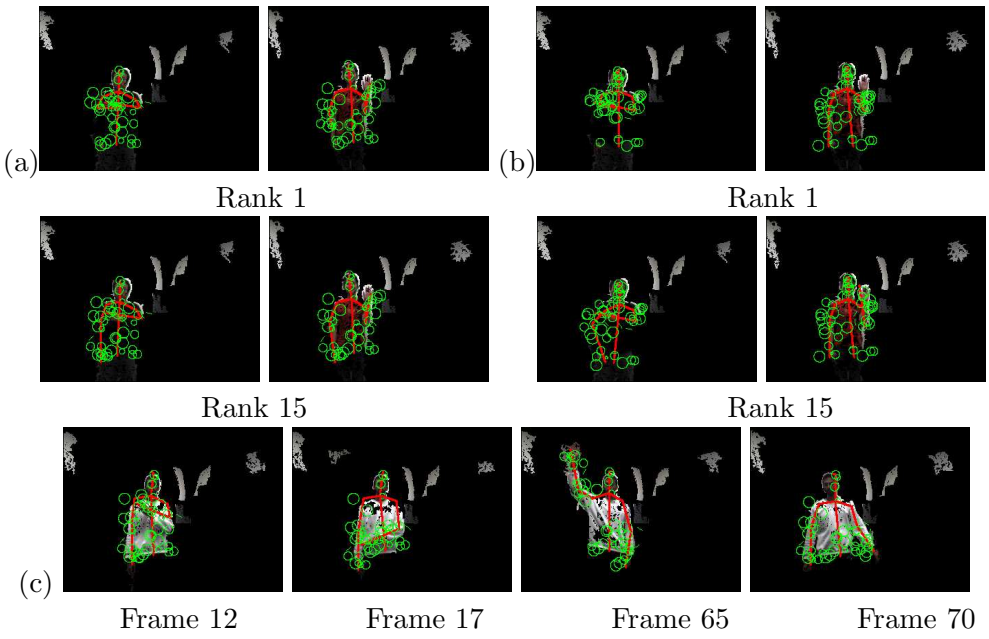


Figure 5.4: (a) Branch based detector (b) Part based. (c) Rank 15 results on a sequence.

joint is no longer anchored by a face-detector. Also we see that using tracking smoothed out errors in the right hand tip for Rank 5 solutions.

There is a larger difference between tracking and detection when it come to processing time. For tracking, the processing time per frame devoted to computing candidates ranges from 532ms to 3s depending on how many pixel change between frames as indicated by the difference image. The average processing time per frame in single image detection was about 6.5s.

5.6.4 Distribution Analysis

We also evaluate our objective function by estimating the average log likelihood of the ground truth pose configuration on the test images (i.e $\frac{1}{T}\sum_t \log P(\hat{\mathbf{X}}|\mathbf{I}_t)$ as proposed

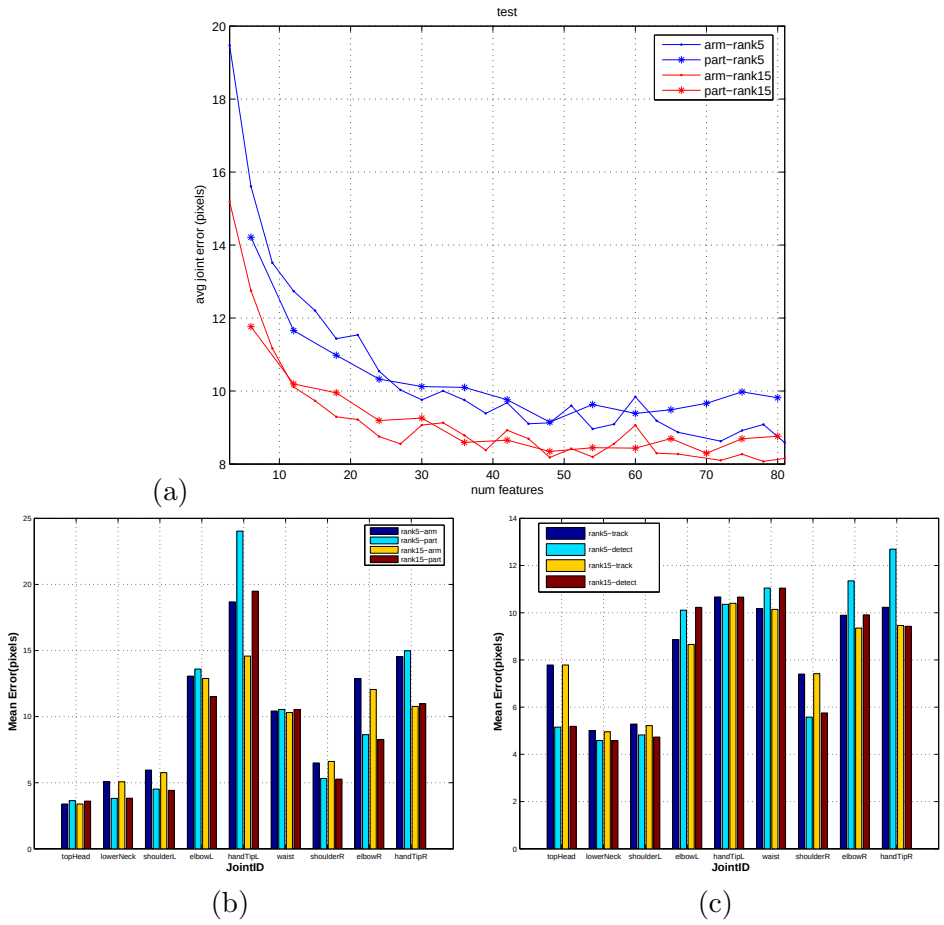


Figure 5.5: Statistics for pose estimation in single frames (a,b) and a sequence (c)

in[62]. We can turn our objective function into a true probability using a sigmoid function $\frac{e^{H(x)}}{1+e^{H(x)}}$ [29]. We then estimate the log probabilities of the ground truth using an MCMC sampler and Parzen window estimation with Gaussian kernels. The average log probabilities of the test set for each iteration is shown in Figure 5.6.

We also can visualize the quality of these distributions by computing the marginal probability of each joint from the samples generated from the MCMC sampler. We modulate the marginal distributions with different colors for each joint and superimpose

P_{td}	Iter 1	Iter 2	Iter 3	Iter 4
91.02	78.78	72.93	76.28	72.94

Figure 5.6: Log-probabilities of images given model

on each other in a manner similar to that of [62]. Examples of the resulting distribution are shown in Figure 5.7(b).

For comparison, we consider the manually designed, top-down, 2D image likelihood, used in [46]:

$$P_{td} \propto \exp(-d^2_{champfer}/2\sigma^2_{champfer} + -d^2_{fg}/2\sigma^2_{fg}) \tag{5.12}$$

Here, $d_{champfer}$ is the average distance of a model based edge to the closest Canny edge and d_{fg} counts the number of mistakes in matching a model predicted foreground mask to a measured one. Using an MCMC sampler we compute the the marginal joint distributions shown in Figure 5.7(a). Clearly, the joint distributions computed using our learned objective function in Figure 5.7(b) form better localized joint positions. This is quantified in Figure 5.6, in which we compute average log probability over our test set.

These results show that our learned part-based objective function is better able to localize joint position, without explicitly modeling limb interaction. Our distribution is crisper and therefor better for localization. Furthermore, since this metric is part based, the optimum can be found efficiently. The likelihood proposed in equation 5.12 is optimized by heuristically generating candidates and evaluating them.

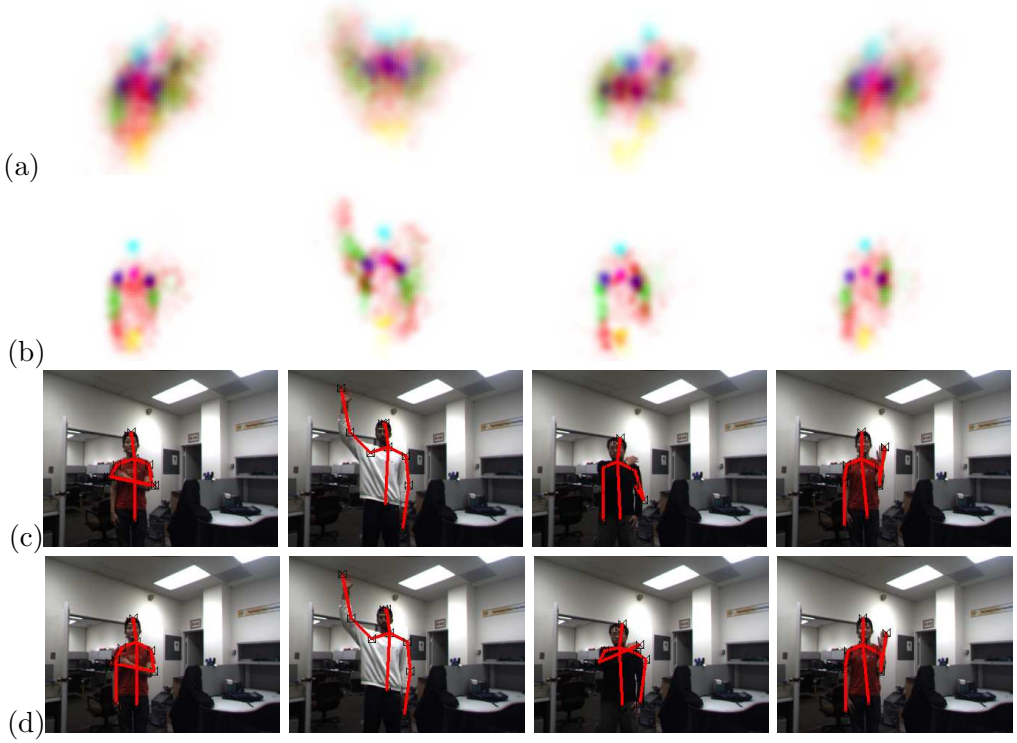


Figure 5.7: In (a) the joint distributions are shown as derived from equation (5.12), while in (b) distribution derived from our learned objective function. In these plots, red, green and blue, corresponds to hand tip, elbow, and shoulder joints respectively. Cyan, magenta, and yellow, correspond to the top head, lower neck, and waist joints respectively. The optimal solutions (i.e. Rank 1) according to our learned objective function is shown in (c) and the Rank 40 solution is shown in (d).

5.7 Discussion

In this chapter, we developed a method to automatically construct an objective function from annotated image data. We proposed a set of generic, model based, features and used real-valued AdaBoost with Domain Partitioned weak learners to learn a strong detector. Our ability to efficiently recover local optima was used to generate negative samples for the training procedure. Our results suggest that the recovered objective function generalizes over multiple people with different clothing and appearances.

Our results suggest that exhaustively searching solvable but less discriminating objective functions for local optimum can in fact generate good candidate pose configurations. These candidate poses can be used to initialize a searches based on more more computationally expensive evaluation criteria. This includes top down likelihoods such as those described in Chapter 7.

Chapter 6

Stereo 3D Pose Tracking

In the previous chapters, we described methods for pose estimation in single camera systems. To a large degree, the effectiveness of these systems is contingent on the reliability and discriminative power of the image observations. Stereo and depth based sensors offer additional measurements in the from of depth information that can be used in the estimation of human poses.

In this chapter, we describe a system to track the arms of a user using stereo imagery and an optimization framework. The input to this system is provided by the Bumble Bee Stereo Camera from PointGrey. This camera, shown in Figure 6.1 produces stereo depth images of size 640×480 at 48FPS. We track the movement of a user by parameterizing

Figure 6.1: BumbleBee®stereo camera from PointGrey®

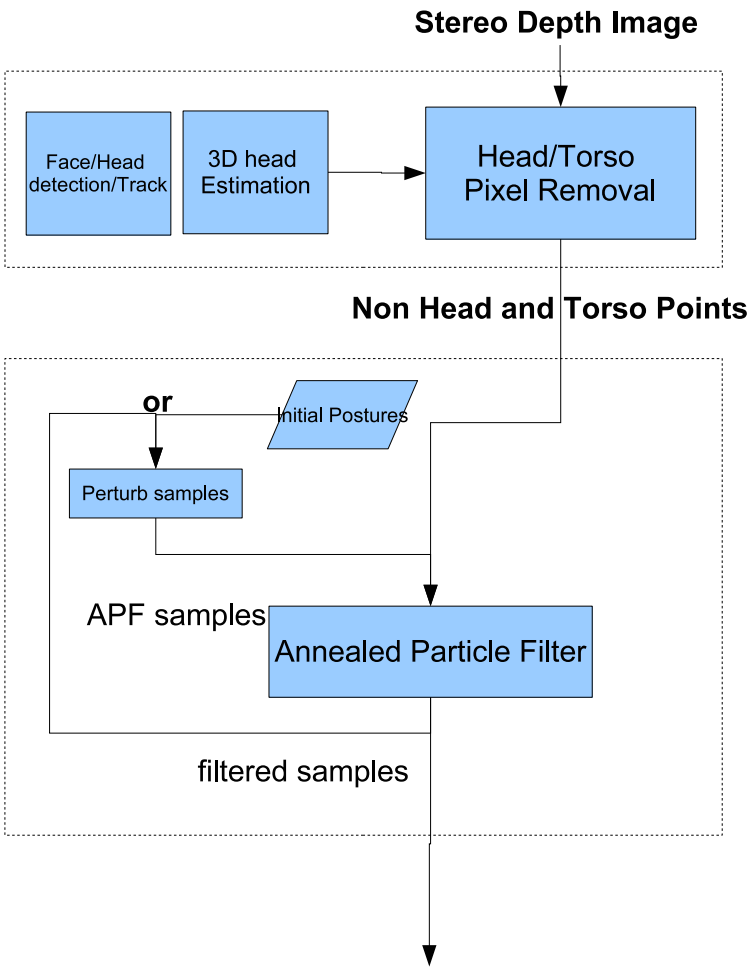


Figure 6.2: Overview of the Stereo Arm Tracking System

an articulated upper body model using limb lengths and joint angles. We then define an objective function that evaluates the saliency of this upper body model with a stereo depth image. We track the arms of a user by numerically maintaining the optimal upper body configuration using an annealed particle filter [20].

In the rest of this chapter we further elaborate on this approach. In section 6.2 we introduce the annealed particle filter. We then describe our system in detail in section 6.3. In section 6.4 we show quantitative results and conclude in section 6.5.

6.1 Related Work

The use of depth information from a stereo sensor has been explored in a number of works including [19][32][93][33]. In works such as in [[[33]]] and [70] primarily single view methods are used in conjunction with uncalibrated 3D data recovery methods to infer 3D poses.

While these works make use of image features, depth measurements can be used directly. In [32] iterative closest point (ICP) is used to track a human model using direct optimization methods. In [19], an efficient ICP algorithm that first aligns individual ridged parts, is used to track a pose initialized using a hashing method. In[7] the rigid parts of a body are aligned with depth points using belief propagation. In [93], information in depth images and silhouettes is used in a learning framework to infer poses from a database.

The main tool used in this system is the annealed particle filter (APF) [20]. This tool allows one to numerically compute the optimum of an arbitrary objective function using a stochastic search. In particular, the APF is initialized with a collection of samples that represent the domain of the objective function. These samples are then perturbed and stochastically sampled according to a smoothed version of the original objective function. Particles are then iteratively resampled according to increasingly sharper versions of this objective function until they converge on the global optimum.

As the main task is to track the arms of a user interfacing with a machine, we assume the user is facing the camera and standing upright. This allows us to more easily segment the head and torso from the body and focus the computational effort on arm localization.

Function APF ($\{\mathbf{X}^k\}$,Ψ)
/* performs the APF on configurations in set X,
  using objective function Ψ
*/
initialize β
do
    perturb samples in $\{\mathbf{X}^k\}$
    weight the samples in $\{\mathbf{X}^k\}$ with the objective funtion,Ψ^β
    resample $\{\mathbf{X^k}\}$ with new weights
    sharpen objective function by increasing β
while samples not converged to single optimum

Figure 6.3: The Annealed Particle Filter

The use of a numerical optimizer allows one to more flexibility model and design search criteria. Besides smoothness, few limitations are imposed on the form of the objective function. Thus, a large degree of flexibility is afforded. This includes top-down model alignment metrics. This added flexibility comes at a computational cost, as arbitrary functions can be difficult to optimize efficiently. In particular, many particles may be necessary if the objective function is not sufficiently convex. In the case of stereo input, however, an objective function is designed that provides a good tradeoff between accuracy and efficiency.

6.2 Annealed Particle Filter

The annealed particle filter [20] is a method to find a optimum of a function. It combines the concept of simulated annealing [61] with that of particle filtering. The algorithm is shown in Figure 6.3.

The main idea behind this filter is to start with a set of particles in the domain of the function to optimize. These particles can then be perturbed and weighted by the objective

function. We then can draw samples from the resulting set with probability proportional to their weights. Perturbing and sampling in this manner allows the resampled particles to concentrate about the local optimum. To prevent particles from being trapped in a local optimum, an annealing coefficient (i.e. β) is applied to the objective function. This term has the effect of amplifying peaks, as β increase, and attenuating peaks or smoothing out the objective function when β is small. During the APF process, β is initialized with a small value thereby smoothing out the objective function. Between iterations, β increases, thereby accentuating the global optimum. This allows particles to explore the entire domain of the objective function and gradually find their way to the global optimum.

6.3 Formulation

The overall system is shown in Figure 6.2. In this system we keep track of the user's head location within an image using an appearance based face/head detector and tracker. The location of the head in the world can then be computed from the depth map. Once the head is anchored, we can identify foreground pixels that belong to the head and torso and separately track the arms. This is done by searching about either their last know position or a set of predefined postures.

6.3.1 Stereo Input Images Processing

The first stage in processing the stereo information consists of finding and tracking the head of the user in the environment. This information is found by tracking the head

position within one of the stereo cameras using the face detection module shown in Figure 3.1.

Once the head is identified, points corresponding to the face or torso are removed from the depth image. Since we are assuming the user is approximately upright, this is accomplished by placing a rectangular 3D box about the center of the head position. The size of the box is proportional to the size of the head. As shown in 6.4, all points inside this box are considered either head or torso pixels and removed. The remaining *non-torso* pixels, $\mathbf{I}_{nt}$, are considered points on the arm and are used by the annealed particle filter.

While this processing is somewhat ad-hoc, it effectively removes pixels that may confuse the arm tracking system.

6.3.2 Stereo Arm Tracking

The APF forms the core of the arm tracker. In particular, we define an articulated model consisting of arms anchored at the computed 3D head location. We then try to find the arm configuration that can account for the most non-torso or head pixels.

The upper body model used is shown in Figure 6.5. It is parameterized by the angles and limb lengths shown. This includes the lengths of the forearms and upper arms, width of shoulders, and the upper arm and lower arm angles. This is a total of 11 degrees of freedom.

The saliency of this articulated model, ϕ, against the processed depth image, $\mathbf{I}_{nt}$, is based on the number of unique points the upper body model can account for. This is accomplished by projecting select fixtures on the arms of the upper body model into the image. Non-torso points in a radius of about 15 pixels about these projected points in

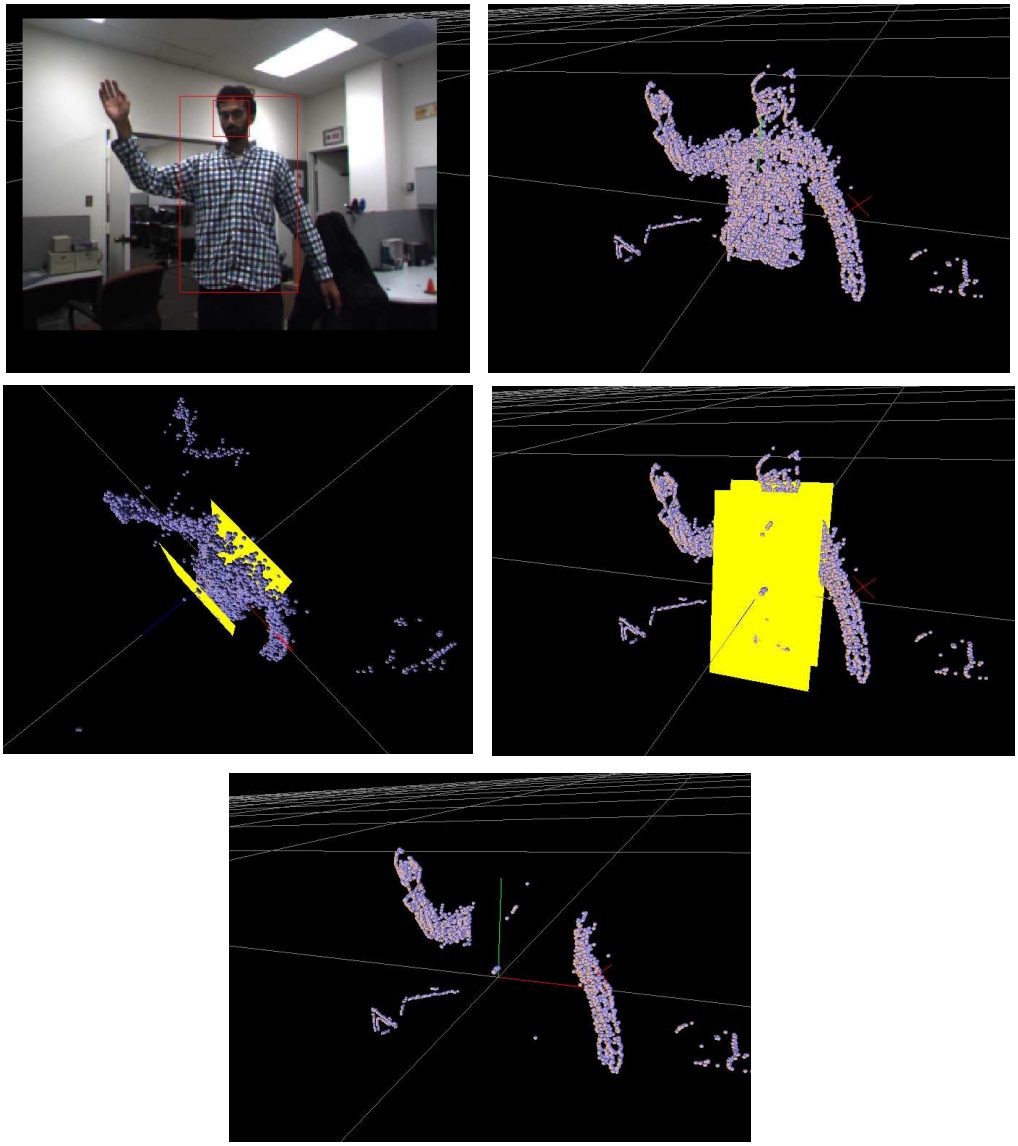


Figure 6.4: In the first row the stereo input is shown. In the second row a box is placed about the head center to remove head and torso pixels. The result is shown in the third row.

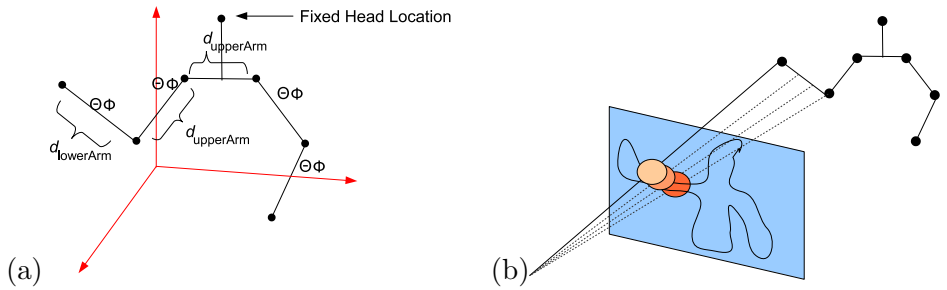


Figure 6.5: In (a) the articulated model. In (b) the process in which depth points are assigned to the model.

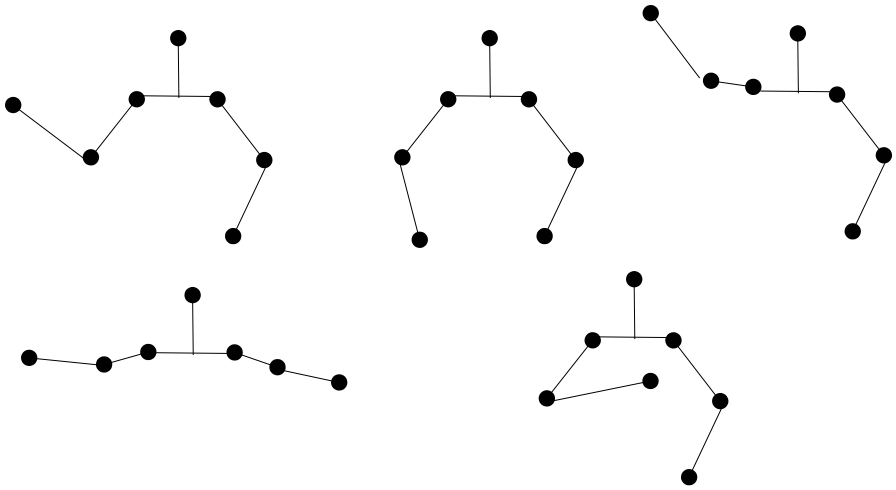


Figure 6.6: Postures used to Initialize the APF

the image plane that are also within .05 units in depth of the fixtures are counted. We ensure depth points are not counted twice by removing points as they are assigned to a fixture. This is illustrated in Figure 6.5.

The number of non-torso points found can then be used as the objective function, ϕ, in the APF. We thus seek to find the upper body that accounts for the most non-torso points.

6.3.3 APF Initialization and Re-Initialization

To start the APF an initial set of samples is needed. In this application we draw samples from a set of 200 predefined postures, some of which are shown as shown in Figure 6.6.

From this initial set the APF is able to converge on a solution. In subsequent frames we can initialize the APF with the set of particles obtained from the previous frame. This allows us to focus the search about the previous posture, assuming continuous motion. To facilitate recoveries from tracking failures, we force the APF to restart from the predefined postures every 10 frames.

6.4 Results

We demonstrate this algorithm in an annotated 14 frame sequence. As shown in Figures 6.7 the user moves his arms in a waving gesture. Here, we superimpose all samples to show the distribution of particles. From this figure we see the entire set of particles follows the user's arms.

Quantitative results are shown in Figure 6.8. Here we report *Rank* results with respect to the ground truth associated with the frames. The rank results were computed with respect to the difference in 3D joint locations. Thus the *RankN* solution is the solution amongst the highest scoring N particles whose 3D joints are closest to the ground truth. In Figures 6.8(a,b) we show the average error of each joint in the overall sequence. We show the average distance between the projected joint locations of the estimated pose and ground truth and the average difference in depth separately. In Figures 6.8(c,d) we show the average errors at each frame in the sequence.

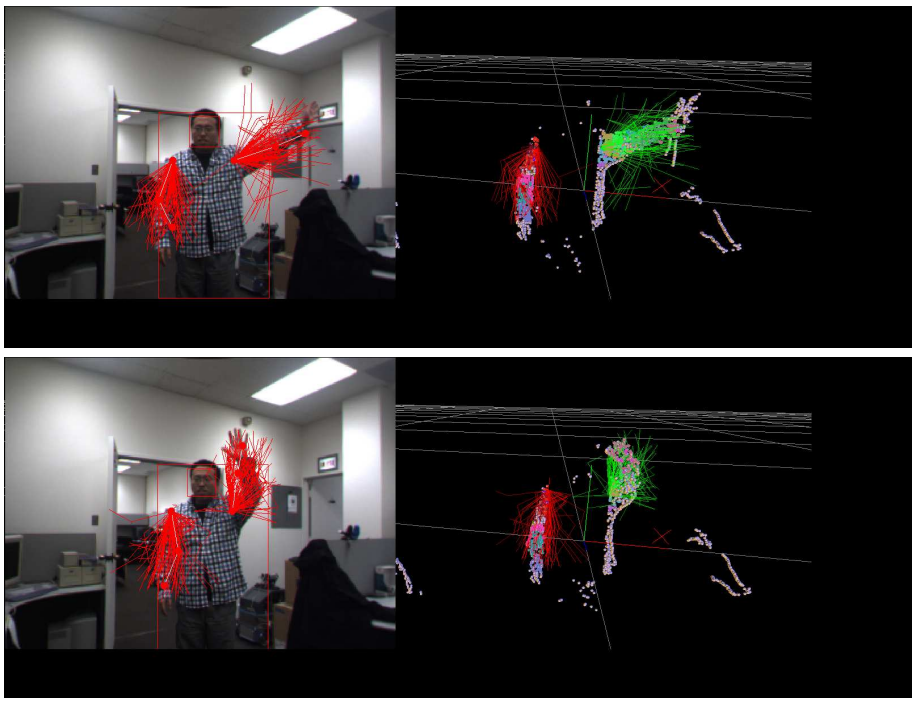


Figure 6.7: Test images and the associated particles from the APF.

From this we see the discrepancy between the $Rank1$ solution and $Rank64$ solution is not that significant. This is because the objective function is designed to yield a single optima. We also note that the error decreases over the sequence as the APF converges onto the correct pose.

The processing time devoted to the APF ranges from 400 to 500ms per frame.

6.5 Discussion

In this chapter, we presented a system that can track the arms of a user from stereo images using an annealed particle filter. This shows that a manually designed cost function together with an optimum found via the APF can be used to track the arms of a user.

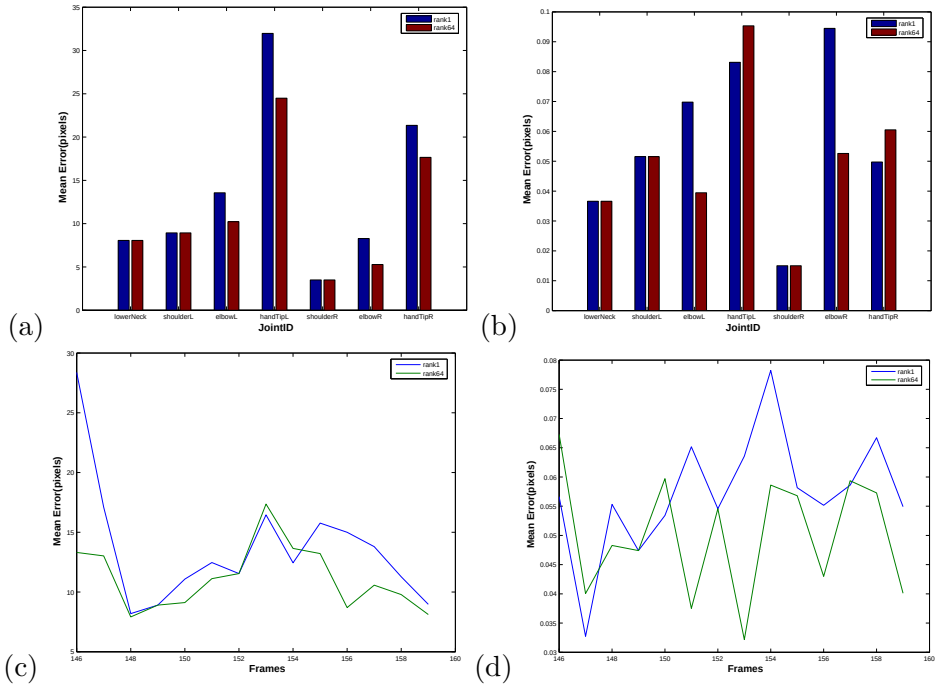


Figure 6.8: In(a) the average joint error for each joint projected into the image over the sequence. In (b) the average depth error for each joint. In (c,d) the average joint error at each frame in sequence.

In this approach we used a simple bounding box to separate the arms from the torso and defined an objective function to evaluate the segmented result. While this is effective when the user is upright, it can be problematic in more general postures. Our use of an initial set of posture also limits the generality of this system. We address these limitation in Chapter 7, by designing an algorithm which uses an objective function that evaluates the entire pose and does not require an initial set of predefined postures.

The performance of this system is largely depended on the quality of the depth image. Since we are using a stereo based sensor, good depth information is only available highly textured areas. This effective requires users to where clothing with a significant amount of texture. In the following chapter we make use of a Range sensor. Using time of

flight technology, textureless areas is ont an issue. Also this camera offers superior depth information although at a lower resolution.

Chapter 7

Pose Estimation with a Real-Time Range Sensor

Depth sensors bridge the gap between single and multi-view systems by providing 3D measurements from a single viewpoint. A key enabling factor is the recent development of affordable real-time depth-sensing cameras (Fig. 1.2). These sensors produce images where each pixel has an associated depth value. Depth input enables the use of 3D information directly, which eliminates many of the ambiguities and inversion difficulties plaguing 2D image-based methods.

In this chapter, we estimate and track articulated human poses in sequences from a single view, real-time range sensor. In particular, we are using the SR3000 from MESA, which produces images of size 176×144 at 25fps. We use a data driven MCMC approach to find an optimal pose based on a likelihood that compares synthesized depth images to the observed depth image. To speed up convergence of this search, we make use of bottom up detectors that generate candidate head, hand and forearm locations. Our Markov chain dynamics explore solutions about these parts and thus combine bottom up and top down processing. The current performance is 10 frames per second. We

provide quantitative performance evaluation using hand annotated data. We demonstrate significant improvement over a baseline ICP approach.

7.1 Related Work

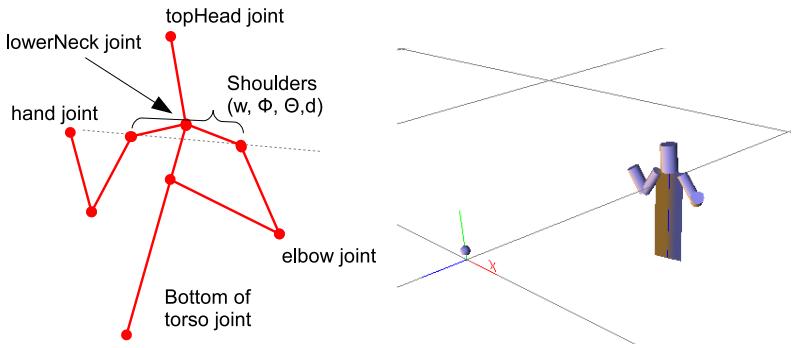
While methods designed for use with stereo sensors such as [93][19][32] can be used with range sensors, many approaches have been specifically designed for use with range sensors.

In particular, in [99], a coarse labeling of depth pixels is followed by a more precise joint estimation to estimate poses. In [94], control theory is used to maintain a correspondence between model based feature points and depth points. The work in [67] makes use of a part based alignment metrics, together with an articulated pose prior, which is optimized using loopy belief propagation. This is further refined using ICP.

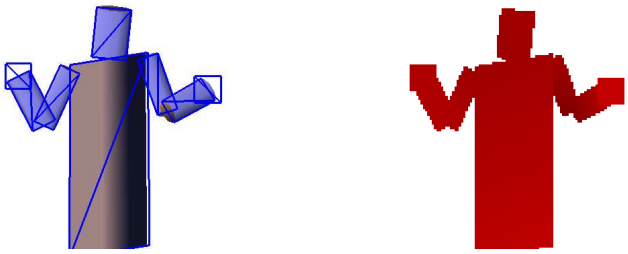
In our work, we find poses by optimizing a generative likelihood that accounts for an observed depth image directly. We solve this problem by combining both top down and bottom up processing in a data driven MCMC framework to find an optimal pose using only depth imagery. We do not need a database of poses nor a large training step. We also do not rely on precise segmentation of the depth streams, which is often problematic in sequences with significant sensor noise or motion blur. Thus, our system is able to track effectively, and is robust to discontinuous motion and tracking failures.

7.2 Representation

We model the body as a skeleton with fixed width cylinders attached. The skeleton itself is modeled as a graph whose vertices are the joints, and edges are the limbs as shown



(a) 3D Pose Skeleton (b) Synthesized Pose(3D)



(c) Projected Pose (2D) (d) Rasterized Pose(2D)

Figure 7.1: Representation of Poses

in Fig. 7.1(a). The joints for the hands, elbows, top of head, lower neck, and bottom of torso are parameterized by their 3D positions. The position of the head and bottom of the torso are defined relative to the lower neck position. The shoulders' joints are defined in 3D relative to the lowerNeck joint. They are parameterized by: w, the distance between them, θ/ϕ, the orientation of the line passing between them, and d, the position of this line along the line between the lower neck, and bottom of torso joints.

This model is defined in the coordinate system aligned with the camera. Thus, depth is along the optical axis and depth information is separated from the image positions. Because depth measurements are more noisy than image plane measurements, parameterizing in this way, allows the solution space to be explored more effectively.

The limbs, head, and torso are fixed width cylinders attached to this skeleton. The cylinder lengths, however are scaled to allow their ends to coincide with their corresponding joints. For each subject, the skeleton and cylinder dimensions are measured from a simple initial training pose.

The main step in evaluating pose fit quality with an underlying depth image consists of estimating a depth image from the given model. This is done by rendering the model defined above into a depth buffer.

To render the pose efficiently we attach a cylinder to each limb on the model. Following this, we find the occluding boundaries of each corresponding cylinder. Given a pinhole camera, these boundaries correspond to a pair of line segments. From this pair of line segments we can render a pair of triangles into the depth buffer, and interpolate depths at the endpoints of the line segments.

This approximation works well for cylinders parallel to the image plane. However, forearms can become orthogonal to the image plane (pointing gestures). To account for this case, we render hand positions as squares at a single depth location. The size of this square is determined by the projected width of the corresponding cylinder. This process is illustrated in Fig. 7.1.

7.3 Formulation

We denote the skeleton as $\mathbf{X}$, the depth image as $\mathbf{I}$, and the depth image rendered from the view point of the camera as $\tilde{\mathbf{I}}$. In order to find a human pose $\mathbf{X}$ in a range image $\mathbf{I}$, at time t we seek to find the pose that optimizes the likelihood:

$$\mathbf{X}_t = \arg\max p(\mathbf{X}|\mathbf{I}_t) = \arg\max p_{obs}(\mathbf{I}_t|\mathbf{X}) p_{prior}(\mathbf{X}) \tag{7.1}$$

where p_{obs} and p_{prior} are the observation likelihood and prior respectively. The observation likelihood is based on estimating an expected range image from $\mathbf{X}$ and comparing the rendered and observed depth images. The prior is used to give impossible poses zero probability.

This optimization is accomplished using a data driven MCMC framework and the Metropolis Hastings (MH) Algorithm outlined in Fig. 7.2. Here, we generate samples

Function $\mathbf{X}_t$ = search($\mathbf{X}_{t-1}$, $\mathbf{I}_t$)
/* *Computes:* $\mathbf{X}_t = \arg\max p(\mathbf{X}|\mathbf{I}_t)$ */
$\mathbf{X}_t := \mathbf{X}_{t-1}$
$\mathbf{X}^0 := \mathbf{X}_{t-1}$
for i = 1 to N
  Sample $\mathbf{X}^i$ from $q(\mathbf{X}^i|\mathbf{X}^{i-1})$
  Sample u from Uniform$(0, 1)$
  if ($p(\mathbf{X}_t|\mathbf{I}_t) < p(\mathbf{X}^i|\mathbf{I}_t)$) then
    $\mathbf{X}_t = \mathbf{X}^i$
  if ($u < A(\mathbf{X}^{i-1}, \mathbf{X}^i)$) then
    $\mathbf{X}^i = \mathbf{X}^{i-1}$

Figure 7.2: Pseudo-code for Data Driven MCMC Based Search

from a proposal distribution, $q(\mathbf{X}^i|\mathbf{X}^{i-1})$ and keep track of the optimum under $p(\mathbf{X}|\mathbf{I})$.

This is an iterative process in which we perturb the current sample $\mathbf{X}^{i-1}$, evaluate, and then either accept or reject it with a likelihood given by:

$$A(\mathbf{X}^{i-1}, \mathbf{X}^{i}) = \min(1, \frac{p(\mathbf{X}^{i}|\mathbf{I})q(\mathbf{X}^{i-1}|\mathbf{X}^{i})}{p(\mathbf{X}^{i-1}|\mathbf{I})q(\mathbf{X}^{i}|\mathbf{X}^{i-1})}) \tag{7.2}$$

The generated samples form a distribution that represents the likelihood, of which we have maintained the maximum. While it is possible to measure convergence properties to determine a stopping criterion, in this work we assume convergence after a fixed number of iterations for each subproblem. This leads to a maximum number of iterations of 4200 and is further detailed in section 7.3.5.

The convergence speed in this process is dependent on the size of the solution space and degree to which the proposal distribution concentrates samples around likelihood modes.

To speed up convergence, we design effective proposal mechanisms detailed in section 7.3.4 and we search over sets of parameters separately. In particular, our proposal mechanisms make use of candidate parts, depth points, and the found pose in the previous frame. By generating proposals through combining information from these sources, we are able to effectively search the solution space.

The overall approach is shown in Fig. 7.3. We first update the head, torso and shoulder parameters while preserving the parameters associated with the arms. In this step we make use of candidate head positions as described in section 7.3.3.1. This estimate for the head and torso is used in the forearm candidate detection of section 7.3.3.2. Following this we can search for the skeleton using the dynamics described in section 7.3.4.

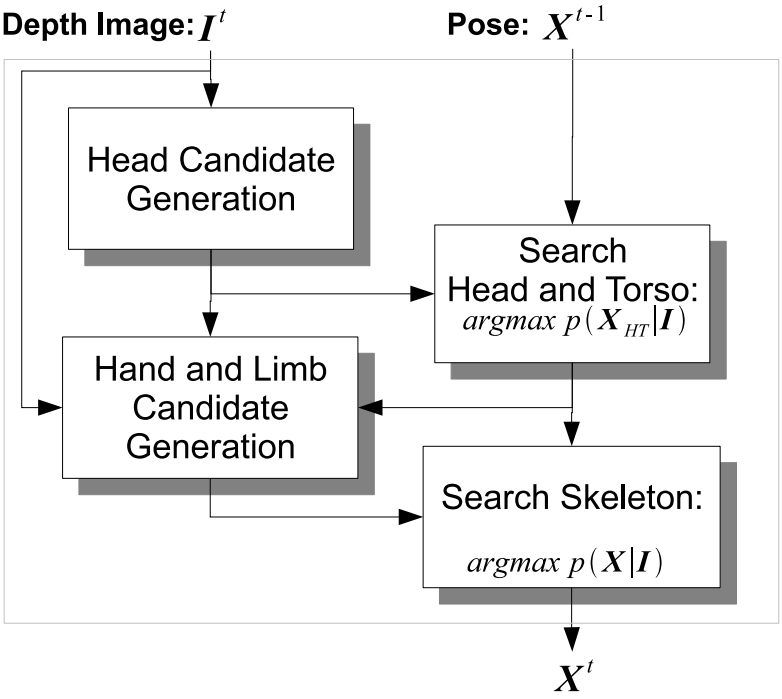
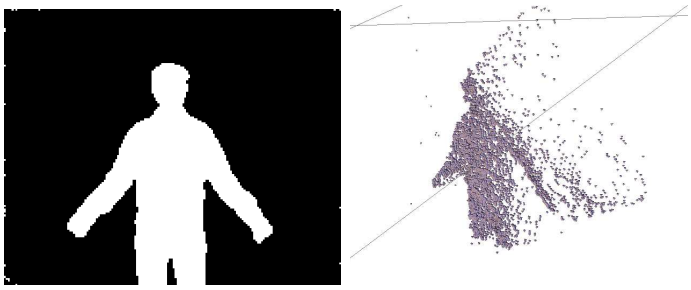


Figure 7.3: Estimation of a Human Pose in a Depth Image

The optimal pose found using this approach, $\mathbf{X}_{max}$, is used to initialize the process in the next frame. By tracking a pose this way, we are able to maintain the optimum under the likelihood. Since the proposal distribution makes use of part candidates found throughout the image, we are able to combine both bottom-up and top-down processing and remain robust to discontinuous motion and tracking failures.



(a) (b)

Figure 7.4: Silhouette(a) and Depth Data (b)

7.3.1 Observation Likelihood

The observation likelihood of a pose defined in section 7.2 is based on rendering a synthesized depth image into a buffer and computing its difference with the observed depth image. In particular, our observation likelihood is:

$$p_{obs}(\mathbf{I}|\mathbf{X}) \sim \exp(-(\quad \lambda_1\phi_s(\tilde{\mathbf{I}},\mathbf{I}) + \lambda_2\phi_d(\tilde{\mathbf{I}},\mathbf{I}) + \lambda_3\phi_{dt}(\tilde{\mathbf{I}},\mathbf{I}))) \tag{7.3}$$

In this equation $\mathbf{X}$ represents our body model, $\mathbf{I}$, the observed depth image, and $\tilde{\mathbf{I}}$ the rendered/expected depth image from $\mathbf{X}$. In this metric, we separate the overall saliency into terms that depend on the foreground silhouettes, ϕ_s, and depth information, ϕ_d and ϕ_{dt}.

Separating depth (3D) and silhouette (2D) terms this way is important because we can assign weights based on their reliability. A pixel is considered part of the foreground silhouette if its depth is closer than a threshold, D_{max}. In Fig. 7.4, we see the absolute depth value is susceptible to measurement errors as well as motion blur, whereas the silhouette is very stable in the image plane. Also, foreground silhouette information is

important for body configurations that do not vary significantly in depth, but in which the limbs are still visible.

The term ϕ_s counts the number of pixels which are different between foreground silhouettes:

$$\phi_s(\tilde{\mathbf{I}}, \mathbf{I}) = \sum_i f(\tilde{I}_i, I_i) \tag{7.4}$$

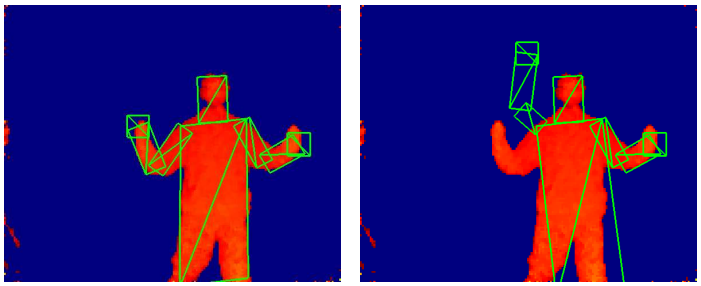
$$f(a,b) = \begin{cases} 1 & if a < D_{max} \oplus b < D_{max} \\ 0 & otw \end{cases} \tag{7.5}$$

The term ϕ_d computes the sum of the thresholded squared differences between the observed and estimated depth images:

$$\phi_d(\tilde{\mathbf{I}}, \mathbf{I}) = \sum_i f(\tilde{I}_i, I_i) w_i \tag{7.6}$$

$$f(a,b) = \begin{cases} (a-b)^2 & if |a-b| < d_{thresh} \\ d_{thresh}^2 & otw \end{cases} \tag{7.7}$$

Because depth measurements tend to be noisy about depth discontinuities, we weight each term in the sum by $w_i = 1/(1 + \alpha D_i)$, where D_i is the magnitude of the Sobel edge at the corresponding depth pixel. In our experiments $\alpha = .0001$.



(a) (b)

Figure 7.5: High (a) and Low (b)Scoring Poses

The term ϕ_{dt} counts the number of pixels missed in depth. In particular, it computes the number of pixels in a pair of depth images whose difference is greater than a predetermined threshold:

$$\phi_{dt}(\tilde{\mathbf{I}}, \mathbf{I}) = \sum_i f(\tilde{I}_i, I_i) w_i \tag{7.8}$$

$$f(a,b) = \begin{cases} 0 & if|a-b| < d_{thresh} \\ 1 & otw \end{cases} \tag{7.9}$$

The term ϕ_{dt} allows us to explicitly penalize pixels missed in depth, whereas ϕ_d ensures smoothness of the observation likelihood in depth. In our work $d_{thresh} = .1$.

Examples of high and low scoring poses over a sample depth image are shown in Fig. 7.5.

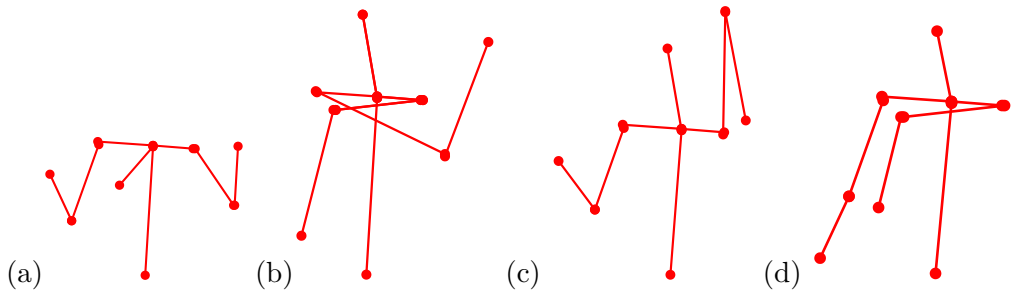


Figure 7.6: Classes of Impossible Poses: (a) Top of head falls below lower neck, (b) Upper arms crossing, (c) Elbows pointing up, (d) Arms crossing the torso and bending down

7.3.2 Prior

The prior term, p_{prior}, in equation (7.1) assumes that limb lengths are Gaussian distributed about average lengths, and assigns zero probability to poses that are highly unlikely to correspond to actual poses. It is thus of the form:

$$p_{prior}(\mathbf{X}) = p_{limb}(\mathbf{X}))(1 - p_{impossible}(\mathbf{X})) \tag{7.10}$$

Where

$$p_{limb}(\mathbf{X}) \sim \exp\left(-\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathbf{X}} \frac{(|\mathbf{x}_i,\mathbf{x}_j| - l_{ij})^2}{2\sigma^2}\right) \tag{7.11}$$

Here $\mathbf{x}_i$ and $\mathbf{x}_j$ correspond to a pair of joints that form a limb whose average length is l_{ij}.

The term $p_{impossible}$ returns a constant if a pose is not possible and a zero otherwise. Ideally, this can be enforced by determining if $\mathbf{X}$ violates the physical constraints imposed on the joints of a human. As a first approximation to this, we consider impossible poses to be those whose projections are exemplified by the classes shown in Fig. 7.6. This includes those poses whose projection causes the top of the head to fall below the lower neck, as

shown in Fig. 7.6(a) or in which the upper arms cross, as shown in Fig. 7.6(b). We also include poses where the arms bend in unlikely ways. In particular, the class shown in Fig. 7.6(c) includes poses where the elbow is above its corresponding shoulder and hand. The class in Fig. 7.6(d) includes poses which cause the upper arm to cross the torso and form an angle less then 135° with its corresponding forearm.

7.3.3 Part Detection

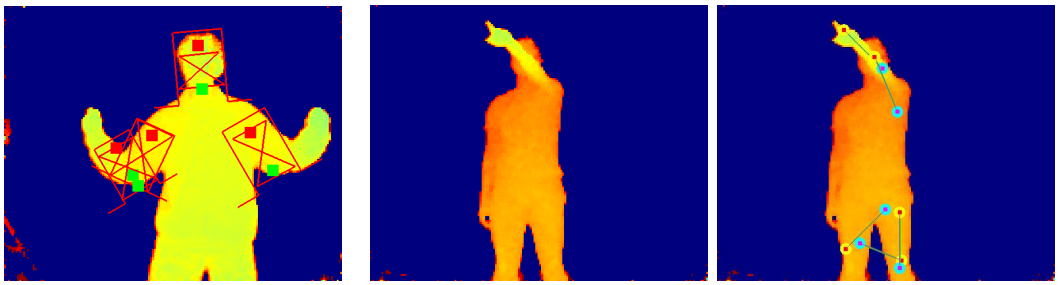
Our estimation algorithm finds body poses that optimize equation (7.1). To aid in this search we make use of bottom up detectors to find candidate head and forearm positions. This bottom up processing occurs at each frame and speeds up convergence of the MCMC algorithm presented in section 7.3.4.

7.3.3.1 Head Detection

To find candidate head positions, we search for the outline of a head shape in the Canny edges of the depth image. At each position and orientation in the image we determine the outline of a fronto-parallel head, situated at a distance given by the underlying depth value. This is illustrated in Fig. 7.7(a). If the head shape sufficiently overlaps foreground pixels, we grade it according to:

$$s = \frac{1}{|BP|} \sum_{\mathbf{x} \in BP} D_{dist}(\mathbf{x}) \tag{7.12}$$

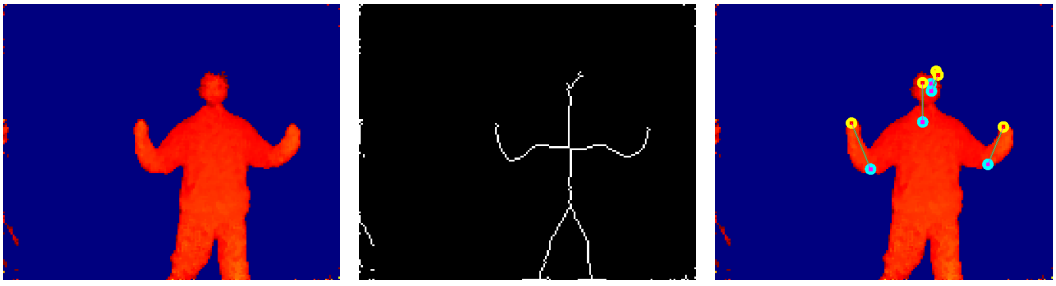
where BP is the set of points in the outline of the head, and D_{dist} is the distance transform of the Canny edges in the depth image.



(a) Head Candidates

(b) Pointing Hand Candidates

Profile Hand candidates:

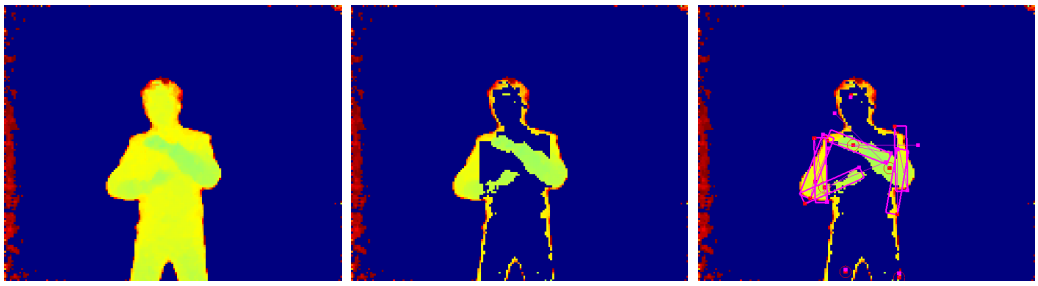


(c) Original

(d) 2D Image Skeleton

(e) Candidates

Rectangle Detection:



(f) Original

(g) Head and Torso Pixels Removed

(h) Candidates

Figure 7.7: Part Candidate Generation

We locate the first candidate head, h_1, by finding the position and orientation that optimizes (7.12). The second candidate, is the head pose that optimizes (7.12) and also differs from h_1 by at least 20 pixels and 10 degrees in orientation. Subsequence poses are found in a similar manner. This is illustrated in Fig. 7.7(a). We keep all such candidates.

7.3.3.2 Forearm Candidate Detection

To find hand candidates, we find the endpoints of the foreground image skeleton. As shown in Fig. 7.7, this works well for profile arms outside the body trunk[17].

We also find other hand candidate points by finding depth points that are relatively close to the camera. This is done by maintaining a list of hand positions that are close to the camera but are also at least 20 pixels apart. This heuristic works when the arm points to the camera and the user faces the camera, which is common.

From these hand candidates, we estimate the elbow position by first finding the orientation, θ, of the forearm in the image plane. This orientation is taken from a rectangle with one end anchored at the hand tip position and also minimizes the average distance of edge points to its boundary.

Using depth information we can determine the 3D location of the hand tip, $\mathbf{x}_{hand}$ candidate, as well as a point, $\mathbf{x}'$, along the direction of θ. From these two points in 3D we can estimate the elbow using the forearm length, $l_{forearm}$, as:

$$\mathbf{x}_{elbow} = \mathbf{x}_{hand} - \hat{d} * l_{forearm} \tag{7.13}$$

where $\hat{d}$ is a unit vector located at $\mathbf{x}_{hand}$ and in the direction of $\mathbf{x}'$.

We also find forearms by segmenting the head and torso pixels. Given an estimate of the head and torso we can label all depth points that are within a threshold to the head or torso cylinders as torso pixels. When these pixels are removed we can better find candidate forearms.

Candidate forearms are subsequently found by considering all positions and orientations in the range image. At each position and orientation, we select two depth points as the major axis of the forearm cylinder. Using these two 3D points as well as the known length and width of the forearm we can hypothesize a forearm in 3D. The outline of this cylinder can be projected back in the image. If there is a sufficient number of foreground pixels in this outline we further evaluate how well it fits the underlying depth pixels by computing the average distance between depths within the cylinder's outline and the plane formed by the occluding line segments.

Multiple candidates are found by repeatedly finding the rectangle that minimizes this average distance and also differs from previous candidates by 20 pixels and 30 degrees.

7.3.4 Markov Chain Dynamics

A critical step in generating samples using data driven MCMC is how the samples are perturbed. In this work, a new sample, $\mathbf{X}^i$, is generated from a previous sample, $\mathbf{X}^{i-1}$, by selecting one of the following methods:

Random Walk Here we generate $\mathbf{X}^i$ by perturbing the parameters of $\mathbf{X}^{i-1}$ under a Gaussian distribution.

Snap to Previous Pose In this move a limb is assigned its position in the previous frame. After this alignment, the updated parameters are perturbed by Gaussian noise.

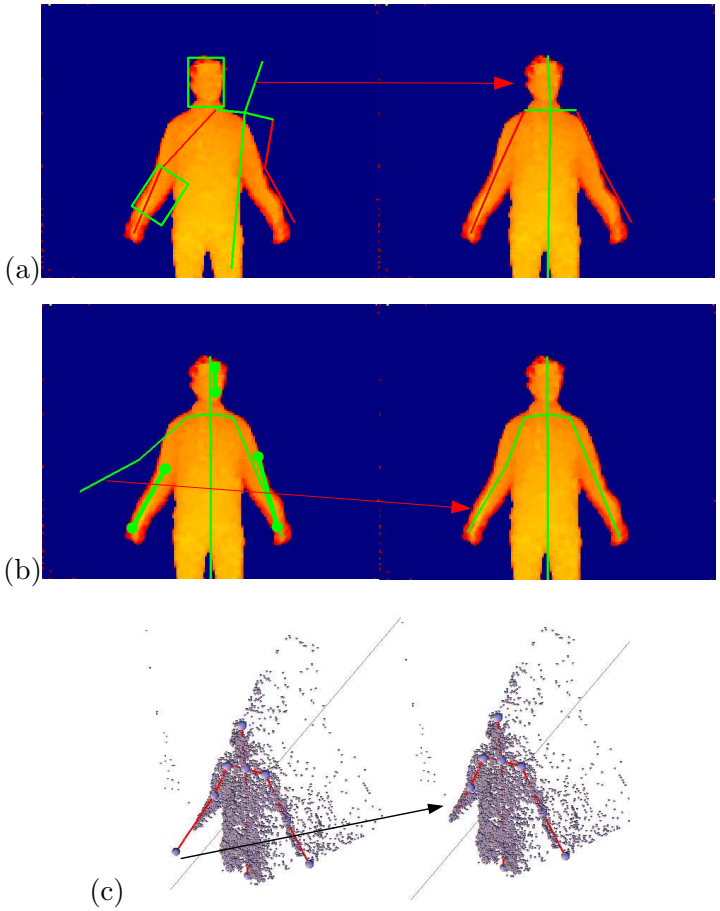


Figure 7.8: Markov Chain Dynamics: (a)Snap to Head (b) Snap to Forearm (c) Snap to Depth

Snap to Head Here we align the head with that of one of the candidate head positions selected with equal probability. This is done by aligning the top head joint. The lower neck joint is aligned at random. We also randomly adjust the torso to be directly under the head. This is illustrated in Fig. 7.8(a), where the head is aligned with a head candidate. After this alignment, the updated positions are perturbed by Gaussian noise.

Snap to Forearm In this proposal we first randomly select a candidate limb or a pair of candidate limbs. We then assign the pose to its 2D hand positions. The depth assigned is either that of a nearby pixel, the average depth in a window about the hand, or its previous depth. The corresponding elbow is either assigned its position from $\mathbf{X}^{i-1}$ or the estimated elbow position. This is illustrated in Fig. 7.8(b), where a forearm is aligned with a forearm candidate. We also, at random swap the hand and elbow position, place the elbow directly behind the hand the length of the forearm, or place the elbow at the midpoint of the hand and corresponding shoulder. After this alignment, the arm is perturbed by Gaussian noise.

Snap to Depth Here, we select at random either a hand, elbow, lower neck, top head, or bottom torso joint. We then adjust its depth by either: assigning it the depth of a nearby depth point, or computing the average depth in a window about the point with equal likelihood. In the case of the top head joint, we also can assign it the depth of the lower neck. This is illustrated in Fig. 7.8(c), where the hand joint is snapped to a depth point under it.

7.3.5 Optimizing using Data Driven MCMC

To obtain better convergence properties, we do not optimize over all parameters simultaneously. Given the high dimensionality of the solution space, a full search would require a prohibitively large number of iterations. Instead, we optimize over groups of parameters.

We first update the head, torso and shoulder parameters while preserving the parameters associated with the arms with 600 iterations of the MH algorithm in Fig 7.2. After this we perform 600 iterations on both arms, then 200 iterations only on the parameters associated with the right arm, followed by 200 iterations on the parameters associated with the left arm. We then search for head/torso, followed by the left arm, followed by the right arm twice. This gives us a total of 3600 iterations. This, together with 600 MCMC iterations devoted to estimate the head and torso prior to hand and forearm candidate generation (see Fig. 7.3) gives us 4200 total iterations.

7.4 Evaluation

To evaluate our system, we make use of annotated test sets to compute performance bounds as well as compare it to standard approaches used to track poses in range data. In these experiments we assume the model parameters are known and fixed.

7.4.1 Comparative Results

In our comparative analysis we evaluate our system on annotated test sets. Our dataset consists of four general categories of motions. The first category is Single Arm Profile motions (SAPr). This consists of one arm moving such that the arm is mostly lateral

to the image plane. This includes motions such as waving. We consider a single arm pointing toward the camera in Single Arm Pointing (SAPt). We consider both arms moving in Two Arm Motion (TAM) and motion dominated by the body in Body Arm Motion (BAM). Our data sets includes four different subjects with different limb widths and lengths. Example images of each category along with the skeletons found by our system are shown in Fig. 7.9.

These data sets have been manually annotated using specially designed software tools that allow us to visualize the depth data in 3D and position joints accordingly. We designed these tools using OpenGL/glut.

Using this data, we can evaluate our performance quantitatively as shown in Fig. 7.10. As a comparison, we show results for the ICP-based algorithm described in [32]. In this ICP implementation we use a cylinder based pose model parameterized by a skeleton with fixed limb lengths and widths. As ICP is primarily a tracking algorithm, we consider two re-detection schemes: In the first scheme, ICP-LP, if the error that ICP minimizes is greater than a threshold, we use assume the tracker failed and use the last recovered pose. In the second scheme, ICP-GT, we manually re-initialize the model to the ground truth whenever the maximum difference between their corresponding joints exceeds 15 pixels. This re-initialization occurs at the start of each frame. Because we are using ground truth data, ICP-GT represents the performance of ICP using the best pose re-detection possible.

In computing Fig. 7.10 we consider a frame in which the maximum joint error between the pose and the ground truth in the image plane is less then 15 pixels to be a success. The statistics shown are for frames that are successful in each motion category, as well as

for all frames combined (ALL). In Fig. 7.10 the top row shows the results for our system, while the bottom row show the results for ICP-GT. The left column shows image errors, the middle column shows depth errors, and the right column shows the success rates

From these results, we see that our system works quite well for frames in which poses were recovered. For SAPt, we note that the elbow positions are often occluded by the hands. While this affects their estimation accuracy, in this class the hand tips, which are localized well, are of greater significance. Our system does lose track, as is indicated by success rates less then 100%. This usually happens when the arms are very close to the body, completely occluded, or motion is very quick. In these cases, the system can lose track of one or both arms, however, it is able to recover.

The overall success rates for each system is shown in Fig. 7.11. Our approach had a success rate of 0.930 whereas the rate of ICP-LP was only 0.169 . Here we see ICP alone, without addressing tracking failures has little chance of performing well. The success rate was 0.907 for ICP-GT. Recall that ICP-GT manually reinitializes to the correct pose when its estimate is too different from the ground truth, and thus represents the best possible re-detection with an ICP-based tracker. Even in this case, our recovery rates are slightly higher. This demonstrates the effectiveness of our tracking and bottom up processing in automatically recovering from tracking failures.

Our system is also able to track at higher levels of accuracy for recovered poses. The overall average error for our system is 2.56 pixels in the image and 0.036 in depth. In contrast ICP-GT with manual re-initialization had errors of 3.13 pixels in the image and 0.050 in depth. These results are significant with a p-value of less then 0.1. This demonstrates the modeling accuracy of our likelihood measures.

7.4.2 System Evaluation

To evaluate the system we compute its performance in terms of success rates and joint errors as a user moves away from the camera and parallel to the camera. In these sequences, the user stands at a specified position relative to the camera and moves his arms. At each point, we compute the average distance between estimated pose and the annotated ground truth as the joint error. Success rates are computed as before.

Distance From Camera In this sequence, the performance as the users moves away from the center of the camera is evaluated. The path the user takes is shown via the vertical line in Fig. 7.12 as *Distance from Camera* and the performance is shown in Fig. 7.13. Points in the plot are taken to the starting point shown on the path. From these plots we see that the 3D joint error is relatively constant, while the success rates decrease with the distance from the camera. This show that when the system is able to find poses, its accuracy is reasonably stable. The loss of performance can be attributed to the part detectors, whose accuracy is contingent on there being enough pixels to make local measurements.

Distance Along Camera In this sequence, the performance as the users moves away from the center of the camera is evaluated. The path the user takes is shown in Fig. 7.12 as *Distance from Camera* and the performance is shown in Fig. 7.13. In this range arm motion was constrained to be in the field of view of the camera. Here we see that performance was fairly uniform across the path.

7.4.3 Limitations

From these results, we see that our system works quite well for frames in which poses were recovered. However, our system has difficulties, when the arms are fully occluded. Our system will try to explain the current image assuming the arms are visible. If an arm is fully occluded, it will try to align the arm with another part of the body. This is illustrated in Fig. 7.15. Other difficulties occur when the low level detectors completely miss their targets. This can happen if there is too much motion blur in the frame, or the size of the corresponding parts are too small. Since we search about the pose in the previous frame, we generally do not need the detectors to work all the time when motion is slow. When motion is fast, however, and the detectors fail, our system is unable to find the optimal configuration within the number of iterations available.

7.5 Discussion

In this chapter, we developed a system that combines bottom-up and top-down processing using a data driven MCMC framework on range images. We have developed an effective likelihood based on efficiently rendering hypothesized depth and comparing it to the observed depth image. We also penalize impossible pose configurations. We have also designed robust bottom up part detectors that allow the system to automatically recover tracking failures. Currently our system runs at approximately 10fps in a single threaded framework running on 32-bit 3GHz-Xeon processor with 8GB of Ram. Most of the processing is devoted to rasterizing and computing differences between depth buffers.

We plan on making use of General Purpose Graphics Processing Units (GPGPU) for further improvements in both speed and modeling accuracy. In particular, we can process multiple parallel Markov chains and rasterize more complex limb models. In addition, while model parameters such as lengths and limb widths are currently measured in a training frame, we plan on automating this process either in a calibration sequence or within the first few frames of an arbitrary sequence. This idea is developed in Chapter 8.

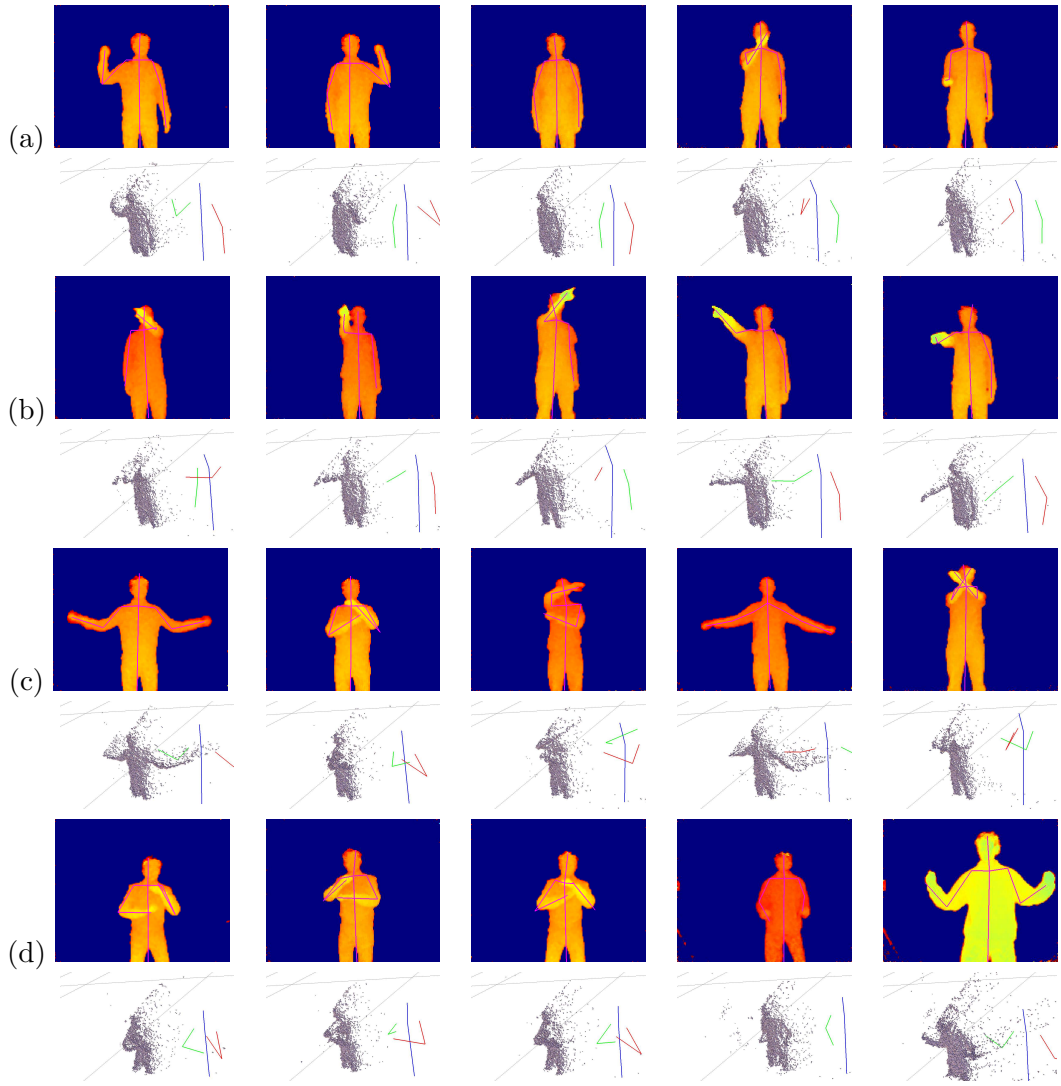


Figure 7.9: Examples:(a)Single Arm Profile (SAPr)(b)Single Arm Pointing (SAPt)(c)Two Arm Motion(TAM)(d)Body Arm Motion(BAM)

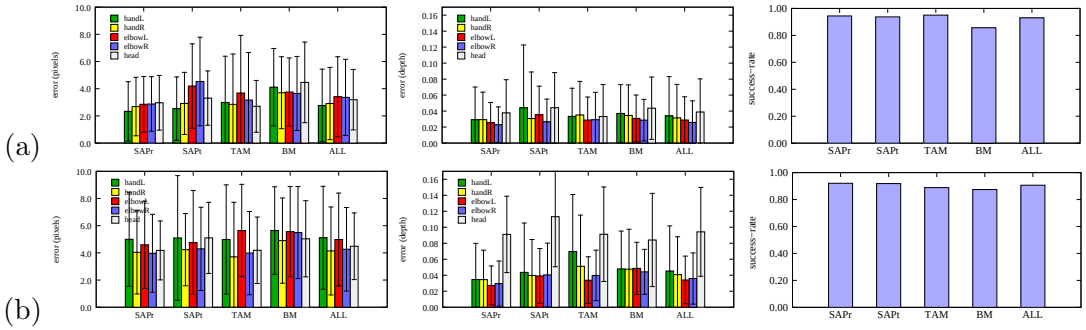


Figure 7.10: Quantitative Evaluation of Motion Types: (a) Data Driven MCMC, (b) Iterative Closest Point w Ground Truth Re-Initialization.

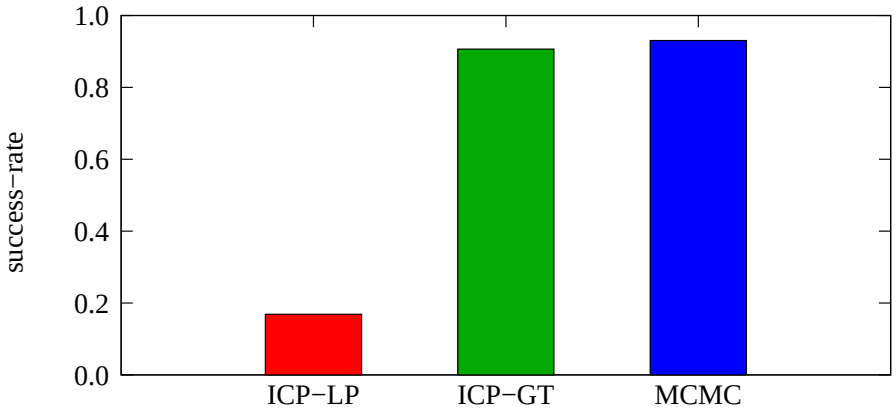


Figure 7.11: Success rates for tracking systems

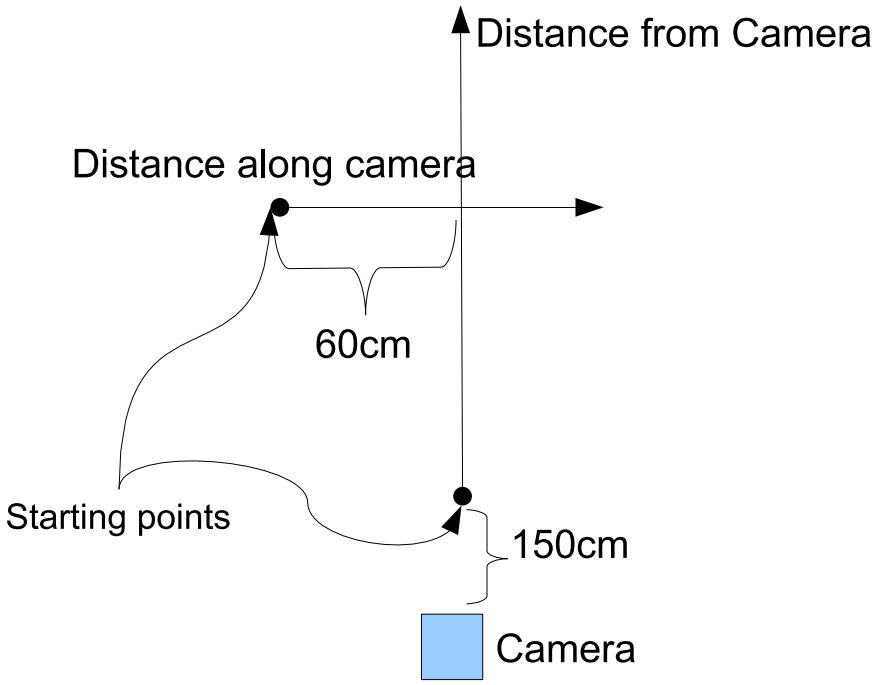
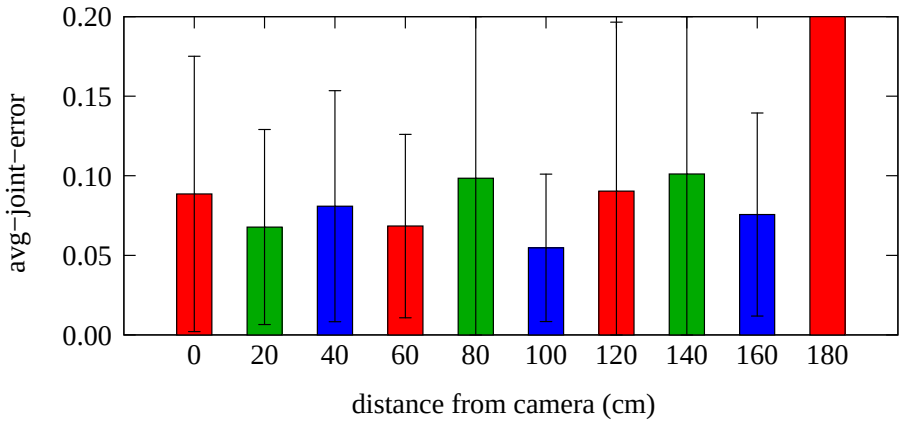
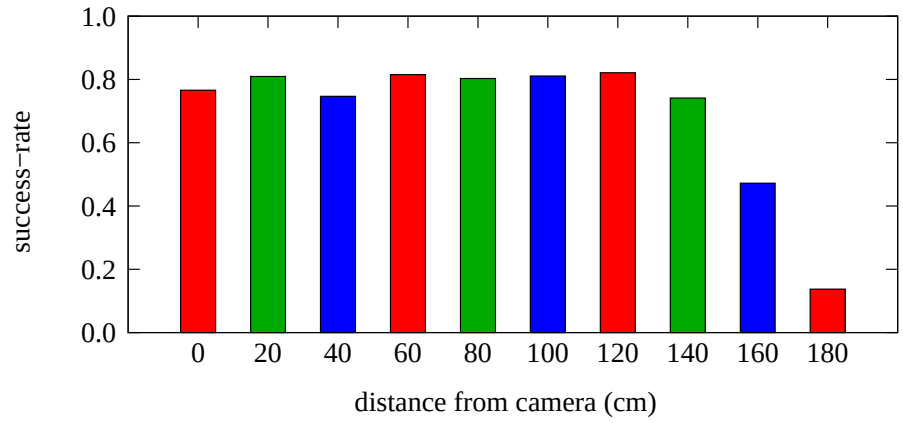


Figure 7.12: Paths of Evaluation

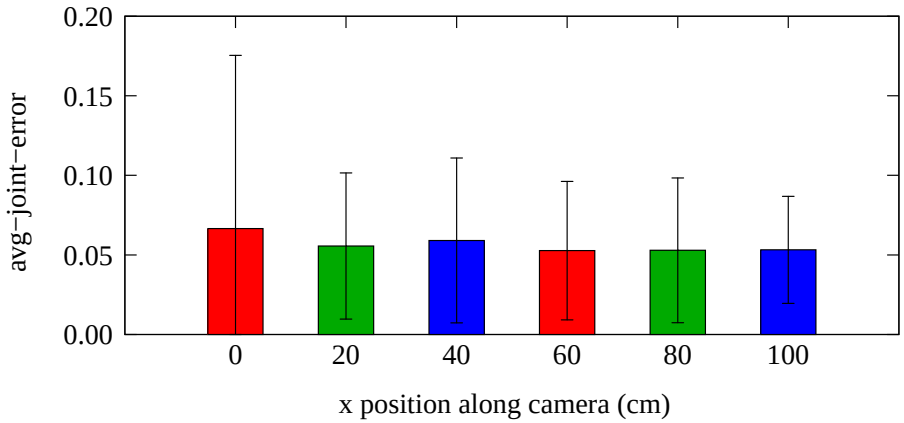


(a) Joint Error

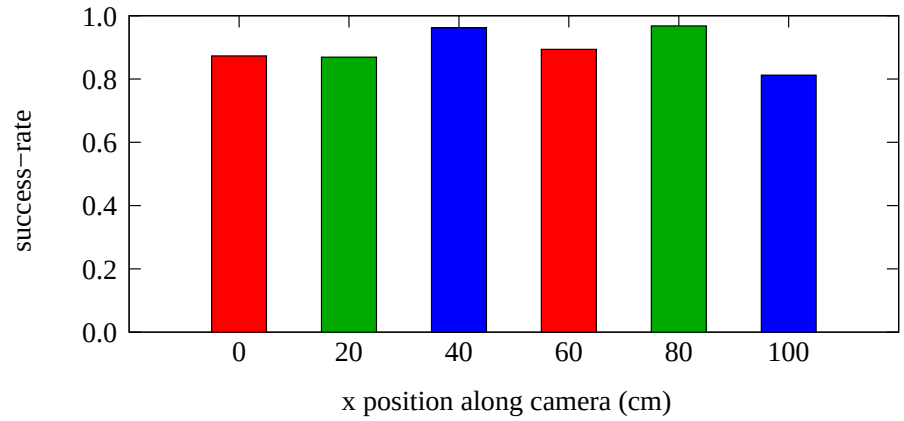


(b) Success Rate

Figure 7.13: Performance vs distance from Camera (relative to starting point on path)



(a) Joint Error



(b) Success Rate

Figure 7.14: Performance vs distance along Camera (relative to starting point on path)

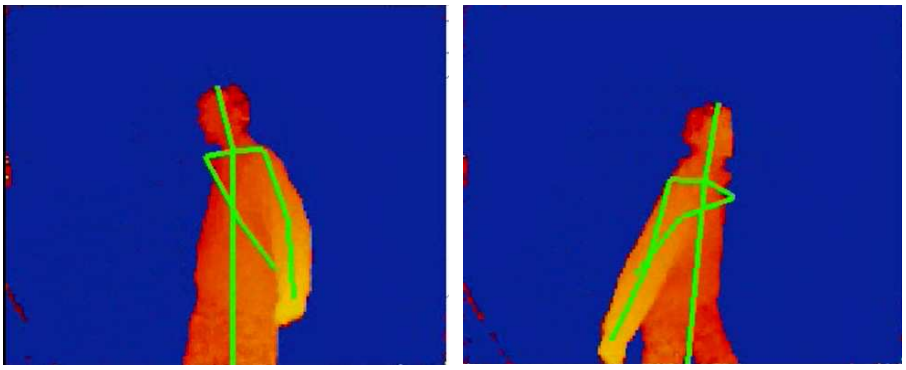


Figure 7.15: Occlusion Failures

Chapter 8

Model Parameter Estimation

In the previous chapter we assumed the lengths and widths of the human were known and fixed. Given these model parameters, we were able to find the pose of the user in each frame of a sequence. While the performance of this tracking system is robust, it does depend on the the model parameters used. This is especially true on subjects whose dimensions vary significantly from the model parameters employed, as in the case of tracking a slender subject on a model appropriate for a sumo wrestler.

In this chapter we extend the tracking framework to automatically estimate these model parameters from a training sequence. Using a Bayesian framework, these parameters are estimated by alternating between estimating the pose and model parameters in each frame in the sequence independently and then estimating the overall model parameters. While the recovered model parameters are useful in measurement related tasks, we show that tracking performance is also affected. Having good estimates of these model parameters results in improved tracking performance.

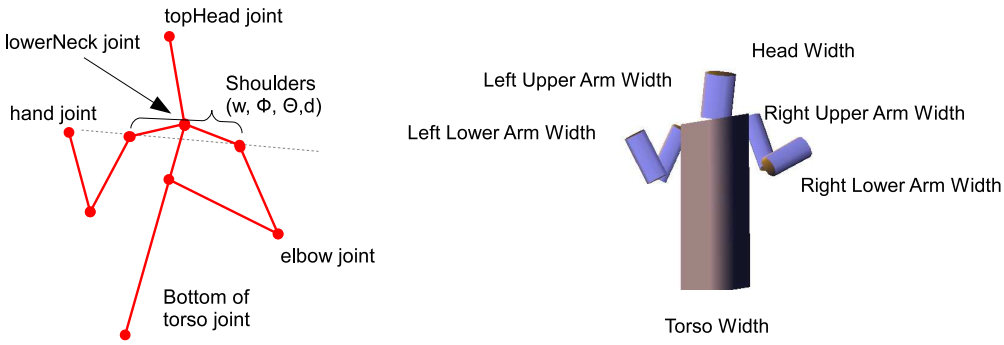


Figure 8.1: Model Parameters

8.1 Formulation

In Chapter 7, we estimated the pose of a specific skeletal model, $\mathbf{X}_t$, in an range image $\mathbf{I}_t$. We now seek to find both the skeleton in each frame as well as the shape of the body. The shape of a model includes the width and length of the cylinders comprising its limbs. We denote the extended parametrization of the pose as $\hat{\mathbf{X}}_t$. We also denote the fixed prior of these model parameters, illustrated in Figure 8.1, as $\mathbf{M} = [\mathbf{W}, \mathbf{L}]$. Note that $\hat{\mathbf{X}}_t$ now contains both skeleton as well as shape information (as lengths and widths). We denote the shape component of $\hat{\mathbf{X}}_t$ as $\mathbf{M}_t$.

To estimate pose parameters we seek to find the optimal configurations of poses and shape and model parameters over a sequence of images. These can be found as the optimum of a likelihood:

$$\{\hat{\mathbf{X}}_\mathbf{t}\}_{t=1}^{N}, \mathbf{M} = \arg\max \prod_t p(\hat{\mathbf{X}}_\mathbf{t}, \mathbf{M}|\mathbf{I}_t) \tag{8.1}$$

Using Bayes' rule:

$$\begin{aligned}
\prod_t p(\hat{\mathbf{X}}_\mathbf{t}, \mathbf{M}|\mathbf{I}_t) &= \prod_t p(\mathbf{I}_t|\hat{\mathbf{X}}_\mathbf{t}, \mathbf{M}) p(\hat{\mathbf{X}}_\mathbf{t}, \mathbf{M}) \\
&= \prod_t p(\mathbf{I}_t|\hat{\mathbf{X}}_\mathbf{t}) p(\hat{\mathbf{X}}_\mathbf{t}|\mathbf{M}) p(\mathbf{M}) \\
&\sim \prod_t p(\mathbf{I}_t|\hat{\mathbf{X}}_\mathbf{t}) p(\hat{\mathbf{X}}_\mathbf{t}|\mathbf{M}) \\
&= \prod_t p(\mathbf{I}_t|\hat{\mathbf{X}}_\mathbf{t}) p(\hat{\mathbf{M}}_\mathbf{t}|\mathbf{M}) \qquad (8.2)
\end{aligned}$$

Here we assume the image, $\hat{\mathbf{I}}$ only depends on $\hat{\mathbf{X}}_t$ and that all shapes are equally likely (i.e $p(\mathbf{M})$ is constant). Furthermore we model the distribution $p(\mathbf{M}_t|\mathbf{M})$ as independent Gaussian in each component (i.e. in each model length and width dimension).

To find the optimum of the likelihood in equation 8.2 we alternate between computing $\hat{\mathbf{X}}_t$ in each frame while fixing $\mathbf{M}$ and then fixing $\hat{\mathbf{X}}_t$ while updating $\mathbf{M}$.

8.2 Parameter Estimation

To find the optimal model parameters in equation (8.1), we make use of sampling based methods to maintain the optimum of the distribution in equation (8.2). In particular we alternate between computing the pose in each frame while fixing $\mathbf{M}$ and then fixing the poses and computing $\mathbf{M}$. This allows us to process each image independently and, from the estimated shape in each frame, find the optimal set of model parameters. The model parameters, $\mathbf{M}$ are computed as the sample mean of the estimated shape in each frame, $\mathbf{M}_t$. This is illustrated in Figure 8.2.

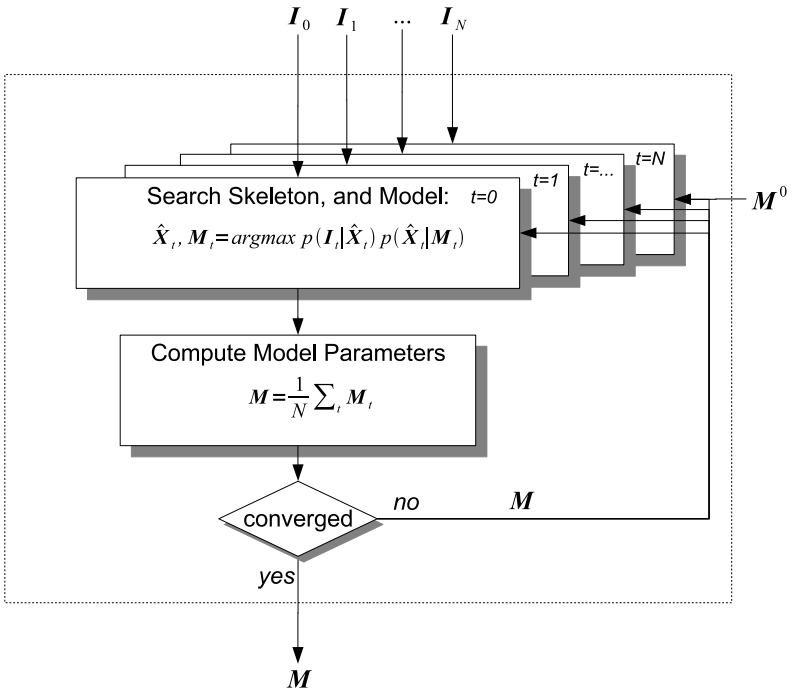


Figure 8.2: Model Parameter and Pose Estimation

To find the optimal pose and model parameters in each frame we can make use of the data driven MCMC framework described in Chapter 7 extended to search over widths as well. This is illustrated in Figure 8.3. Here we first find the optimal configuration based on initial estimate of the model parameters. Following this we optimize over groups of joints and widths using DDMCMC.

Recall that DDMCMC generates samples under a distribution by generating a new sample, $\mathbf{X}^i$, from a previous sample, $\mathbf{X}^{i-1}$. The dynamics governing this are the the same as in section 7.3.4, for parameters associated with the skeleton. In perturbing the widths, however, we only consider Random Walk perturbations.

8.3 Evaluation

To evaluate the performance of our model parameter estimation framework we make use of annotated sequences of 5 test subjects. These sequences include a training sequence followed by a longer test sequence consisting of the motions used in Chapter 7.

From the training sequences we estimate the model parameters for 5 test subjects. Each of the resulting models are used to track the annotated test sequences of each subject. The the performance is shown in the matrices in Figure 8.4.

From this analysis we see that overall performance is dependent on the model parameters used in tracking. Lower errors and higher success rates on the diagonals of the matrix in Figure 8.4 show that having the correct model parameters results in better performance. This is most evident in the subjects "D" and "E". These subjects had shape significantly different from the others. As a result their performance was most impacted by using the model parameters trained on the other subjects.

8.4 Discussion

In this chapter we presented a framework to estimate model parameters from a sequence. This accomplished by iteratively estimating the pose and model parameters in each frame of the sequence and then averaging the recovered model parameters. To find the pose and model parameters, we extended the data driven MCMC framework for pose estimation discussed in Chapter 7 to include the model widths. We have shown that tracking performance is affected by the set of model parameters used and having good estimates of these parameters results in improved tracking performance.

The estimation of these model parameters takes a few seconds approximately per frame. While this processing is not at interactive rates, the parameters are not expected to change for a given user. We can thus use this method in a calibration phase use on the initial frame(s) before processing using our pose estimation system.

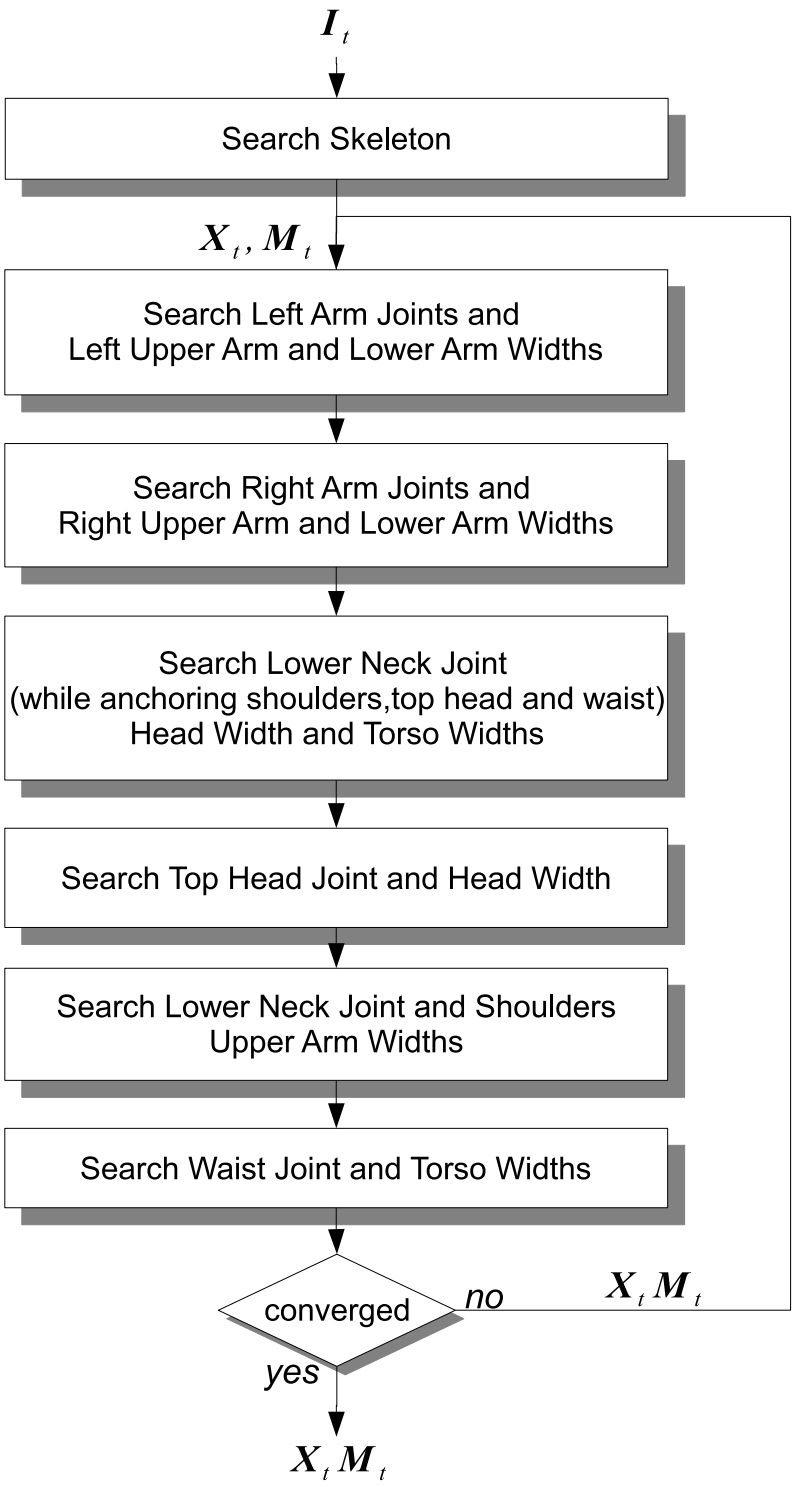


Figure 8.3: Optimum in Frame Likelihood

Success Rate

sequence \ model	A	B	C	D	E
A	0.89	0.89	0.84	0.27	0.79
B	0.98	0.99	0.98	0.37	0.96
C	0.91	0.93	0.90	0.56	0.82
D	0.76	0.71	0.67	0.83	0.33
E	0.46	0.46	0.39	0.16	0.72

Image Errors

sequence \ model	A	B	C	D	E
A	3.58	3.81	4.53	6.88	4.52
B	2.63	2.40	3.04	4.55	3.92
C	3.73	3.25	3.72	7.86	4.48
D	4.70	5.26	4.95	3.80	6.52
E	4.90	5.86	5.96	8.90	4.50

Figure 8.4: Performance over sequences of Specific Users

Chapter 9

Conclusions and Future Work

In this work we have developed methods to extract models and estimate the pose of a human in images and image sequences taken from a single view point. We have considered single view color cameras, stereo cameras and a range sensor. For single view color cameras we have developed an efficient limb detection and tracking framework. We have also designed a framework to extract joint locations in using efficient and exhausted search strategy. We have designed a boosting framework to learn saliency measures from labeled training data.

For stereo sensors we track the movement of a user by parameterizing an articulated upper body model using limb lengths and joint angles. We then use an annealed particle filter to find the optimal pose in the stereo depth image.

For range sensor data, we developed a system that combines bottom-up and top-down processing using a data driven MCMC framework on range images. We have developed an effective likelihood based on efficiently rendering hypothesized depth and comparing it to the observed depth image.

9.1 Future Directions

While we have a considered different approaches to pose estimation, there exist many areas for further study. Some directions of future work include improving extending our methods to account for clothing. Clothing is a difficult thing to model, as it adds additional degrees of freedom to an already high dimensional search space. Also, the use of more realistic limb shapes, instead of rectangles and cylinders, can be used to improve modeling accuracy.

Our methods have be largely focused on find the pose of the upper body of a single user. In addition to estimating the full body, another important direction would be to consider the estimation of multiple interacting people. While one could have multiple instances of our our single person frameworks, multiple people interacting in ways that causes occlusions and limb interaction, such as in dancing or boxing, introduces additional complexities.

References

[1] Ankur Agarwal and Bill Triggs. 3D human pose from silhouettes by relevance vector regression. In *CVPR*, pages II:882–888, July 2004.

[2] Ankur Agarwal and Bill Triggs. Monocular human motion capture with a mixture of regressors. In *VHCI*, pages III: 72–72, 2005.

[3] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. In *Siggraph*, 2005.

[4] A. O. Balan, L. Sigal, and M. J. Black. A quantitative evaluation of video-based 3D person tracking. In *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 349–356, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Alexandru O. Balan and Michael J. Black. The naked truth: Estimating body shape under clothing. In *ECCV (2)*, pages 15–29, 2008.

[6] A.O. Balan, L. Sigal, M.J. Black, J.E. Davis, and H.W. Haussecker. Detailed human shape and pose from images. In *CVPR07*, pages 1–8, 2007.

[7] Olivier Bernier. Real-time 3D articulated pose tracking using particle filters interacting through belief propagation. In *ICPR (1)*, pages 90–93, 2006.

[8] Alessandro Bissacco, Ming-Hsuan Yang, and Stefano Soatto. Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In *CVPR*, 2007.

[9] Gary R. Bradski. The OpenCV Library. *Dr. Dobb's Software Tools for the Professional Programmer*, November 2000.

[10] Matthieu Bray, Pushmeet Kohli, and Philip H.S. Torr. Posecut: Simultaneous segmentation and 3D pose estimation of humans using dynamic graph-cuts. In *ECCV*, pages II: 642–655, 2006.

[11] Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *CVPR*, pages 8–15, Santa Barbara, CA, June 1998.

[12] Marcus A. Brubaker and David J. Fleet. The kneed walker for human pose tracking. In *CVPR*, 2008.

[13] Tat-Jen Cham and James M. Rehg. A multiple hypothesis approach to figure tracking. In *CVPR*, pages II: 239–245, 1999.

[14] German K.M. Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *CVPR*, pages I: 77–84, 2003.

[15] Kiam Choo and David J. Fleet. People tracking using hybrid monte carlo filtering. In *ICCV*, pages 321–328, 2001.

[16] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.

[17] Pedro Correa, Ferran Marqués, Xavier Marichal, and Benoit M. Macq. 3d posture estimation using geodesic distance maps. *Multimedia Tools Appl.*, 38(3):365–384, 2008.

[18] David Demirdjian. Combining geometric- and view-based approaches for articulated pose estimation. In *ECCV (3)*, pages 183–194, 2004.

[19] David Demirdjian, Teresa Ko, and Trevor Darrell. Untethered gesture acquisition and recognition for virtual world manipulation. *Virtual Reality*, 8(4):222–230, 2005.

[20] J. Deutscher, A. Blake, and I.D. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR*, pages II: 126–133, 2000.

[21] Jonathan Deutscher and Ian. D. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61(2):185–205, Feb 2005.

[22] M. Dimitrijevic, V. Lepetit, and P. Fua. Human body pose detection using bayesian spatio-temporal templates. *Computer Vision and Image Understanding*, 104(2):127–139, November 2006.

[23] Ahmed M. Elgammal and Chan-Su Lee. Inferring 3D body pose from silhouettes using activity manifold learning. In *CVPR (2)*, pages 681–688, 2004.

[24] Pedro Felzenszwalb and Daniel Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[25] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *CVPR*, pages 2066–, 2000.

[26] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, Cornell University, September 2004.

[27] Vittorio Ferrari, Manual Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008.

[28] Alexandre R.J. François. Software architecture for computer vision. In G.Medioni and S.B. Kang, editors, *Emerging Topics in Computer Vision*, pages 585–654. Prentice Hall, 2004.

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Special invited paper. additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[30] Dariu Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[31] Dariu M. Gavrila and Larry S. Davis. 3D model-based tracking of humans in action: A multi-view approach. In *CVPR*, pages 73–80, 1996.

[32] Daniel Grest, Jan Woetzel, and Reinhard Koch. Nonlinear body pose estimation from depth images. In *DAGM-Symposium*, pages 285–292, 2005.

[33] Feng Guo and Gang Qian. Human pose inference from stereo cameras. In *WACV*, page 37, 2007.

[34] Ismail Haritaoglu, Davis Harwood, and Larry S. Davis. W4s: A real-time system for detecting and tracking people in 2 1/2-D. In *ECCV*, page I: 877, 1998.

[35] Daniel Heckenberg. Performance evaluation of vision-based high dof human movement tracking: A survey and human computer interaction perspective. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 156, Washington, DC, USA, 2006. IEEE Computer Society.

[36] David C. Hogg. Model-based vision: A program to see a walking person. *IVC*, 1(1):5–20, February 1983.

[37] Nicholas R. Howe. Evaluating lookup-based monocular human pose tracking on the humaneva test data. In *EHuM: Evaluation of Articulated Human Motion and Pose Estimation*, 2006.

[38] Nicholas R. Howe, Michael E. Leventon, and William T. Freeman. Bayesian reconstruction of 3D human motion from single-camera video. In *NIPS*, pages 820–826, 1999.

[39] Sergey Ioffe and David A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, 43(1):45–68, June 2001.

[40] Michael. Isard and Andrew Blake. Condensation — conditional density propagation for visual tracking. *IJCV*, 29:5–28, 1998.

[41] Shanon X. Ju, Michael J. Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. of the 2nd Int. Conf. on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[42] Ioannis A. Kakadiaris and Dimitri Metaxas. Three-dimensional human body model acquisition from multiple views. *IJCV*, 30(3):191–218, December 1998.

[43] Mun Wai Lee and Isaac Cohen. Human upper body pose estimation in static images. In *ECCV*, pages II:126–138, 2004.

[44] Mun Wai Lee and Isaac Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *CVPR*, pages 334–341, 2004.

[45] Mun Wai Lee and Isaac Cohen. A model-based approach for estimating human 3D poses in static images. *PAMI*, 29(6):905–916, 2006.

[46] Mun Wai Lee and Ramakant Nevatia. Body part detection for human pose estimation and tracking. In *WMVC*, pages 23–23, 2007.

[47] Rui Li, Ming-Hsuan Yang, Stan Sclaroff, and Tai-Peng Tian. Monocular tracking of 3D human motion with a coordinated mixture of factor analyzers. In *ECCV (2)*, pages 137–150, 2006.

[48] Xiaoming Liu, Ting Yu, Thomas Sebastian, and Peter Tu. Boosted deformable model for human body alignment. In *CVPR*, 2008.

[49] A.S. Micilotta, E.J. Ong, and R. Bowden. Detection and tracking of humans by probabilistic body part assembly. In *BMVC05*, 2005.

[50] Ivana Mikic, Mohan M. Trivedi, Edward Hunter, and Pamela C. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003.

[51] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV*, pages Vol I: 69–82, 2004.

[52] T.B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 103(2-3):90–126, November 2006.

[53] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[54] Greg Mori. Guiding model search using segmentation. In *ICCV*, pages 1417–1423, 2005.

[55] Greg Mori and Jitendra Malik. Recovering 3D human body configurations using shape contexts. *PAMI*, 28(7):1052–1062, 2006.

[56] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, pages II:326–333, Washington, DC, 2004.

[57] Daniel D. Morris and James M. Rehg. Singularity analysis for articulated object tracking. In *CVPR*, pages 289–296, Santa Barbara, CA, June 1998.

[58] Ryuzo Okada and Stefano Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. In *ECCV (2)*, pages 434–445, 2008.

[59] Ralf Plänkers and Pascal Fua. Articulated soft objects for video-based body modeling. In *ICCV*, pages 394–401, 2001.

[60] Eunice Poon and David J. Fleet. Hybrid monte carlo filtering: Edge-based people tracking. In *MOTION '02: Proceedings of the Workshop on Motion and Video Computing*, page 151, Washington, DC, USA, 2002. IEEE Computer Society.

[61] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[62] Deva Ramanan. Learning to parse images of articulated bodies. In *NIPS*, pages 1129–1136, 2006.

[63] Deva Ramanan and David A. Forsyth. Finding and tracking people from the bottom up. In *CVPR*, pages II: 467–474, Madison, WI, 2003.

[64] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Strike a pose: Tracking people by finding stylized poses. In *CVPR (1)*, pages 271–278, 2005.

[65] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):65–81, 2007.

[66] Xiaofeng Ren, Alexander C. Berg, and Jitendra Malik. Recovering human body configurations using pairwise constraints between parts. In *Proc. 10th Int'l. Conf. Computer Vision*, volume 1, pages 824–831, 2005.

[67] J. Rodgers, D. Anguelov, H.C. Pang, and D. Koller. Object pose detection in range scan data. In *CVPR06*, pages II: 2445–2452, 2006.

[68] Rémi Ronfard, Cordelia Schmid, and Bill Triggs. Learning to parse pictures of people. In *ECCV (4)*, pages 700–714, 2002.

[69] Rómer Rosales and Stan Sclaroff. Learning body pose via specialized maps. In *NIPS*, pages 1263–1270, 2001.

[70] Rómer Rosales, Matheen Siddiqui, Jonathan Alon, and Stan Sclaroff. Estimating 3D body pose using uncalibrated cameras. In *CVPR (1)*, pages 821–827, 2001.

[71] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[72] Gregory Shakhnarovich, Paul A. Viola, and Trevor J. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–757, 2003.

[73] H. Sidenbladh, M.J. Black, and D.J. Fleet. Stochastic tracking of 3D human figures using 2d image motion. In *ECCV*, pages II: 702–718, 2000.

[74] Hedvig Sidenbladh and Michael J. Black. Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1-3):183–209, 2003.

[75] Hedvig Sidenbladh, Michael J. Black, and Leonid Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *ECCV (1)*, pages 784–800, 2002.

[76] Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J. Black, and Michael Isard. Tracking loose-limbed people. In *CVPR*, pages I: 421–428, Washington, DC, 2004.

[77] Leonid Sigal, Michael Isard, Benjamin H. Sigelman, and Michael J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *NIPS*, pages 1539–1546, 2003.

[78] Black M. J. Sigal L. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical report, Brown University, 2006. CS-06-08.

[79] C. Sminchisescu. 3D human motion analysis in monocular video techniques and challenges. In *AVSBS06*, pages 76–76, 2006.

[80] Cristian Sminchisescu and Allan D. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In *ICML*, 2004.

[81] Cristian Sminchisescu and Bill Triggs. Covariance scaled sampling for monocular 3D body tracking. In *CVPR*, pages 447–454, Kauai, Hawaii, December 2001.

[82] Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *I. J. Robotic Res.*, 22(6):371–392, 2003.

[83] Cristian Sminchisescu and Bill Triggs. Kinematic jump processes for monocular 3D human tracking. In *CVPR (1)*, pages 69–76, 2003.

[84] Erik B. Sudderth, Alexander T. Ihler, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. In *CVPR (1)*, pages 605–612, 2003.

[85] Camillo J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *CVIU*, 80(3):349–363, 2000.

[86] Norimichi Ukita, Ryosuke Tsuji, and Masatsugu Kidode. Real-time shape analysis of a human body in clothing using time-series part-labeled volumes. In *ECCV (3)*, pages 681–695, 2008.

[87] Raquel Urtasun, David J. Fleet, Aaron Hertzmann, and Pascal Fua. Priors for people tracking from small training sets. In *ICCV*, pages 403–410, 2005.

[88] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, Kauai, Hawaii, 2001.

[89] Paul A. Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV*, pages 734–741, 2003.

[90] Yang Wang and Greg Mori. Multiple tree models for occlusion and spatial constraints in human pose estimation. In *ECCV (3)*, pages 710–724, 2008.

[91] Matthias Wimmer, Freek Stulp, Stephan J. Tschechne, and Bernd Radig. Learning robust objective functions for model fitting in image understanding applications. In *BMVC06*, page III:1159, 2006.

[92] Masanobu Yamamoto, Akitsugu Sato, Satoshi Kawada, Takuya Kondo, and Yoshihiko Osaki. Incremental tracking of human actions from multiple views. In *CVPR*, pages 2–7, 1998.

[93] Hee-Deok Yang and Seong-Whan Lee. Reconstruction of 3D human body pose from stereo image sequences based on top-down learning. *Pattern Recognition*, 40(11):3120–3131, 2007.

[94] Kikuo Fujimura Youding Zhu, Behzad Dariush. Controlled human pose estimation from depth image streams. In *CVPRW*, pages 1–8, 2008.

[95] Quan Yuan, Ashwin Thangali, Vitaly Ablavsky, and Stan Sclaroff. Parameter sensitive detectors. In *CVPR*, pages 1–6, 2007.

[96] H. Zhang, W. Huang, Z. Huang, and L. Li. Affine object tracking with kernel-based spatial-color representation. In *CVPR*, pages I: 293–300, 2005.

[97] Tao Zhao and Ramakant Nevatia. 3D tracking of human locomotion: A tracking as recognition approach. In *ICPR (1)*, pages 546–551, 2002.

[98] Long Zhu, Yuanhao Chen, Yifei Lu, Chenxi Lin, and Alan L. Yuille. Max margin and/or graph learning for parsing the human body. In *CVPR*, 2008.

[99] Youding Zhu and Kikuo Fujimura. Constrained optimization for human pose estimation from depth sequences. In *ACCV (1)*, pages 408–418, 2007.