Context-aware Hybrid Approach to Monocular Pose Tracking

Olivier St-Martin Cormier

April 2020

Centre for Intelligent Machines

Department of Electrical and Computer Engineering

McGill University

Montréal, Canada

Thesis Committee

Frank Ferrie

Paul Kry

Jeremy Cooperstock

A thesis submitted to McGill University in partial fulfillment

of the requirements of the degree of

Doctor of Philosophy

Abstract

Human pose tracking is a useful tool with multiple applications in a variety of areas ranging from medical, artistic, to industrial. Existing approaches are often impractical as they require complex instrumentation and specialized hardware. In this research we develop a computational theory by which high degree of freedom articulated human pose tracking can be accomplished from observations of a single color image stream. This is supported by a detailed empirical analysis of its implementation through an extensive set of experiments of synthetic data, publicly available datasets, and live video capture sequences.

We posit that, while ambiguous, occluding contours carry enough information to track fully articulated human pose over time in most real world cases. We propose a hybrid computational framework based on both a classical particle filter tracker and a deep learning approach to test our hypothesis. Deep learning is used for state recognition, while the particle filter is used to interpolate and maintain consistency over time. Formally, the input to our tracker is a time-series of images from a monocular camera capture system and the output is the time-varying pose configuration of an articulated 3D human model.

The amount of data required to train a convolutional neural network to perform pose tracking, and the complexity in collecting such data, presents an important problem. To circumvent this, we propose a two pronged approach. First, we use features (silhouettes) instead of full image information as the input to the network, which both reduces the complexity of the input data and makes the tracker texture invariant. The second characteristic of our training strategy is to rely on synthetic data, obtained from a generative model. These two tactics allow a large amount of data to be generated and enable training of the model. We show experimentally that the use of synthetic data generalizes to real-world data.

For the particle filter, we rely on a metric to compare the silhouettes of objects and evaluate the validity of candidate poses obtained both from the neural network and from basic motion models based on physics. To select the most appropriate metric for use with

human shapes, we provide a careful review of methods presented in the literature and present a systematic approach to test their accuracy in order to select the best one.

Another component necessary for the research presented herein resides in determining how to store and access various types of time-based information. We present a novel database architecture, which allows asynchronous access to data and enables predictions to be made from recorded data.

All choices made in the implementation of our framework are based on carefully presented experiments on synthetic data to ensure that each individual component operates as well as possible. Notable aspects that are evaluated include pose propagation motion models and particle resampling strategies. Synthetic data is also used to validate our initial hypothesis through the use of both a small scale articulated model and a virtual human model. We end our research by applying our tracker to real-world human motion sequences. We use both sequences recorded in front of a green screen and sequences recorded in natural environments to evaluate both silhouette extraction algorithms and tracker performance.

We find that when silhouettes are correctly segmented, it is indeed possible to track pose at high degrees of freedom (37 in our experiments) – a significant advance to the state of the art. While this supports the ability of our approach to generalize better than other state of the art trackers, it is currently limited by segmentation accuracy on standard datasets, making for less favorable comparison. Still, we believe that this works marks a promising approach to tracking complex 3D objects from monocular image sequences which will only improve with better figure/ground separation.

Résumé

Le suivi de postures humaines est un outil utile avec de multiples applications dans une variété de domaines allant du médical, artistique au secteur industriel. Les approches existantes sont souvent difficilement applicable car elles nécessitent une instrumentation complexe et du matériel spécialisé. Dans cette recherche, nous développons une théorie par laquelle un suivi de pose humaine articulée à haut degré de liberté peut être accompli à partir des observations d'un flux d'images monoculaire. Ceci est soutenu par une analyse empirique détaillée de sa mise en oeuvre à travers un vaste ensemble d'expériences avec des données synthétiques, des ensembles de données accessibles au public et avec de vrais videos.

Nous postulons que, bien qu'ambigus, les contours de silhouettes contiennent suffisamment d'informations pour suivre la pose humaine articulée au cours de la plupart des cas présents dans des séquences videos réelles. Nous proposons un cadre hybride basé à la fois sur un système classique de suivi des filtres à particules et sur une approche d'apprentissage en profondeur pour tester notre hypothèse. L'apprentissage en profondeur est utilisé pour la reconnaissance d'état, tandis que le filtre à particules est utilisé pour interpoler et maintenir la cohérence temporelle. Plus formellement, l'information à l'entrée de notre système est une série chronologique d'images provenant d'une caméra monoculaire et l'information à la sortie est la pose d'un modèle humain 3D articulé.

La quantité de données nécessaires pour entrainer un réseau convolutionnel afin d'effectuer le suivi de pose, et la complexité associée à la collecte de ces données, présente un problème important. Pour contourner cela, nous proposons une approche à deux volets. Tout d'abord, nous utilisons des silhouettes au lieu d'image complètes comme intrant sur le réseau, ce qui réduit à la fois la complexité des données d'entrée et rend le système invariant à la texture. La deuxième caractéristique de notre stratégie d'apprentissage est de s'appuyer sur des données synthétiques, obtenues à partir d'un modèle génératif. Ces deux tactiques permettent de générer une grande quantité de données et permet l'apprentissage du modèle. Nous montrons expérimentalement que l'utilisation des données synthétiques est applicable

aux données du monde réel.

Pour le filtre à particules, nous nous appuyons sur une métrique pour comparer les silhouettes des objets et évaluer la validité des poses candidates obtenues à la fois à partir du réseau neuronal et à partir de modèles de mouvement basés sur la physique. Pour sélectionner la métrique la plus appropriée pour une utilisation avec des formes humaines, nous fournissons un examen minutieux des méthodes présentées dans la littérature et présentons une approche systématique pour évaluer leur précision afin de sélectionner la meilleure.

Un autre élément nécessaire à la recherche présentée ici est de déterminer comment enregister et accéder à divers types d'informations temporelles. Nous présentons une nouvelle architecture de base de données, qui permet un accès asynchrone aux données et permet de faire des prédictions à partir de données enregistrées.

Tous les choix effectués dans la mise en oeuvre de notre système sont basés sur des expériences soigneusement présentées sur des données synthétiques afin de garantir que chaque composant fonctionne aussi bien que possible, individuellement. Les aspects notables qui sont évalués comprennent les modèles de mouvement, les modes de propagation de particules, ainsi que les stratégies de ré-échantillonnage des particules. Des données synthétiques sont également utilisées pour valider notre hypothèse initiale en utilisant à la fois un modèle articulé à petite échelle et un modèle humain virtuel. Nous terminons nos recherches en appliquant notre traqueur à de vraies séquences de mouvements humains. Nous utilisons à la fois des séquences enregistrées devant un écran vert et des séquences enregistrées dans des environnements naturels pour évaluer à la fois les algorithmes d'extraction de silhouette et les performances du traqueur.

Nous constatons, lorsque les silhouettes sont correctement segmentées, qu'il est en effet possible de suivre une pose avec un large nombre de degrés de liberté (37 dans nos expériences) – une avancée significative en comparaison avec d'autres approches. Bien que cela démontre la capacité de notre approche à se généraliser mieux que d'autres traqueurs, notre approche est actuellement limitée par la précision de la segmentation de silhouettes, ce qui

est problématique sur les ensembles de données standard, cela rend les comparaisons moins favorables. Néanmoins, nous pensons que ce travail marque une direction prometteuse pour suivre des objets 3D articulés complexes à partir de séquences d'images monoculaires qui ne peut que s'améliorer avec une meilleure extraction de silhouette.

Acknowledgments

I would first like to thank Professor Frank P. Ferrie for his continuous support and valuable guidance. I also need to thank the other members of my thesis committee, Professor Paul Kry and Professor Jeremy Cooperstock, whose great support and insightful comments have tremendously helped in the research process. I would like to thank McGill University as a whole for making this research possible.

Besides my advisors, I would like to thank my fellow lab-mates, Amir Abbas Haji Abolhassani, Andrew Phan, Michael Smith, Yanyan Mu, and Prasun Lala who have helped create a stimulating research environment. I am especially grateful to Prasun Lala for proof-reading and providing insight for various written documentation. I also thank my colleagues from the University of British Columbia and Université Laval, with whom it was a pleasure to work with. I am also thankful to the open source community, which facilitates collaboration and support between people all around the world.

Last but not the least, I would like to express gratitude toward my parents, Micheline St-Martin, and Michel Cormier, for supporting me throughout this research and life in general.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Thesis

The fundamental problem addressed by this research is the inference of complex behavior given noisy, limited, and ambiguous observable data. To expend on this, let us start by defining the issues we encounter. The noise in the measurement results from inherent imperfections of the measuring process itself. These limitations are due to the fact that it is impossible to capture the sum of all information related to the observed process. Both of these problems could be alleviated by either using better sensors or more sensors, but the goal of this research is to cope with these restrictions. The final issue we tackle is ambiguity. Assuming that a system can be modeled, we define ambiguous problems as those where the function mapping from parameter space to measurement space is non-injective, and is not necessarily surjective. What we mean by that is that any point in the parameter space maps to a point in the measurement space. However, not all points in the measurement space necessarily map back to the parameter space, and that points that do map back may not have a unique mapping. This means a given measurement can be produced by multiple different configuration of the parameters.

The main thesis we put forward here is that such a class of problem can be addressed

by the regularization of the incomplete data based on a set of prior models and assumptions rooted in a general understanding of the observed system's context. To do this, we use a model-based approach which reduces the complexity of the system to a finite set of parameters. This model should be accurate enough to allow us to make predictions of future states based on the current and previous states. We can then take the incomplete and ambiguous measurements and compress them into a representation that encodes the parameters we are trying to evaluate, while discarding other parameters we are not concerned with. The problem then turns into a fusion task as we combine our observations and our expectations into a unified representation that allows us to obtain an appropriate estimate of the state of the observed system.

To experimentally test our theory, we choose to develop a high degree of freedom human posture tracking system with image sequences recorded from a single color camera. Human pose tracking is an ideal problem choice as it presents all of the issues discussed above. All cameras present a certain level of noise and the finite resolution presents a limit to the amount of data we can capture. The use of a single camera further limits our perception of the model being tracked and introduces ambiguity due to occlusion. This occlusion can stem from either objects in the scene or self occlusion of the human model itself. Human motion is also quite complex and provides ambiguous cases depending on how it changes with respect to the camera position.

We embed the understanding of context in the tracking task by using established knowledge of physics and priors based on known biomechanical properties of the tracked subjects. We demonstrate that relying on this richer context allows us to reduce the pose tracking space and provide better tracking results. We use a particle filter to fuse the predictions from our model of the system and the limited recorded measurements. To reduce the complexity of the fusion task, we opt to use a generative model that relies on the silhouette of the tracked model. We leverage the flexibility of generating synthetic data to both show that the approach proposed herein is viable and that appropriate synthetic data can generalize to real

world scenarios.

The use of silhouettes is motivated by the desire to reduce the influence of varying texture due to clothing and lighting conditions in order to make the tracking system more easily generalizable. The choice of silhouettes is based on work by Koenderink et al. [1984] who have shown that the visual contour of an object can be used to infer a lot of information about the curvature and surface geometry of the object generating the contour. This framework allows us to demonstrate experimentally that, by adding such contextual information and prior knowledge, it is possible to track complex three dimensional articulated pose from limited silhouette information.

While the research is geared toward the specific application of observing the articulated 3D structure of a human with the use of a single monocular camera, the proposed solution is shown to generalize to other analogous scientific problems through small-scale experimentation on non-human articulated structures. To further show the flexibility of our method, results with other data modalities such as stereo vision systems and depth sensing cameras are also presented. We use the small scale model of Chapter 4 to show that the addition of one or more additional cameras greatly improves the tracking results. Section 2.3 shows how depth information can be integrated into our tracking framework.

1.2 Background

The research topic discussed herein was inspired by a project titled "Collaborative, Human-focused, Assistive Robotics for Manufacturing" (CHARM), funded by General Motors Research as part of the NSERC CRD program. This project consisted of the development of a robotic assistant for human workers in an assembly line context. A photo representing the type of environment considered by CHARM is presented if Figure 1.1. One important piece of this system was the visual tracking of the human worker. The project began with the use of an expensive and intrusive VICON system, that involves an array of infra-red cameras

detecting and tracking reflective markers placed on the subject being tracked. The second development iteration of the project saw the VICON system replaced by a much cheaper and non-intrusive array of Microsoft Kinect sensors [Phan and Ferrie, 2015]. This second iteration led to the publication of a paper describing the specific system [Cormier et al., 2015]. Figure 1.2 shows how the tracking data was displayed back to the human worker. At the highest level, the motivation is to further reduce the cost and complexity of the tracking system by determining whether the array of Kinect sensors can be replaced by one or more standard color cameras. On a more scientific level, we show that we can trade measurement complexity for model complexity and thus that a richer model that includes contextual information can compensate for lower resolution measurements. There exists a broad range of potential applications for simple, cheap, and non-intrusive human motion capture. These range from safety critical human-robot interaction to artistic purposes in visual effects for movies and games. A number of publications resulted from the CHARM research project [Cormier and Ferrie, 2016; Cormier et al., 2015; Gleeson et al., 2015, 2013a,b; Haddadi et al., 2013; Hart et al., 2014, 2018; Phan and Ferrie, 2015], more information can be found at `http://charm.sites.olt.ubc.ca/`

1.3 Contributions

Investigating the topics related to the problem stated earlier allows us to explore the state of the art from a more critical and focused point of view. This enables us to determine some of the shortcomings of the currently employed methodologies and make improvements in certain areas. We will now discuss some of these contributions.

The first of these resides in a computational framework for the inference process that can combine predictions generated from a model of the system with the recorded data from the real system. While this topic has been investigated by others, we feel that it still lacks a solution appropriate to the tracking problem outlined above, which involves both high

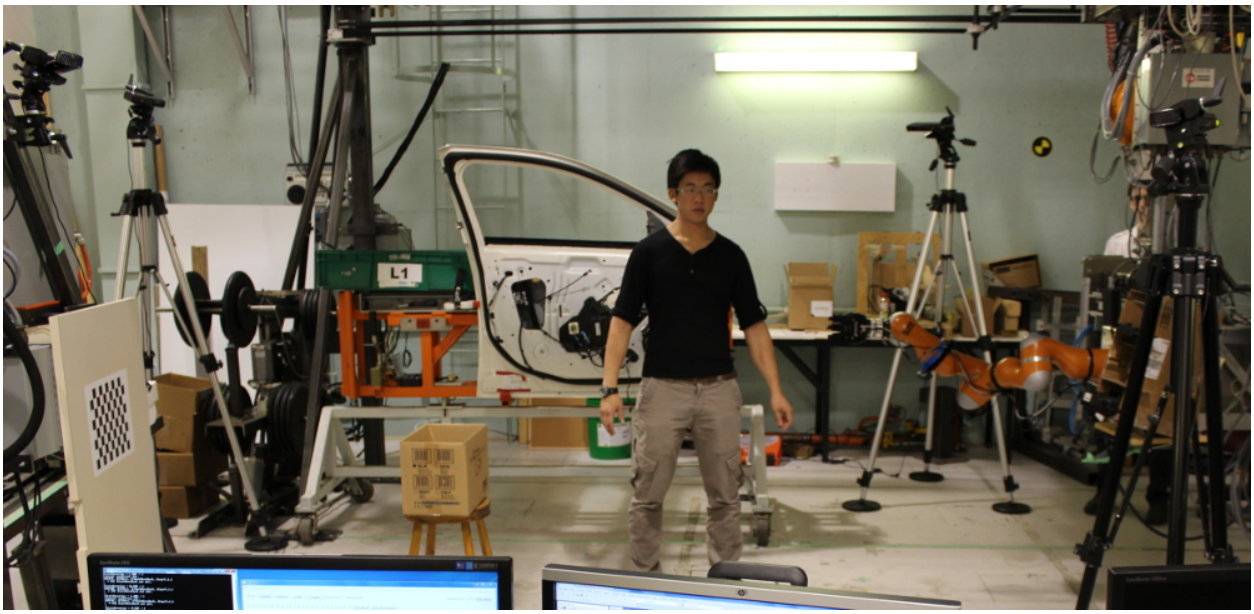


Figure 1.1: Photo from a CHARM project experiment showing the type of industrial environment considered.

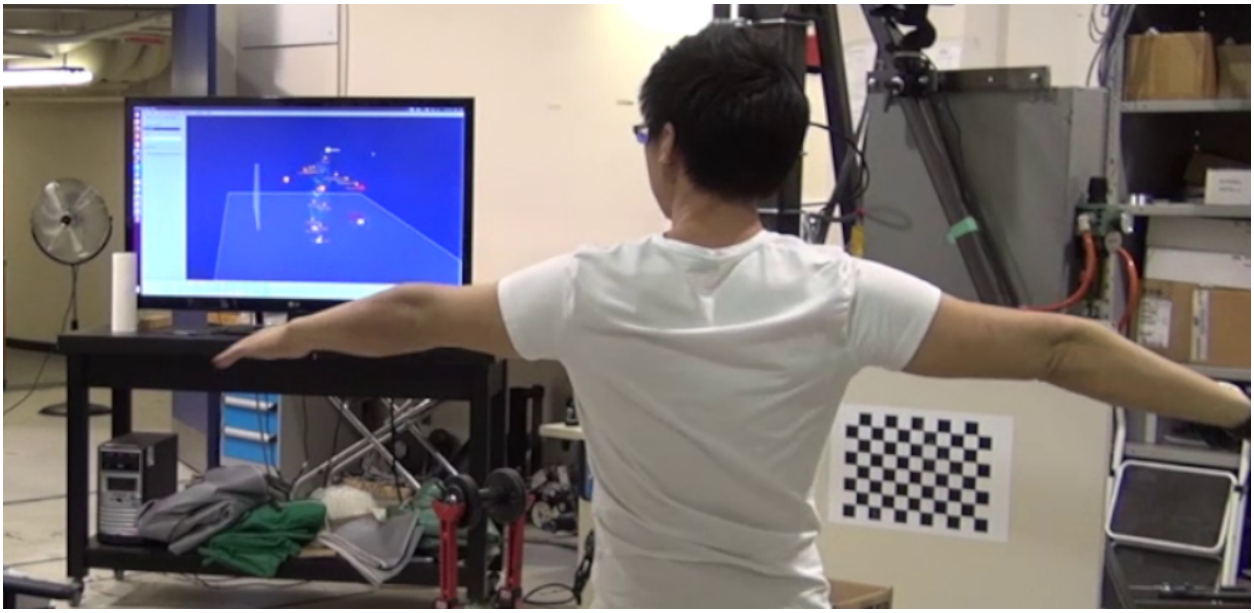


Figure 1.2: Photo from a CHARM project experiment showing .

dimensional data and types of data that are not necessarily directly relatable. We present an hybrid method based on particle filtering and machine learning to fuse image data and modeled data. This is discussed in Section 2.4 and Chapter 5.

The second contribution involves determining to what extent low dimensional descriptors such as silhouettes can represent complex deformable three dimensional shapes with high number of degrees of freedom. We explore this area by providing a survey of methods used in the literature to compare the similarity of silhouettes and determine how well they correlate to changes in the pose of the model generating the silhouette. We detail a set of experiments, and present their results in Section 2.3.

The third and arguably most practical contribution stems from finding ways to maintain a consistent representation of the state of the modeled system over time. This contribution resides in the development of the Situational Awareness DataBase (SADB) system. This system was originally developed as part of the CHARM project to store, maintain, and distribute contextual information that can then be used by other components of the system or by other networked systems that require the human pose or other recorded pieces of information. This system allows the simple organization of multiple types of time-coded data, and provides a straightforward means to retrieve values at any desired timestamp by proposing a novel data access framework that includes interpolation and extrapolation.

Another contribution is the use of deformable 3D mesh models, in contrast to most methods in the classical literature that rely on a combination of cylinders and spheres to model humans. The use of parameterized photo-realistic 3D models based on biomechanical data provides a more accurate prediction of shape and appearance of both a human and its motion. The other advantage of using meshes is the possibility of using 3D scanning to better model the humans being tracked.

An additional significant contribution is that we show how traditional priors can be replaced by synthetic data throughout the process of developing and evaluating our tracker. Synthetic data rendered from our human model is first be used to determine the best metric

to compare silhouettes. Synthetic data is also used to systematically tune the numerous parameters of our tracker to maximize accuracy. It is also used to train a convolutional neural network (CNN) as part of the hybrid particle filter human tracking approach and is used to evaluate the performance of our complete tracker.

The final contribution and potentially most significant from a practical point of view is the development of the complete tracking system itself, which, as shown in Chapters 4 and 5, can be used to track a wide variety of model types.

1.4 Overview

As a large part of this research is experimental in nature, a key part of the work resides in devising an experimental test bed on which to test and prove our ideas and approaches and compare them to the state of the art. The simplified block diagram of the framework we implemented is shown in Figure 1.3. As our research focuses on each of these components, the block diagram also presents a good overview of how the research is structured. The filtering block is drawn with a dashed outline as it is not necessarily a component in itself, but rather integrated into both feature extraction and pose estimation.

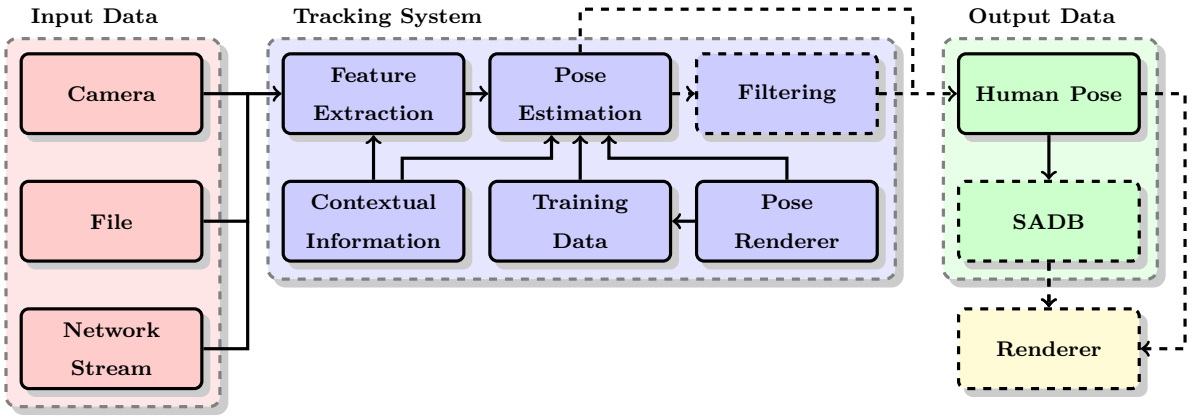


Figure 1.3: Block diagram of the experimental system

The main flow of the tracking process moves from left to right, from the input data on

the left side of the block diagram to the renderer at the bottom right, which allows us to visualize the results. The bulk of the research is focused on the blocks in the blue rectangle.

The topic of feature extraction is explored in depth in Section 2.1. As stated earlier, we found that silhouettes provide a viable feature for the task at hand. Feature extraction thus consists of segmenting input images into a foreground silhouette and a background mask. A large part of Section 2.1 is used to evaluate different metrics to compare silhouettes. This, in turn, enables us to demonstrate that silhouettes do encode the articulated pose of the skeleton. We also describe different methods to extract silhouettes from image sequences as well as the common issues encountered with these methods.

Pose estimation consists of looking at the extracted features, predictions from our model, as well as contextual information to determine what the most likely pose of the model is. This is both a data fusion task, as different types of data are used, and an optimization problem to find the pose. The most obvious way to tackle both aspects at once is through the use of a filtering approach. Two types of filters that are commonly used for similar applications are the Kalman filter [Kalman, 1960] and the particle filter [Salmond and Gordon, 2005]. Both of these approaches require us to map the high dimensional input data to a lower dimensional space defined by the parameters of the system's model, in our case, the pose vector. Filtering allows us to integrate the results over time, which decreases ambiguity. A discussion of other approaches is presented in Section 1.5. The choice of particle filtering is further discussed in Section 2.4.

The pose renderer is a 3D rendering system that renders deformable mesh models given a pose vector. It is used both to evaluate candidate poses from the filter as well as generate training data for CNN-based approaches that are discussed in Section 2.4, as well as in Chapter 5.

Contextual information is required to determine ways in which to use knowledge of the environment to generate priors for the model parameters. Based on a general understanding of the physics acting within the world from Newton's laws of motion [Newton et al., 1833] and

the kinematic equations presented by Whittaker [1904], we can write a set of equations that can be used as a time evolution model of the system; these will be described in Section 2.4. By starting with information recorded about the environment, a simulation approach can be employed to extrapolate and generate expected values and better interpret the state of the world by making predictions. This results in motion models that can be used by the particle filter. This means that starting from limited knowledge, we can predict the next physically plausible states of the environment, including the motion of humans within the observed world. These motion models are presented in Section 2.4. The contextual information also includes models of humans. These models consist of both a geometric mesh model, required to render virtual silhouettes, and a skeletal model used to both deform the mesh model and compute the output of the pose estimation. Geometric models for both human and non-human subjects are described in Chapters 5 and 4, respectively. Other useful sources of contextual information are camera calibration and background models. The SADB block represents the Situational Awareness DataBase system that was previously mentioned in the contributions section.

A complete system is obtained by integrating the components into a complete tracking framework, which will be presented in Chapter 4. In Chapter 5, this framework is tested through the use of standard datasets to obtain meaningful comparison to the state of the art. The completed framework also enables us to go further into specific research areas by enabling more precise experimentation. We have also devised means to experiment and determine, from a more practical standpoint, the limits of each component individually.

As all these experiments result in a large number of videos and plots, only selected results are included in this document, while most are not included because of space constraints. The complete results can be found on a companion website located at `http://cim.mcgill.ca/~olivier/thesis/`.

To summarize, we find that while ambiguity is inevitable, the shape of the occluding contour of a deformable object can be used to estimate the articulated pose of the object.

The use of an appropriate metric and the introduction of contextual information allows us to find instantaneous hypotheses. Integrating these hypotheses over time enables us to converge to the most likely solution. In cases where ambiguity is unresolvable, our model can provide a physically meaningful temporary solution until ambiguity is resolved and we can converge back to a better solution.

1.5 Related Literature

As the topic of articulated human posture tracking involves a variety of sub-areas, we have segmented the general topic into three discrete components to facilitate the review [Moeslund et al., 2006]: *initialization*, *tracking* and *pose estimation*. Some authors also describe a fourth component [Moeslund et al., 2006], which operates at a higher level for functions such as identity recognition or gesture detection. The current research considers some of the implications of initialization, and focuses on the tracking and pose estimation aspects. As identity recognition and gesture detection are separate topics, relying more on a semantic understanding of the scene, we choose not to cover them in the current research.

The global surveys of human pose tracking and motion capture presented by Moeslund and Granum [2001] and Moeslund et al. [2006] detail many different approaches presented in the literature. Two classes of methods are presented: active and passive sensing. Active sensing involves the instrumentation of the tracked subjects by requiring them to wear sensors [Andrews et al., 2016], which may be problematic, impractical, or even impossible in real-life situations. The current thesis thus only focuses on passive sensing. Within the passive tracking literature, a large number of papers published in the past few years are aimed at cameras with depth sensors (for example, the approach published by Shotton et al. [2013]). This is mostly due to the increasing availability of devices like the Microsoft Kinect [Lieberknecht et al., 2011; Fallon et al., 2012; Shotton et al., 2013; Wang et al., 2013]. While these new devices present a good approach by providing much denser measurement than in-

tensity cameras, they do not present a better way to make use of contextual information. As stated earlier, we determine the feasibility of a human posture tracking with a monocular camera configuration by putting more emphasis on methods which use similar capture setups.

The initialization component consists of determining the best way to start the system in a valid state, in order to allow successful tracking in subsequent frames. The first thing that needs to be initialized is a model of the subject, which includes the kinematic structure, usually in the form of skeletal model, and appearance model, which usually encompasses both the shape and texture. Most papers define the pose (or state) of a skeletal structure as either the angles of each bone with respect to their parent or as the absolute position and orientation of each bone with respect to a global reference frame. The number of degrees of freedom (DOF) comprising the skeletons varies greatly between publications. It ranges from two [Isard and MacCormick, 2001], when assuming that the human is standing on the ground and tracking only the position on the floor, to 40 degrees of freedom [Szczuko, 2014] when tracking a complex skeleton with a large number of rigid bodies. The goal here is to find the minimum number of DOF to reduce the dimensionality of the search space while maintaining a skeleton capable of matching the motion of a real human skeleton. Scale is also an important factor. For example, the hand tracker presented by Wang et al. [2013] describes a 28 DOF model of a hand. While this is valid for tracking in a video sequence where only a hand is present, modeling a hand to this level of detail in a full body tracker is not reasonable as the pose space would be prohibitively large and the resolution of the input data would most likely not convey enough information to recover pose properly. The physical characteristics of the rigid bodies in the skeleton are usually assumed to be known and static over the tracking period. These characteristics include the length, the mass, the inertia tensors, and the joint angle limits. The exact values selected are usually based on biomechanics [del Rincón et al., 2011] or selected manually. To model the shape of humans, most methods rely on the combination of simple geometric shapes such as spheres and cylinders

[Isard and MacCormick, 2001; Klinger and Arens, 2009; Vondrak et al., 2013; Sedai et al., 2013; Szczuko, 2014]. Other methods rely on either generic deformable 3D mesh models or mesh models acquired from 3D scanning [Stoll et al., 2010]. The final major aspect of initialization is to determine the initial pose. Most of the methods assume that the initial pose of the skeleton is known a-priori. Iterative closest point (ICP) can also be used to find a set of initial matches between a rendered model and the first image frame [Wang et al., 2013]. While most likely not unique due to ambiguity, these matches provide an adequate set of candidates to initialize trackers.

The tracking part is where most of the low level computer vision algorithms are required. As with most computer vision applications, the first stage of the tracking systems presented in the various surveyed publications consist of a feature extraction process. While CNN based tracking methods use learning of convolution kernels to determine appropriate features, most of non-CNN methods rely on a mixture of silhouette [Güdükbay et al., 2013; Klinger and Arens, 2009; Vondrak et al., 2013; Brubaker et al., 2010; Sedai et al., 2013], intensity edges, and texture descriptors [Duff et al., 2011; del Rincón et al., 2011]. The hand tracking algorithm presented by Wang et al. [2013] combines the silhouette, a histogram based color model, as well as edges into a single error measure. One of the reviewed algorithms [Szczuko, 2014] applies a face detection algorithm to distinguish the front from the back of a human and thus remove a source of ambiguity. As stated earlier, silhouettes, while being a simple feature, encode a large amount of information about the shape of the generating model [Koenderink et al., 1984]. This is most likely the reason why silhouettes seem to be the most common feature used in the literature; more emphasis is put on this type of feature. Silhouette extraction consists of segmenting the silhouette of a moving object over a background, which results in a binary image.

Segmentation algorithms can be classified into three categories [Collins et al., 2000]: "temporal differencing" methods, which aim to select pixels that change between frames, "background subtraction" methods, which segment the image based on the difference be-

tween the current frame and a learned background model, and "optical flow" methods, that rely on detecting the continuous motion of objects over a sequence of frames. The algorithm described by Collins et al. [2000], used for the silhouette extraction of Güdükbay et al. [2013] and Brubaker et al. [2010], consists of a hybrid approach combining a static background subtraction algorithm with temporal smoothing. A slightly more complex background segmentation algorithm is presented by Stauffer and Grimson [1999], where the background model is learned as a mixture of Gaussian components, which solves problems caused by periodic temporal variation in a scene and allows the model to update itself continuously. A similar type of segmentation algorithm presented by Isard and MacCormick [2001] proposes the computation of the log-likelihood ratio of a pixel belonging to the foreground or the background. This provides a probability map of how likely a pixel belongs to the foreground. The segmentation method proposed by Elgammal et al. [2002] also provides a foreground probability map. A compression step can be added to the segmentation stage [Sedai et al., 2013] to reduce the amount of data and extract distinct features of the silhouette. A discrete cosine transform is used to compress the data and only the most influential components are retained. This method is somewhat similar to what is used by the JPEG (Joint Photographic Experts Group) image compression algorithm. A simpler silhouette segmentation method is presented by Shotton et al. [2013], where a depth camera is used to segment foreground objects based on a depth threshold. The mixture of Gaussian components methods described above are also stated to work well when the depth value is added to the measurement vector [Stauffer and Grimson, 1999]. The only downside to depth-based methods is that they require more specialized input devices. Szczuko [2014] also proposes a simplified silhouette extraction method by engineering the environment. A green screen background is used to allow simple chroma keying segmentation. The idea proposed by del Rincón et al. [2011] to minimize the computational cost of silhouette extraction consists of using a Kalman filter to track a bounding box around the region of the image containing the human. This reduces the total number of pixels that need to be considered. There is not

yet a definitive solution to the problem of foreground segmentation. More recent state of the art approaches employ convolutional neural networks to segment images, but still require human intervention [Song et al., 2018].

Finally, the pose estimation aspect of human posture tracking consists of using the measurements from the tracking part as well as a set of constraints to extract the most likely pose. Most of the papers surveyed use a particle filter [Salmond and Gordon, 2005; Thrun, 2002] approach to track the 3D posture over time. This technique has the advantage of explicitly requiring a temporal evolution model to predict how the pose evolves over time as well as the ability to consider multiple potential solutions, which makes the tracker more robust to perturbations. Three basic types of motion priors are used within the pose tracking literature. The first proposes the assumption that the evolution of a human's pose over time can be modeled as a Gaussian diffusion process [Sedai et al., 2013]. While this is clearly not a valid global assumption, the proponents of this approach argue that it is valid locally (over short intervals of time). The second approach uses a learned model based on motion capture data to create a pose transition probability table [Klinger and Arens, 2009; Vondrak et al., 2013]. The last type of prior is derived from a physical simulation to ensure physically valid pose transitions [Vondrak et al., 2008; Wang et al., 2013]. While each of these methods may stand on their own, most papers use a somewhat hybrid approach to predict pose transformation over time. Regardless of the selected motion prior, the computationally expensive part of the problem resides in the particle likelihood estimation [Vondrak et al., 2008]. Other published approaches to tracking the solution rely on a grid search followed by gradient descent to find the pose [Plänkers and Fua, 2003]. This approach works well with a lower number of dimensions, but the search space becomes intractable for tracking problems with a larger number of dimensions. We take a closer look at such an approach in Chapter 4. Other authors rely on adding stochastic characteristics to a standard particle filter [Mitchelson and Hilton, 2003] to track problems with larger dimensionality. To improve the accuracy of the tracking results, a method proposed by Deutscher et al. [2000] uses simulated

annealing to drive the solution toward a global solution. A similar annealing approach is employed by Szczuko [2014]. Classical human tracking papers by Wren and Pentland [1998, 1999] propose a method of tracking that relies on the detection of blobs and the use of physically based constraints. Human behavior is also modeled to provide tracking constraints. In contrast to the filter-based methods, Shotton et al. [2013] propose a system which determines the pose independently in each frame by using randomized decision forests to learn a mapping from the input to the pose space. This method has the advantage of being computationally efficient and attains a performance of approximately 200hz, but requires a large amount of training data to initialize the classifiers. Another novel tracking approach is presented by Belagiannis et al. [2014] where the human pose is not represented as a skeleton. The pose is presented as a conditional random field, where each bone is represented by an independent variable. Random sample consensus (RANSAC) can also be used as a means to track human pose over an image sequence [Duff et al., 2010]. Physics simulation can be used alongside RANSAC as a way to determine if a pose candidate is an inlier or an outlier to a simulated motion trajectory. The most common tracking problem mentioned in the literature is caused by non-rigid deformations of the model due to certain types of clothes (and hair). Stoll et al. [2010] describe a system capable of cloth simulation to better predict the deformation of the silhouette of the tracked person. This method requires a very precise skeletal model as well as a detailed 3D mesh model of the subject. While interesting, this is somewhat beyond the scope of the current thesis, where tight fitting clothes are assumed. The major downside of most presented tracking algorithms is the high execution cost stated to be as high as a few minutes per frame [Wang et al., 2013].

The idea of using physics simulation as a prior for computer vision tracking has been proposed since the nineties [Metaxas and Terzopoulos, 1993], where it was used as the prediction model for a Kalman filter. Another use of simulation was presented even earlier by Pentland and Williams [1989], who used simulation as a means to generate physically meaningful animations based on a simulated environment by using modal analysis. This method, closely

related to the finite element method of simulation, used multiple levels of representation to distort the shape of a deformable model. Taking these ideas a step further, a system capable of generating physically meaningful animations based on a simulated environment and simulated agents was presented by Tu and Terzopoulos [1994]. Most publications considered in this review rely on commercially available simulation frameworks, but some rely on manually solving a restricted simulation system over time [Brubaker et al., 2009, 2010]. A good introduction to physical simulation theory and implementation is presented by Baraff [2001]. The most commonly used engines are Bullet Physics [Coumans et al., 2013], NVidia PhysX [Nvidia, 2001], Open Dynamics Engine (ODE) [Smith et al., 2005], and the Crisis Physics Library [Vondrák, 2005]. The comparison of multiple simulation engines proposed by Erez et al. [2015] shows that Bullet and PhysX have the best performance, as of this writing, of the reviewed engines, depending on the test cases. It must be noted that Erez et al. [2015] show that simulation engines are more or less interchangeable, without needing to rewrite a large part of the project. In order to maintain the pose of the skeleton in the physics simulator synchronized with the recovered pose, "virtual" forces need to be applied to the rigid bodies composing the simulated skeleton. Some algorithms use a proportional integral derivative (PID) controller to regulate those forces [Wang et al., 2013; Vondrak et al., 2008, 2013].

An alternative, and more recent, approach to human pose tracking is machine learning with convolutional neural networks (CNN). More commonly known as deep learning, it can be seen as either a completely independent approach to pose tracking, or as a potential solution to any component of traditional approaches. The deep-learning methods are formulated as either regression problems [Toshev and Szegedy, 2014] where the output is the pose vector, or as classification problems where, often, the 2D locations of joints are expected to be found [Wei et al., 2016]. Many methods also formulate the problem as a combination of regression and classification by separating the tracker into different stages working toward the solution [Moreno-Noguer, 2017; Mehta et al., 2017]. Park et al. [2016] present a method that is

somewhat in between by using a single network to extract both 2D and 3D joint positions at the same time. This network consists of a few convolutional layers, common for both 2D and 3D, followed by 2 sets of fully connected layers for the 2D and 3D tasks. While a single network is used, the 3D estimation fully connected layers are also connected to the output of the 2D pose estimator, which makes it an hybrid 2 stage framework. Similarly, most of the published approaches rely on 2D joint location as a first step in the 3D pose inference problem. The reason behind the common usage of 2D joints in recent literature is due to the existence of off-the-shelf solutions [Wei et al., 2016]. Some approaches then rely on neural networks to project the 2D joint positions into a set of 3D joint positions by representing joint positions, both in 2D and 3D, as Euclidean distance matrices [Moreno-Noguer, 2017]. This reduces the effect of scale by representing points in relation to other points.

As an extension to 2D joint positions, the monocap tracker [Zhou et al., 2018] relies on a CNN to extract heatmaps of possible positions of the joints in the 2D image. 3D poses are then built as a weighted combination of 3D poses from a learnt dictionary using an expectation maximization optimization process. A similar approach is presented by Zhou et al. [2016] where 2D joint location heatmaps are used to find 3D pose, represented as a mixture of learnt basis poses. Likewise, the Vnect tracker proposed by Mehta et al. [2017] uses a CNN trained on annotated data from a variety of sources, including the Human3.6M dataset [Ionescu et al., 2014], to extract the 2.5D position of joints in the image as a heatmap, by trying to also estimate the depth. A second CNN is used to fit a 3D kinematic skeleton to the extracted 2.5D joint locations. Temporal smoothing is also applied to obtain a temporally stable solution. The main advantage of using a heatmap-based representation for 2D joint location is that these heatmap images can be used directly as the input of a second CNN to find the 3D position [Mehta et al., 2017], instead of a fully-connected network. The heatmap-based approach can be taken a step further by representing the 3D pose as a heatmap as well [Moreno-Noguer, 2017]. This is done by discretizing the space into fixed-size voxels and using deconvolutional layers to estimate likelihood of each joints being located into each voxel.

Other features such as depth [Du et al., 2016] and motion [Tekin et al., 2016] are also used as sources for CNN-based pose extractions. Pre-existing heightmap extraction algorithms can be used to augment the rgb image from a monocular camera into and rgbd image [Du et al., 2016]. This rgbd image is then used as the input to the 3D estimation CNN. Raw pixels can also be used directly as the input of the CNN. For example, the method proposed by Grinciunaite et al. [2016] merged a sequence of rgb frames into a high dimensional block, which is fed into the CNN. The hope is that by using multiple frames, temporal relations are also learned by a standard CNN without the complexity of recurrent neural networks (RNN). Rogez and Schmid [2016] demonstrate that synthetic data can be used to train a CNN and that such a CNN can generalize to real data by replacing the background of images containing a human in a known pose.

An even simpler framework is presented by Chen and Ramanan [2017], where 3D joint position ground truth from motion capture datasets is used with an off-the-shelf 2D joint detector [Bogo et al., 2016] to record correspondences between 2D and 3D. Once enough relations are recorded, the inference problem is formulated as a simple nearest neighbor problem.

The common thread through or most of the deep learning literature is that better results are obtained when going from some feature to 3d pose rather than trying to learn 3D pose from raw pixel values. While most published works rely on 2D joint positions, other features have been successfully used. This is a good indication that using silhouettes as a feature can yield interesting results. By first showing that variation of occluding contours are directly correlated to changes in the articulated pose of the object generating them, we show that silhouettes encode the underlying state of the object. This makes them prime candidates to be used as the input of a CNN. The main advantage of our approach compared to purely machine learning approaches is that combining a CNN with a particle filter enables us to explicitly keep track of multiple hypotheses, which helps resolve ambiguity over time. The particle filter also allows us to integrate more understanding of the context by using

assumptions of how objects move. We can change some of these assumptions based on context without the need to retrain a CNN for different conditions.

Other topics related to deep learning that are explored in this thesis include transfer learning [Pan and Yang, 2010], which can enable the usage of pre-trained network [Simonyan and Zisserman, 2014; Szegedy et al., 2017] as a basis for the task we are interested in. To properly use candidates from a CNN for the particle filter, we need a way to obtain a set of plausible candidates instead of a single result. The simplest way to obtain such a set of particles is to add a Gaussian component to the result of the CNN, thus allowing the particle filter to characterize the shape of the error surface around the predicted result. This, however, does not solve the ambiguity problem. In order to obtain different candidates directly from the network and get greater variety which may allow us to resolve ambiguity, we can use "dropout as a Bayesian approximation" [Gal and Ghahramani, 2016] by retaining the dropout layers when predicting results. A third way is to run the trained CNN on slightly distorted variations of the input silhouettes.

To evaluate the performance of the system we are proposing, and be able to compare our results with those of other authors, we need to quantify our results by using the same metrics as those used in the literature. Most papers with quantitative results run their algorithms against one of four public datasets, the HumanEVA dataset [Sigal and Black, 2006], the Human3.6m dataset [Ionescu et al., 2014], the KTH Multiview football videos dataset [Schuldt et al., 2004], or the silhouette database from Vlasic et al. [2008]. The HumanEVA and Human3.6m datasets seem to be the most widely used and are probably the most interesting for the current thesis as they consist of a series of video scenes where human subjects move within a static room, which simplifies segmentation. Furthermore, Human3.6m is of slightly greater interest as it includes segmentation sequences, which may enable us to test the actual tracker without the need for foreground-background segmentation. We show that the provided segmentation is similar in apprearance to what we extract from the HumanEVA dataset. The video sequences of all of these sets are accompanied by motion tracking data

that can be used as ground truth to quantify the error of the tracking systems. The papers that do not rely on the aforementioned datasets usually require specific measurement modalities that may not be present in those datasets. This makes comparing the result of these algorithms with others harder. The recently released McGill-Reparti Artificial Perception Database [Phan, 2015] provides synchronized depth and intensity images from four different vantage points as well as Vicon marker locations for multiple sequences containing one or more humans. Another evaluation approach that we use is to render synthetic video sequences to test the trackers. This method, presented by Shotton et al. [2013], allows direct comparison between the original pose used to render the scene and the recovered pose from the tracker. Szczuko [2014] proposes an error metric that consists of computing the sum of squared differences between the pose vector from the tracker and the ground truth pose vector. This approach is problematic as the ground truth needs to have the exact same format as the tracker pose vector. A more practical error metric proposed by Sigal and Black [2006] consists of computing the average Euclidean distance between a set of points located on the human model and the position of physical markers measured by the motion capture system. This error metric provides a measurement distance between a recovered pose and a measured pose in a single frame. Using a moving average over a video sequence provides a way to both measure error and visualize how the error evolves over time. The standard deviation of the instantaneous error can also be computed over a sequence of frames. This metric is interesting because it is independent of the underlying model used for the tracking. For this reason, it seems that most recent papers use it [Moreno-Noguer, 2017; Zhou et al., 2018].

We begin our work by following classical computer vision approaches. Through a set of carefully designed experiments, we demonstrate cases where these approaches succeed as well as cases where they fail. We then integrate deep convolutional neural network techniques to replace certain component algorithms, such as the propagation of particles in our particle filter. Similar to approaches that use joint position heat maps as the input to the CNN used

for 3D pose extraction, we feed the silhouette of the model being tracked as the input to the CNN. This work results in a tracking method that blends traditional approaches and deep learning. As stated in the literature [Yasin et al., 2016], one of the hardest parts for human pose estimation is obtaining accurate image data with associated 3D pose. To resolve this issue, we use synthetic data throughout our work to both train and experiment. Ultimately, we compare our tracker to state of the art trackers by using standard datasets.

1.6 Thesis Organization

The thesis is organized in six chapters. Chapter 1 contains the introduction, which contains everything up to this point. This includes the problem statement, the thesis, a discussion of the inspiration for the research, a description of the contributions of our research, an overview of the work done, and the relation of our research with the literature.

Chapter 2 provides a more technical discussion of the sub-topics involved in our research. We start by looking at how features are extracted from images in Section 2.1. Section 2.3 then describes how we select an appropriate metric to compare these features. Finally Section 2.4 details the tracking algorithm by presenting the motion priors, resampling strategies, and methods to compute the results.

The third chapter presents an overview of the technical details of the implementation, a description of the content and purpose of the companion website, and a discussion of how we handle the different types of data involved in our complete experimental framework.

The next two chapters describe the experiments we use to evaluate our approach and report their results. First, Chapter 4 uses purely synthetic, non human data, to both demonstrate validity of our proposed tracking algorithm and tune various parameters by observing their influence on the performance of the global system. Chapter 5 then applies our approach to human data to show its limits. Chapter 5 is also where we introduce machine learning approaches to overcome certain of those limits.

Finally, Chapter 6 presents a summary of the results in Section 6 and provides a roadmap for future work that may improve the performance of our tracker in Section 6.1.

Chapter 2

Theory & Methodology

This chapter details the main components of the research involved in the implementation of a pose tracking framework. We start by detailing the type of features we use and how we extract them from the input images. We then determine how well these features correlate to the information we want to extract and explore different metrics that can be used to evaluate this correlation. We then investigate how these measurements can be integrated over time to estimate the state of the observed system. We end this chapter with a discussion of how all of data we are using can be represented.

2.1 Feature Extraction

As stated in the introduction, we opt to rely on silhouette as the main feature. There are multiple reasons and assumptions that led to the choice of this feature.

The first reason is based on the fact that a silhouette is the projection of a 3D object on a 2D plane. This projection is usually a non-injective surjective function, that is, with a set of camera parameters a 3D shape will cast a known silhouette given, but a given silhouette can be generated by multiple 3D shapes. However, there is a direct correlation between the 2D projection and the 3D shape of the object. This is why shape from silhouette algorithms [Laurentini, 1994] can be used to estimate the "visual hull" of an object from multiple views

of the silhouette. This visual hull is the intersection of the projection cones between the 3D object and their 2D silhouettes. Given enough viewpoints, this visual hull becomes similar to a piecewise convex hull of the 3D model. We thus make the assumption that the silhouette should encode enough information about the shape of humans to enable us to make predictions based on it. We will test this assumption in Section 2.3.

The second assumption is that the use of silhouettes removes ambiguity caused by variations in textures and colors. The information we are interested in is the pose. While this cannot be directly observed, we posit that the occluding contour of the model contains more information about its pose than texture does.

The last reason for selecting the silhouettes as a feature is a practical one. We choose silhouettes because there are multiple well known algorithms and methods to extract them from images of different types. This section will discuss some of these algorithms.

The selection and implementation of the silhouette extraction step is closely related to the results of the experiments that are presented in Section 2.3. A high level overview of the silhouette extraction process is shown in Figure 2.1.

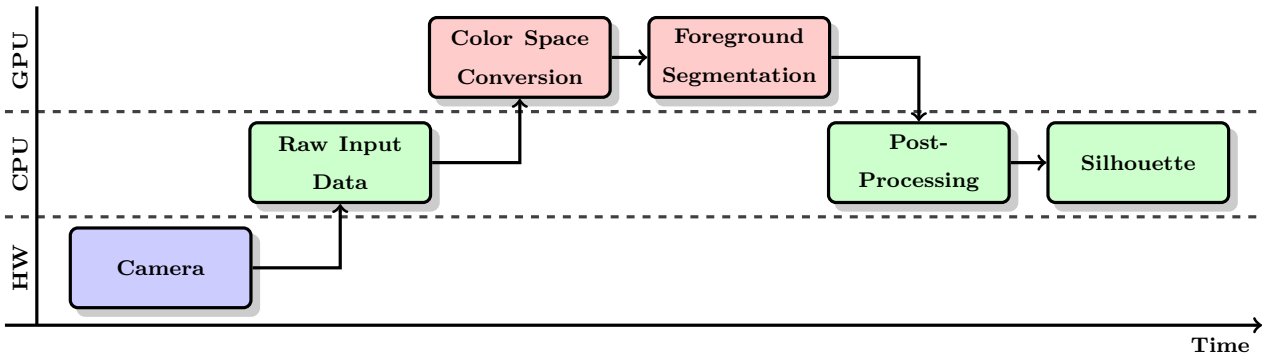


Figure 2.1: Overview of the silhouette extraction process

The foreground segmentation step relies on a learned background model consisting of a Gaussian mixture model for each pixel [Stauffer and Grimson, 1999], which provides a binary segmentation image. The post-processing stage is where we extract the actual silhouette. As we are evaluating metrics that rely on a chain-coded representation of object shape in Section

2.3, we also need to compute a chaincode representation of the silhouette. A simple example of a segmented binary image is shown in Figure 2.2(a). A labeling algorithm is then used to find individual blobs in the images, as colored in Figure 2.2(b). Depending on the type of background, we grow the foreground region by a small number of pixels to merge foreground regions close to one another and ensure that the complete silhouette is detected properly. The silhouette is extracted by finding the largest contiguous foreground region and pruning the other regions (Figure 2.2c). The chain-code is computed by first selecting a boundary pixel, as seen colored red in Figure 2.2(d). We arbitrarily choose to use the pixel closest to the top-right corner of the image as the start of the chain-code. Traveling clockwise along the border, as shown in 2.2(e), provides the chain-code as a linked list of border pixel coordinates.

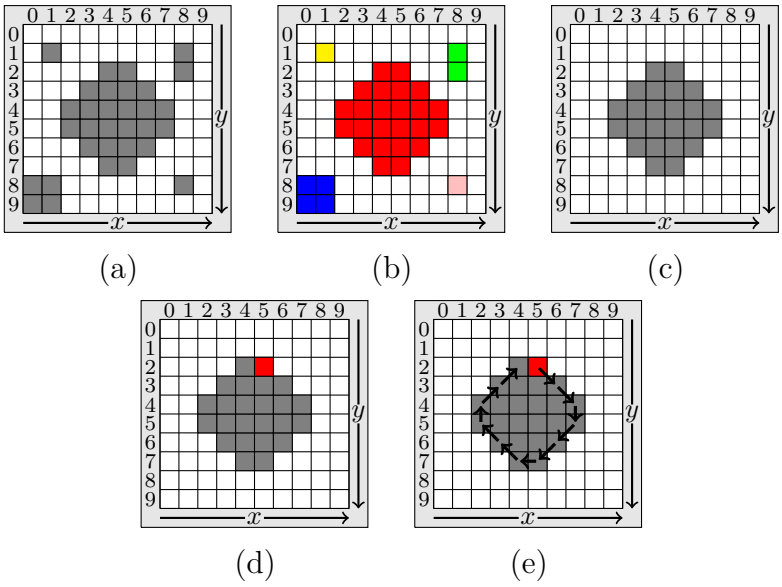
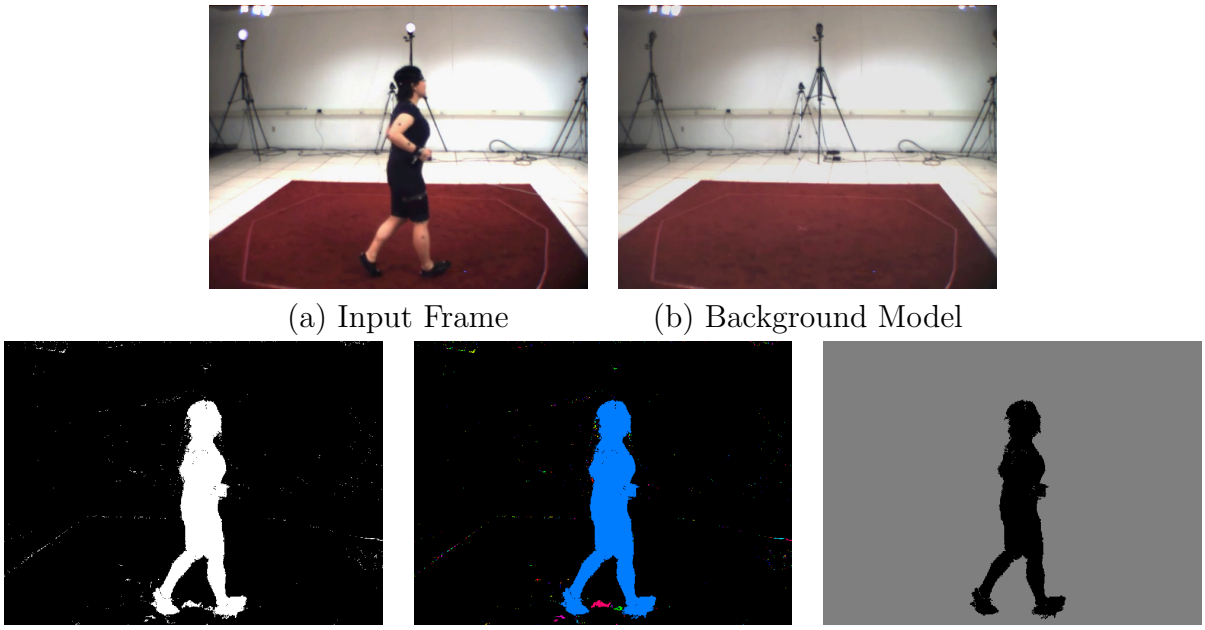


Figure 2.2: Example Binary image (a) being labeled (b), then pruned (c). Given a start pixel (d), the chain code representation of the outline is computed (e).

Figure 2.3 provides a example of the silhouette extraction process on a frame from the HumanEVA dataset [Sigal and Black, 2006]. Given a captured input frame (Figure 2.3(a)), the background model (Figure 2.3(b)) is used to label pixels as either foreground or background and obtain a binary foreground image (Figure 2.3(c)). A labeling algorithm (Figure 2.3(d)) is then used to find and extract the largest foreground blob (Figure 2.3(e)).



(a) Input Frame (b) Background Model

(c) Segmentation (d) Labeling (e) Output

Figure 2.3: Example feature extraction with frame 100 of the HumanEVA [Sigal and Black, 2006] Jog 2 sequence

While the algorithm described earlier performs adequately in most cases, there are some issues that affect certain frames. Figure 2.4 provides sample frames from the HumanEVA dataset where three common issues are encountered. Shadows, as seen in Figure 2.4(a), are a very important issue, depending on the lighting configuration and can corrupt parts of the silhouette as they are usually detected as foreground. Figure 2.4(b) shows a frame where a silhouette is not completely extracted, resulting in a missing limb. This occurs when a region of pixels in the tracked subject is similar enough to the background model and are thus not detected as foreground. This causes a hole in the silhouette preventing the full extraction. Slight camera motion due to vibration can result in cases similar to that of Figure 2.4(c), where a part of the background gets labeled as foreground. Sudden lighting changes can also result in mislabeling similar to Figure 2.4(c).

To prevent these issues when experimenting with the tracking system of Section 2.4, we chose to record our own testing sequences with controlled lighting in front of a green screen. The steps involved in extracting the silhouette are similar to those of Figure 2.3, but we can

(a) Shadows (b) Labeling Error (c) Camera or lighting Error

Figure 2.4: Segmentation issues in various HumanEVA [Sigal and Black, 2006] sequences

use off-the-shelf video editing tools to obtain the initial segmentation. Figure 2.5 provides a graphical representation of the modified process.

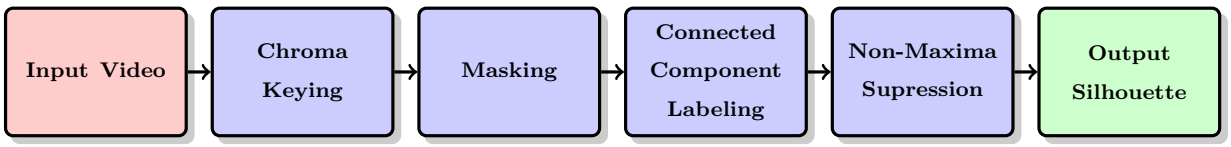


Figure 2.5: Green screen silhouette extraction process

The major difference is the chroma keying that replaces the Gaussian mixture background model and the addition of a masking stage to remove scene elements that lie outside of the green screen. The chroma keying is done in Natron, a free and open-source video compositor. The Natron graph used for the chroma keying step is shown in Figure 2.6.

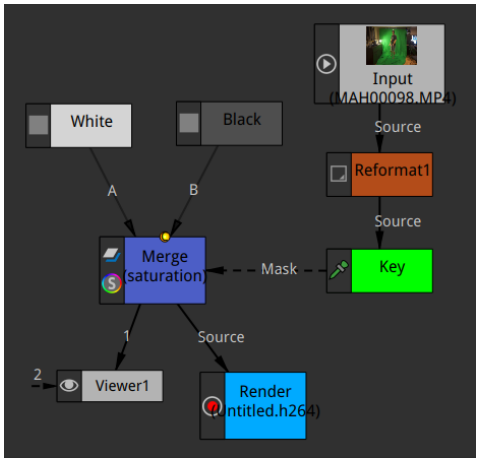
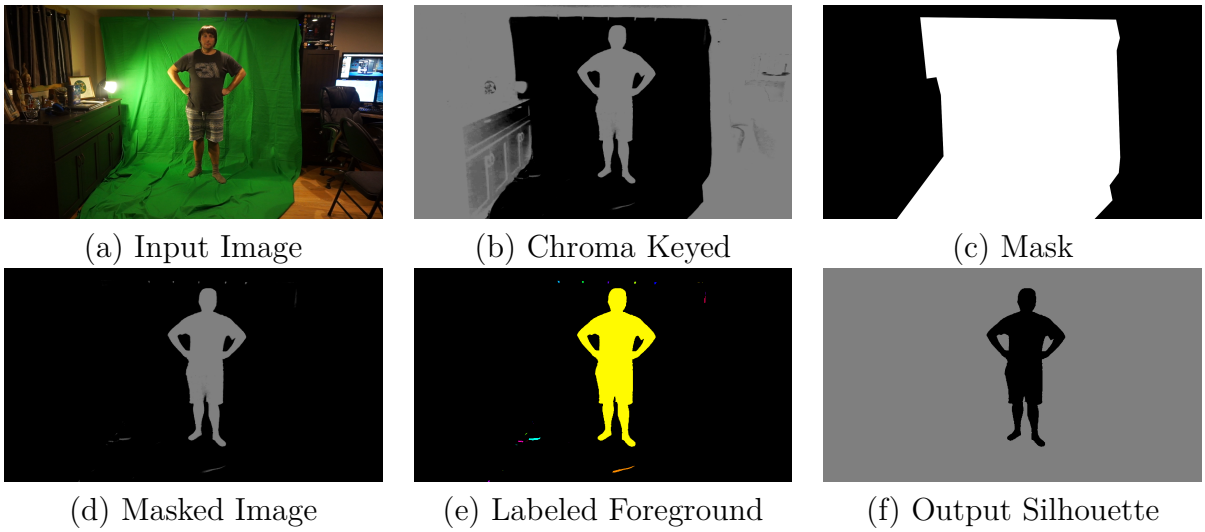


Figure 2.6: Natron graph used for chroma keying

Figure 2.7 shows a frame as it is processed through the different stages shown in Figure 2.5 as well as the manually created mask (Figure 2.7(c)) used for the masking stage.



(a) Input Image (b) Chroma Keyed (c) Mask

(d) Masked Image (e) Labeled Foreground (f) Output Silhouette

Figure 2.7: Segmentation issues in various HumanEVA [Sigal and Black, 2006] sequences

While not practical for general application of the tracking algorithm, the use of chroma-keying provides a much cleaner segmentation which allows more flexibility during the testing process. Figure 2.8 presents a selection of binary silhouettes from our green screen sequences.

Figure 2.8: Sample frames from our green screen sequences

2.2 Human Model

To enable experimentation, we construct a deformable mesh model of a human with an internal skeletal structure. To this end, Figure 2.9 and accompanying Table 2.1 provide a graphical representation of the skeleton and a list of the joints as well as the number of degrees of freedom for each. It is important to note here that there are 3 redundant degrees of freedom at the root joint. This is because 3 DOF are used for the global model rotations which are used by both the particle filter tracking the root position and the filter tracking the pose. The pelvis also accounts for local rotations around the pelvis. In most cases these redundant DOF could be merged into one, but the implementation is simpler when kept separate. Figure 2.10 provides examples of rendered silhouettes with different poses from our model.

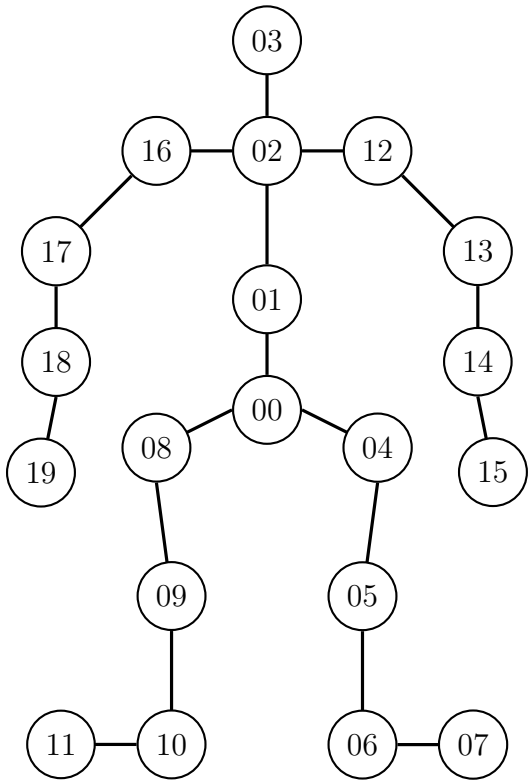
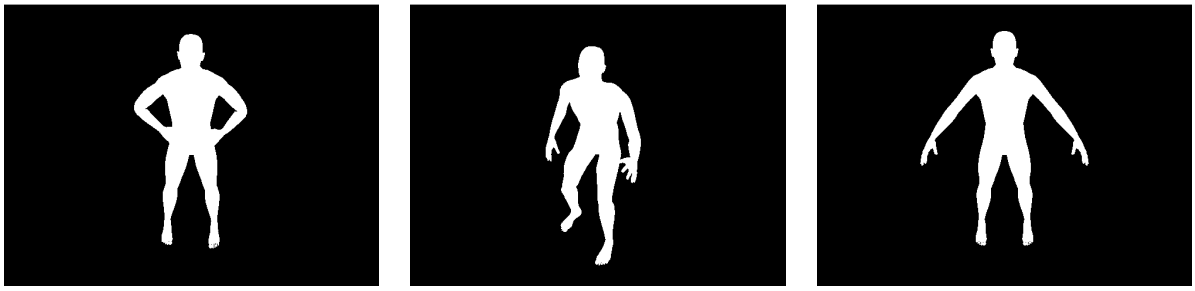


Figure 2.9: Proposed skeletal model

Index	Parent	Name	DOF
00	–	Root / Pelvis	6
01	00	Abdomen	1
02	01	Thorax	3
03	02	Head	3
04	00	Right Hip	0
05	04	Right Thigh	3
06	05	Right Shin	1
07	06	Right Foot	1
08	00	Left Hip	0
09	08	Left Thigh	3
10	09	Left Shin	1
11	10	Left Foot	1
12	02	Right Shoulder	0
13	12	Right Upper Arm	3
14	13	Right Forearm	2
15	14	Right Hand	2
16	02	Left Shoulder	0
17	16	Left Upper Arm	3
18	17	Left Forearm	2
19	18	Left Hand	2
		Total DOF	37

Table 2.1: Proposed human skeletal structure

We rely on MakeHuman [Team, 2001] to create a mesh model that we will be able to deform based on the pose of the skeleton. The skeleton, coupled with our rendering system and a silhouette shader allow us to generate virtual silhouettes, such as the three example silhouetes shown in Figure 2.10.



(a) (b) (c)

Figure 2.10: Example rendered silhouettes

2.3 Metric selection

While we focus on the application of shape matching to the task of human posture tracking over image sequences in the current research, the evaluation of the image similarity metrics should be useful for any application that requires comparisons of object outlines.

Most human tracking and human pose extraction papers describe the metric they use to measure distance between observed data and expected data, but few provide the rationale behind the choice of one metric over others and even fewer provide comparisons to other metrics. We now provide a comparison of widely used metrics and provide a better understanding of which types of metrics are more appropriate for the task of comparing human silhouettes. The intention is thus to implement these metrics and compare them, from both theoretical and practical standpoints.

A survey of some shape comparison metrics is provided by Veltkamp [2001]. While there is some overlap between the metrics presented here and those by Veltkamp [2001], we go beyond simply describing the selected metrics and systematically evaluate them. A survey of

silhouette metrics applied to human shape is presented by Poppe and Poel [2006], but only evaluates three metrics and lacks a definition of the region of convergence of these metrics.

When comparing metrics, many methods employ a specific dataset to measure the metric's performance. A good example of this practice in an other domain is presented by Cohen et al. [2003]. The application of shape matching to human tracking involves a variety of other components that can skew results; we thus choose to evaluate metrics without relying on a specific dataset and instead use synthetic data.

Most of the results presented in this section have been published at the 13th conference on Computer and Robot Vision [Cormier and Ferrie, 2016], but some additions that were late for the publication were added.

2.3.1 Evaluated Metrics

This section presents an overview of the studied metrics, providing a general description of each. A more in depth look at each metric can be found in the papers referenced.

Many of the methods presented by the original authors are neither translation nor scale invariant; we attempt to alleviate these problems by cropping the silhouettes and scaling the resulting images to a uniform size. For methods that rely on a chain-coded representation of the silhouette edge, we resample the chain code to a fixed number of points. These constraints remove some notion of scale and translation from the metrics, thus simplifying comparison between silhouette comparison metrics.

Hu Moments [Rosales and Sclaroff, 2000; Hu, 1962]

Hu moments are a set of image descriptors that are simple to compute and represent many aspects of the evaluated shape such as area, centroid, and orientation. For example, the raw moment of an image are computed as shown in Equation 2.1, where $\mathrm{I}(x, y)$ is the pixel

intensity at the given x and y coordinates,

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \quad . \tag{2.1}$$

These raw moments can be used to find the area as M_{00} and the centroid as

$$\{\bar{x}, \bar{y}\} = \left\{ \frac{M_{01}}{M_{00}}, \frac{M_{10}}{M_{00}} \right\} \quad . \tag{2.2}$$

From the raw moments, we can use Equation 2.3 to compute the translation invariant central moments as

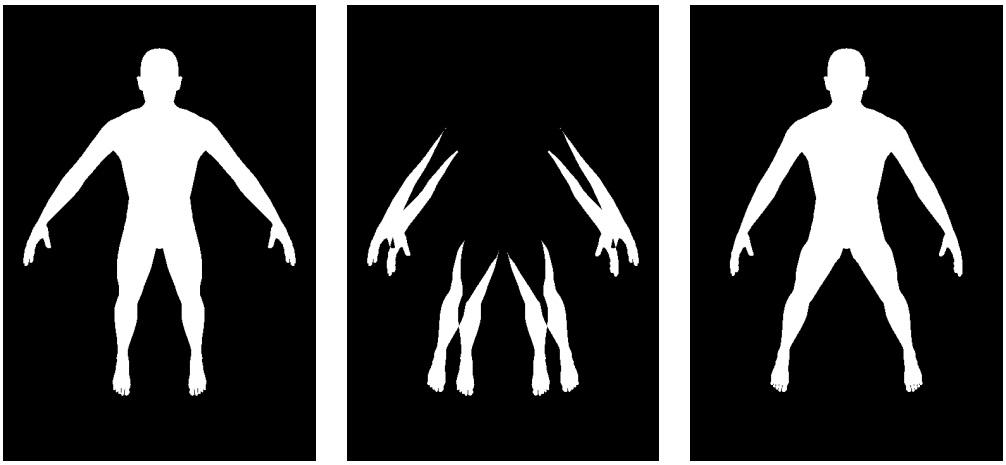
$$\mu_{pq} = \sum_{x} \sum_{y} (x - \bar{x})^p (y - \bar{y})^q I(x, y) \quad . \tag{2.3}$$

From these central moments, we can build up higher level moments that are scale and rotation invariant. The equations for these can be found in the original paper presenting the Hu moments [Hu, 1962]. While image moments can cause a lot of ambiguity, they can be used to successfully match silhouettes with classes of known pose [Rosales and Sclaroff, 2000]. They estimate pose by using an expectation-maximization algorithm to cluster the Hu moments for a set of pose classes. The main advantage of this metric is that negative space within the main shape blob is properly identified as the metric is computed on the pixel values instead of on the border pixels. Figure 2.10(a) provides an example of two large negative space regions inside a silhouette. Once the Hu moments are extracted, the distance between two silhouettes is computed as the Euclidean distance between the moment vectors.

Pixel Count [Wang et al., 2013; Güdükbay et al., 2013]

Another simple metric involves the computation of the number of pixels that differ between two silhouettes. This method requires good camera calibration or image matching, as both silhouettes need to be aligned. Cropping the bounding box of the silhouette and scaling to a common size, as stated previously, also helps in aligning silhouettes. This metric can also

handle negative space within silhouettes, as discussed in the description of the Hu moments. For two silhouettes S_1 and S_2 represented by binary images of dimensions m by n, we can compute the distance with Equation 2.4, where $\oplus$ is the exclusive or (xor) operation, and $S[x, y]$ is the value at pixel location (x, y). Figure 2.11 provides a graphical example of the computation of the exclusive or version of two binary silhouette images S_1 and S_2.



S_1 $S_1 \oplus S_2$ S_2

Figure 2.11: Example computation of the exclusive or version of two binary silhouette images S_1 and S_2.

$$D(S_1, S_2) = \sum_{x=0}^{n} \sum_{y=0}^{m} S_1[x, y] \oplus S_2[x, y] \quad . \tag{2.4}$$

Chamfer Distance [Howe, 2004]

This metric is computed solely from the contour of the silhouette. To do so, a chain code representation of the silhouette edge is extracted by traveling along the edge of the silhouette and recording the visited pixels. The distance between two chain code sets S_1 and S_2 can be computed via the modified Haussdorf distance defined in Equation 2.5, where $d(p, q)$ is the Euclidean distance between two points in the chain codes. A proper Hausdorff distance

would take the maximum instead of the sum,

$$D(S_1, S_2) = \sum_{p \in S_1} \min_{q \in S_2} d(p, q) \quad . \tag{2.5}$$

Turning Angle [Howe, 2004]

This metric requires the computation of a turning angle diagram, which is generated by recording the angle between successive points in the chain code. This diagram is said to better encode the shape of the silhouette than simply using the chain code point locations. The distance between two silhouettes can be computed as the sum of squared differences between the turning angle representations of each silhouette.

Distance Signal [Dedeoğlu et al., 2006]

The chain code representation of the contour is used for this metric, but an extra piece of information is added. Instead of simply looking at the position of the contour points, this metric considers the distance between each point and the center of mass of the silhouette. The center of mass $(\bar{x}, \bar{y})$ of a silhouette represented by a binary image S_i can be computed either with Equation 2.6 or via the central Hu moment,

$$\bar{x} = \frac{\sum_{x=0}^{n} \sum_{y=0}^{m} xS[x, y]}{\sum_{x=0}^{n} \sum_{y=0}^{m} S[x, y]}, \bar{y} = \frac{\sum_{x=0}^{n} \sum_{y=0}^{m} yS[x, y]}{\sum_{x=0}^{n} \sum_{y=0}^{m} S[x, y]} \quad . \tag{2.6}$$

The distance between two shapes S_1 and S_2 with precomputed distance signals DS_1 and DS_2 is computed as the sum of absolute differences, as shown in Equation 2.7, where n is the number of points in the chain code,

$$D(S_1, S_2) = D(DS_1, DS_2) = \sum_{i=0}^{n} \left| DS_1[i] - DS_2[i] \right| \quad . \tag{2.7}$$

Shape Contexts [Belongie et al., 2002; Mori and Malik, 2006]

Shape contexts are 2D histograms that can be computed at any point along the edge of a silhouette. These histograms encode the relation between the point at which it is computed and all other points of the chain code by recording the angle and distance to the other points. The shape context implicitly encodes the local curvature and shape of the silhouette. This metric is translation invariant by design and can be made scale invariant by scaling the silhouettes and resampling the chain codes to a fixed number of points [Agarwal and Triggs, 2006]. The "distance" between two shape contexts p and q is computed by the use of the following χ^2 test:

$$d(p,q) = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_p(k) - h_q(k)]^2}{h_p(k) + h_q(k)} \quad , \tag{2.8}$$

where $h_i(k)$ is the value of the k^{th} bin of the histogram at point i.

Computing a distance between two silhouettes involves finding a one to one mapping between the points on each silhouette that minimizes the sum of distances between the two sets [Belongie et al., 2002]. This type of matching is a full bipartite graph combinatorial optimization problem that can be solved via the Kuhn–Munkres algorithm [Kuhn, 1955].

A computationally simpler method of computing the distance between two chain code sets is to use Equation 2.5. This alleviates the need to run the graph optimization step at the cost of not obtaining a one to one mapping. We refer to this simplified version as a greedy set matching strategy.

2.3.2 Metric Evaluation

Before comparing each of the metrics, one must first determine what the characteristics of a good metric are. Veltkamp [2001] presents the following three desirable properties for shape matching metrics:

- **Metric Property**: The distance between any two silhouettes should always be pos-

itive and a minimum distance should only be obtained when comparing a given silhouette to itself. The distance metric should also respect the triangle inequality, as presented in Equation 2.9,

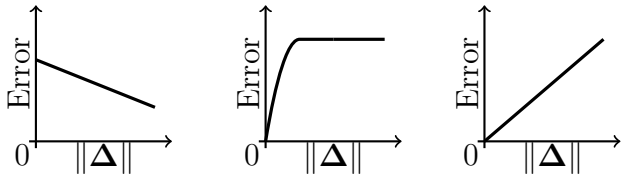
$$D(S_1, S_2) + D(S_2, S_3) \geq D(S_1, S_3) \quad . \tag{2.9}$$

- **Robustness Property**: Distance measures should be robust to the kind of noise that are often encountered in computer vision problems. Such sources of noise include discretization errors, blur, and occlusion.
- **Invariance Property**: The metric should be invariant to classes of transformations that are expected to be encountered.

Note that the invariance property is very important for image matching, but less so for the task of human posture tracking. This is due to the fact that we extract a subwindow containing the silhouette, which we scale to a fixed size. This reduces the effect of translation and scale on the silhouette. The human silhouettes we observe are also mostly standing upright, so rotation is not as present as when tracking other types of objects.
A stronger criterion can be added to the Metric Property by requiring metrics to increase monotonically as we move away from the correct solution; however, it is not expected that all metrics behave that way over the entire pose space. We can sample the error manifold to determine the size of the neighborhood around the correct solution where the metric behaves monotonically. To test this, we sample a given number of points at a distance $\|\mathbf{\Delta}\|$ and compute statistics of the recorded errors. By gradually increasing $\|\mathbf{\Delta}\|$, we obtain a curve of how each metric behaves as a function of distance in pose space. From this curve, we can determine if each error respects the Metric Property.

We expect to see three basic metric behavior types, as represented by example plots in Figure 2.12. The first (Figure 2.12(a)) is a metric that does not respect the Metric Property as the error does not increase with $\|\mathbf{\Delta}\|$, and is thus not an applicable metric. As all metrics

that are tested herein have already been shown to work by the original authors, none of the results should resemble Figure 2.12(a). The second and third types of metrics (Figures 2.12(b) and 2.12(c)) are both valid with respect to the properties listed previously, but the metric in 2.12(c) is superior in cases where a gradient descent method is used to find the solution as the error increases linearly over the range of $\|\mathbf{\Delta}\|$.



(a) Decreasing (b) Plateauing (c) Linear

Figure 2.12: Three types of possible metric behaviors

Interestingly, the plot shown in Figure 2.12(c) is simply a linear plot of the distance in pose space (Error=$\|\mathbf{\Delta}\|$). This allows us to compare metrics by measuring if the metric values correlate well with the distance in pose space. As we are rendering the poses, we know the value of $\mathbf{\Delta_{21}}$ exactly. This makes it possible to directly compute the Pearson product-moment correlation coefficient [Pearson, 1895] between $D(S_1, S_2)$ and $D(\mathbf{P_1}, \mathbf{P_2})$.

The Robustness Property was investigated by testing the metrics with the addition of noise. We simulate two common problems found in human tracking applications: ground shadows and camera noise. These sources of noise change the overall shape of the silhouette and cause a reduction in the performance of most metrics. Figure 2.13 shows the silhouettes for six test cases that are considered.

- Ground Shadows (Figures 2.13(b) and 2.13(e)): While there exist a multitude of algorithms to segment the silhouette from an image, most of them suffer from difficulties dealing with shadows [Moeslund et al., 2006]. A blob is thus added below the feet of the silhouette to simulate shadows cast on the ground.

- Camera Noise (Figures 2.13(c) and 2.13(f)): Camera noise may also affect the accuracy of the silhouette segmentation. This type of noise also affects the segmentation and

may cause mis-labeled pixels that result in variations along the edge of the silhouette. To simulate this, we can randomly displace pixels along the edge.

The invariance property can be explored by testing the metrics under different scene configurations. We run two sets of tests, one with the 3D human model facing the camera (Figures 2.13(a), 2.13(b), and 2.13(c)) and the other with the camera looking at the model sideways (Figures 2.13(d), 2.13(e), and 2.13(f)).

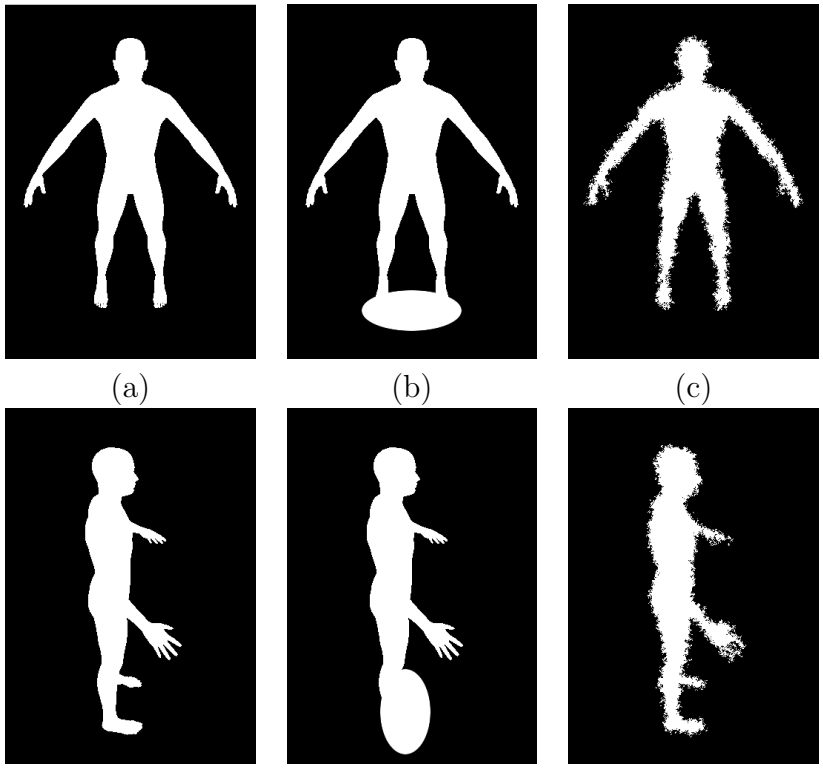


Figure 2.13: Reference silhouettes used for the tests, viewed from the front (a,b,c) and from the side (d,e,f). Shadows (b,e) and noise (c,f) are added to test robustness.

2.3.3 Experimental Methodology

As stated earlier, the initial experiment involved recording the values of the metrics as we move away from the correct solution. To compute the distance $\|\mathbf{\Delta}\| = D(S_1, S_2)$ between two poses, we use the L^2 norm of the difference between the S_1 and S_2, as shown in Equation

2.10. This distance could be measured in degrees, but as we are dealing with normalized poses (all values of the pose vector are between 0 and 1), the actual units are related, but not exactly degrees, we thus call the unit "normalized pose space unit" (NPSU). As we are looking at a 37 degrees of freedom model, the summation goes from 1 to 37.

$$\|\mathbf{\Delta}\| = D(P_1, P_2) = \sqrt{\sum_{d=1}^{37} |P_{1d} - P_{2d}|^2} \tag{2.10}$$

The absolute maximum distance between two poses occurs when $\mathbf{P_1} = [0, 0, ..., 0]$ and $\mathbf{P_2} = [1, 1, ..., 1]$, when we obtain $\|\mathbf{\Delta}\| = D(\mathbf{P_1}, \mathbf{P_2}) = 6.08$. To simplify testing procedure, we kept the values of the first 3 DOF constant at zero. This choice eliminates the effect of the translation components, which have little to no effect on the rendered silhouettes because of the scaling and cropping discussed earlier. For the first few tests, we set $\mathbf{P_1} = [0.5, 0.5, ..., 0.5]$ as a reference pose and vary $\mathbf{P_2}$, which means that the maximum distance between $\mathbf{P_1}$ and $\mathbf{P_2}$ is halved to 3.04.

Metric Values are thus recorded as $\|\mathbf{\Delta}\|$ is increased from 0 to 3.04. For each increment step of $\|\mathbf{\Delta}\|$, a few hundred sample measurements of the metrics are computed in order to calculate the minimum and maximum values of the error as well as average value and standard deviation. These values are presented in Figure 2.17, with the average as a black line, error bars representing the standard deviation and a gray-shaded region to show the range of metric values between the minimum and maximum. From these results, it is possible to determine the region in which each metric is monotonically increasing and record it in Table 2.2. The Pearson product-moment correlation coefficient between the error function and $\|\mathbf{\Delta}\|$ is computed to further describe the behavior of the metric. To do so, the error manifold of each metric is sampled at a few hundred randomly selected locations within $\|\mathbf{\Delta}\| \in [0, M]$ to generate a set of measurements X. For each of these measurements, the value of $\|\mathbf{\Delta}\|$ is recorded to obtain a set Y. The correlation between these two sets is computed according to Equation 2.11. The correlation coefficients are also recorded in Table

2.2.

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{2.11}$$

To obtain fair results, the value of M is selected as the largest monotonically increasing region of the metrics in the given test. For example, the correlation coefficients reported for the front of the clean case (Column 3) of Table 2.2 are computed with $M = \max(0.53, 0.61, 0.76) = 0.76$.

The other measure that is recorded to compare the metrics is the time required to compute the distance between two silhouettes. Similar to the other tests, the distance between a few thousand silhouettes is computed and the execution time is recorded. This allows the computation of the average time and standard deviation. The absolute values alone have little meaning as they are mostly dependent on the hardware used, but the relation between them provides an indication of performance. These results are also be reported in Table 2.2.

To get a more precise understanding of how the metrics behave, we also conduct a set of experiments that records the error values as we move a distance δ from the reference pose along each DOF independently. While the full set of results is not reported here, it is available on the companion website. Figure 2.14 demonstrates what we would find in an ideal case, where the metric increases proportionally to the pose distance.

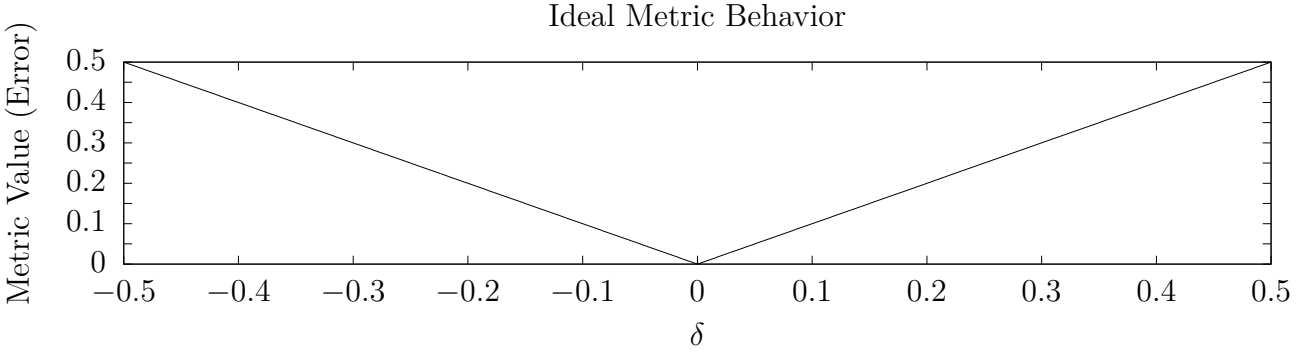


Figure 2.14: Ideal value of a metric between two poses as the distance between those two poses changes. The distance is measured as described in Equation 2.10

The reality is that none of the metrics we are evaluating behave ideally in all cases. For

example, Figure 2.15 shows how the pixel count metric behaves when only the 25^{th} degree of freedom is moved. This behavior, while not ideal, demonstrates that the metric is a good representation of the change in distance, when the distance is within a small region around zero.

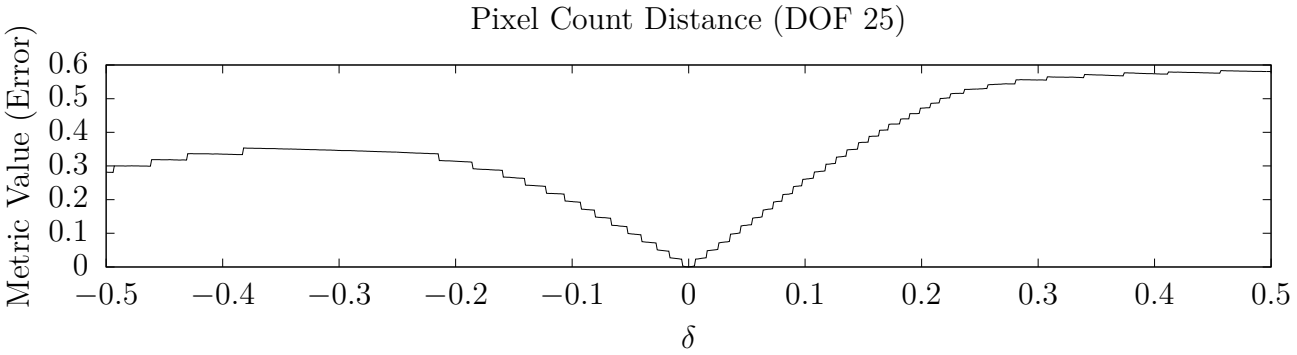


Figure 2.15: Value of the pixel count metric between two poses as the distance between those two poses changes, while changing a single component of the pose vector. The distance is measured as described in Equation 2.10

Occlusion and ambiguity can also result in metric behaviors that are not smooth and change abruptly. Such an example is shown in Figure 2.16. In this example, the metric still behaves properly, but only in a very small region around zero.

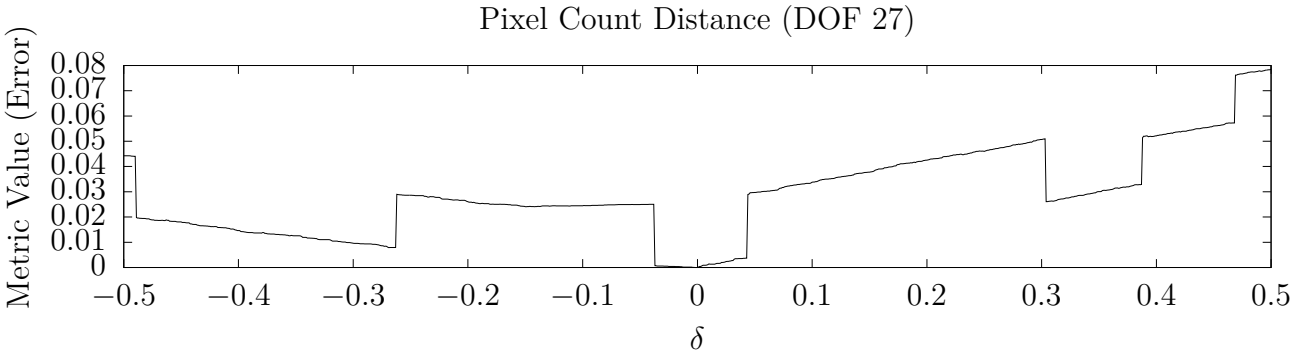


Figure 2.16: Value of the pixel count metric between two poses as the distance between those two poses changes, while changing a single component of the pose vector. The distance is measured as described in Equation 2.10

The final test, which acts as a validation, involves once again the measurement of the metric values as a function of distance in pose space. For this experiment, both poses are

Metric	Clean				Floor Shadow				Noise				Run Time (μs)	
	Monotonic Region		Correlation Coefficient		Monotonic Region		Correlation Coefficient		Monotonic Region		Correlation Coefficient			
	Front	Side	Front	Side	Front	Side	Front	Side	Front	Side	Front	Side	Average	Std.Dev.
Hu Moments	0.53	0.38	0.168	0.337	0.08	0.30	0.117	0.201	0.08	0.38	0.039	0.338	93.70	19.60
Pixel Count	0.61	0.46	0.730	0.746	0.61	0.46	0.747	0.741	0.61	0.76	0.727	0.748	0.59	0.08
Chamfer Distance	0.61	0.76	0.688	0.718	0.61	0.76	0.686	0.709	0.61	0.76	0.681	0.714	4300.88	1071.97
Turning Angle	0.61	1.37	0.672	0.700	0.00	0.61	0.415	0.666	0.08	0.76	0.099	0.572	37.22	6.28
Distance Signal	0.76	0.76	0.796	0.743	0.00	0.76	0.208	0.658	0.76	0.76	0.797	0.718	0.42	0.06
Shape Contexts														
Greedy Matching	0.61	0.46	0.714	0.684	0.61	0.46	0.691	0.659	0.61	0.46	0.703	0.681	30.75	6.86
Bipartite Matching	0.61	0.46	0.744	0.692	0.61	0.46	0.728	0.664	0.61	0.46	0.726	0.677	38.07	6.73

Table 2.2: Summarized results of the similarity metric experiments, both monotonic region size and correlation coefficients are measured in pose distance, as described in Equation 2.10.

selected randomly, instead of keeping the reference pose constant. This should demonstrate how the metrics behave on arbitrary poses and thus confirm wether the results obtained in the previous tests are a good representation of the overall behavior of each metric.

2.3.4 Results & Conclusions

Figure 2.17 shows the plots of the error versus the pose-space distance for each metric, when applied to the first tested case with the reference silhouette shown in Figure 2.13(a). The other plots have not been included because of space constraints, but can be found on the aforementioned website. The relevant information from the plots are recorded and summarized in Table 2.2.

As stated earlier, the goal here was not to determine whether the metrics are valid in general cases, as this has already been done by the original authors, but to systematically determine how appropriate each metric is to the specific task of articulated human posture tracking from silhouettes.

The first result that stands out from Table 2.2 is that Hu moments only increase monotonically over a very small region of the pose space. While they may be good metrics for matching between shape classes where the value of the metric changes by a large value, they are not appropriate for matching slight variations in the pose of humans. The monotonic region measure also allows us to notice that the turning angle metric is very sensitive to the

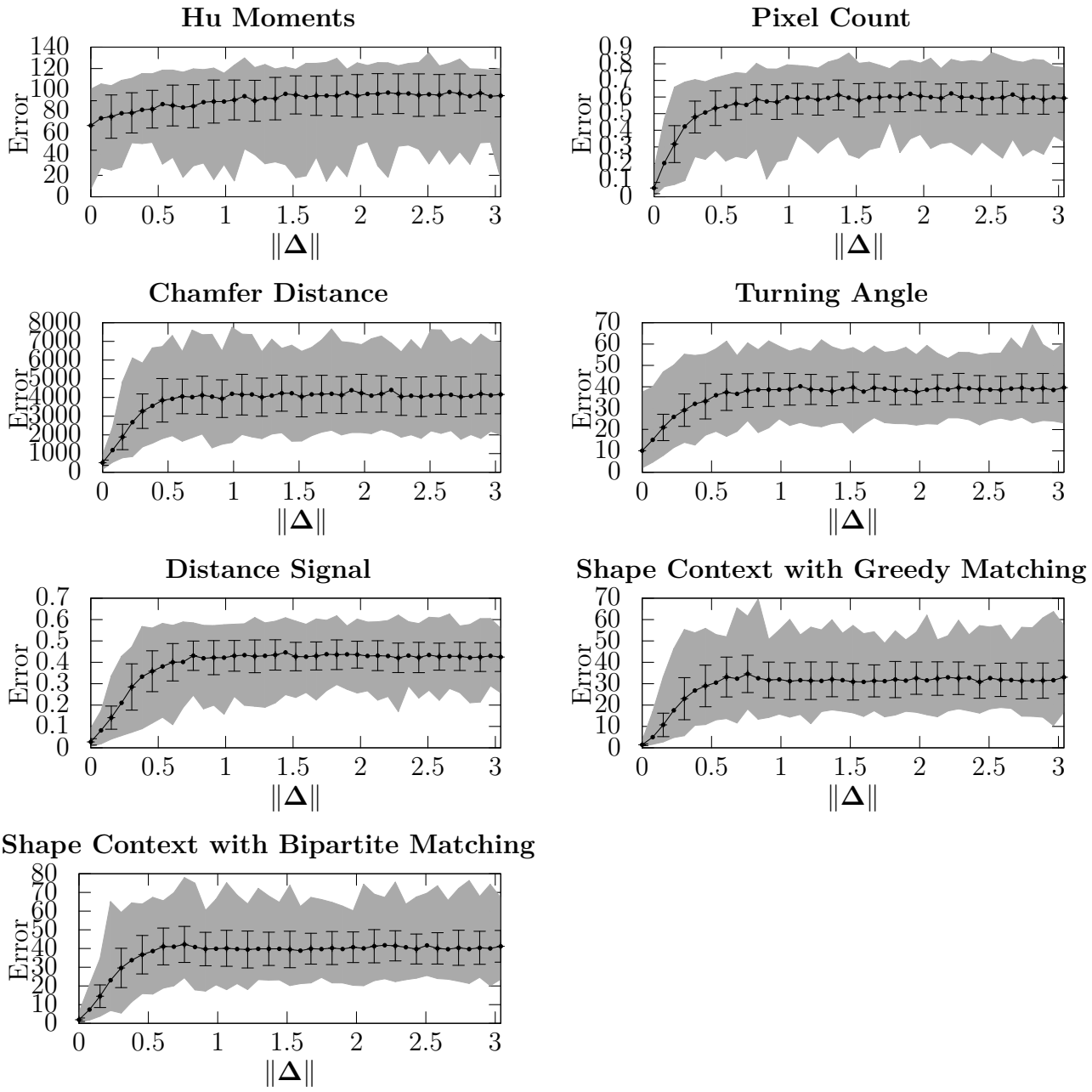


Figure 2.17: Results for the basic case observed from the front. The error values recorded here are the recorded metric values. The exact units of each metric are irrevelant here as we are interested in the behavior of the metric. Results of the other considered cases are available on the website, and are summarized in Table 2.2.

presence of noise or shadows. Similarly, the distance signal is seen not to be robust in the presence of shadows. Disregarding these three metrics, we are left with the pixel count, the chamfer distance, and the shape contexts. The chamfer distance is monotonic over a larger region than the other two, especially when the human is not facing the camera. However, the pixel count metric has a higher correlation coefficient in all tested cases.

The last element in Table 2.2 that can be used to compare the metrics is the runtime. The first interesting result to note here is that while the type of matching for shape contexts has little to no effect on monotonicity, and only a marginal effect on the correlation to pose-space distance, there is a 20 percent difference in the run times between the two methods. The trade-off between the performance of the metrics and the execution time difference leads us to believe that the added complexity of full bipartite graph matching is not necessary to obtain adequate results. There is a sharp contrast between the run times of the pixel count metric and that of the chamfer distance. The fact that the chamfer distance has a large run time was expected as it is the only one of the algorithms with an $\mathcal{O}(n^2)$ complexity with n being the number of points in the chain code. The complexity of the pixel count is $\mathcal{O}(n)$ with n the number of pixels in the silhouette. The difference in run time is further explained by the fact that the pixel count only requires integer and bitwise operations, whereas the chamfer distance relies on floating point distance computation.

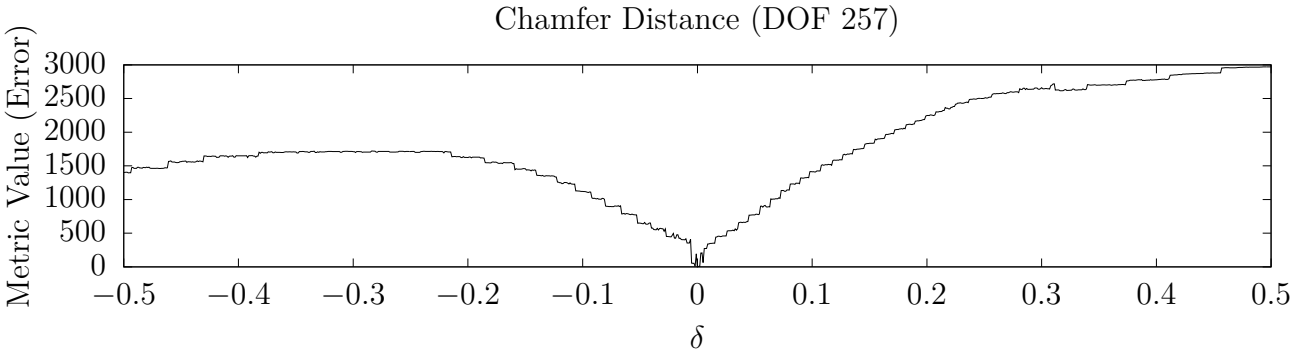


Figure 2.18: Value of the chamfer distance metric between two poses as the distance between those two poses changes, while changing a single component of the pose vector. The distance is measured as described in Equation 2.10

By looking at the results from the test where we move along each DOF independently and comparing the plots for the pixel count and chamfer distances, we can see that the pixel count distance varies more smoothly with δ and presents less small variations (that appear as noise on the curves), which further indicates that the pixel distance is more appropriate for our purposes. This can be seen when comparing the plot in Figures 2.15 and 2.18. Again, the complete set of results is available for inpection on the companion website.

The last step before concluding is to test whether the metrics behave in the same way for any given poses. We confirm this by generating a set of random pairs of poses at known distances and record the resulting distances. Figure 2.19 shows the result of this experiment for the pixel count distance and Figure 2.20 shows the results for the chamfer distance. Each point in these scatterplots represent a pair of random poses. When we compare these two scatter plots with the corresponding curves in Figure 2.17, we observe the same behavior and roughly the same metric values. This confirms that both metrics behave in a consistent manner for any pose pair. The plots for the other metrics are not as relevant here, but are available for viewing on the companion website.

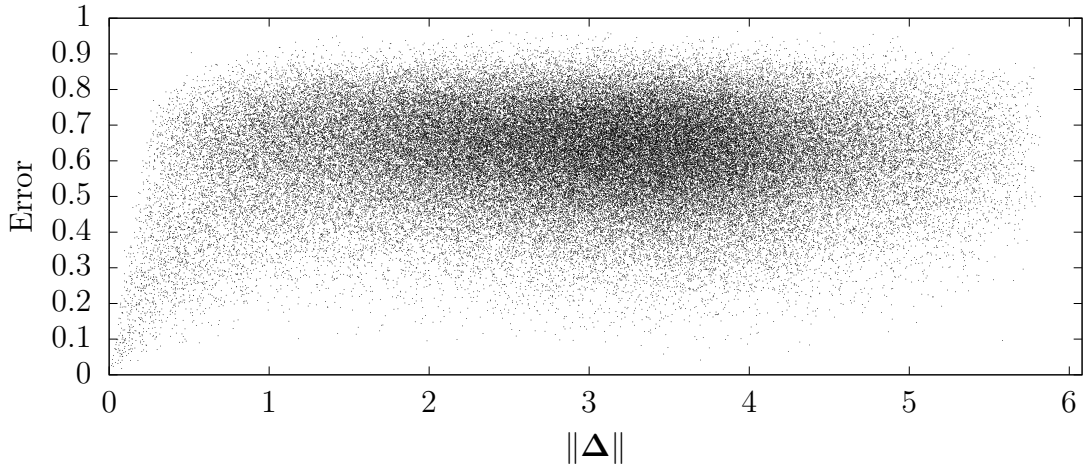


Figure 2.19: Pixel count metric value for pairs of random poses.

To conclude, the key finding of this investigation is that all of the metrics discussed perform reasonably well in ideal conditions, but only the pixel count metric, the chamfer

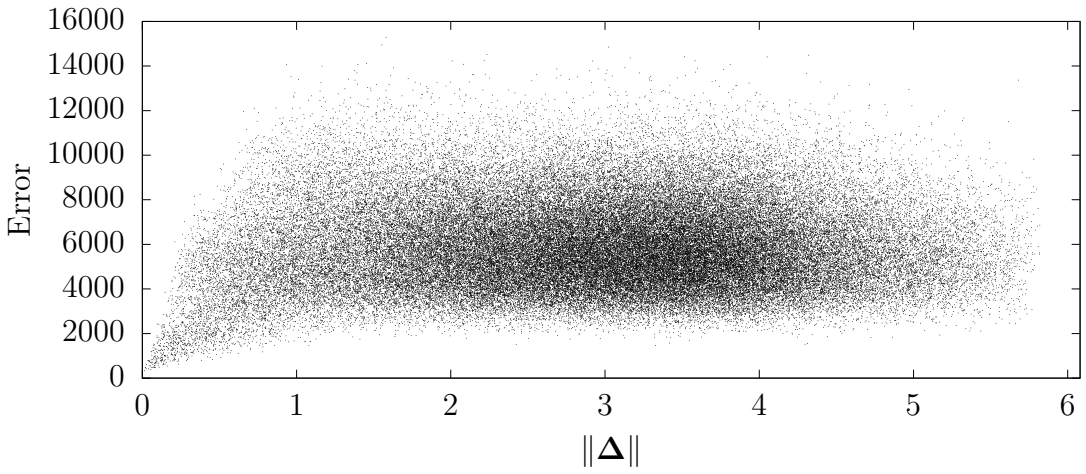


Figure 2.20: Chamfer disatnce metric value for pairs of random poses.

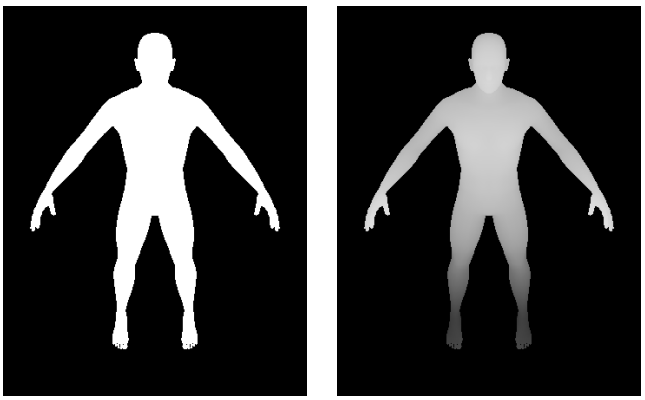
distance, and the shape contexts are robust to the types of noise that are commonly encountered in human posture tracking applications and are thus appropriate for such applications. Furthermore, with a higher correlation to pose-space distance and a lower computation time, the pixel count distance is deemed to be slightly superior to the other metrics.

These conclusions were arrived at by producing a systematically varied dataset under experimental control and probing the strengths and weaknesses of the metrics, rather than relying on a particular existing (and limited) dataset.

Another aspect we were hoping to use these results for was to determine if the pose space could be easily discretized without losing accuracy. The results provided in Figure 2.17 show that even the slightest perturbation in the pose vector results in noticeable changes in the metric value. This indicates that any level of useful discretization would result in a loss of precision.

2.3.5 Silhouette and Depth

A possible improvement over using the pixel count is to consider depth information in addition to silhouette image, if this information is available. Figure 2.21 provides an example of a silhouette image with depth information. The segmentation of the silhouette can be



(a) (b)

Figure 2.21: Silhouette (a) and depth image silhouette (b)

executed using the algorithm described in Section 2.1, either with or without the addition of depth to the RGB vector used for the Gaussian mixture model.

To experiment with the addition of depth, we compute the sum of squared differences (SSD) between the captured and segmented depth image and the depth buffer of the rendered silhouette. The mathematical form of the computed SSD is shown in Equation 2.12, for two depth images D_1 and D_2 of dimensions m by n,

$$D(D_1, D_2) = \sum_{x=0}^{n} \sum_{y=0}^{m} \Big(D_1[x, y] - D_2[x, y] \Big)^2 \quad . \tag{2.12}$$

Performing the same experiments as described for non-depth metrics to determine the size of the monotonically increasing region and the correlation coefficients yields the two plots shown in Figure 2.22 and the numerical results reported in Table 2.3.

	Clean				**Floor Shadow**				**Noise**			
	Monotonic Region		**Correlation Coefficient**		**Monotonic Region**		**Correlation Coefficient**		**Monotonic Region**		**Correlation Coefficient**	
Metric	Front	Side	Front	Side	Front	Side	Front	Side	Front	Side	Front	Side
Pixel Count	1.064	0.760	0.680	0.725	1.216	0.760	0.654	0.725	0.684	0.760	0.684	0.724
Depth SSD	0.988	0.912	0.645	0.682	0.760	0.912	0.605	0.680	0.760	0.912	0.626	0.670

Table 2.3: Results of the metric evaluation tests for depth

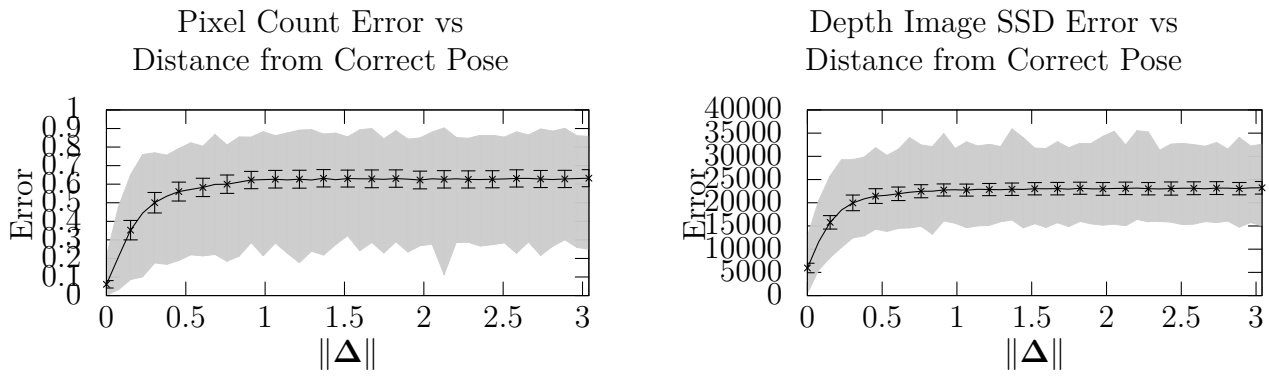


Figure 2.22: Silhouette (a) and depth image silhouette (b)

Looking at the size of the monotonic regions provided by both depth and non-depth metrics, we see that there is no significant difference in performance gained by considering depth when the model is viewed from the front. When viewed from the side, however, using depth provides a significantly larger monotonic region which would translate into a more robust metric and thus better performance, as it attenuates the effect of ambiguity. The most important result is the small monotonic region for the depth SSD metric when viewed from the front in the presence of floor shadows, as they greatly increase the SSD, thus skewing results. While the pixel count metric has a slight edge in terms of correlation coefficient in all cases, the correlation coefficients are very similar whether depth is considered or not. The conclusion here is that the addition of depth can help in certain cases, but the difference in results is not significant. More research may be required to determine if there exists more efficient depth-based metrics than a sum of squared differences. However, metrics that do not require depth present adequate results without requiring specialized hardware.

The results presented in this section confirm our assumption that continuous changes in the pose vector result in continuous changes in the silhouette, within a small region around a pose (within approximately 0.5 distance in normalized pose space). This means that once a reasonably accurate solution is found, we can either use gradient descent to refine this solution or use a filtering approach to improve accuracy and maintain it over time. The initialization part of the next section discusses different strategies that can be used to obtain

this initial solution. Silhouettes are thus a viable feature to use for the purpose of human pose tracking.

2.4 Tracking

Now that we can extract features and have a metric to compare them, we look at how to keep track of pose vectors by integrating information over time.

For this task, we rely on a classical particle filter approach. The canonical formulation of a particle filter is shown in Equation 2.13. In this equation, x_t is the state vector, which in the case of human tracking, is the pose vector we are interested in finding. The term $y_{1;t}$ is the sequence of measurements from time 1 to time t, which in our case are the extracted silhouettes since the start of the video sequence. The likelihood term is the probability of observing a given measurement y_t, given a state vector x_t. The likelihood is where we apply the silhouette comparison metric described earlier. The temporal prior is where we need to model the expected change in state vector between frames. Finally, the posterior is simply the result from the previous frame.

$$\underbrace{p(x_t|y_{1:t})}_{\text{Posterior at } t} \propto \underbrace{p(y_t|x_t)}_{\text{Likelihood}} \int \underbrace{\overbrace{p(x_t|x_{t-1})}^{\text{Temporal Prior}} \overbrace{p(x_{t-1}|y_{1:t-1})}^{\text{Posterior at } t-1} dx_{t-1}}_{\text{Predictive Density}} \tag{2.13}$$

A practical description of a generic particle filter, provided by Salmond and Gordon [2005], shows that there are five main aspects that need to be considered when thinking about particle filtering: initialization, particle weighting, particle propagation, particle resampling, and result computation. We will now go over each of these components to describe them in more detail.

2.4.1 Initialization

For any particle filter to keep track of something correctly, a proper initialization is required. The better the initialization, the more rapidly the filter will converge to the correct solution. In simple cases, a random sampling of the search space is enough for the filter to converge. For more complex systems, a large number of random particles would be necessary for convergence. A more accurate initial hypothesis is thus desirable for proper convergence.

We use a gradient descent method, starting from known common poses, to generate a set of initial candidate poses. We will see through the experiments in Chapter 4 that this approach is appropriate for simpler models, but the experiments in Chapter 5 show that it only works in ideal cases when working with more complex models. For such models, we turn to machine learning to obtain the initial pose estimate. Chapter 5 provides a discussion of the network architecture we use to generate pose candidates. We use the network to detect key poses and the particle filter is used for interpolation between the detected keyframes.

As a quick experiment to determine if a CNN can capture the mapping between pose and silhouette, we designed a simple network with the architecture shown in Figure 2.23. We trained this network with synthetic silhouettes generated from random pose vectors. More information about the generation of silhouettes for training will be presented in Chapter 5. We then tested the network by giving it a pose vector and compared the output silhouette generated by the network to the rendered silhouette generated from the same vector. Figure 2.24 presents comparisons between the output of the network, in the bottom row, and the ground truth silhouettes, in the top row. This simple experiment shows that even with a relatively simple network architecture, we can learn the mapping between pose and silhouette. Chapter 5 will present a network that does the opposite by estimating the pose vector from the silhouette. We will discuss in Section 6.1, the possibility of replacing the renderer from our framework with a generative neural network such as the one presented here.

2.4.2 Particle Weighting

Figure 2.25 provides an overview of the steps involved in computing the likelihood of a pose candidate in the form of a block diagram. To re-iterate, each particle in the filter is a pose candidate. As such, each pose candidate represents a possible configuration of the mesh model and can be used to generate an artifical silhouette, as described in Section 2.2. The first step in computing the weight of a candidate particle thus consists in deforming the mesh model in accordance with the pose vector. By rendering a silhouette of the model we can

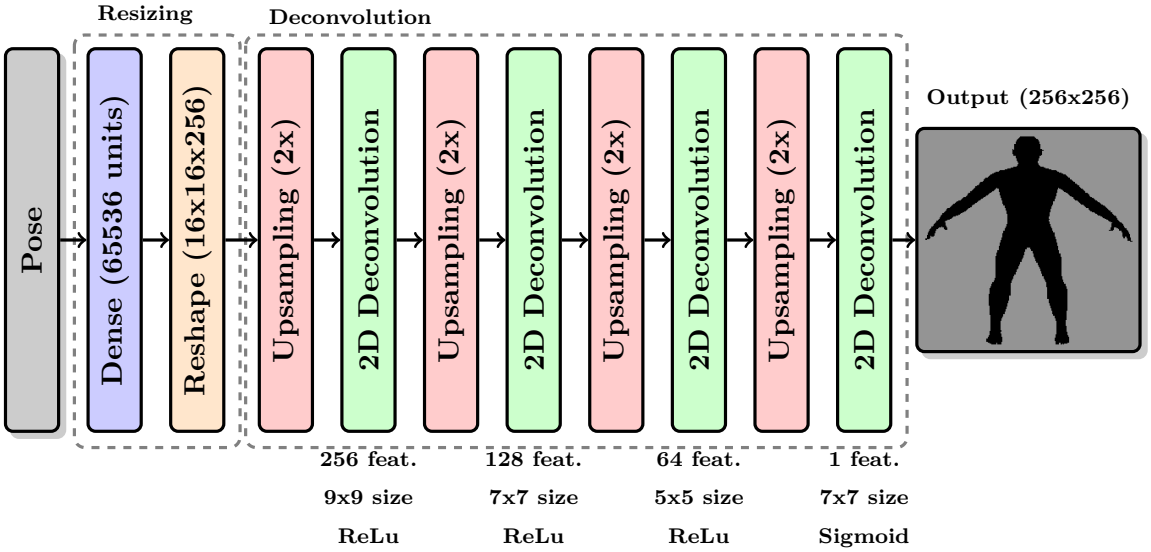


Figure 2.23: Experimental Deconvolutional Neural Network architecture.

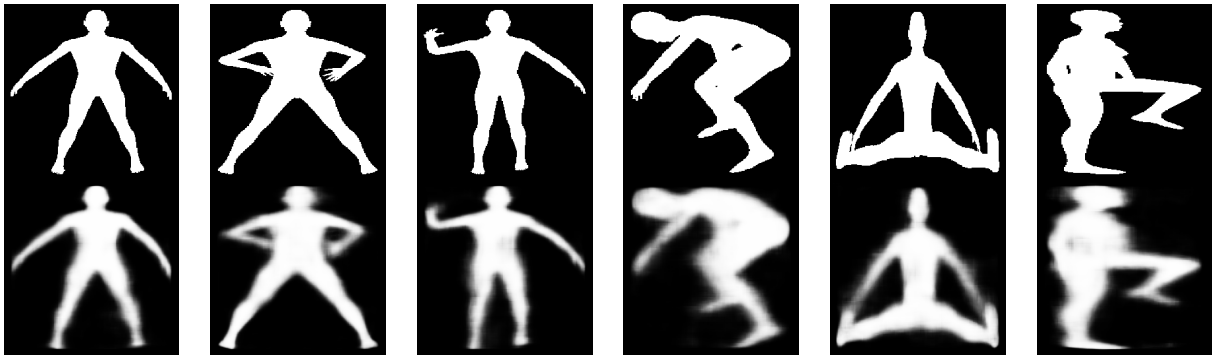


Figure 2.24: Example comparisons between the output of the network (bottom row) and ground truth silhouettes (top row).

then apply the similarity metric described in Section 2.3 to obtain a likelihood score that we can use as a weight.

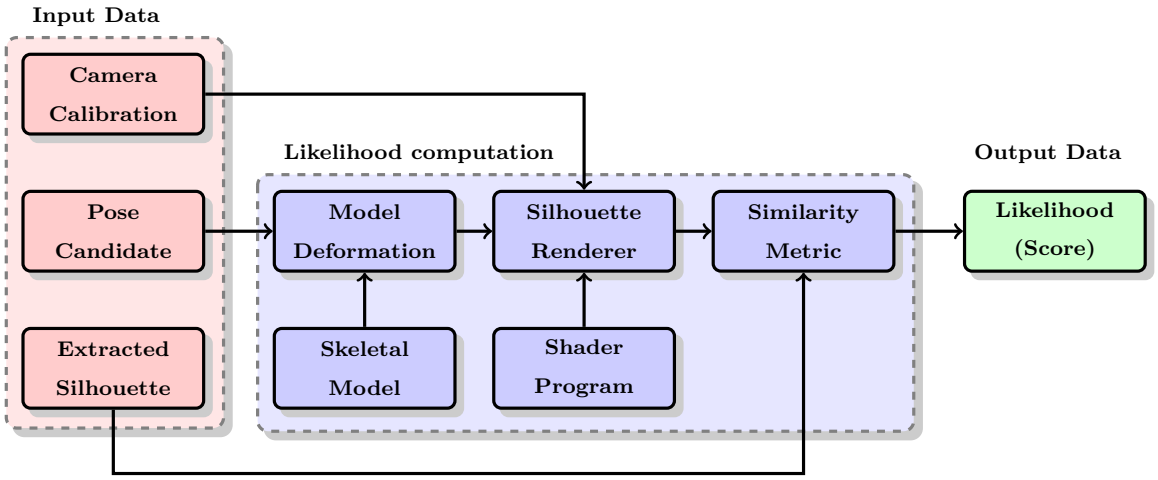


Figure 2.25: Block diagram of the particle weight computation process

Some approaches, such as the one presented by Canals et al. [2009] propose different ways to map the similarity measure into meaningful particle weights. All of the proposed methods rely on setting the particle weights to a value proportional to the similarity metric. To accentuate the convexity of the probability distribution, the particle weights w^k can be computed from the exponential of the similarity measure, as shown in Equation 2.14, where z_t^k is the similarity measure of the k^{th} particle at time t,

$$w_t^k = \frac{\exp(1 - z_t^k)}{\sum_{k=0}^{N-1} exp(1 - z_t^k)} \quad . \tag{2.14}$$

The fractional part of Equation 2.14 (dividing by the sum of the weights) is there to normalize the particle weights so that they sum to 1.

2.4.3 Particle Propagation - Dynamics Models

There are multiple approaches to modeling and predicting the state of the observed system over time. We now list the different motion models used throughout the literature to determine the most appropriate one for the task of pose tracking. Sadly, not all publications

provide detailed descriptions of how particles are propagated over time [Agarwal and Triggs, 2006]. This is why we provide a quick survey of the most widely used propagation methods, then we select methods to evaluate based on the goals and requirements of the current work.

Gaussian Diffusion

This simple motion model, based on the assumption that human motion is continuous over a small interval of time, is widely used in the literature [Duff et al., 2011; Sedai et al., 2013; Vondrak et al., 2013]. Equation 2.15, where Σ is a diagonal covariance matrix, shows how states are assumed to evolve over time,

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \mathcal{N}_t(0, \Sigma) \quad . \tag{2.15}$$

One surprising point demonstrated by Yang et al. [2005] is that the choice of random number generation algorithm can also influence the convergence speed and accuracy of the particle filter, depending on the symmetry of the random samples drawn from the Gaussian distribution.

First Order Motion

We work under the assumption that the pose evolves in a continuous manner, we do not however know if the rate of change of the pose is also continuous or if the if the acceleration is continuous. For this reason, we consider first and second order models, which allow us to consider the rates at which the pose changes over time. This type of motion model considers both the position and velocity of particles [Vondrak et al., 2013]. Given a state vector $\mathbf{x}$ and a velocity $\dot{\mathbf{x}}$, the basic form of this model is shown in Equation 2.16,

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \dot{\mathbf{x}}_t + \frac{\Delta t^2}{2} \cdot \mathcal{N}_t(1, \Sigma_{\mathbf{x}}) \quad . \tag{2.16}$$

The velocity of the particle can be computed in three ways. The first is to use the finite

difference approximations of velocity by using the two previous states in the form shown in Equation 2.17,

$$\dot{\mathbf{x}}_t = \mathbf{x}_{t-\Delta t} - \mathbf{x}_{t-2\Delta t} + \Delta t \cdot \mathcal{N}_t(1, \Sigma_{\dot{\mathbf{x}}}) \quad . \tag{2.17}$$

The second way of obtaining $\dot{\mathbf{x}}_t$ is to again use a finite difference approximation, but instead of using the state of each particle individually, we can use the output state of the filter and assume that this approximated velocity should hold for all particles.

The third way is described by Hue et al. [2002] (see also Canals et al. [2009]), who propose augmenting the state vector by the addition of velocity components to obtain $\mathbf{x'}_t = [\mathbf{x_t}, \dot{\mathbf{x}}_\mathbf{t}]$. This provides the ability of directly tracking the velocity at the cost of a larger state space. They also add a Gaussian component to the update equation, as shown in Equation 2.18, where I is the identity matrix the same size as $\mathbf{x}$ and $\mathcal{N}$ is a vector of values drawn from a zero-mean normal distribution,

$$\mathbf{x'}_{t+\Delta t} = \begin{bmatrix} I & \Delta t \cdot I \\ 0 & I \end{bmatrix} \mathbf{x'}_t + \begin{bmatrix} \frac{\Delta t^2}{2} \cdot I \\ \Delta t \cdot I \end{bmatrix} \mathcal{N}_\mathbf{t}(\mathbf{1}, \Sigma) \quad . \tag{2.18}$$

Second Order Motion

Taking this a step further, one can append an acceleration component to the pose and track acceleration as well as velocity. Adding acceleration means that we can better approximate the motion over time, at the cost of more parameters. The resulting pose vector becomes $\mathbf{x''}_t = [\mathbf{x_t}, \dot{\mathbf{x}}_\mathbf{t}, \ddot{\mathbf{x}}_\mathbf{t}]$ with the time propagation Equation 2.19 where α, β, and γ are scaling factors,

$$\mathbf{x''}_{t+\Delta t} = \begin{bmatrix} I & \Delta t \cdot I & \frac{\Delta t^2}{2} \cdot I \\ 0 & I & \Delta t \cdot I \\ 0 & 0 & I \end{bmatrix} \mathbf{x''}_t + \begin{bmatrix} \alpha \cdot I \\ \beta \cdot I \\ \gamma \cdot I \end{bmatrix} \mathcal{N}_\mathbf{t}(\mathbf{0}, \Sigma) \quad . \tag{2.19}$$

Learned Motion Models

There are three main classes of approaches to this motion model. The first is to use a database of recorded motion data to learn the probabilities of a state transition table [Klinger and Arens, 2009]. These probabilities can be used to propagate states to their most likely successors. The second approach is based on an understanding of human motion to improve tracking performance for learned classes of motions [Wren and Pentland, 1999]. The use of exemplar motion capture sequences as a prior to particle propagation is also presented as a motion prior by Vondrak et al. [2013]. Particle states are propagated by finding the k nearest neighbors (closest states) in the database and weighting their successors from the training sequences. The third involves the use of convolutional neural networks (CNN), and will be discussed in more detail in Chapter 5.

As all of these approaches are based on recorded data instead of first principle assumptions, they are only valid when the motion of the model somewhat matches the motion present in the recorded database. For this reason, we use learned models both to detect keyframes and generate pose hypotheses. The particle filter is still required to interpolate between keyframes.

Rigid Body Physical Simulation

With the addition of contextual information such as the position of the ground plane [Duff et al., 2011; Vondrak et al., 2013] and of other objects in the scene, as well as a few assumptions about properties of the object being tracked, we can use rigid body physical simulation to propagate particles forward in time. Skeletal simulation is possible with an appropriate kinematic model. Brubaker et al. [2010] propose a model for a simplified version of the lower body and represent other kinematic joints as Markov models.

To obtain a more varied particle set as well as take into consideration external forces, we need to add some perturbation to the particle states. These perturbations are analogous to the addition of the $\mathcal{N}_{\mathbf{t}}(0, \Sigma)$ term in Equations 2.15 and 2.18. Duff et al. [2011] have

come up with three ways to add such perturbations in a meaningful way. The first is to add a random variation, drawn from a Gaussian distribution, to the pose predicted by the simulator. The second method is to apply a set of randomized external forces to the simulator that will impact the predicted state. Lastly, both noise types can be combined to obtain a more varied particle set.

If simulation is used to propagate states and no forces are applied to the model, we obtain a passive motion [Vondrak et al., 2013, 2008] where only gravity pulls the skeleton down. Active motion models require the addition of virtual forces to pull the model towards a target state [Wang et al., 2013]. To determine which state the model should be drawn towards, an understanding of the purpose of motion is necessary. This understanding comes from a database of motions learned from motion capture sequences [Vondrak et al., 2013, 2008].

The main issue when looking at physics simulation is that in addition to the pose we are interested in, we also need to keep track of the forces acting on each limb. These forces, commonly encoded as screws, include both linear and rotational forces applied to each limb. This greatly increases the pose space to around 100 DOF for the human model we are using. Based on the results of the experiments presented in Section 4.2, we find that the tracker becomes unstable when the number of DOF exceeds 50. The high number of DOF required to keep track of forces acting upon the model thus constitutes too large of a pose space for accurate tracking and would lead to instability. For this reason, we will need to restrict our approach to using the motion models described previously that do not change the size of the pose space as much.

2.4.4 Particle Resampling - Converging towards the solution

Particles need to be resampled either at each time step, or when the number of efficient particles falls below a given threshold, depending on the strategy used. Particle resampling is done in order to prevent either one, or a small subset of the particles, from overpowering

the other particles and controlling the result. When this happens, the particles do not represent the space around the correct solution well. To fix this, we can create a new set of particles by moving the previous set of particles in the vicinity of the computed solution. Multiple approaches to accomplish this are presented by Li et al. [2015].

As mentioned by Li et al. [2015], the selection of an appropriate strategy depends on the application. For this reason, we explore this topic experimentally in Chapter 4 by presenting experiments to explore the differences between the following resampling strategies: multinomial, residual, systematic, stratified, and an hybrid residual systematic approach. We also include the original uniform combing resampling method presented by Salmond and Gordon [2005] and show experimentally that this classic approach outperforms the other methods presented by Li et al. [2015] for the case we are interested in.

To somewhat combat particle impoverishment, we have added a slight modification to the original combing method. The version we use is very similar except that the particles are sorted in order of increasing weight and a configurable percentage of the lowest weighted particles get regenerated randomly (Monte Carlo sampling) or through the use of a CNN. The combing is done in the remaining particles. Figure 2.26 shows a graphical representation of the particle selection process in a similar way as the original from [Salmond and Gordon, 2005]. This is an extreme case with 14 particles where half of the particles are randomly reinitialized. The remaining 7 particles are selected with uniformly spaced combs. Sorting the particles by weight makes it easier to prevent low weight particles from being selected by the combing process. The downside to this modification is the added computational complexity required to sort the particles in the first place.

2.4.5 Result Computation

While perhaps the most crucial part, it is also the simplest. The result is computed as a weighted sum of the particles with their associated weights, as shown in Equation 2.20,

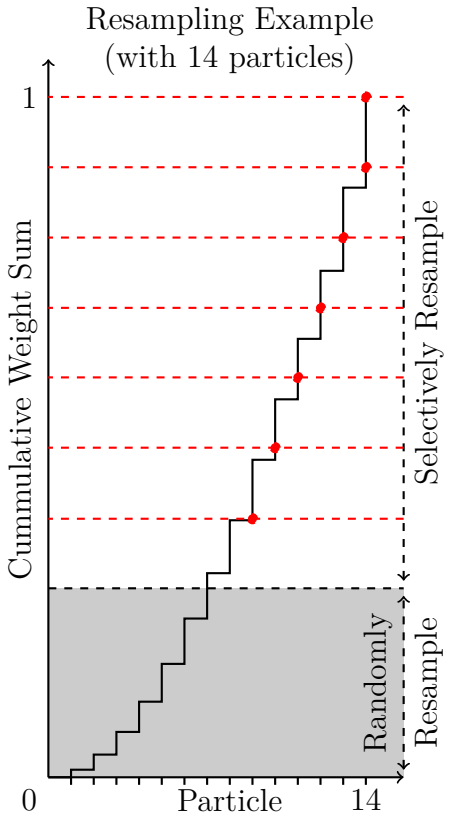
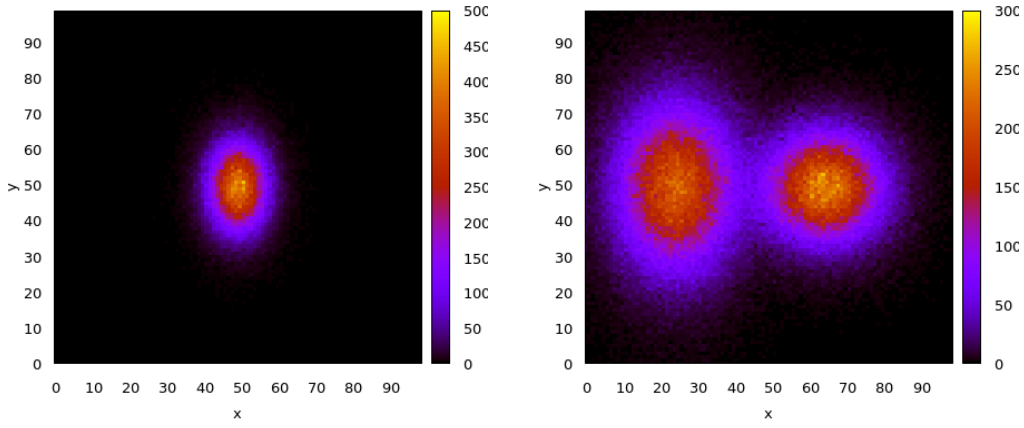


Figure 2.26: Uniform comb particle selection scheme from Salmond and Gordon [2005] with modification to allow random resampling of lowest weight particles.

where w_t^k is the same as in Equation 2.14,

$$\mathbf{r_t} = \sum_{k=0}^{N-1} w_t^k \mathbf{x_t^k} \quad . \tag{2.20}$$

This approach works well when the distribution of particles and their weights support a single hypothesis. Figure 2.27 provides two possible particle density distributions for a theoretical two dimensional particle filter that allows us to visually observe the effect of ambiguity. An unambiguous case where the weighted average works well is shown in Figure 2.27(a). In ambiguous cases like the one shown in Figure 2.27(b), the particle weight density distribution supports multiple hypotheses and the weighted average would provide an incorrect solution between the two hypotheses. Experiments in Chapter 4 show that ambiguity isn't as simple to detect as in Figure 2.27(b) in real world applications, but that good priors can result in convergence in certain ambiguous cases. A specific ambiguous case is presented in Figure 4.30, where we use a two degree of freedom model and plot the metric value over the entire pose space. We see that the distribution is not as easily resolved as that of Figure 2.27(b).



(a) Unambiguous Case (b) Ambiguous Case

Figure 2.27: Two dimensional example particle weight density distribution with and without ambiguity

Chapter 3

Experimental Setup

This chapter focuses on discussing the technical characteristics of the framework we implemented to test our assumptions.

3.1 Implementation

The goal of this section is to describe the system used for the experiments presented. This is both to make it possible to reproduce results as well as to provide context on the environment used.

3.1.1 Software

Being a proponent of the open-source initiative, the initial goal was to only rely on open-source software if possible. The tracking system runs on a Linux operating system and all tools used were open-source. The only issue that was encountered is that the HumanEVA dataset is distributed as a MATLAB program and matrices. Thankfully, the data could be converted using GNU Octave [Eaton and al., 1988].

Most of the tracking program is written in C++ and uses different libraries for the different tasks. Most notably, we rely on OpenGL [SiliconGraphics, 1992] to render images

and OpenCL [KhronosGroup, 2009] for GPU computation. This choice was based on the use of OpenCL and OpenGL interoperability to allow OpenCL to use the OpenGL pixel buffers directly without the need to transfer the rendered images to the CPU memory and back to GPU memory for OpenCL. If we had to rewrite the code base, we would upgrade to Vulkan [KhronosGroup, 2016] to enable homogeneous render and compute buffers, which would simplify the process.

The deep learning parts of the tracker depend on Keras [Chollet, 2015], relying on the Tensorflow [GoogleBrainTeam, 2015] backend. Most of the deep learning is written in Python, as the C API for Keras was found to be unreliable. We then embed the Python scripts in our C++ program to facilitate the transfer of images and matrices between C++ and Python. The small scale experiments presented in Chapter 4 do no rely on the CNN, as the particle filter can track the model without it and the goal is to tune the various parameters of the filter. When starting to experiment with human models in Chapter 5, we quickly realized that the mapping between the silhouette and the pose is less linear than what we see with the small scale model. We propose the use of a convolutional neural network as an additional estimator of the inverse mapping between silhouette and pose, with the particle filter still being used to track the solution over time. Specifics about the architecture and training of the network are thus presented in Chapter 5, where the filter needs the CNN to replace particles the diverge from the solution.

3.1.2 Hardware

The tracking system runs on a single computer with an Intel Core i7 6850K CPU clocked at around 4GHz and 128 gigabytes of system memory. The GPU is an NVidia GeForce GTX TITAN Black with 6 gigabytes of memory, graciously provided by NVidia. System memory can be used by SADB to store large amounts of data, without the need to read and write to physical media. The only other hardware component worthy of note here is a Sony alpha 6000 that was used to record the sequences in front of the green screen.

3.2 Companion Website

As a lot of the experiments required to test our hypotheses involve image sequences, the most practical way to visualize the data is to render video clips. As videos cannot be included in the paper thesis document, we provide summaries of the results here and opt to create a web page to present the video results. This page is located at `http://cim.mcgill.ca/~olivier/thesis/`. In the interest of keeping the thesis succint, the companion website also presents additional experiments that are not thoroughly discussed in the thesis as they are closely related, but somewhat outside the scope of the thesis.

The page is structured as a list of the experiments, separated into four main sections. Clicking on the title of an experiment will open a section containing the data. In many cases, drop-down input selections are available to see different results. The first video at the top of the page is the result of the chroma-keyed sequence, which is a good representation of the capabilities of the tracker. Figure 3.1 shows a screenshot of the top of the website.

The first section presents all the plots from the evaluation of the silhouette comparison metrics. The results of all these plots are summarized in Table 2.2. Between the three experiments, there are almost a 1000 plots that can be viewed in this section. The metric behavior as a function of distance in pose space experiment contains plots from Figure 2.17, as well as all of the other combination of parameters and cases evaluated. Similarly, the generalized metric behavior as a function of distance in pose space plots show how the metrics behave given randomized poses. Finally, the metric behavior when moving along each DOF shows how each DOF affects the metric value.

The second section contains all the results related to the small scale experiments. Most of these experiments follow the order in which they are described in Chapter 4. The only experiment not described in this document is the root motion experiment. This experiment tests the motion of the model itself, instead of changes to the pose of the model. It is thus a bit outside of the focus of the thesis. The results are also unsurprising, as it works as expected.

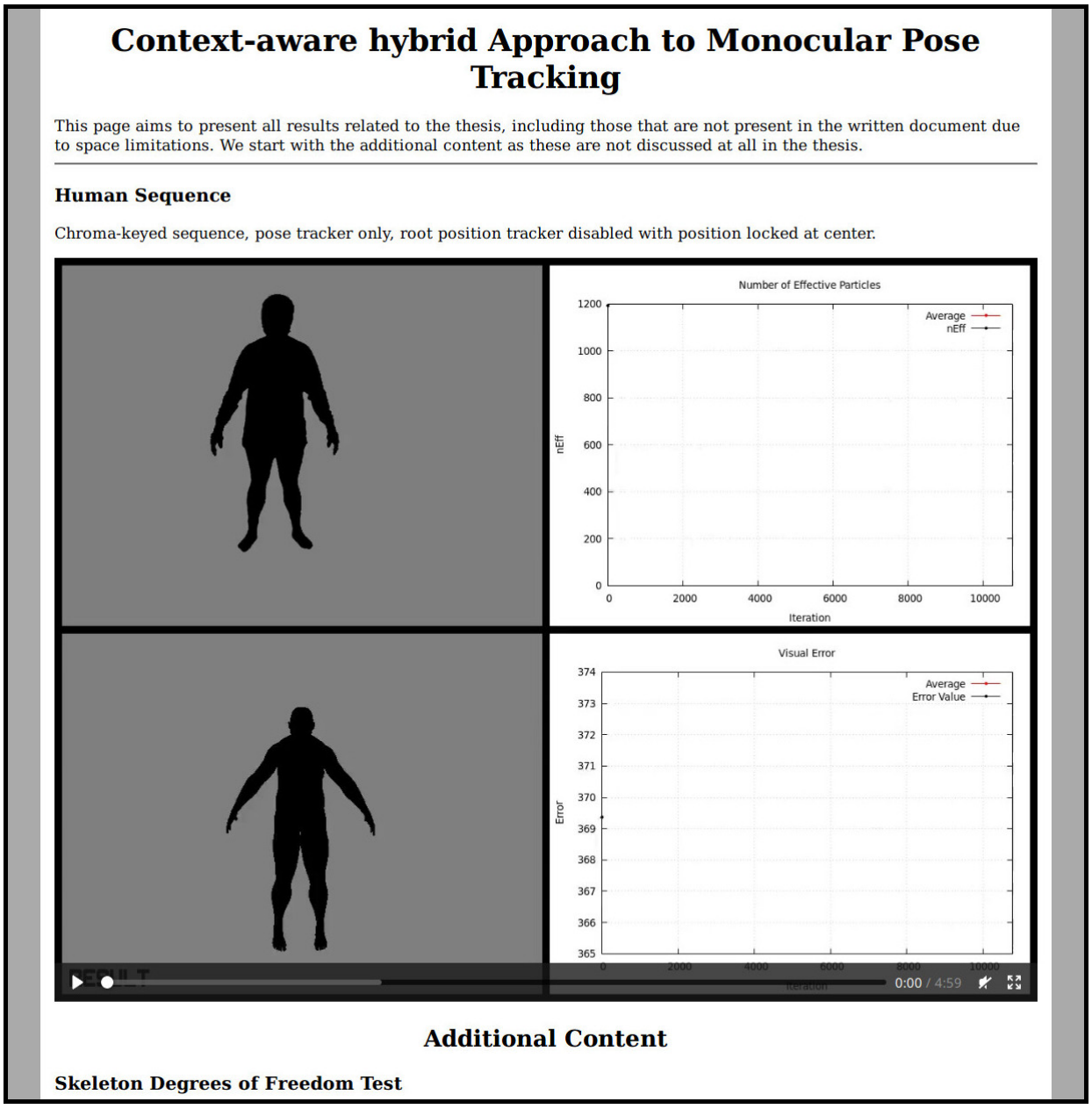


Figure 3.1: Screenshot of the top of the companion website, showing the format of the page.

The third section presents the results of the human experiments, starting with the synthetic experiments, going over early experiments of the chroma-keyed sequence and ending with the HumanEVA experiments. These experiments follow the flow of Chapter 5.

The final section contains experiments that do not fit into any of the other sections and is called "Additional Content". These are a lot of the experiments made to test different components of the framework itself. The skeleton DOF test aims to test the deformable human mesh model and visualize which DOF corresponds to which limb. The physics tests show early experiments with the usage of physics engines by using a kinematic model with capsules as limbs. The 2D particle filter test allowed us to see the particle distribution and the particle weights. The camera calibration experiments were an interesting way to see if we could align our virtual OpenGL world with the real world camera calibration by setting up a simple augmented reality demonstration. The deformation and shader tests were used to test shader programs to make sure we could render all of the required data properly. We also made sure that we could read the HumanEVA dataset properly by rendering the marker locations. This renderer demonstrated some issues with the HumanEVA dataset, which motivated our decision to use the 2D reprojection error metric when we evaluate the performance of our tracker with this dataset. The other video in this section is the experiment we did with multiple cameras. We used this to prove that our approach can scale to more complex capture systems by integrating multiple sources of information.

While the main goal of the webpage is to show results that could not be included in this document, we hope it also demonstrates a level transparency to show what the conclusions we make are based on.

3.3 Situational Awareness Database

Throughout the framework, we need to handle different subproblems that each require different representations. By representations, we mean ways to encode information into a useful

format so that it can be used by different components consistently. Some of the components rely on time series data, while others rely on geometric information, and others on pixel intensity images. In order to do any meaningful computation on this data, we need to address the representation problem.

We briefly present in this section the Situational Awareness DataBase (SADB) as a novel solution to represent arbitrary data in a consistent semantic manner. While there may have been other, simpler, solutions to work around the representation problem for the pose tracker, a solution like SADB is required for larger scale projects such as CHARM [Cormier et al., 2015], where a large number of agents need to work in collaboration towards a common goal. It also presents interesting oportunities to separate the computational load between multiple compute nodes. Some work has been done to port SADB to a ROS [Quigley et al., 2009] node as part of the CHARM project, but the current implementation remains a standalone server. Although this is a significant implementation component of the overall pose tracking solution we propose, we do not examine SADB here in detail as we aim to focus on the tracking aspect itself.

The SADB system is designed as a blackboard system, where all clients can read or write. The implementation loosely follows the guidelines presented by Corkill et al. [1987], but also borrows from modern NoSQL database design [Foundation, 2005; Sanfilippo and Noordhuis, 2009; WEB, 2009].

Figure 3.2 provides a block diagram summary of the basic architecture of the SADB system. To be consistent with the nomenclature used in most of the blackboard architecture literature, the clients are called knowledge sources or sinks, and the general SADB Controller is denoted as the control shell.

At its core, the SADB control shell is implemented such that information can be accessed on a real-time basis by both programs and human operators. Two access interfaces are presented, a native SADB interface, which implements all of the data access functions, and a simplified HTTP interface which allows humans to more intuitively monitor the operation

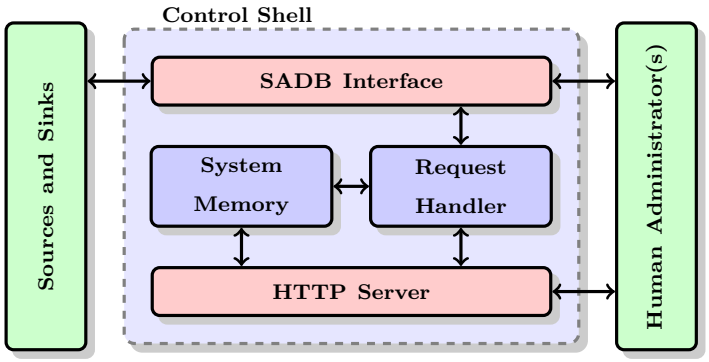


Figure 3.2: Block diagram of the Situational Awareness DataBase (SADB) framework

and state of the database.

As agents produce and consume information at different rates, synchronization between these agents can quickly become an issue. To solve this problem, values stored on the blackboard are discretely time-coded, and agents may request information at arbitrary timestamps through a variety of methods.

The simplest way is to return the value at the timestamp closest to the requested one. This essentially creates a piece-wise constant function from the data points. The better, albeit slightly more computationally expensive, way to answer those requests is by using interpolation (or extrapolation) to generate intermediate values, based on recorded values. A variety of interpolation algorithms are presented and can be selected by the clients, based on the type of requested information. Other timestamp access approaches are also available such as simply retrieving the latest value or the value either before or after a given target timestamp. This handling of time-based information provides a variable granularity that allows each client to receive information at their own required rates.

A key point of SADB is that it can be used with visualization tools to see the state of the system at any timestamp and to play back events. Figure 3.3 shows how a laser scan of an environment stored in SADB can be retrieved and rendered. Similarly, Figure 3.4 shows a skeleton rendered from pose data also stored in SADB. While these two pieces of information are recorded by different algorithms, both can be stored and referenced in the

same coordinate system and time code, which means that they could be rendered either together or separately. This serves two purposes. First, it can be used to inspect events after they have occurred by looking at all of the available information related to the event. Second, it facilitates the analysis of the algorithms individually by enabling one to isolate only the parts of information generated by the algorithm one is evaluating. Figures 3.3 and 3.4 are examples taken from the CHARM project.

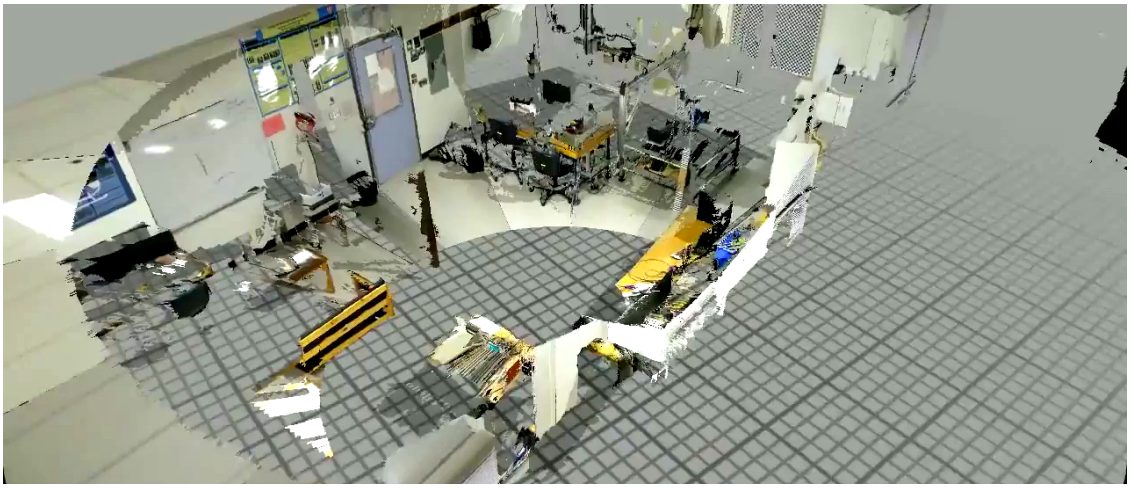


Figure 3.3: Screenshot of rendered geometric data read from SADB.

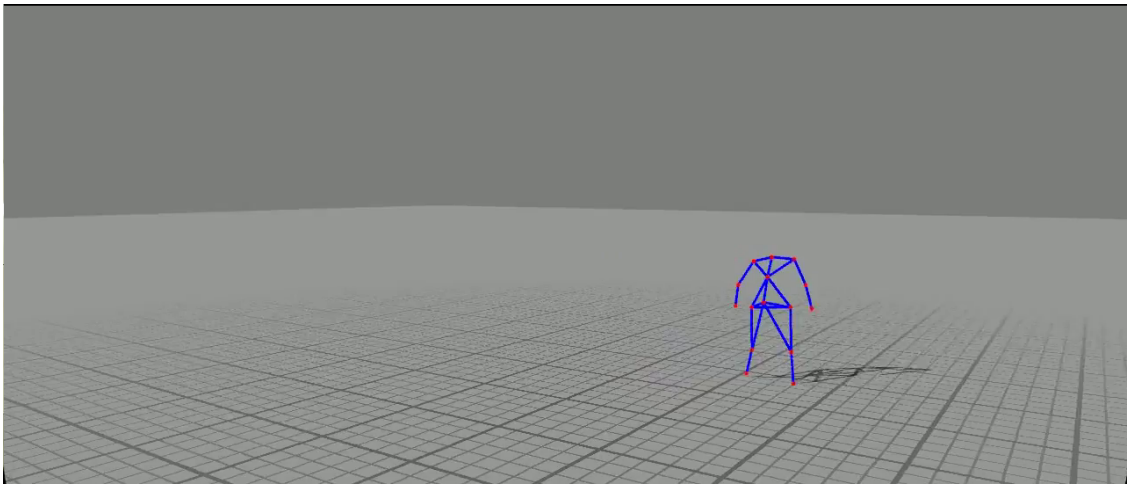


Figure 3.4: Screenshot of rendered pose data read from SADB.

As we focus on the tracking problem itself, no further discussion of the technical details of SADB are presented in this thesis. We invite the reader to consult an earlier publication

in the conference on Computer and Robot Vision [Cormier et al., 2015] to learn more about this system. The system in its entirety is publicly available at `http://www.cim.mcgill.ca/~apl/database/sadb/`.

Chapter 4

Small Scale Experiments

This chapter describes and reports the results of small scale experiments designed to validate our main assumptions and determine if our framework can generalize to human models. The results from this chapter are crucial for tuning the various parameters of our model and, in turn, make it possible to do the experiment with a complete human model that will be presented in the following chapter.

To begin experimenting with various combinations of propagation, resampling, and evaluation methods, we devise a set of fully controlled small scale experiments. These experiments consist of tracking a virtual snake-like shape evolving over time. Figure 4.1 shows a diagram of the model that will be used, with the mesh drawn in blue and the skeleton in red. This configuration enables arbitrary variations in the number of DOF in the skeleton, by adding or removing bones, without changes to the mesh model.

As shown in Figure 4.2, bones in our model are defined by 3 parameters: the index of the parent joint p, the distance from the parent joint l, and the relative orientation with respect to the orientation of the parent q. This simple bone model allows the representation of any articulated structure, from the snake-like model we are using for our small-scale experiments, to the complete human model that will be described in the next Chapter.

It must be noted that any rotation around the main axis is ambiguous if none of the other

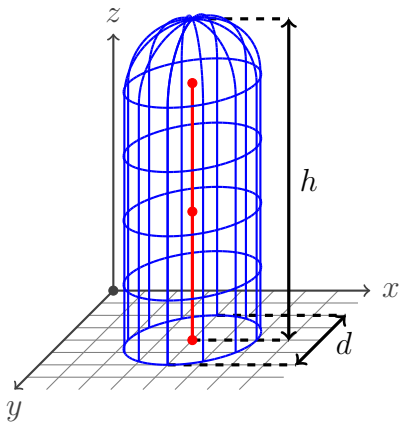


Figure 4.1: Graphical representation of the snake-like model that will be used in the small scale experiments.

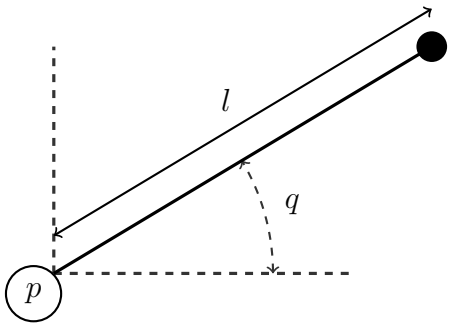


Figure 4.2: 2D graphical representation of three parameters of a bone.

DOF are in a configuration that make it unambiguous. This makes this small scale problem harder, but also makes it more similar to the kind of ambiguity that occurs naturally with human silhouettes, where the distinction between front and back is ambiguous.

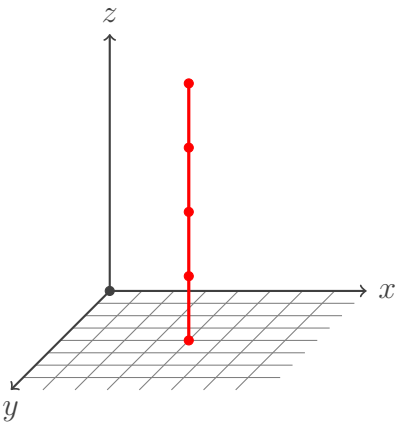
Del Rincón et al. [2011] propose the use of two distinct filters, one to track the root position of the model and the second to track the pose. Our experimental configuration assumes that this type of approach will be used, which allows us to simplify experimentation by keeping the position of the root joint constant. This enables us to experiment on the pose tracking particle filter exclusively.

Figure 4.3 provides diagrams of the cases that will be used in the small-scale experiments. Each of these cases introduces specific issues that can arise in real sequences.

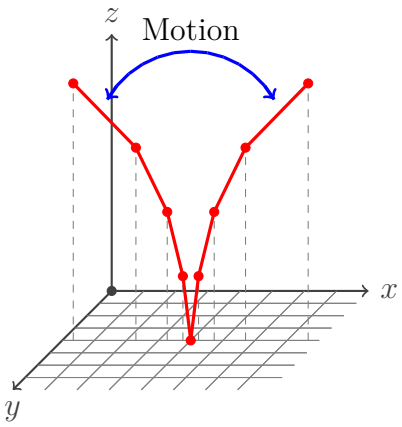
4.1 Stability Experimentation

This first experiment consists of using the basic configuration (Figure 4.3(a)) and executing the tracking algorithm on a motionless sequence to measure the amount of drift and error that accumulates over time. This allows us to experimentally determine approximate values for the various parameters of the tracker and compare the stability of the different motion priors. We arbitrarily choose to run the tracker for 100 frames before collecting measurements. These collected measurements consist of the following:

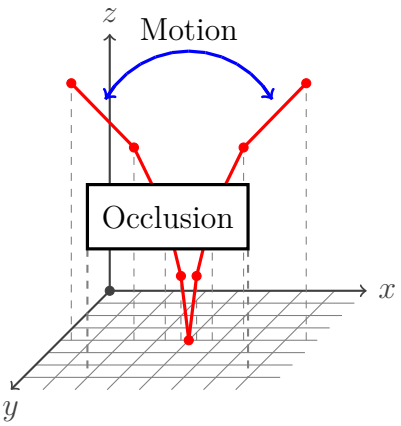
- **Image Data:** Three images are recorded in order to visually assess the tracker's performance. Figure 4.9 shows what these images look like for the small-scale experiments.
  - Input image: a copy of the rgb image that is fed to the tracker. In the small-scale experiments, we directly render the segmented image, so only the segmented image is recorded.
  - Segmented image: the binary segmented image of the input, which allows us to determine the cause of some problems such as mislabeling or occlusion.



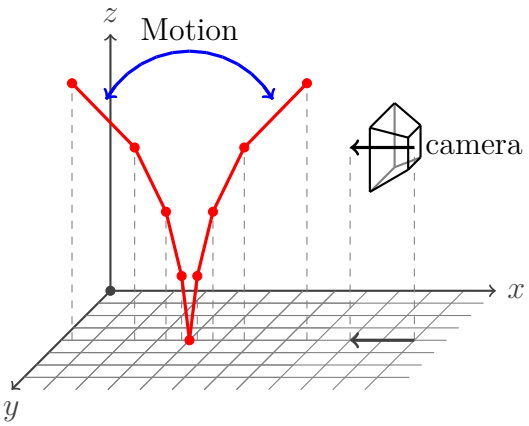
(a) Basic Configuration



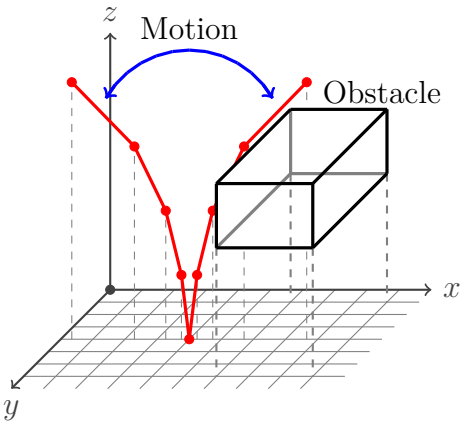
(b) Simple Motion



(c) Occlusion Configuration



(d) Ambiguous Configuration



(e) Restricted Pose Space Configuration

Figure 4.3: Small scale experiment configurations

  - Output Image: the output pose from the tracker is rendered to obtain a graphical representation of the output pose.

- **Pose Error:** Measured in degrees as a 2-norm (Euclidean) distance in pose space between the output from the tracker and the ground truth.
- **Visual Accuracy:** Value of the visual metric computed between the input image and the rendered output from the tracker.
- **End Effector Position:** Tracked position of the model's endpoint
- **End Effector Error:** Distance between the tracked end-effector position and the ground truth.
- **Particle Pose:** The pose and value of the metric of each particle is recorded. This information may be useful as we can look at histograms of the particle distribution to determine the cause of certain issues such as ambiguity.
- **Particle Statistics:** Statistics such as particle dispersion and the number of efficient particles provide some information about how far particles are distributed with respect to the tracker's output and how many particles contribute to the solution, based on their weights.

The first trials consist of varying the amount of particles used for the particle filter and recording the average visual accuracy (metric value of the tracker's output) and pose error over the tracking sequence. Figure 4.4 provides plots of these results for a Gaussian prior. In these plots, the black dots represents the recorded points, and the red curve is a fit to those points. The fitted curve is not accurate, but provides a general representation of the behavior. As expected, there is an increase in the visual accuracy and decrease in the pose error as more particles are considered. This is expected because a larger particle set provides a better coverage of the metric hypersurface around the solution, which, in turn, leads to a more accurate evaluation of the local maximum.

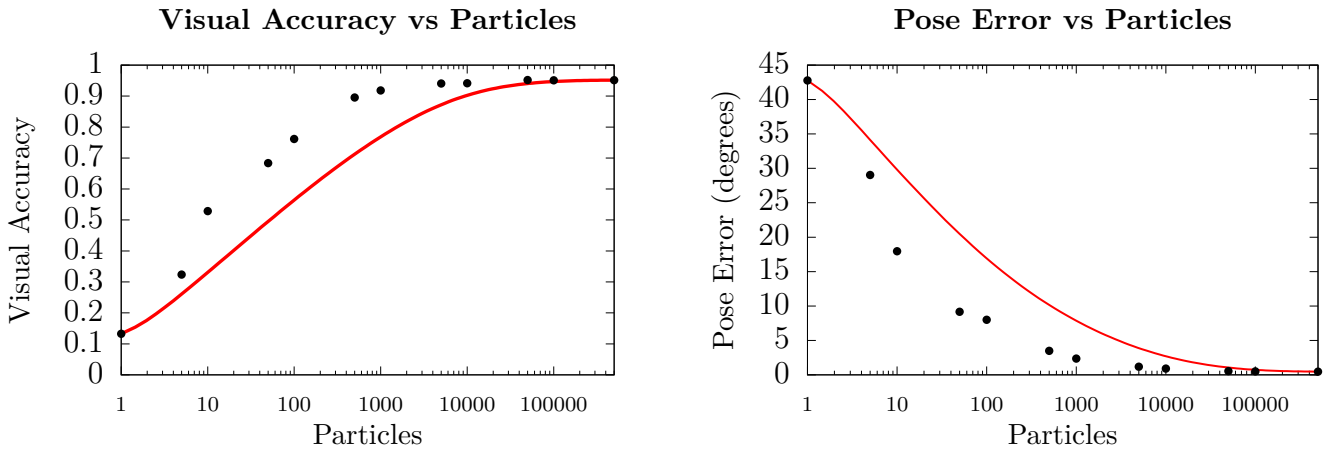


Figure 4.4: Stability experiment results for the small scale experiments with a Gaussian prior.

Figure 4.5 shows the results for the same Gaussian prior, with the addition of a gradient descent refining of the particle states. The results obtained with this approach are significantly worse than the results obtained from the Gaussian prior alone. This is due to the gradient descent getting stuck in local maxima of the visual accuracy and not converging toward the global maximum.

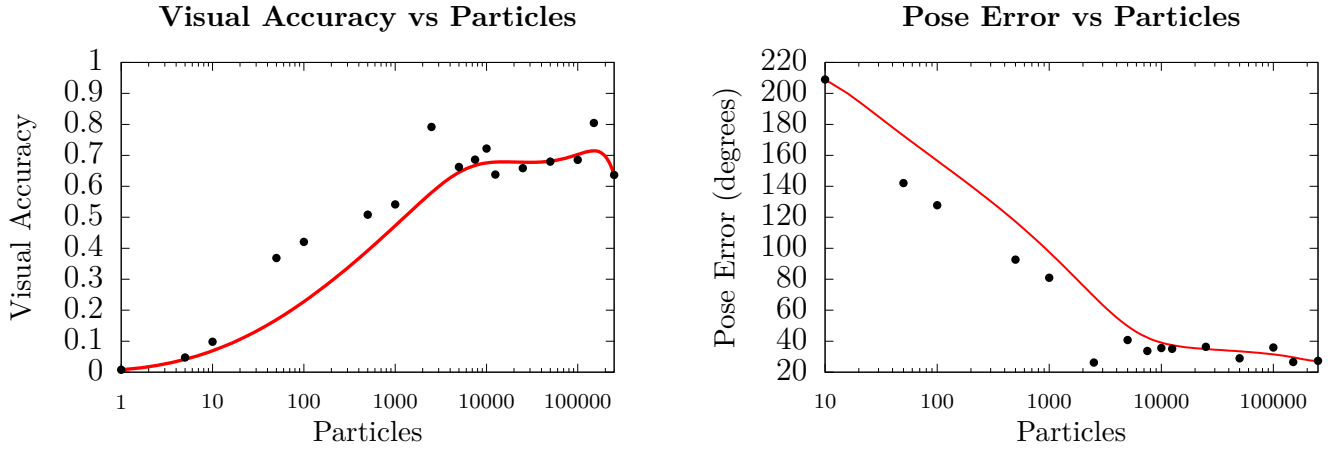


Figure 4.5: Stability experiment results for the small scale experiments with a Gaussian prior and gradient descent. Note here that the plot starts at 10, as the tracker fails to converge for smaller numbers of particles.

The two next figures (Figures 4.6 and 4.7) provide results for first and second order

motion models respectively. With a large number of particles, the results of both are similar to those of the Gaussian prior, but the pose error is higher for smaller numbers of particles.

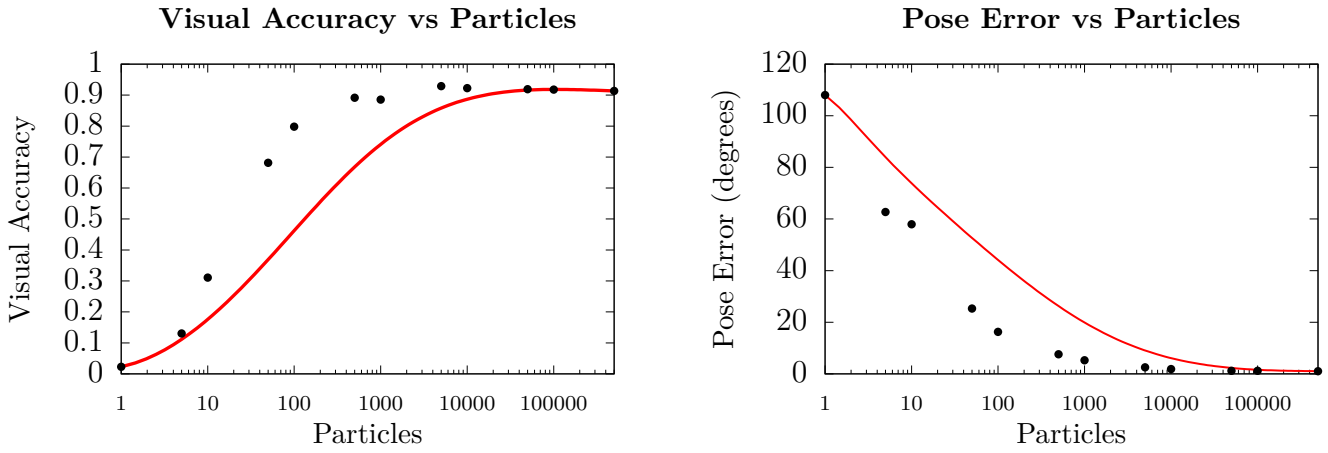


Figure 4.6: Stability experiment results for the small scale experiments with a 1st order motion model.

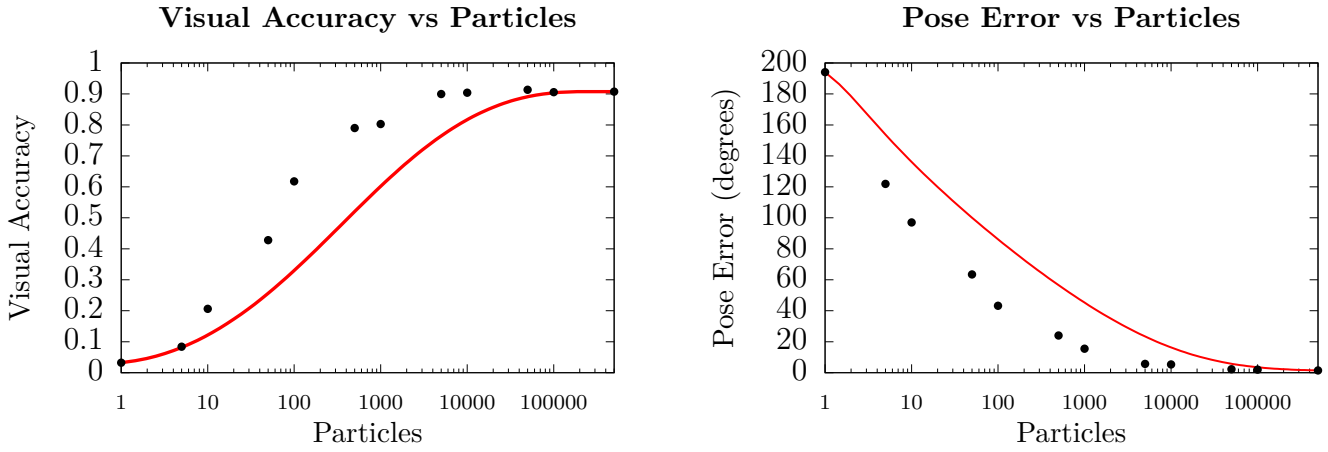


Figure 4.7: Stability experiment results for the small scale experiments with a 2nd order motion model.

The next parameter we can explore is the effect of is the standard deviation of the Gaussian distribution. Figure 4.8 shows that the pose error grows linearly with the standard deviation. This is expected as larger values means that the distance between particle states will grow more rapidly, which in turn means that less particles will lie in the vicinity of

the correct solution, thus resulting in a poorer approximation of the shape of the metric hypersurface around the solution. While the results presented here are not quite exciting as there is no motion, running this test for the next experiments should prove more insightful as we expect motion speed to induce maxima and minima in the visual accuracy and pose error plots respectively. This is because when the motion speed falls within the standard deviation, the motion will be better captured as the particles will be propagated to the correct position.

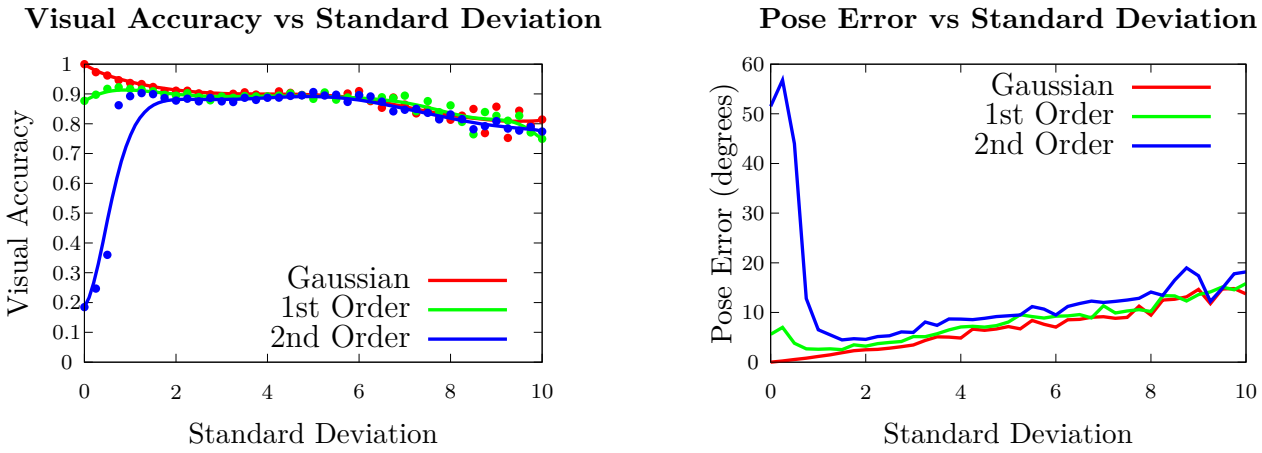


Figure 4.8: Influence of the standard deviation on the stability experiment results for the small scale experiments without motion.

The reason why the second order motion model starts out with lower accuracy, and conversely higher pose error, is due to the fact that only the pose was initialized with ground truth values. This was done so that all three motion models start with the same information. We can, however, see that increasing the standard deviation allows the tracker to explore enough candidates to find and lock onto the correct solution. This again shows the importance of choosing appropriate parameters and the need for a good initialization.

The most important result of this first set of experiments is that the visual accuracy is inversely proportional to the pose error. While this was predicted by our study of the metrics in Section 2.3, it shows that the assumptions we made hold for models other than humans.

4.2 Base Motion Experiment

Figure 4.3(b) shows a graphical representation of the configuration used for this experiment and Figure 4.9 shows a selection of five frames representative of the sequence that will be used for the current experiment. This experiment provides the best case scenario for testing motion priors.

frame 1 frame 50 frame 62 frame 70 frame 100

Figure 4.9: Selected frames from the simple motion sequence.

One experiment we can conduct with the motion sequence is to vary the number of degrees of freedom. We can do this by keeping the same mesh model, but adding more bones to the model shown in Figure 4.1. This adds more degrees of freedom to the model and allows more varied motion. Figure 4.10 shows the results of such an experiment. There are two interesting things to note about this. First is the general tendency of the visual accuracy to decrease as more DOF are present and inversely as the pose error increases. This effect is not as pronounced as initially expected. This is likely due to the redundancy in DOF, where an error at a given DOF can be compensated by the next DOF in the chain. The second is the behavior with smaller numbers of DOF caused by improper choice of standard deviation, where small changes in the pose vector of the snake with a low number of DOF causes larger pixel changes. The other important note is that the second order prior completely fails when the number of DOF is higher than fifty. It is currently not known why this occurs, but as we are dealing with a 36 DOF model, this issue is of low priority and left for future work.

Looking at Figure 4.11, we see a clear demarkation in the visual accuracy between the three priors, where the second order prior is approximately 10% better than the first order prior, which is in turn approximately 10% better than the Gaussian prior. While this result

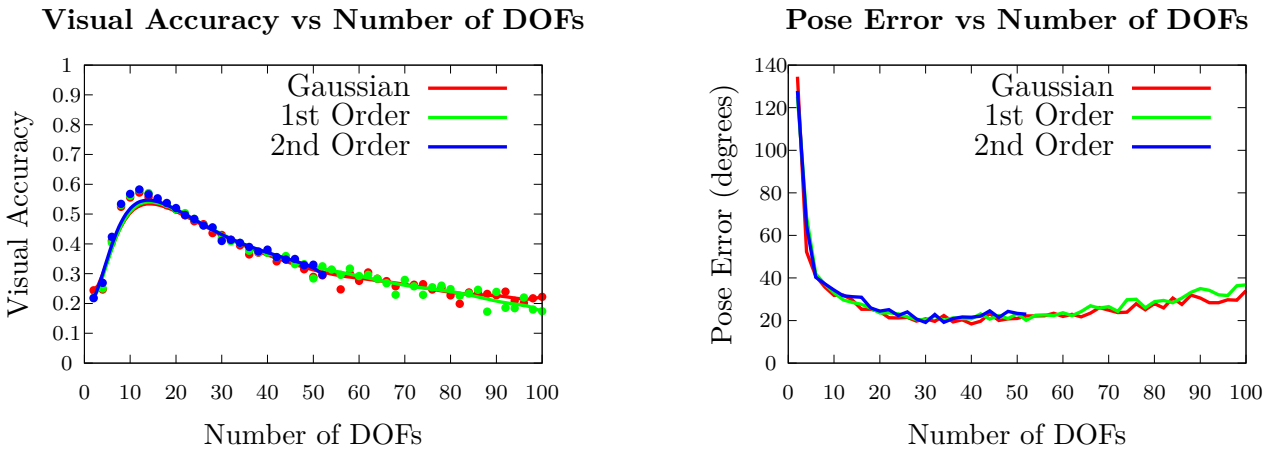


Figure 4.10: Motion Experiment with different number of degrees of freedom.

carries onto the pose error, the difference between the Gaussian and second order priors is only around 0.5 degrees, or a difference of around 3%.

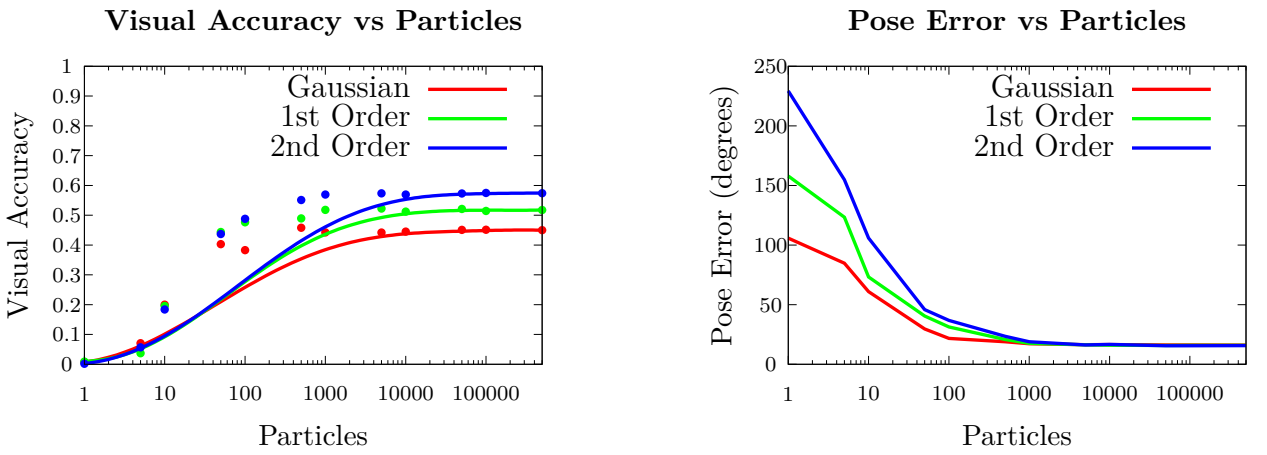


Figure 4.11: Stability experiment results for the small scale experiments with a Gaussian prior and Gradient descent

As discussed earlier, the standard deviation sweep provided in Figure 4.12 shows a peak where the standard deviation is large enough to allow the particles to properly cover an area large enough that it includes the solution, but not so large that the particle density becomes too low.

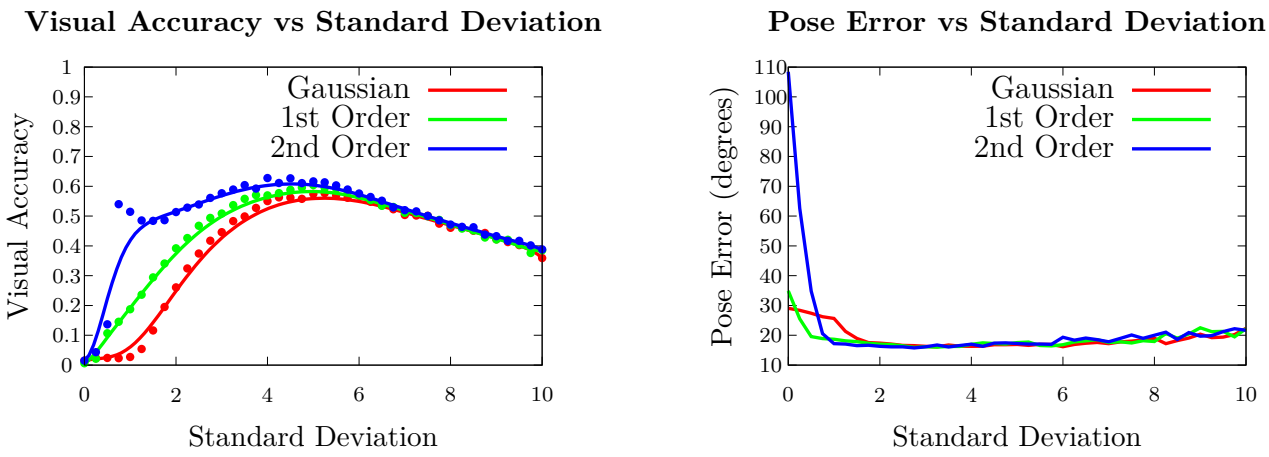


Figure 4.12: Influence of the standard deviation on the stability experiment results for the small scale experiments with simple motion

4.3 Occlusion Experiment

For this experiment we can look at two distinct cases. First is positive occlusion, where the target is occluded by an object that is detected as foreground. This would mostly happen with occluding objects that move during the sequence and cannot be included in the background model fast enough. The second type is negative occlusion, where the target is occluded by an object detected as background. This would happen when the background model contains objects that lie between the camera and the tracked object. The sequences are generated in the same way as the motion sequences, with the addition or removal of a rectangle, as seen in Figure 4.13. We use the values for the standard deviation that were found to provide the best results in the motion experiments.

In both cases, the occlusion has an impact on the silhouette similarity metric proportional to the area covered. While we explore the issue of occlusion here, the sequences we will use for human testing should contain little to no occlusion. Figures 4.14 and 4.15 show the results of these experiments. It is interesting to see that in this case, as expected, higher order models provide more accurate results by assuming that the rate of change of the pose is continuous, so that the pose will continue its trajectory rather than moving randomly.

Positive Occlusion Negative Occlusion

Figure 4.13: Selected frames from sequences with occlusion

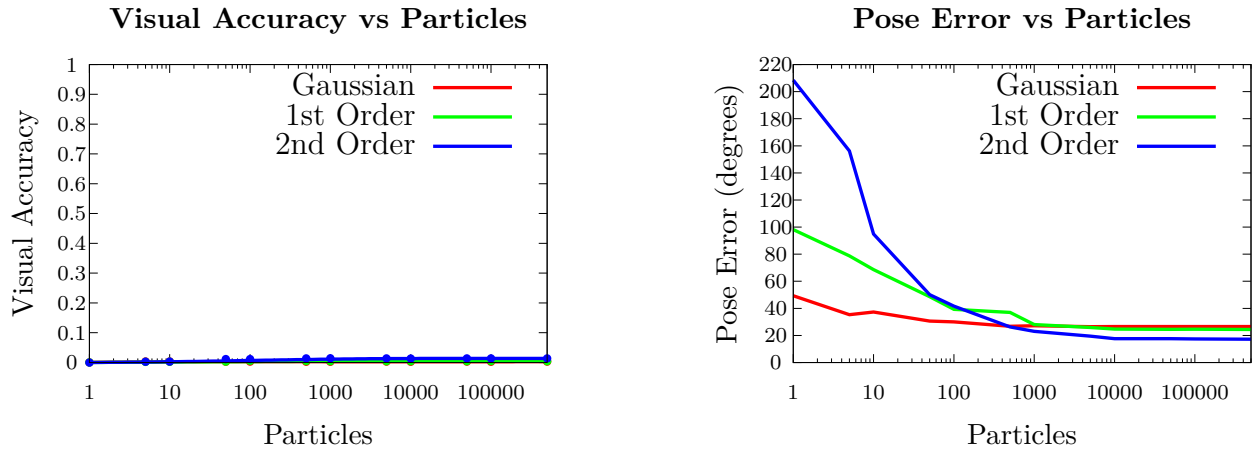


Figure 4.14: Influence of the number of particles on the small scale experiments with simple motion and positive occlusion

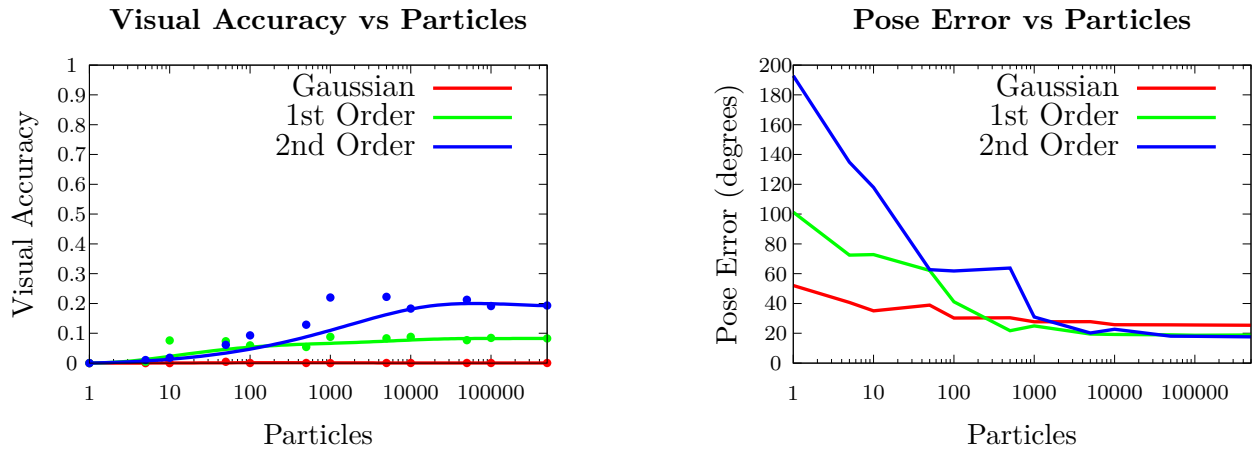


Figure 4.15: Influence of the number of particles on the small scale experiments with simple motion and negative occlusion

4.4 Ambiguity Experiment

As shown in Figure 4.3(d), if the motion of the snake is along the main axis of the camera, there is a lot of ambiguity as very few pixels change. Figure 4.16 show two frames from the ambiguous sequence. Those two frames are equivalent to Frame 1 and Frame 100 from Figure 4.9, with the camera rotated 90 degrees around the z axis. There is a notable scale difference between the two images, but those correspond to the two extrema of the motion. The scale change between consecutive frames is minimal when the snake is upright and gradually increases as it bends either way.

We can start by looking at the particle sweep plots in Figure 4.17 to see that all three motion priors struggle with this experiment. However, the second order motion is better at finding solution that are visually close to the correct solution.

To better understand ambiguity, we can look at the particle distribution in the form of what we will call temporal histograms. These plots, such as those in Figure 4.18, present how particles are distributed over time. As we are looking at a model with 36 DOF, tracked by a particle filter that considers a large number of particles, the amount of data to visualize has a very high dimensionality and is thus hard to present. We look to reduce the dimensionality by

(a) Snake bent toward the back (b) Snake bent toward the front

Figure 4.16: Selected frames from the ambiguous motion sequence

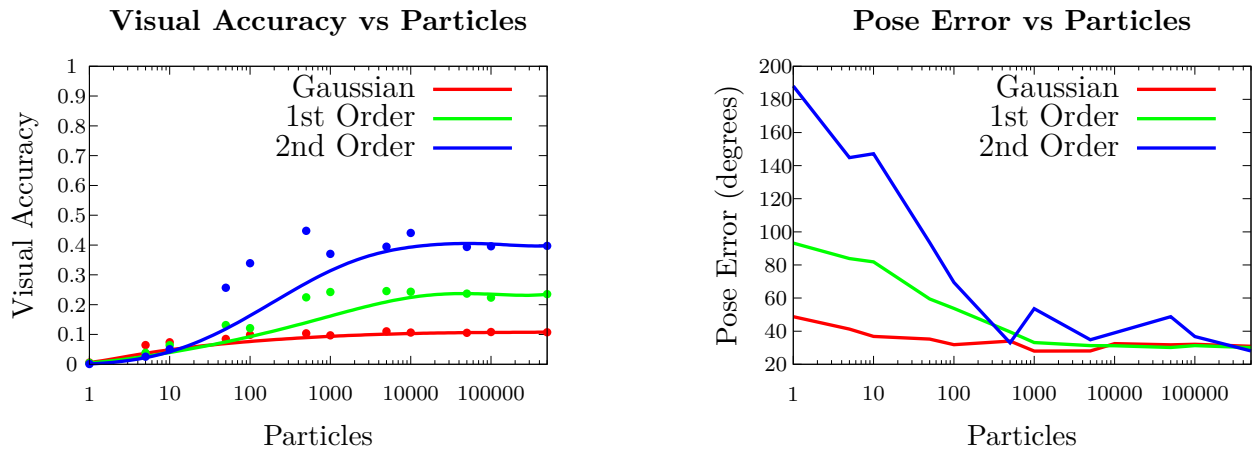


Figure 4.17: Influence of the number of particles on the small scale experiments with ambiguous motion

choosing a different representation. We reduce the 36 DOF to a three dimensional vector by computing the position of the end effector from the pose vector. To present the position of all the particles, we opt to use a histogram representation. We get one histogram for the position of the end effector along each axis. Only the histograms for the X and Y axis will be presented because of space constraints, but all three are available on the companion website. To show how the distributions evolve over time, we stack the histograms horizontally. Because we are stacking them, we cannot draw the histograms with bars, we instead use colors to show the number of particles in each bin. Looking at the plots in Figure 4.18, the horizontal axis is time, measured in frames. The vertical axis is the position of particles in a normalized coordinate system to restrict the range to $[0, 1]$. This is why the position axis has no unit. Each column of the plot is the histogram of the particle positions at the frame number corresponding to the column. The scale on the right of the plots shows how the colors relate to the number of particles in each of the histogram bins. The histograms were normalized so that the bin with the largest number of particles is one.

Before looking at the results for the ambiguous motion sequence described above, we can go back to the sequence without motion to generate Figure 4.18 by plotting the temporal histograms of the particle distributions. In Figure 4.18, the snake does not move and the camera is placed on the Y axis looking straight at the model, so ambiguity is mostly along the Y axis as motion along this axis would be hard to detect. This is seen in the plot for the Y axis as the histograms show that the particles are more spread out. An example with a lot of ambiguity can be seen in Figure 4.24, where ambiguity causes the filter particles to scatter.

While we can visually inspect the histograms and determine along which dimension ambiguity is more pronounced, we can also compute the entropy of the histograms to get a measure of the amount of uncertainty. Figure 4.19 shows the entropy associated with the histograms of Figure 4.18. As discussed earlier, the entropy along the X axis is much lower than that along the Y axis. The entropy spikes after initialization and then decreases as

resampling and propagation help to stabilize and converge toward the solution.

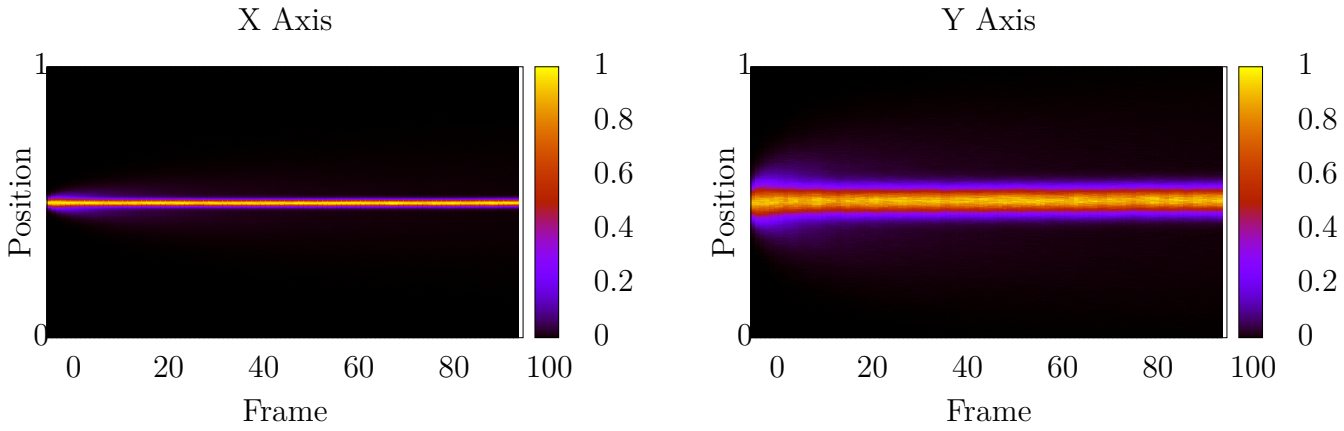


Figure 4.18: Histogram of end effector position of particles for a sequence without motion

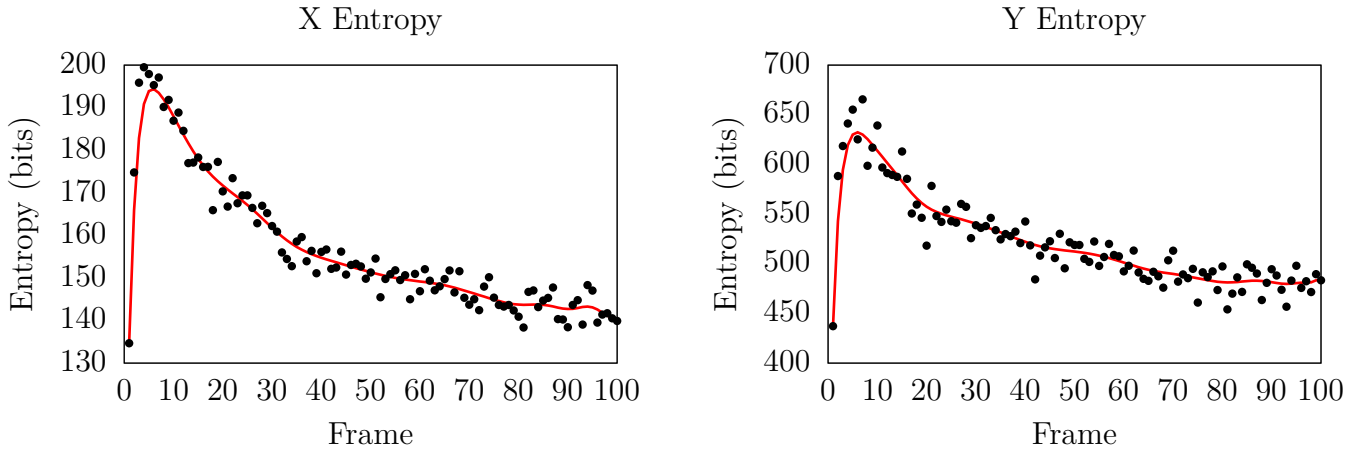


Figure 4.19: Entropy associated with the histograms of Figure 4.18

Looking at a sequence with motion, we see that there is more ambiguity along all axes, but that there is still more ambiguity along the camera's main axis (Y in this case) as evident from the entropy plot in Figure 4.21. Figures 4.20 and 4.21 show the histograms and associated entropies, respectively. These plots stem from using a Gaussian diffusion motion model.

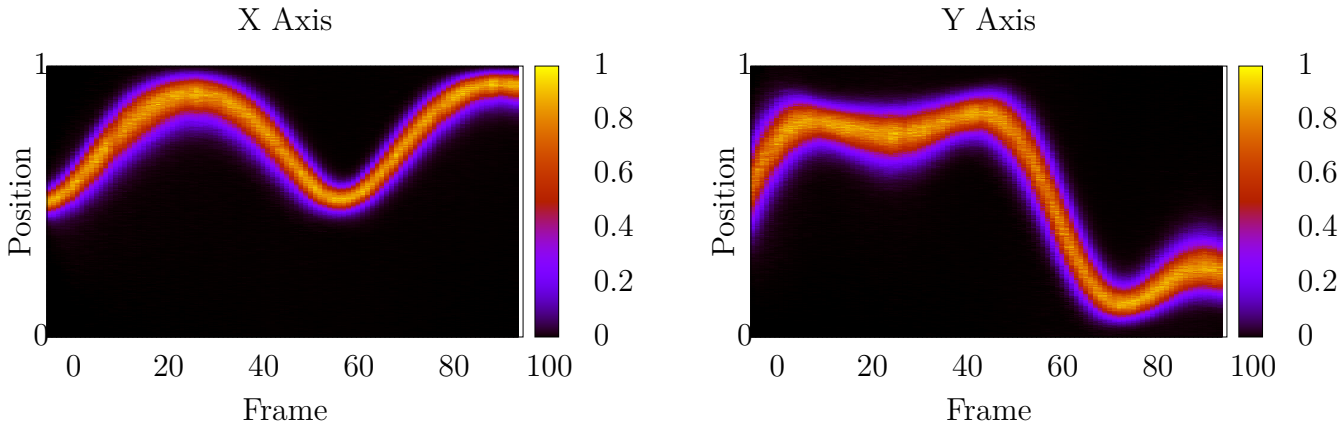


Figure 4.20: Histogram of end effector position of particles for a sequence with simple motion

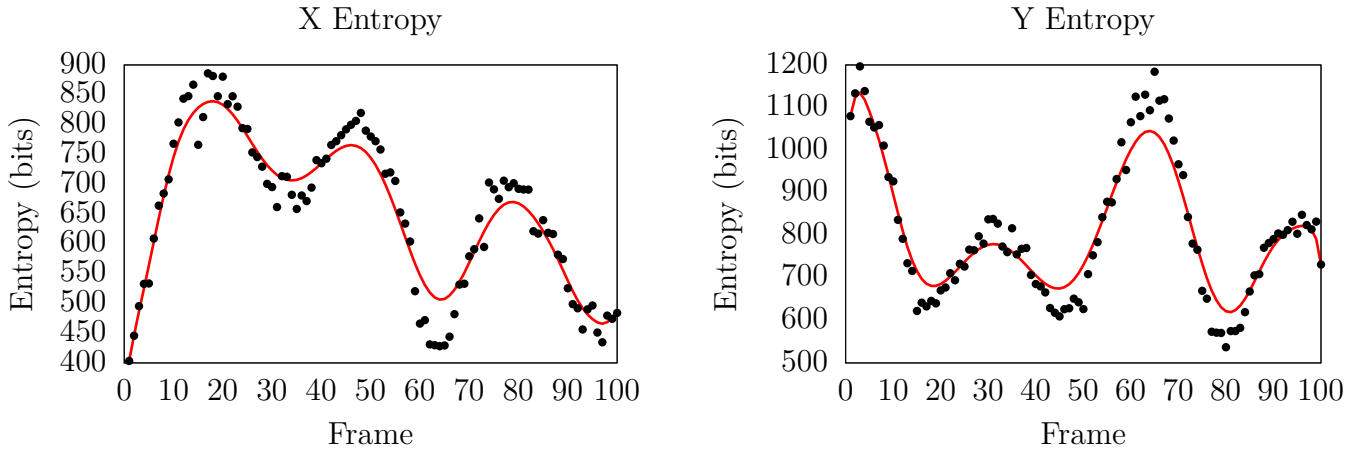


Figure 4.21: Entropy associated with the histograms of Figure 4.20

It is interesting to note that the entropy increases when the direction of motion changes as it takes a few frames for the tracker to move particles along the right path and stabilize. If we now add the ambiguous motion by moving the end effector back and forth along the Y axis, we expect to see a sinusoidal shape in the plot of the Y position, and a straight line for the X position.

Figures 4.22 and 4.23 show the histograms and associated entropy plots when using a Gaussian prior. It can clearly be seen that the tracker fails to estimate the position along the Y axis in this case. If it worked properly, we would expect to see something that looks sinusoidal. An interesting thing to note is that due to the nature of the particle filter, we do not see more than one peak in the histogram, as particles only cover the area around where the tracker thinks the solution should be. This shows that detecting ambiguity is not as simple as in the hypothetical case shown in Figure 2.27. Looking at the entropy of the histogram does not further indicate the presence of ambiguity as the entropy for both axis are in the same range as those in Figures 4.19 and 4.21.

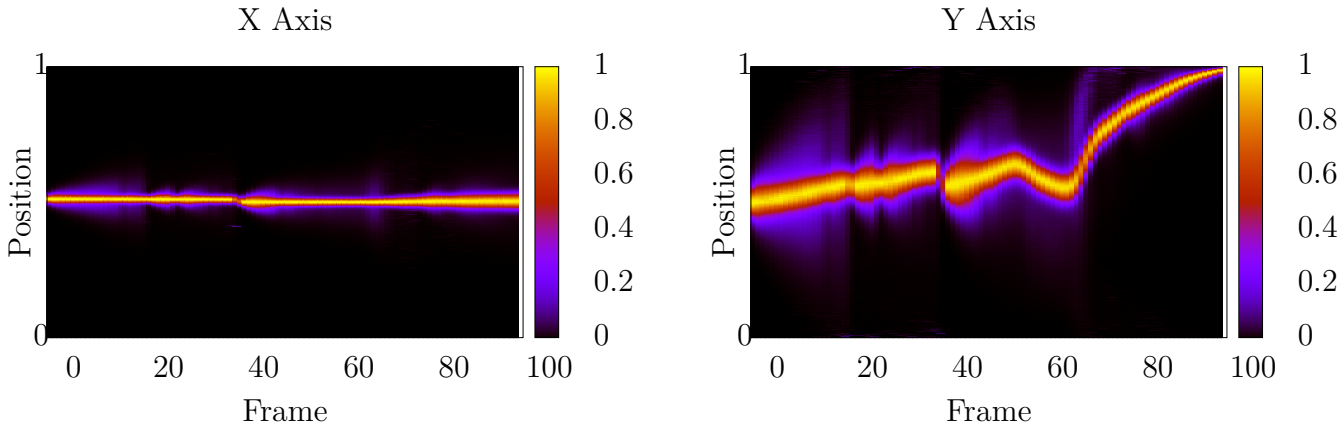


Figure 4.22: Histogram of end effector position of particles for a sequence with ambiguous motion, using a Gaussian motion prior.

When switching to a first order motion model, we get significantly better results. Figures 4.24 and 4.25 show that the particle set is better at approximating the motion, where we see a more sinusoidal shape to the motion along the Y axis. The particles are however more

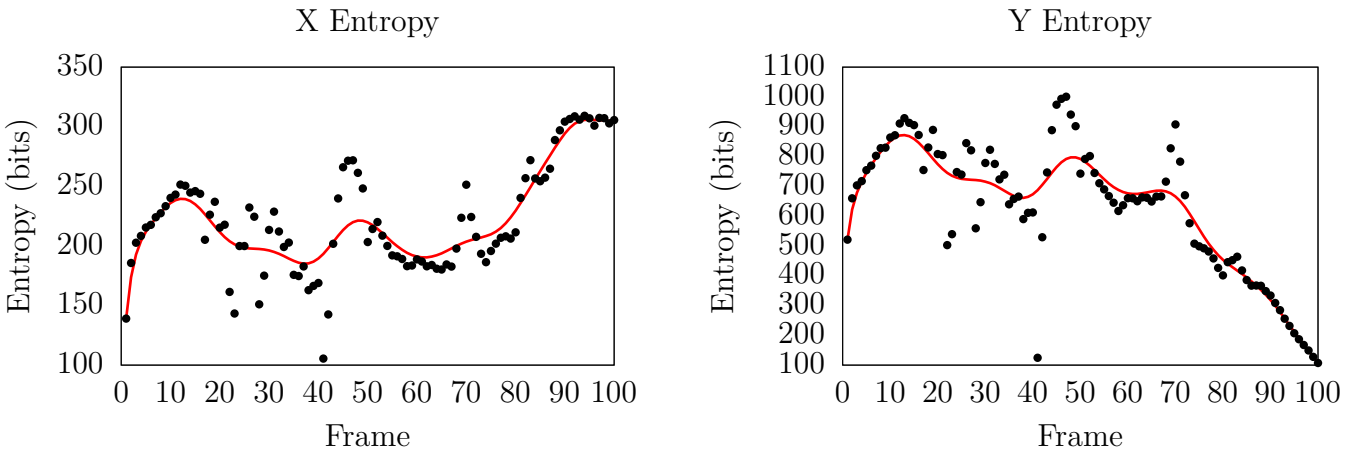


Figure 4.23: Entropy associated with the histograms of Figure 4.22

spread out. This gives a better coverage of the metric manifold, but also results in more ambiguity, which results in a higher entropy, as seen in Figure 4.25.

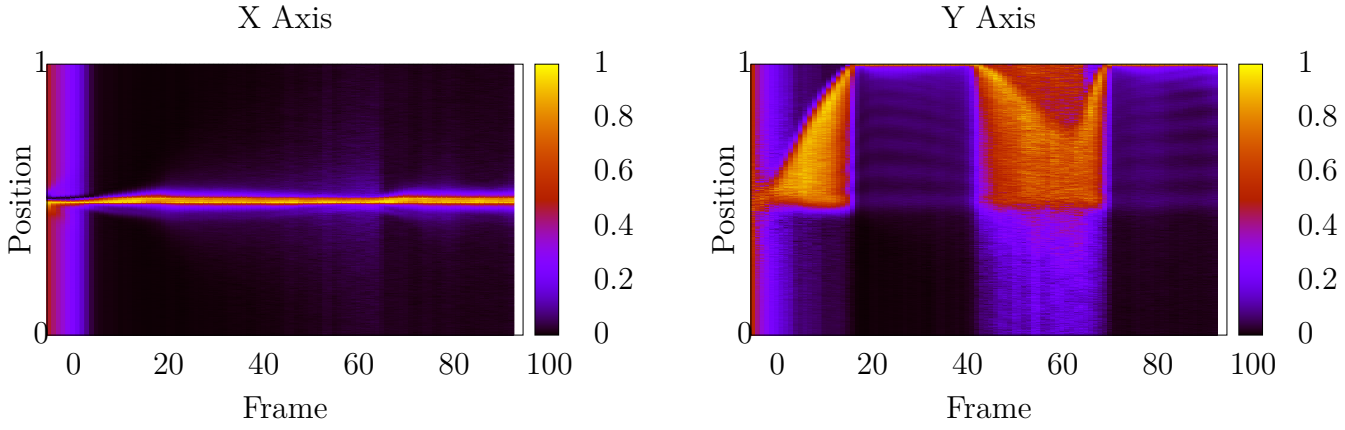


Figure 4.24: Histogram of end effector position of particles for a sequence with ambiguous motion, using a first order motion prior.

Finally, when using a 2nd order motion model, we get improved results over the Gaussian prior, but the tracker still struggles. This can be seen in Figures 4.26 and 4.27. After a few iterations of resampling, the tracker seems to stabilize after approximately 65 frames, where the particles become less spread out and more concentrated around possible solutions. Ambiguity is still present and the resulting pose from the tracker is still incorrect.

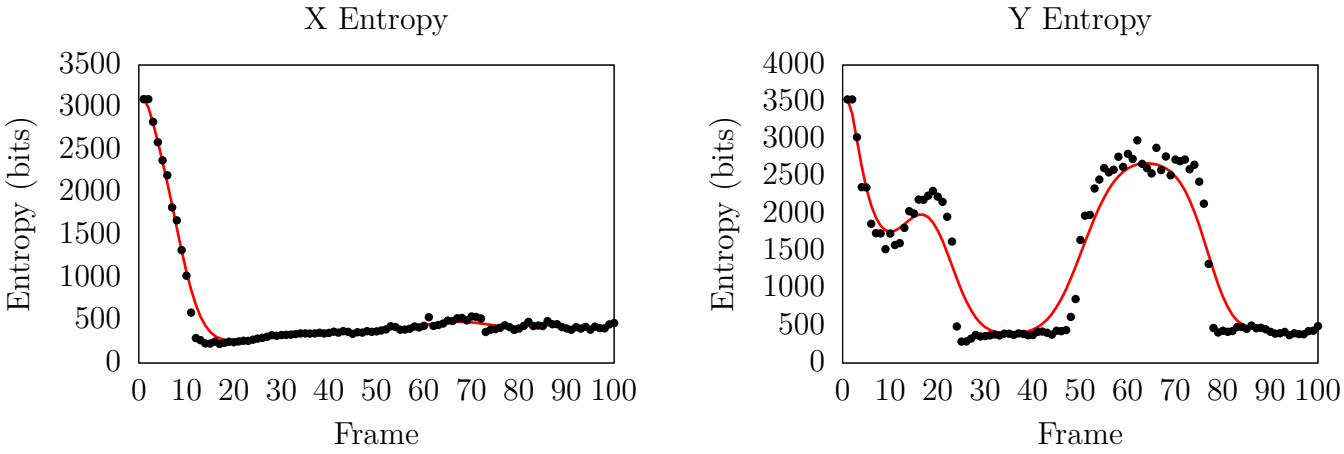


Figure 4.25: Entropy associated with the histograms of Figure 4.24

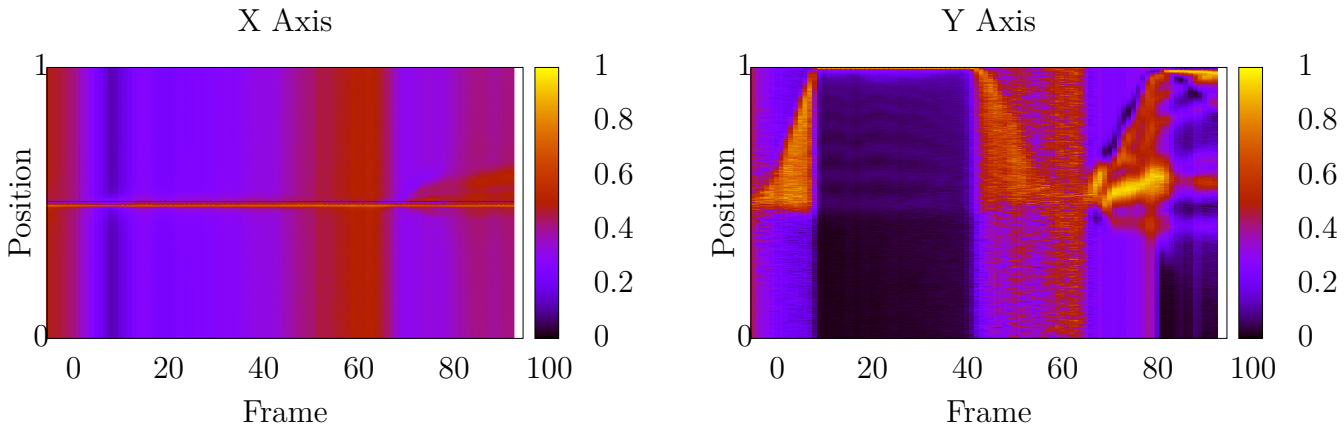


Figure 4.26: Histogram of end effector position of particles for a sequence with ambiguous motion, using a second order motion prior.

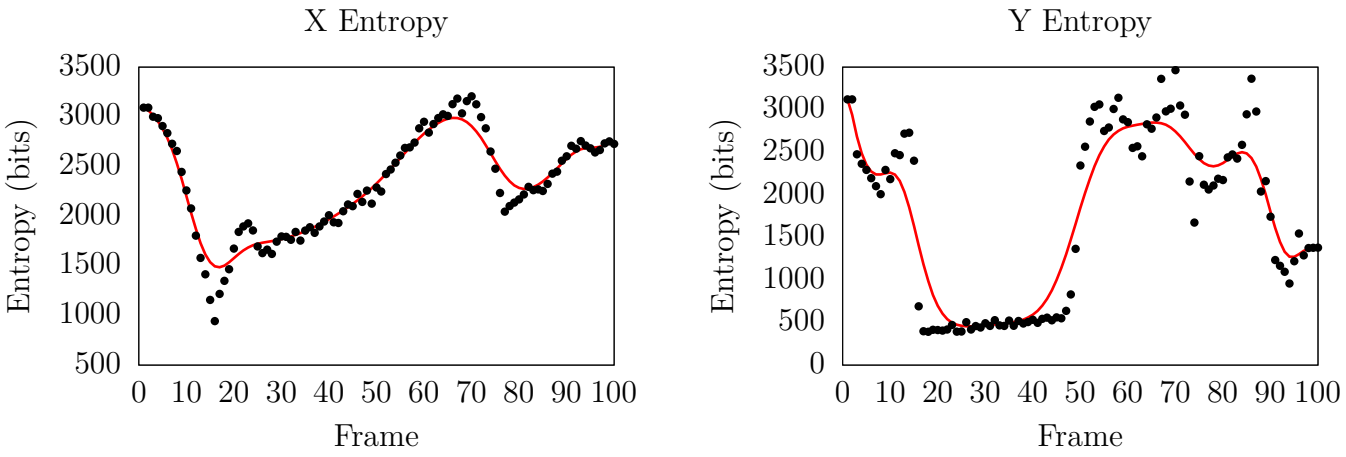


Figure 4.27: Entropy associated with the histograms of Figure 4.26

The goal of these experiments was to determine if it was possible to determine when ambiguity is present by looking at the distribution of the particles and the entropy of the histograms. We see that this is possible, more ambiguity does result in a higher entropy, but the way particles are distributed does not provide a clear solution on how to mitigate the effects of ambiguity. We were hoping for the histogram to show a distribution similar to that of Figure 2.27, but that is not the case. This demonstrates that ambiguity is a limiting factor of our proposed system.

4.5 Initialization Error Experiment

The previous experiments relied on all particles being initialized to the correct pose. While a useful approach for experimenting with other parameters, we do not expect to find an exact initial pose for real-world human motion sequences.

To experiment, we started a series of tracker sequences where the particles were initialized with an increasing distance from the correct solution. As with previous experiments, we ran the tracker for 100 frames and recorded the usual measurements.

Figure 4.28 demonstrates that as the distance between the real pose and the initial pose

of the tracker increases, the average pose error increases. The visual accuracy plot shows that for any possibility of tracking, the initial pose should be within fifty degrees of the correct pose, but closer is better.

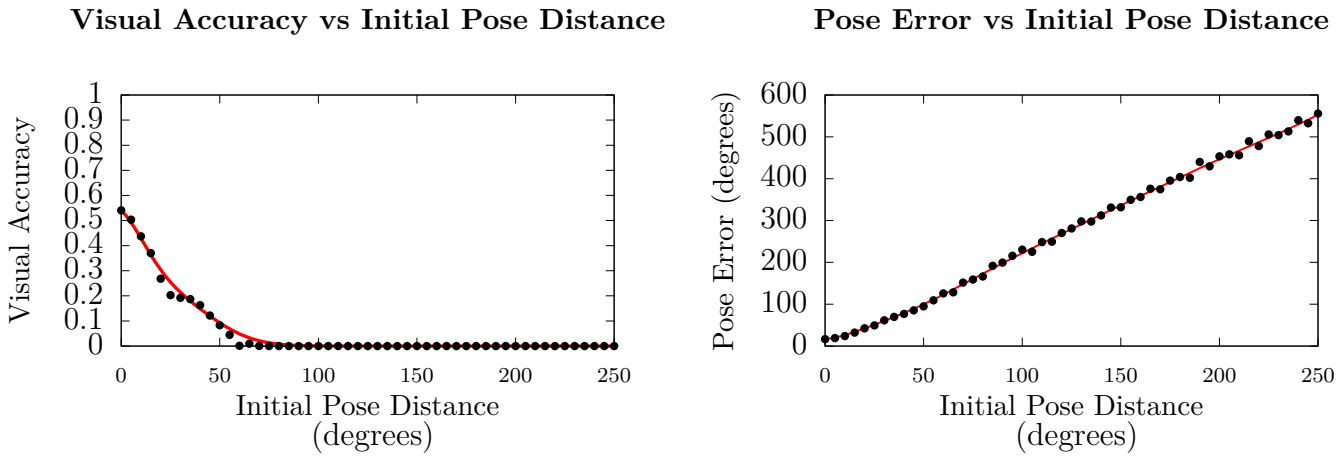


Figure 4.28: Stability experiment results for the small scale experiments with a Gaussian prior and Gradient descent

4.6 Resampling Methods

To compare resampling methods, we used the stationary sequence of Figure 4.3(a) and applied the different algorithms. We chose this sequence as particle drift would have a greater effect on the results, thus providing a better test case for the resampling. To better characterize the behavior over time of the particles, we varied the resampling percentages and also tried using a percentage based on how many particles have a major contribution to the solution. This measure is known as the number of efficient particles [Canals et al., 2009] n_{eff}, as shown in Equation 4.1, where n is the number of particles and w_k is the weight of the k^{th} particle.

$$n_{eff} = \left(\sum_{k=1}^{n} w_k^2 \right)^{-1} \tag{4.1}$$

For each resampling algorithm, we get a set of measurements that can be plotted as in Figure 4.29, where the black points are the actual measurements, the red lines are a spline fit to the measurements and the blue line represents the value obtained with the use of the number of efficient particles. The blue line was plotted this way as the percentage of resampled particles varies throughout the sequence and thus the measurement cannot be plotted to a single point on the graph. To save space and make the data easier to understand, Table 4.1 provides a summary of the results. As with other results omitted from this document, the complete set of graphs is available on the companion website. It is clear from these results that the uniform comb filter greatly outperforms the other algorithms for this tracking task.

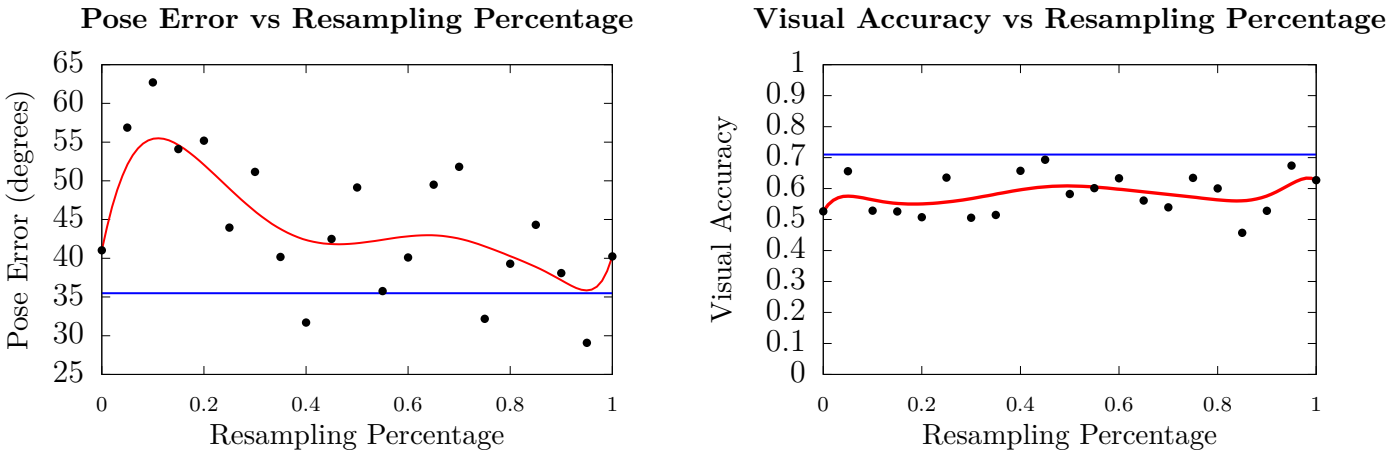


Figure 4.29: Results for the stratified resampling algorithm

4.7 Metric Surface Experiment

First, before going into the description of this experiment, please excuse the graphics presented in Figure 4.30 as they were taken from video files. These were not initially meant to be included in thesis as they stem from an early experiment, but the results proved to be more informative than initially expected. While the following plots may be hard to see in

Algorithm	Pose Error (degrees)					Visual accuracy				
	Min.	Max.	Avg.	Std Dev	n_{eff}	Min.	Max.	Avg.	Std Dev	n_{eff}
Comb	5.64	6.61	6.14	0.27	6.23	0.90	0.92	0.91	0.01	0.90
Multinomial	31.61	67.69	44.90	9.60	37.05	0.47	0.78	0.59	0.08	0.60
Residual	38.69	82.90	57.38	11.45	41.01	0.45	0.72	0.54	0.07	0.63
Systematic	0.00	62.59	41.66	16.18	48.98	0.00	0.72	0.53	0.19	0.57
ResidualSystematic	29.31	70.16	45.36	11.30	39.68	0.47	0.73	0.57	0.07	0.64
Stratified	29.08	62.71	44.23	8.74	35.49	0.46	0.69	0.58	0.06	0.71

Table 4.1: Summarized results of the resampling experiments results

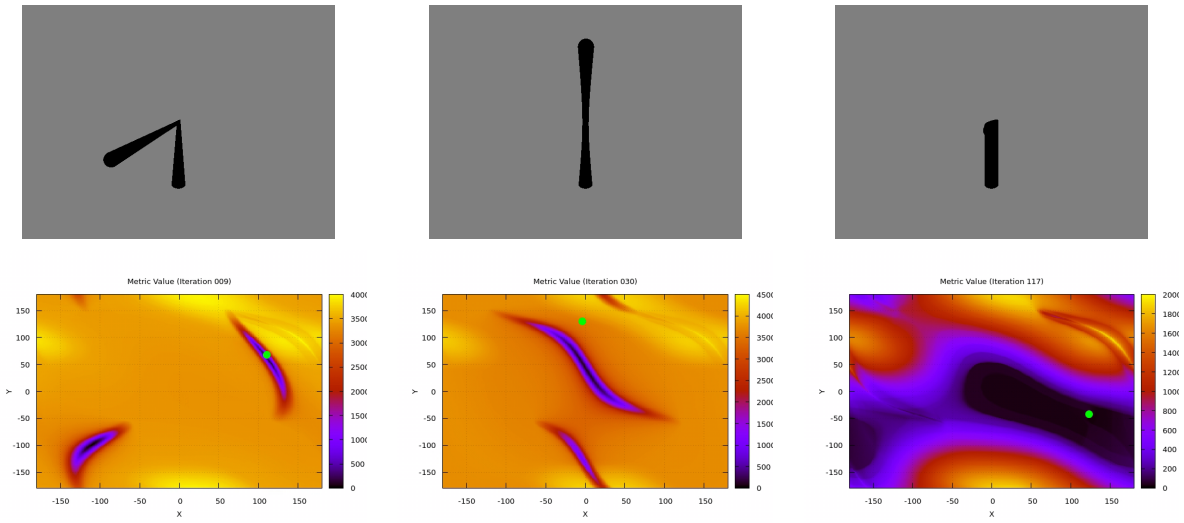
print, the videos on the companion website should prove to be easier to inspect.

Whereas the results in Section 2.3 focused on the local behavior of the metric value, the current experiment looks at the global behavior of the metric value. The use of a 2 degree of freedom model allows us to visually inspect the entire metric surface by computing the value of the metric for all combinations of [integer] pose angles (-180^o to 180^o) and plotting the resulting surface as a heatmap. The green circle overlayed on top represents the real pose of the model.

The goal here was to determine how much ambiguity is present during the tracking sequence and how well the metric is able to capture the information we require. For tracking purposes, the tracker is trying to find the minimum of the metric surface, represented by darker regions. In the ideal case, there should be a single global minimum on the metric surface and the green circle should lie exactly at the center.

As any rotation around the Z axis cannot be resolved when the snake is pointing straight up, we are obviously not looking at such an ideal case, and we can assume that tracking a human will also present such ambiguity issues. As we can see in the frames reported in Figure 4.30, ambiguity is almost always present as there are often multiple regions where the metric value dips to low values.

Figure 4.30 shows three interesting cases. Column (a) shows a case where the metric captures the situation well. The solution resides in a clearly defined local minimum. There is some visible ambiguity, but as long as the initial estimate is in the vicinity of the solution,



(a) Frame 9 (b) Frame 30 (c) Frame 117

Figure 4.30: Metric Surface Experiment Results.

even a simple gradient descent can find the correct solution. The second column displays a case where the metric does not correctly represent the situation, as the solution (green circle) does not lie within one of the minima. This is an issue as in this case, the metric is not valid. Luckily, when looking at the entire video, we see that this occurs only for a few frames and that the tracker should be able to recover. The last column shows a case where ambiguity is extremely high. The metric value correctly encodes the situation as the value is small over a wide range of poses. As this only occurs at specific frames, the particle filter should be able to either maintain a proper solution, or be able to recover easily once the ambiguity is resolved.

This can be seen at Frame 117, where almost any values along the center horizontal line could be a solution. We can see in some short instances such as at Frame 30, that the minima of the metric do not capture the correct solution represented by a green circle. The motion model of the particle filter is what enables it to continue tracking it's current solution and move over the ambiguous "zones". This short video took over a day to generate.

This experiment also confirms the need for the particle filter as the solution cannot be found from a single frame in all cases. The temporal smoothing provided by the motion model

should allow the tracker to maintain a consistent solution even in the case of ambiguity.

Chapter 5

Human Experiments

This chapter presents experiments with a complete human model. The previous chapter has shown that our tracker works with the number of DOF required to track a human pose, and that it can even provide good approximations in the presence of occlusion and ambiguity. We thus start with the same tracking framework, but quickly find that some of the assumptions break under the higher complexity of the human model. We thus introduce a deep-learning based method of generating candidate particles for the particle filter which greatly improves the performance.

5.1 Initialization

When tracking on non-simulated monocular videos of humans, ground truth is not available to initialize the tracker. For this task, we opted to turn toward deep learning and convolutional neural networks (CNN). As seen earlier, the initialization does not need to be perfect, but should be as close as possible. In fact, as long as the the Gaussian component of our tracker covers the area of the solution within a few frames after initialization, our tracker should converge.

Our aim here is to train a network to learn the mapping between silhouette and pose. To this end, we propose a convolutional neural network similar in structure to that presented

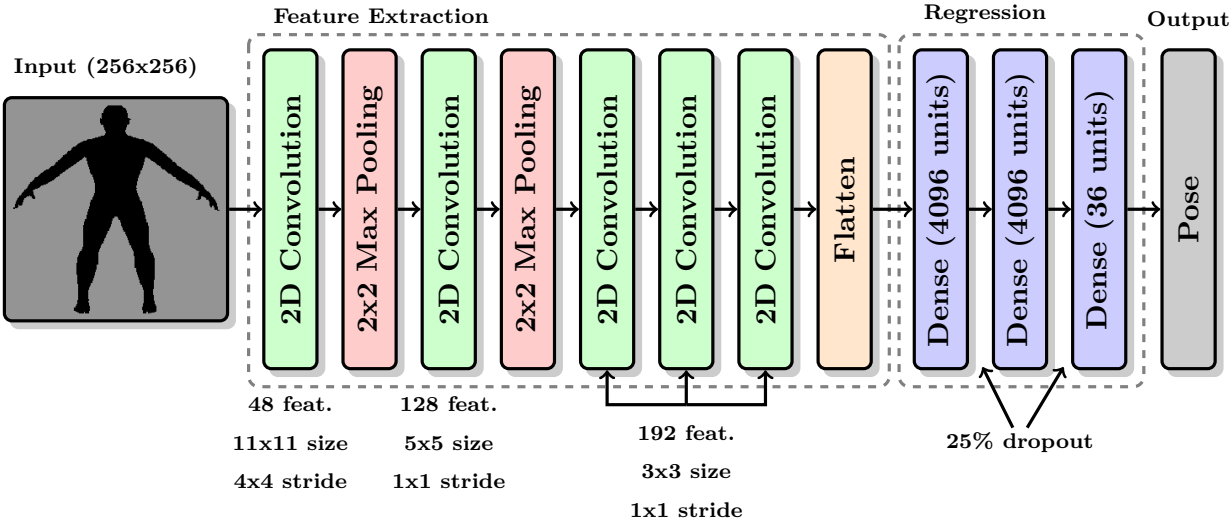


Figure 5.1: Convolutional Neural Network (CNN) architecture, all activation functions are rectified linear units (ReLU).

by Toshev and Szegedy [2014], with the major distinction of using a silhouette as the input instead of the full color image from the camera. Having shown that silhouettes have a strong correlation with the pose, using binary images as the input reduces the dimensionality of the data at the first layer, which simplifies the complexity of the network as a whole. This also provides a texture-invarient network, which greatly reduces the amount of training data required to learn the mapping. Another aspect of our approach, which enables more efficient training, is the use of synthetic training data. As we have a generative model which can generate a silhouette from a pose vector, we can directly use that data to learn the mapping between the two. Figure 5.1 provides a high level overview of the network architecture. Machine learning can also be integrated directly as part of the particle filtering framework by either replacing the randomly resampled particles from Figure 2.26 with candidate poses obtained from a pre-trained deep-learning network, or by replacing low weight particles during the propagation stage.

In order to obtain different candidates from the network and allow our tracker to evaluate a wider range of candidates, we can use one of three approaches. The first, and simplest way to add variation to the candidates is to add a Gaussian component to the output of

the CNN. This is similar to what the particle filter does to increase coverage internally. The second technique is to distort the input silhouette before sending it through the network, which would result in different outputs. The last approach is to use "dropout as a Bayesian approximation" [Gal and Ghahramani, 2016] by retaining dropout on the output fully connected layers when predicting results. This means that every time a silhouette goes through the CNN, a different output pose candidate is obtained. While all three approaches provide more variety in the resulting candidate poses, the last two methods are more interesting as they may provide candidates that are not necessarily all close to one another. This means that the network may be able to provide a solution to ambiguity by exploring more likely candidates even if they are not in the same vicinity. We opt to use the third method as the choice of algorithm used for distorting the silhouette may introduce a bias.

Training

Remembering the results of Section 2.3, we know that the metric error and the pose distance are closely correlated. This allows us to train the network using the mean square error (MSE) in pose space as the training loss function. This prevents us from having to render the pose output from the network in order to compute the loss function, thus greatly reducing training time.

Initial experimentation with a synthetic mesh model provided encouraging results when executed with synthetic data, but proved inadequate with real data. In order for the model to generalize as well as possible, we use multiple 3D human models to generate more varied silhouettes. To this end, one hundred randomized human models were generated by altering key parameters such as height, weight, muscle mass, sex, and age within MakeHuman [Team, 2001]. Figure 5.2 provides a selection of a few such models to show the type of resulting variety.

Due to ambiguity and self occlusion in the training data, we do not expect the network to reach a zero error with any amount of training. Before deciding on the network shown in

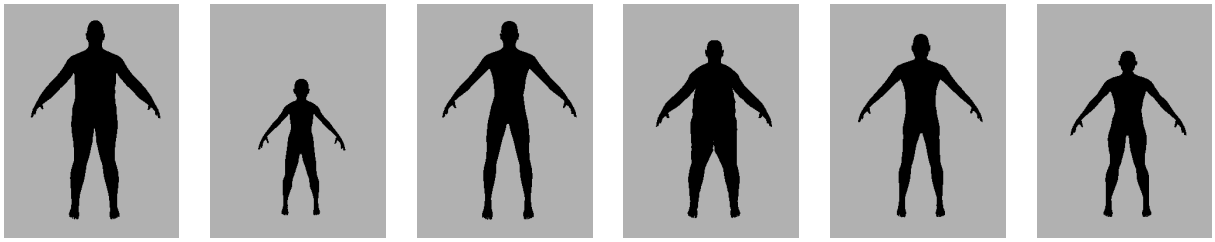
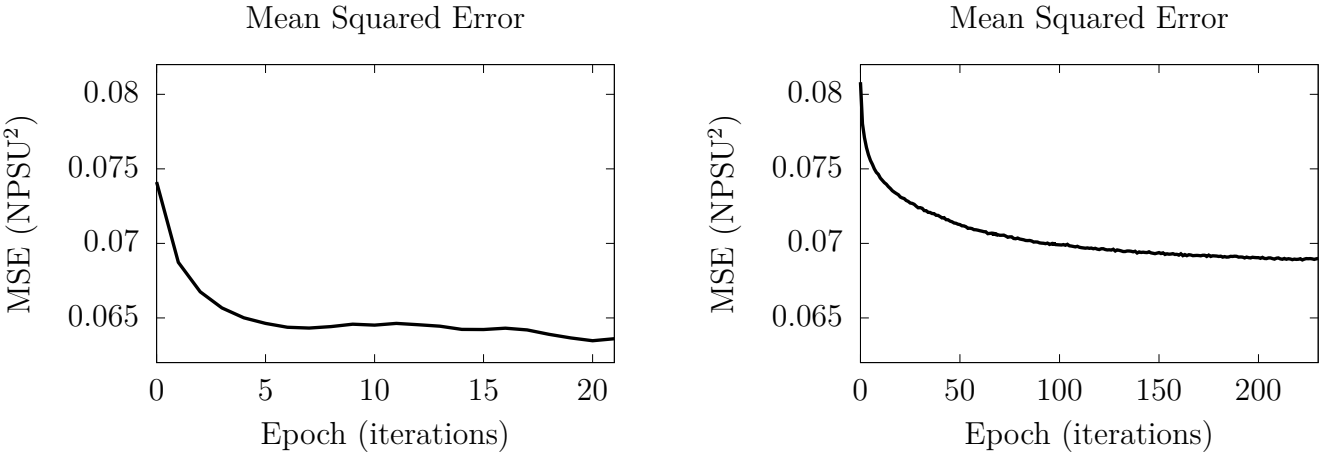


Figure 5.2: Sample models used for training

Figure 5.1, we tried to use transfer learning by using well-known pre-trained networks such as VGG16 [Simonyan and Zisserman, 2014] with the output softmax layer replaced with a sigmoid layer of the size of our pose vector.



(a) Our Network

(b) VGG16 [Simonyan and Zisserman, 2014]

Figure 5.3: Means squared error vs epoch. $NPSU^2$ stands for normalized pose space unit squared.

Figure 5.3(a) shows the loss function of our network and Figure 5.3(b) shows the loss function when training vgg16 for the same type and amount of training data. The error is measured in normalized pose space unit squared ($NPSU^2$). This is a measure of Euclidean distance (2-norm) in a normalized pose space where each component lies within the range 0 and 1. We have normalized our pose space by setting a minimum and maximum degree value for each DOF based on bio-mechanical data. When training the network we use a 36 degree of freedom pose vector. We have also experimented with a 25 DOFs network,

by locking the rotation of the hands and feet to the central position, to speed up training when experimenting. This alternate network was use solely for certain experiments with the chroma-keyed sequences, not for the experiments on the HumanEVA dataset.

To match the expected dimensions, the data when training vgg16 was resized to 224x224 and the silhouette was converted to 3 channels instead of our 256x256 single channel images. During training of vgg16, the convolutional layers were locked and only the top three fully connected layers were trained. We found that, when using transfer learning, the final mean squared error is similar to that of our proposed network, but that a larger number of epoch is required for the loss function to "bottom out" due to computational complexity. For the same reason, our network requires less memory as it contains less parameters and works on single channel binary input data.

5.2 Evaluation

We evaluated our method on four datasets, first is a computer generated synthetic dataset that provides ideal testing conditions as well as ground truth. The second test involves a chroma keyed data set created for the purpose of testing our approach on real data, while eliminating issues related to the silhouette extraction. Sadly, this second dataset does not include ground truth and will be use as a qualitative evaluation only. The final two datasets are publicly available sets used in the literature to compare with other approaches. The selected datasets are HumanEVA [Sigal and Black, 2006] and Human3.6M [Ionescu et al., 2014]. Comparison with other methods is made difficult as the skeletal structure differs between authors, as different authors focus on different aspects. Furthermore, most published results with the HumanEVA dataset stem from methods trained directly on the dataset [Moreno-Noguer, 2017]. We aim to develop a tracker which can generalize to any human silhouette. We hope for better results with the Human3.6M dataset as it includes precomputed background-foreground segmentation.

5.2.1 Synthetic Data

Similarly to the metric selection experiments of Section 2.3, using synthetic data allows us to experiment in ideal conditions to test the base performance of the tracker.

The first experiment consists of running the tracker on a static sequence of images and looking at the output of the tracker. This allows us to characterize the stability of the tracker. Figure 5.4 shows that the pose error starts at a higher value then stabilizes after approximately 20 frames. This behavior is mostly caused by the temporal smoothing aspect of the filter, which requires a few frames before converging and stabilizing close to the solution. The error is once again measured in normalized pose space units ($NPSU^2$).

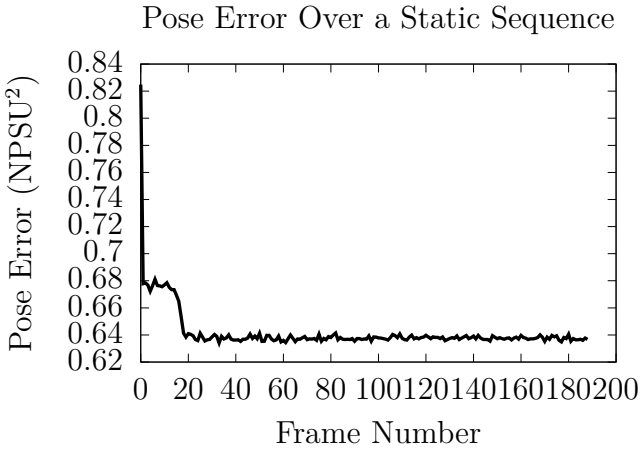


Figure 5.4: Pose error over a sequence containing no motion

In addition to the 3D pose error, we also measure the 2D pose reprojection error. This error is computed as the average distance between the position of certain joint locations of the tracked pose and the positions from the ground truth. We add this error metric as it will be used when evaluating our tracker on the public datasets, as the ground truth of those datasets do not use the same pose vectors, and thus comparing joint locations requires less conversions, which may introduce new sources of error. The joints used to compute this error are the top of the head, the shoulders, elbows, wrists, knees, and ankles. Figure 5.5 provides the plot of the 2D pose error. It is important to note that the 2D results were computed separately. Due to the randomess aspect of our particle filters, the results will not exactly

match. We can however see the same global behavior of a high initial error which quickly stabilizes to a small value.

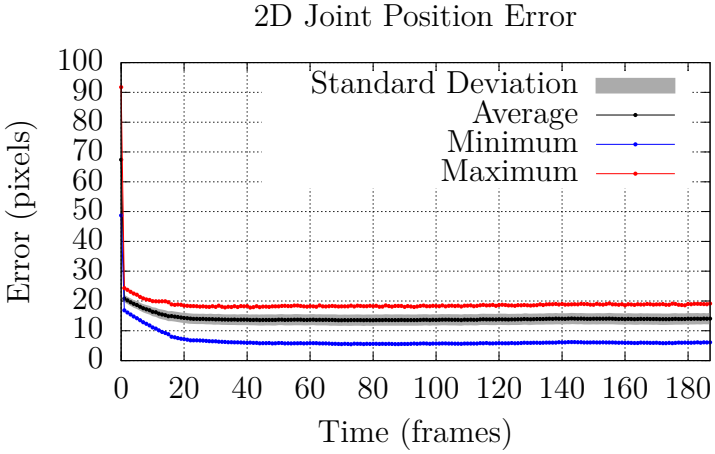


Figure 5.5: 2D Pose error over a sequence containing no motion

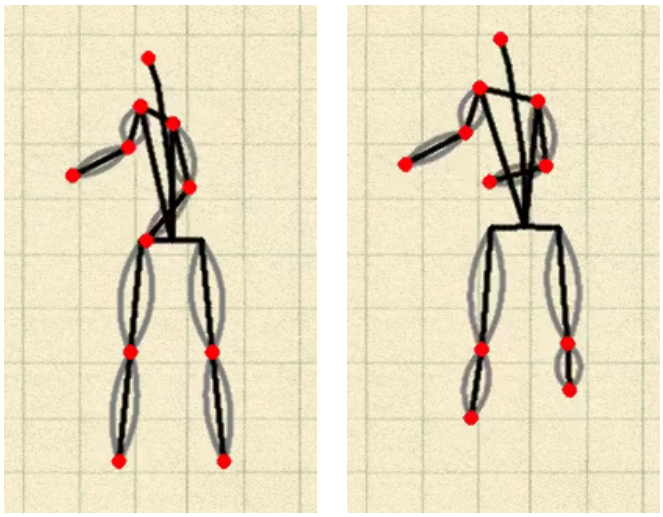
Similar to what was done for the metric selection, we tried the tracker for sequences containing motion on each degree of freedom individually and ran sequences twice, once from the front, once from the side. Again, the complete results are available on the companion website. A summary of the average pose errors of each sequence is reported in Table 5.1. The pose errors are measured as the L^2 norm between the known pose that was used to generate the data and the output pose from the tracker. These pose errors are thus measured in a normalized space and in the range [0,6] as we are looking at a 36 DOF model. The 2D pose error are reprojection errors, measured in pixels. To compute these values, we project the 3D joint positions of the model to a 2D image space and compare the resulting positions with those of the ground truth data. The numbers reported are the average error over all joints, over all frames of the sequence. From these results we determine that our approach works for human models. Looking at the videos shows that there is a noticeable amount of oscillation around the solution. This might be resolved by increasing the temporal smoothing at the cost of a lag when tracking motion.

The surprising result is that the 3D pose error is significantly lower when the model is viewed from the side, while the 2D pose error is higher than when the model is viewed from

	Frontal View				Side View			
	3D Pose		2D Pose		3D Pose		2D Pose	
DOF	Mean	Std.Dev.	Mean	Std.Dev.	Mean	Std.Dev.	Mean	Std.Dev.
00	0.76	0.10	53.86	30.55	0.38	0.15	31.93	8.37
01	0.70	0.09	19.51	10.09	0.40	0.14	63.41	27.09
02	0.70	0.17	47.04	24.32	0.44	0.27	30.37	9.37
03	0.68	0.07	32.51	12.31	0.38	0.15	19.90	6.59
04	0.72	0.08	14.24	5.90	0.39	0.14	34.15	10.26
05	0.67	0.08	19.54	4.34	0.36	0.14	27.81	5.28
06	0.87	0.21	26.75	8.57	0.39	0.19	35.13	15.61
07	0.80	0.20	24.24	10.79	0.43	0.25	49.56	14.41
08	0.77	0.14	22.26	7.16	0.35	0.15	25.20	3.84
09	0.76	0.14	12.63	7.63	0.34	0.14	29.53	2.87
10	0.71	0.07	13.09	4.16	0.34	0.14	15.33	3.91
11	0.53	0.09	15.66	4.18	0.34	0.14	18.77	5.00
12	0.73	0.11	16.53	4.55	0.38	0.18	14.83	4.42
13	0.73	0.08	19.93	4.81	0.44	0.19	34.13	6.68
14	0.72	0.07	16.97	5.37	0.34	0.14	31.19	3.79
15	0.74	0.12	17.31	5.62	0.39	0.16	28.34	3.50
16	0.72	0.09	18.64	4.65	0.41	0.20	27.58	7.48
17	0.70	0.07	14.31	4.52	0.33	0.14	28.76	3.17
18	0.74	0.17	17.25	5.13	0.47	0.21	23.26	4.76
19	0.74	0.16	19.72	4.15	0.47	0.26	30.00	4.35
20	0.76	0.16	19.43	5.61	0.36	0.16	17.36	4.71
21	0.88	0.17	25.26	7.25	0.40	0.22	28.62	6.67
22	0.76	0.13	16.47	4.50	0.34	0.15	28.20	3.04
23	0.73	0.15	20.55	5.41	0.34	0.14	22.95	3.69
24	0.85	0.16	19.82	5.33	0.39	0.21	35.94	6.57
25	0.75	0.12	14.97	4.93	0.34	0.14	27.97	3.34
26	0.82	0.15	20.40	4.96	0.39	0.21	25.51	3.42
27	0.71	0.09	15.59	4.74	0.37	0.19	23.06	3.76
28	0.85	0.15	18.43	5.46	0.43	0.21	19.90	5.72
29	0.71	0.12	16.22	4.36	0.36	0.20	26.53	3.31
30	0.65	0.10	11.36	4.99	0.38	0.18	21.40	3.44
31	0.73	0.12	12.33	5.08	0.37	0.21	23.83	3.56
32	0.81	0.17	15.24	5.07	0.34	0.14	10.45	5.69
33	0.70	0.09	15.02	4.59	0.36	0.19	24.00	5.56
34	0.80	0.17	13.46	5.25	0.34	0.15	9.95	4.83
35	0.78	0.11	14.53	4.59	0.37	0.17	20.87	3.77

Table 5.1: Recorded 2D and 3D pose errors from synthetic data sequences with a single moving joint

the front. We confirm this finding with the HumanEVA dataset as it presents 3 different views and the same behavior is observed. The other interesting result in this table is the higher than expected 2D pose errors. The average 2D pose errors are between 15 and 20 pixels. Figure 5.6 shows a graphical representation of the 2D pose for a frame from the ground truth and from the tracker output. We can notice a significant vertical axis offset simply by looking at the joint positions related to the grid. This offset alone accounts for approximately a 12 pixel error along the vertical axis for most joints. This offset is caused by the particle filter responsible for tracking the root position. More investigation is required to find out why.



Ground Truth Tracker Output

Figure 5.6: Example result from the tracker showing a noticeable offset along the vertical axis.

5.2.2 Chroma-keyed Experiment

We have so far demonstrated that our approach can work on synthetic data. We will now turn our attention to real data and try to determine if our tracker can generalize and cope with the imperfections associated with real data such as noise, inaccurate segmentation, and model mismatch. We begin with a homemade dataset with sequences recorded in front of a green screen to help with segmentation. As this dataset does not include ground truth, we will only use it for qualitative purposes. The goal here is to determine general characteristics of the tracker and determine if it can generalize enough to be functional on real-world data. To save on computation time, we only run the pose tracker and assume that the root position is fixed.

Figures 5.7, 5.8, and 5.9 provide example cropped silhouettes from a chroma keyed sequence. The left columns show the input silhouette, while the right columns show a silhouette rendered from the output of the tracker, for comparison purposes.

The first thing we notice is that the silhouette of the virtual model does not exactly match the silhouette of the subject in the sequence. Using a 3D scan of the model might help in this case, but would not help in making the system more generalizable. While introducing error, this mismatch does not prevent the tracker from capturing a good representation of the pose. By looking at the entire video[1] we notice that the tracker is not exactly stable due to the shape mismatch and that it has more difficulty tracking the lower body, but that overall it can keep track of the general pose and can recover from errors when it fails. We now know that with good segmentation, our model generalizes well enough to provide useful tracking from real-world image sequences.

5.2.3 HumanEVA Dataset [Sigal and Black, 2006]

The humanEVA dataset consists of a variety of sequences containing different activities, performed by 4 different subjects (actors). The data itself is composed of a collection of 4

[1]The first video on the companion website, labeled "Final Experiment"

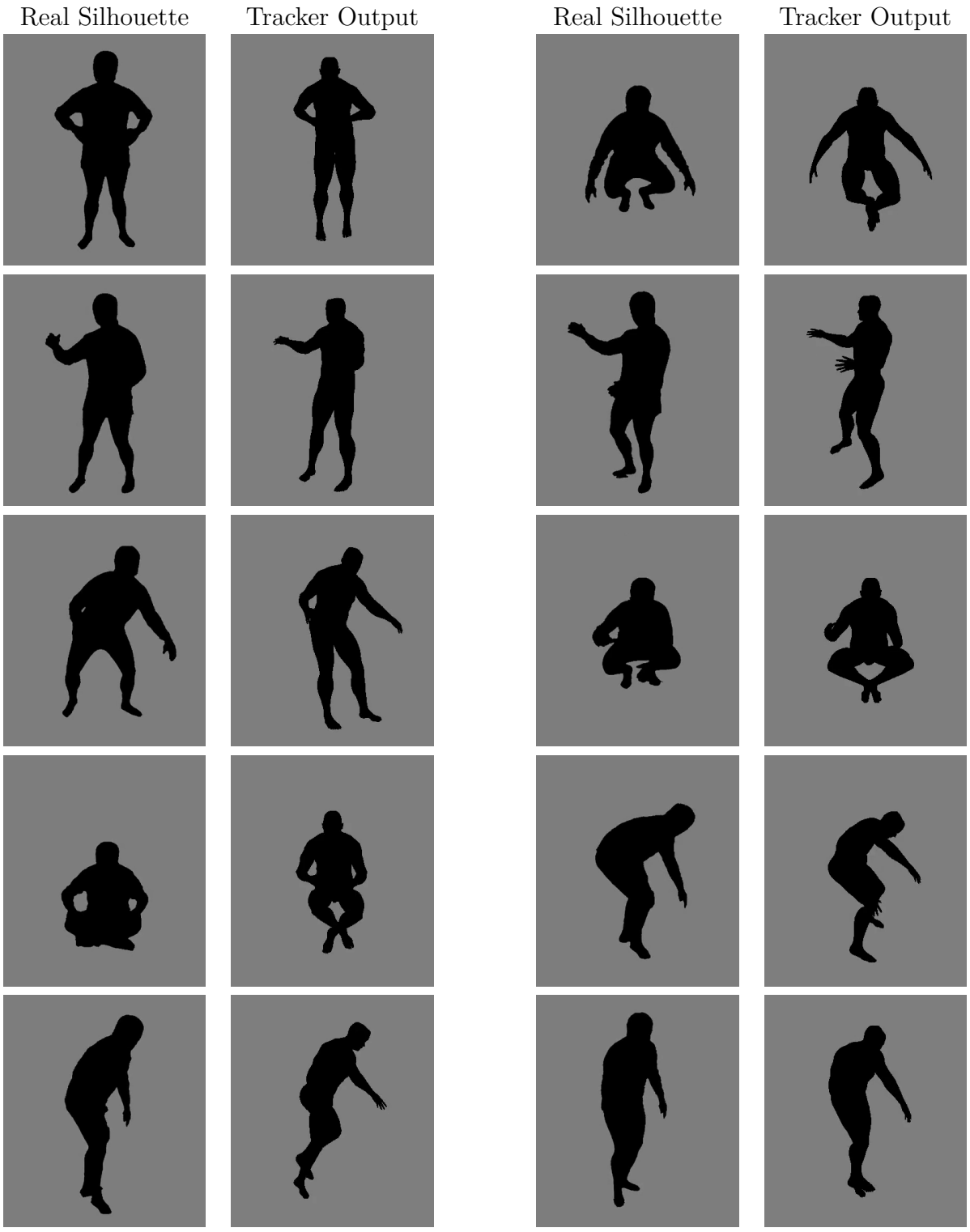


Figure 5.7: Example result from the chroma-keyed sequence (part 1)

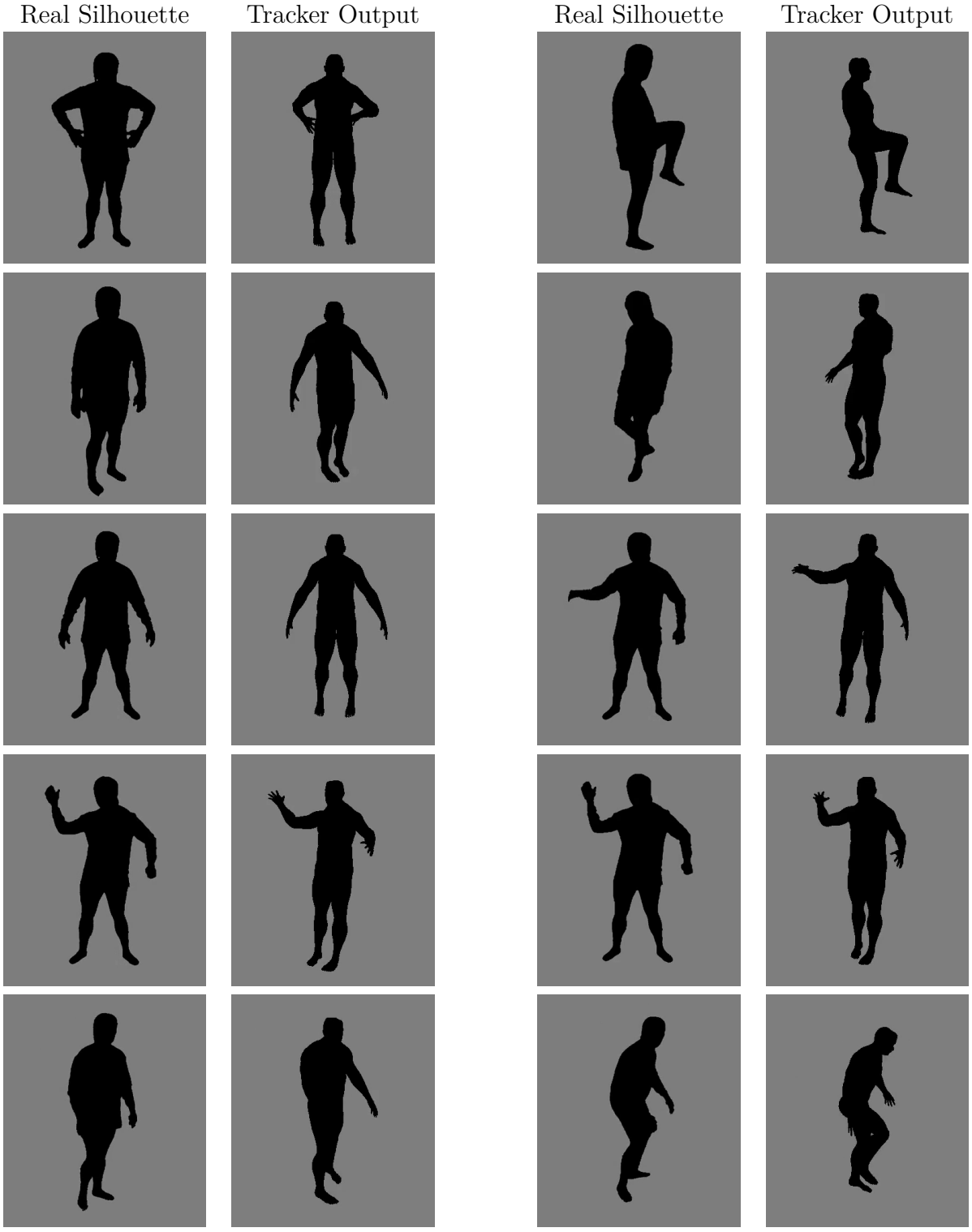


Figure 5.8: Example result from the chroma-keyed sequence (part 2)

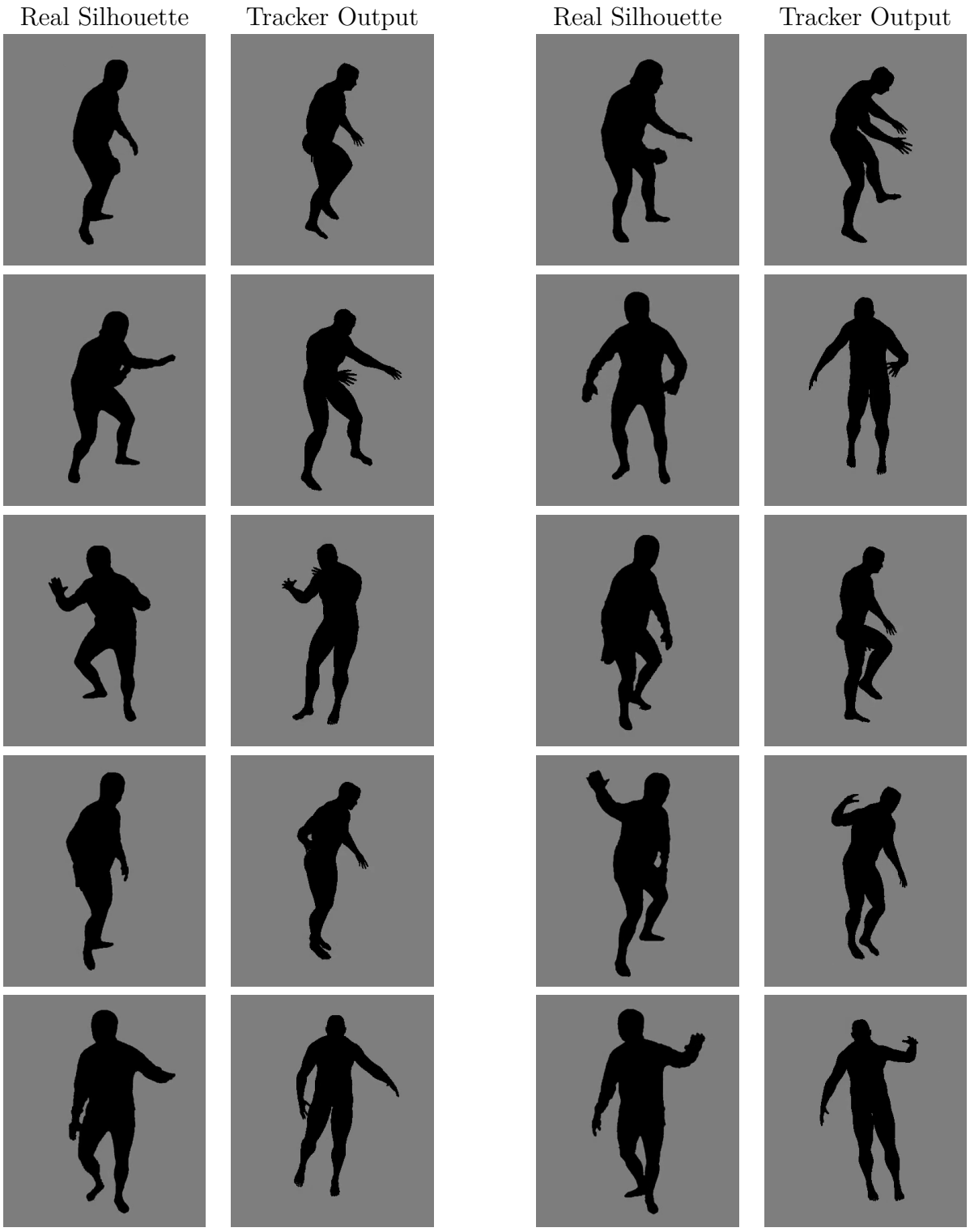


Figure 5.9: Example result from the chroma-keyed sequence (part 3)

grayscale cameras, 3 color cameras, and ground truth pose recorded with a Vicon motion capture system. Figure 5.10 provides sample frames to show how the cameras are located with respect to the subject. While the grayscale images are shown, we will only apply our tracker to the color images as the segmentation algorithm we are using relies on color to determine which pixels compose the silhouette of the subject.

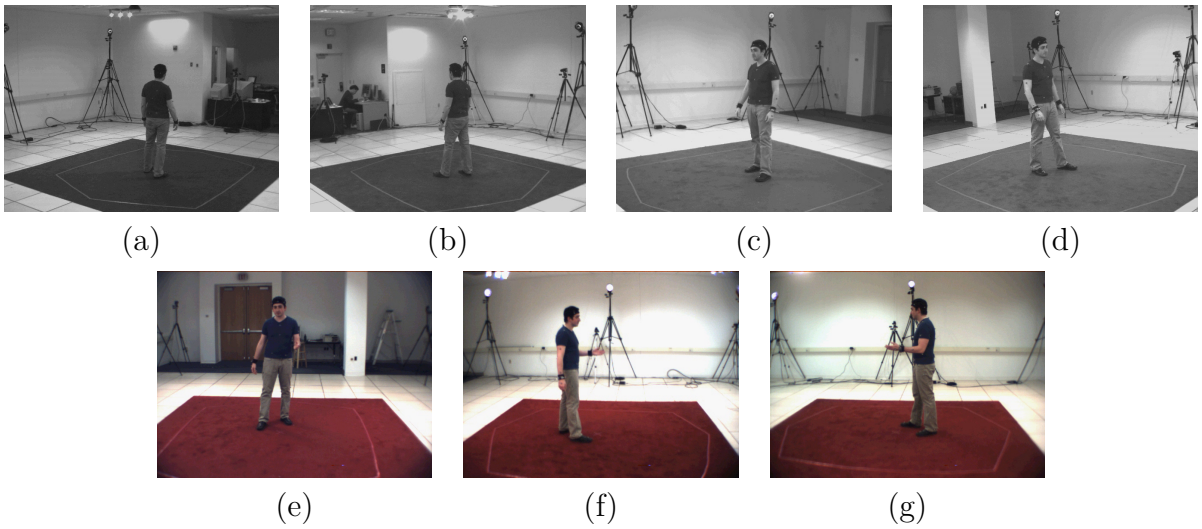


Figure 5.10: Example images from the HumanEVA dataset showing the four grayscale (a-d) and three color cameras (e-g).

Applying our tracker to the various sequences and looking at the output of the tracker, we quickly notice that it fails to properly track the subject's pose. As with most other experiments, only some of the results are reported here, but the full set of videos and graphics is available on the companion website for convenient access. After executing our tracker on the sequences, we compare our results to the ground truth provided in the dataset to obtain a quantitative measure of our tracker's performance. Figures 5.11 through 5.13 provide the error plots for the three cameras of the Gesture-1 sequence with subject 1. The recorded errors are 2D reprojection errors, measured in the same way as the results in Table 5.1, but presented as plots of the error values over time instead of a single average for the entire sequence. The plots also report the standard deviation as well as the minimum and maximum joint reprojection errors. The first thing to notice is that the errors are considerably higher

than what was seen with synthetic data. This is supported by our visual inspection of the output videos, shown on the companion website.

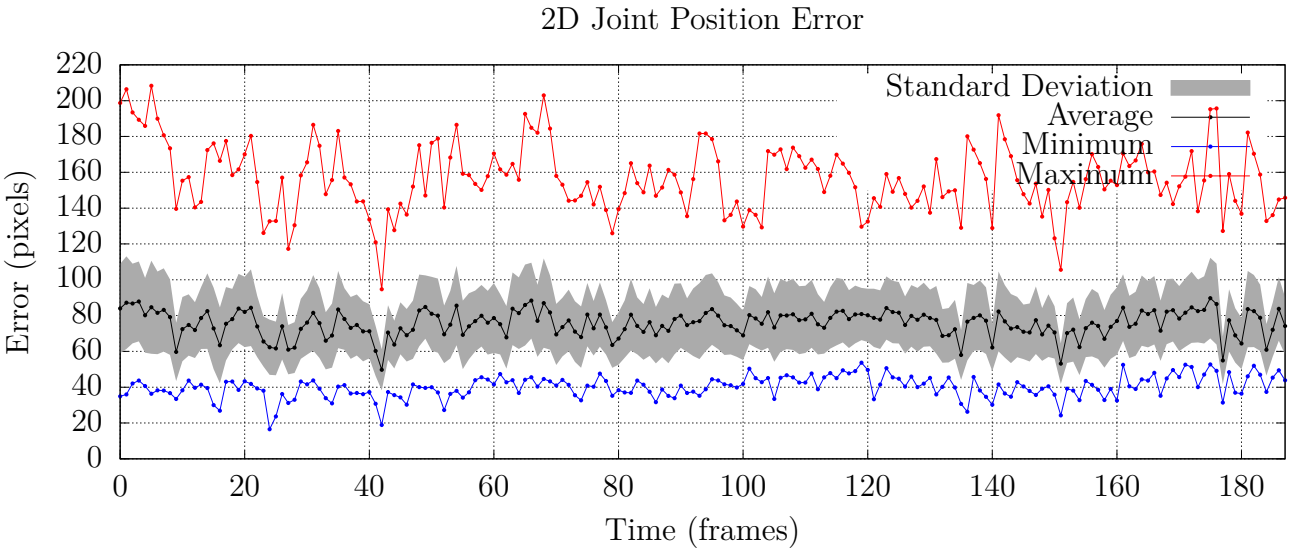


Figure 5.11: 2D reprojection error for sequence Gesture-1 of HumanEVA, with subject 1,viewed from camera 1.

These figures show that the lowest average error is obtained with the second camera, while camera 1 provides the worse results. Looking at the segmentation images, we see clearly that the segmentation is noticeably better with the second camera as there is less background clutter that can be mistaken as part of the silhouette, and vice versa. The third camera, with an error somewhere in the middle shows a worse segmentation than Camera 2, mainly where the feet and the floors shadows are mislabeled, but a much better overall segmentation than Camera 1. Figure 5.14 shows the same segmented frame from the three cameras. In this particular frame, the first camera fails to detect the right arm of the subject. This seems to suggest that errors in segmentation lead to much lower tracking performance.

Table 5.2 provides a summary of the errors obtained in the sequences where ground truth is available. We see the same pattern of Camera 2 providing better results for most sequences other than those with Subject 2. Inspecting the videos with the second subject reveals that the shadows are more important with Subject 2 than with the other subjects and that this causes overall larger errors. All of this seems to support our hypothesis that the silhouette

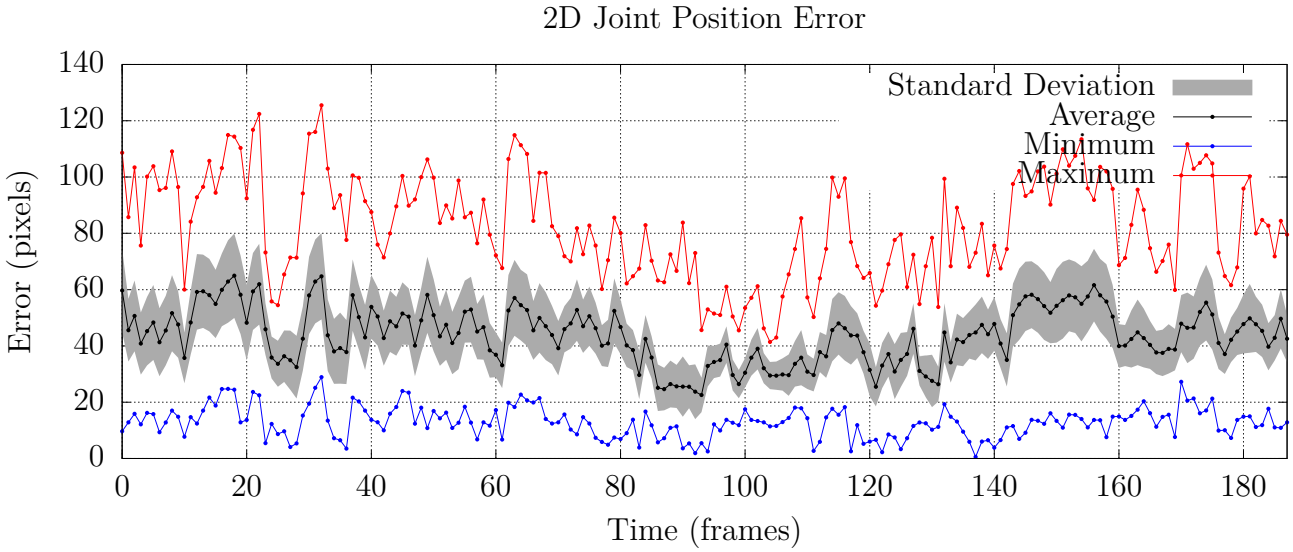


Figure 5.12: 2D reprojection error for sequence Gesture-1 of HumanEVA, with subject 1,viewed from camera 2.

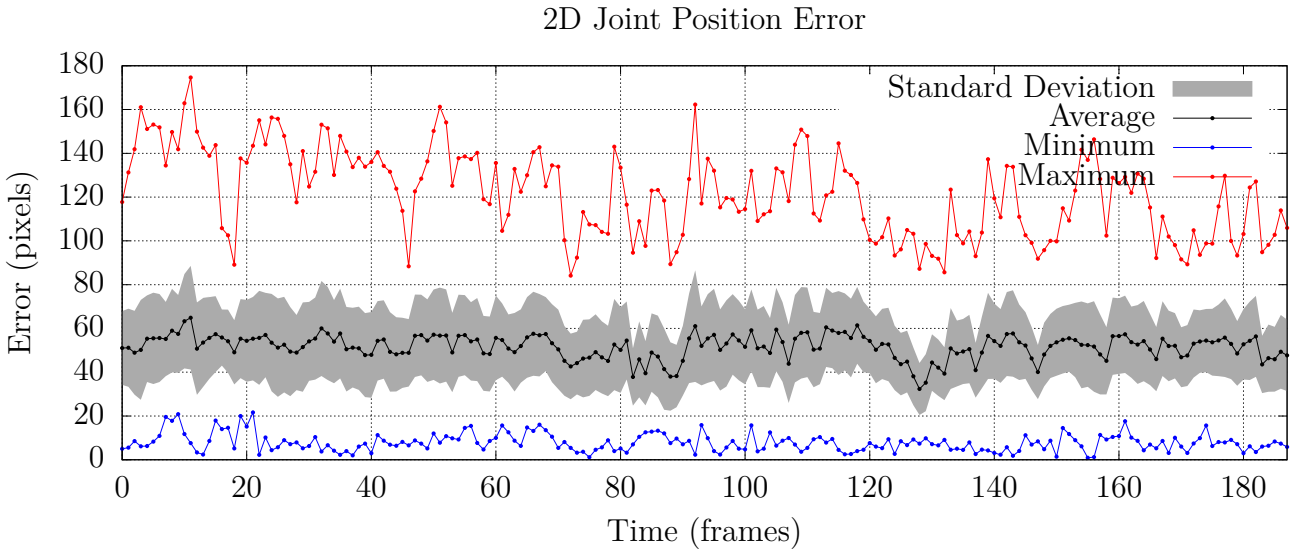


Figure 5.13: 2D reprojection error for sequence Gesture-1 of HumanEVA, with subject 1,viewed from camera 3.

Camera 1 Camera 2 Camera 3

Figure 5.14: Example images from the HumanEVA dataset showing the segmentation from the three cameras color cameras.

segmentation is the limiting factor for our tracker.

		Camera 1		**Camera 2**		**Camera 3**	
Sequence	Subject	Mean	Std.Dev.	Mean	Std.Dev.	Mean	Std.Dev.
Box-1	1	70.43	7.36	48.19	12.45	40.74	10.02
Gestures-1	1	75.41	21.52	45.33	24.39	51.45	23.68
Jog-1	1	81.63	27.01	66.86	24.03	79.32	30.97
Walking-1	1	89.45	30.68	98.96	30.09	89.24	27.37
Jog-1	2	128.18	40.08	133.91	52.16	136.40	51.38
ThrowCatch-1	2	84.58	11.47	81.82	32.10	83.93	43.04
Walking-1	2	109.20	37.02	112.34	37.24	118.57	38.29
Jog-1	3	101.93	32.30	91.15	28.36	86.35	30.55
Walking-1	3	126.42	43.51	121.95	52.83	150.57	83.42

Table 5.2: Recorded 2D pose errors (pixels) from HumanEVA [Sigal and Black, 2006] sequences

5.2.4 Human3.6M Dataset [Ionescu et al., 2014]

While we initially had high hopes for this dataset as it includes segmentation data, inspecting the data itself revealed the same segmentation issues we encountered when segmenting the data from the HumanEVA dataset. Figure 5.15 shows a couple frames where segmentation issues were important and perhaps worse than our Gaussian mixture model applied to some of the HumanEVA sequences. Sadly, Human3.6m does not provide the empty sequences required to train the background models needed to re-segment the data.

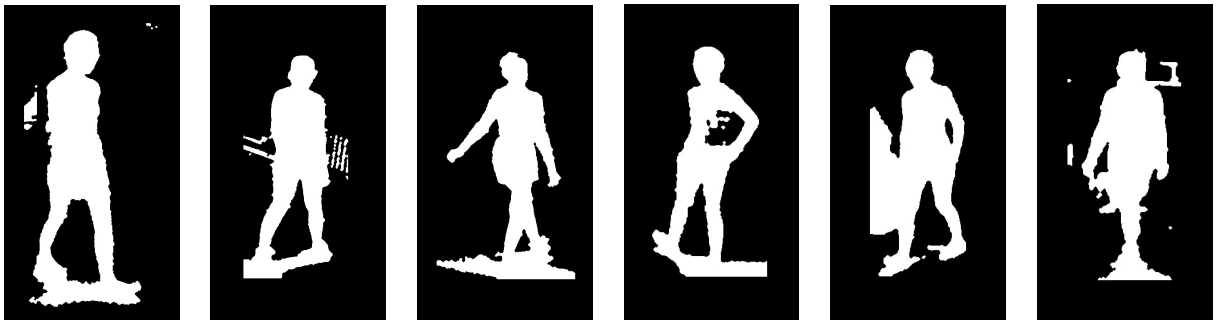


Figure 5.15: Sample Human3.6M [Ionescu et al., 2014] segmentation showing errors.

As the segmentation is no better than that of the HumanEVA dataset, we do not expect functional results and have thus opted not to use this dataset at this point in time.

In conclusion, it is clear that our implementation is limited by the quality of the silhouette segmentation, which is problematic with the sequences in both HumanEva and Human3.6M datasets. The experiments with chroma keyed sequences indicate that with appropriate segmentation, our approach provides good results. It is, however, important to look at the results in the context of the initial goal of this thesis. Our approach, tracking 36 DOFs, generalizes well and works well even on sequences where the human doesn't match the virtual model. To put this in perspective, most deep learning approaches, for example the network presented by Moreno-Noguer [2017], do not generalize and require the network to be retrained for each dataset it is applied to. Thus these approaches have different networks for each specific applications. This requires the capture of new data in each scenario to retrain the network. Our approach was aiming for a single tracking algorithm that could generalize to any video sequence, and we have shown that it is possible. We are still hopeful that improvements in silhouette extraction can further increase the accuracy of our results.

Chapter 6

Conclusion

To conclude, the research presented herein aimed to find a simpler approach to the problem of human pose tracking. Simplicity here refers to a reduced level of instrumentation both with respect to the scene in which the tracking occurs and with respect to the subject being tracked. To this end, our approach requires a single video camera and does not rely on any sensors or markers attached to the model. To accomplish this task, we used a generative model that relies on biomechanical data and returned to the foundations of computer vision to explore representations and features less commonly used in recent literature. The integration of modern approaches such as deep learning allowed us to develop a computational framework that can successfully track complex articulated poses. Through careful experimentation, we have demonstrated that our approach works well in ideal cases, and been able to show under which less than ideal conditions it succeeds, and have shown situations where it fails. Most importantly, the experiments in Chapter 5 allowed us to determine that the accuracy of the feature extraction is the most crucial factor to enable accurate tracking.

The key finding of the thesis is that while occlusions inevitably result in ambiguity, silhouettes are valid features to track highly articulated objects when dealing with monocular images. We explore this area by providing a survey of methods used in the literature to compare the similarity of silhouettes and determine how well they correlate to changes in

the pose of the model generating the silhouette. We detailed a set of experiments, and presented their results in Section 2.3. This careful study of silhouette comparison metrics has shown that for the task of human tracking, computing the exponentiated pixel count difference provides the most reliable metric. Through experimentation in Chapter 4, the particle filter approach we describe in Section 2.4 is shown to be able to resolve ambiguity over time, in most cases.

We have shown that synthetic data can be used throughout the development of a complete tracking system. We have also demonstrated that synthetic data can generalize to real world data given accurately extracted silhouettes. We first used it as part of our silhouette comparison metrics to more evaluate the various approaches in a common, controlled, frame of reference. The second use of synthetic sequences was in Chapter 4 to tune the various parameters of our tracker to improve overall accuracy. The third was in Chapter 5 to generate examples in order to train a CNN to map directly from silhouette space to pose space. This was initially done solely for the purpose of initialization, but ended up being used for particle propagation when tracking human models. Finally, synthetic data was used to evaluate the performance of our tracker under a variety of scenarios.

Another important contribution that stems from our research, from an engineering perspective, is the development of the tracking framework itself. We have shown our framework to be adaptable to different models, and flexible enough to allow different motion priors and particle sampling strategies. As we use the particle filter as a tool to fuse predictions and recorded data, we have also shown that additional sources of information can be added to our framework.

6.1 Further Work

As we have shown that the silhouette segmentation is the biggest limiting factor in the accuracy, and in most case, functionality of our tracker, a more careful review of segmentation

algorithms could prove to be the next logical step toward obtaining a tracker that generalizes more readily to real-world sequences.

Along the same line of thought, standard data augmentation techniques [Witten et al., 2016] targeting known issues with silhouette detection could also be used to provide a larger coverage of silhouettes that might be encountered in real video sequences. These issues include the addition of shadows and noise as well as removing limbs that may be mislabeled when segmenting the silhouette.

As the models used to train the CNN are generated from parameters in MakeHuman [Team, 2001], we could leverage that additional data in a future version. By adding certain parameters to the pose space, we could try and obtain approximations of age, sex, etc. This additional data could also enable us to select a model from a database to maximize similarity to the subject being tracked. By changing the mesh model used to evaluate particles, we should get better tracking performance as the silhouette of the selected model might better match the observed silhouette. This data may also be valuable for certain applications such as targeted advertising, where the additional information could be used to provide advertisements based on the age and sex of the viewer.

As seen in Section 2.4.1, we can use a neural network to generate silhouettes from pose vectors. It may be possible to replace the renderer from our framework with this network. The possible advantage is that by training this network with a wide variety of 3D human models, we could generate a "pseudo" silhouette that would capture how different models would look with the given pose. The generated silhouette would be more of a likelihood map of which pixel should belong to the silhouette than a clear silhouette. Figure 6.1 provides an example of what such a generated silhouette could look like. The pixel count metric could then be replaced by a sum of squared differences to compute the weight of the particles. This could help the system generalize better to different body shapes. The network could also be trained to take into consideration common segmentation issues, as discussed previously.

While outside the scope of the current thesis which aims to work on monocular video

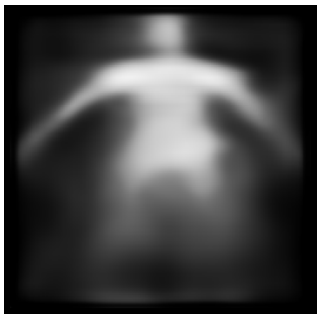


Figure 6.1: Example "silhouette" generated by a neural network that encodes the appearance model of multiple human body shapes.

sequences, we have modified our tracker to work with multiple cameras. The addition of a second camera roughly halves both the pose error and the end effector position error in the small scale tests. As this is not directly related to the goals of this document, the results of these experiments are not reported here, but are available on the companion website. The most important aspect of this experiment is that it shows that our approach is extensible and can integrate information from more sources to reduce ambiguity. While we have only tested this on the small scale model, the use of multiple cameras might improve the tracker for real sequences as well. Without going into too much detail, integrating multiple cameras only requires the rendering of an additional virtual silhouette per addtional camera for each particle in the filter. The major downside is thus that adding a camera roughly doubles the computational cost.

Bibliography

Romer Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, page 721–727. IEEE, 2000.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

Y. Wang, J. Min, J. Zhang, Y. Liu, F. Xu, Q. Dai, and J. Chai. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics*, 32(4): 43, 2013.

U. Güdükbay, I. Demir, and Y. Dedeoğlu. Motion capture and human pose reconstruction from a single-view video sequence. *Digital Signal Processing*, 23(5):1441–1450, 2013.

Nicholas R Howe. Silhouette lookup for automatic pose tracking. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, page 15–22. IEEE, 2004.

Yiğithan Dedeoğlu, B Uğur Töreyin, Uğur Güdükbay, and A Enis Çetin. Silhouette-based method for object classification and human action recognition in video. In *Computer Vision in Human-Computer Interaction*, page 64–77. Springer, 2006.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

Greg Mori and Jitendra Malik. Recovering 3d human body configurations using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1052–1062, 2006.

Leonid Sigal and Michael J Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *Brown Univertsity TR*, 120, 2006.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2014.

D. Salmond and N. Gordon. An introduction to particle filters. *State space and unopbserved component models theory and applications*, page 1–19, 2005.

J.J. Koenderink et al. What does the occluding contour tell us about solid shape. *Perception*, 13(3):321–330, 1984.

Andrew Phan and Frank P Ferrie. Towards 3d human posture estimation using multiple kinects despite self-contacts. In *Machine Vision Applications (MVA), 2015 14th IAPR International Conference on*, page 567–571. IEEE, 2015.

Olivier St-Martin Cormier, Andrew Phan, and Frank P Ferrie. Situational awareness for manufacturing applications. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, page 320–327. IEEE, 2015.

Olivier St-Martin Cormier and Frank P Ferrie. Evaluation of shape description metrics applied to human silhouette tracking. In *Computer and Robot Vision (CRV), 2016 13th Conference on*. IEEE, 2016.

Brian Gleeson, Katelyn Currie, Karon MacLean, and Elizabeth Croft. Tap and push: Assessing the value of direct physical control in human-robot collaborative tasks. *Journal of Human-Robot Interaction*, 4(1):95–113, 2015.

Brian Gleeson, Karon MacLean, EA Croft, and J Alcazar. Human-robot communication for collaborative assembly. In *GRAND Symposium*, 2013a.

Brian Gleeson, Karon MacLean, Amir Haddadi, Elizabeth Croft, and Javier Alcazar. Gestures for industry: intuitive human-robot communication from human observation. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 349–356. IEEE Press, 2013b.

Amir Haddadi, Elizabeth A Croft, Brian T Gleeson, Karon MacLean, and Javier Alcazar. Analysis of task-based gestures in human-robot interaction. In *2013 IEEE International Conference on Robotics and Automation*, pages 2146–2152. IEEE, 2013.

Justin W Hart, Brian Gleeson, Matthew Pan, AJung Moon, Karon MacLean, and Elizabeth Croft. Gesture, gaze, touch, and hesitation: Timing cues for collaborative work. In *HRI Workshop on Timing in Human-Robot Interaction, Bielefeld, Germany*, page 21, 2014.

Justin W Hart, Sara Sheikholeslami, Brian Gleeson, Elizabeth Croft, Karon MacLean, Frank P Ferrie, Clément Gosselin, and Denis Laurandeau. Developing robot assistants with communicative cues for safe, fluent hri. In *Foundations of Trusted Autonomy*, pages 247–270. Springer, 2018.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

Isaac Newton, Daniel Bernoulli, Colin MacLaurin, and Leonhard Euler. *Philosophiae naturalis principia mathematica*, volume 1. excudit G. Brookman; impensis TT et J. Tegg, Londini, 1833.

E.T. Whittaker. *A treatise on the analytical dynamics of particles and rigid bodies: with an introduction to the problem of three bodies*. CUP Archive, 1904.

Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

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

Sheldon Andrews, Ivan Huerta, Taku Komura, Leonid Sigal, and Kenny Mitchell. Real-time physics-based motion capture with sparse sensors. In *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)*, page 5. ACM, 2016.

J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, et al. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2821–2840, 2013.

Sebastian Lieberknecht, Andrea Huber, Slobodan Ilic, and Selim Benhimane. Rgb-d camera-based parallel tracking and meshing. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 147–155. IEEE, 2011.

Maurice F Fallon, Hordur Johannsson, and John J Leonard. Efficient scene simulation for robust monte carlo localization using an rgb-d camera. In *2012 IEEE international conference on robotics and automation*, pages 1663–1670. IEEE, 2012.

M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *Proceedings of the Eighth IEEE International Conference on Computer Vision, ICCV 2001*, volume 2, page 34–41. IEEE, 2001.

P. Szczuko. Genetic programming extension to apf-based monocular human body pose estimation. *Multimedia Tools and Applications*, 68(1):177–192, 2014.

J.M. del Rincón, D. Makris, C.O. Uruñuela, and J-C Nebel. Tracking human position and

lower body parts using kalman and particle filters constrained by human biomechanics. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(1):26–37, 2011.

V. Klinger and M. Arens. Ragdolls in action - action recognition by 3d pose recovery from monocular video. In *International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing*, page 219–223, 2009.

M. Vondrak, L. Sigal, and O.C. Jenkins. Dynamical simulation priors for human motion tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):52–65, 2013.

S. Sedai, M. Bennamoun, and D.Q. Huynh. A gaussian process guided particle filter for tracking 3d human pose in video. *IEEE Transactions on Image Processing*, 22(11), 2013.

C. Stoll, J. Gall, E. De Aguiar, S. Thrun, and C. Theobalt. Video-based reconstruction of animatable human characters. In *ACM Transactions on Graphics (TOG)*, volume 29, page 139. ACM, 2010.

M.A. Brubaker, D.J. Fleet, and A. Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 87(1-2):140–155, 2010.

D.J. Duff, T. Morwald, R. Stolkin, and J. Wyatt. Physical simulation for monocular 3d model based tracking. In *IEEE International Conference on Robotics and Automation*, page 5218–5225, 2011.

R.T. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, et al. *A System for Video Surveillance and Monitoring*, volume 2. Carnegie Mellon University, the Robotics Institute Pittsburg, 2000.

C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time track-

ing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2. IEEE, 1999.

A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, 2002.

Gwangmo Song, Heesoo Myeong, and Kyoung Mu Lee. Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1760–1768, 2018.

S. Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, page 511–518. Morgan Kaufmann Publishers Inc., 2002.

M. Vondrak, L. Sigal, and O.C. Jenkins. Physical simulation for probabilistic motion tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, page 1–8, 2008.

Ralf Plänkers and Pascal Fua. Articulated soft objects for multiview shape and motion capture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1182–1187, September 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1227995. URL `https://doi.org/10.1109/TPAMI.2003.1227995`.

Joel Mitchelson and Adrian Hilton. Hierarchical tracking of multiple people. In *British Machine Vision Conference*, 2003.

J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 2, pages 126–133 vol.2, June 2000. doi: 10.1109/CVPR.2000.854758.

C.R. Wren and A.P. Pentland. Dynamic models of human motion. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 22–27. IEEE, 1998.

C.R. Wren and Alex P. Pentland. Understanding purposeful human motion. In *IEEE International Workshop on Modelling People*, page 19–25. IEEE, 1999.

V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3d pictorial structures for multiple human pose estimation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

D.J. Duff, J. Wyatt, and R. Stolkin. Motion estimation using physical simulation. In *IEEE International Conference on Robotics and Automation*, page 1511–1517, 2010.

D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6): 580–591, 1993.

A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, page 215–222, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4. doi: 10.1145/74333.74355. URL `http://doi.acm.org/10.1145/74333.74355`.

X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Computer Graphics and Interactive Techniques*, page 43–50. ACM, 1994.

M.A. Brubaker, L. Sigal, and D.J. Fleet. Estimating contact dynamics. In *IEEE 12th International Conference on Computer Vision, 2009*, page 2389–2396, 2009.

D. Baraff. Physically based modeling: Rigid body simulation. *ACM SIGGRAPH Course Notes*, 2(1), 2001.

Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 15(49):5, 2013.

Nvidia. Nvidia physx. `https://developer.nvidia.com/physx`, 2001. Accessed: 2013-07-12.

R. Smith et al. Open dynamics engine. `http://www.ode.org/`, 2005. Accessed: 2013-07-12.

Marek Vondrák. Crisis physics library. `http://crisis.sourceforge.net/`, 2005. Accessed: 2014-10-29.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4397–4404. IEEE, 2015.

Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

Francesc Moreno-Noguer. 3d human pose estimation from a single image via distance matrix regression. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1561–1570. IEEE, 2017.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017.

Sungheon Park, Jihye Hwang, and Nojun Kwak. 3d human pose estimation using convolutional neural networks with 2d pose information. In *European Conference on Computer Vision*, pages 156–169. Springer, 2016.

Xiaowei Zhou, Menglong Zhu, Georgios Pavlakos, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Monocap: Monocular human motion capture using a cnn

coupled with a geometric prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4966–4975, 2016.

Yu Du, Yongkang Wong, Yonghao Liu, Feilin Han, Yilin Gui, Zhen Wang, Mohan Kankanhalli, and Weidong Geng. Marker-less 3d human motion capture with monocular image sequence and height-maps. In *European Conference on Computer Vision*, pages 20–36, 2016.

Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Direct prediction of 3d body poses from motion compensated sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 991–1000, 2016.

Agne Grinciunaite, Amogh Gudi, Emrah Tasli, and Marten den Uyl. Human pose estimation in space and time using 3d cnn. In *European Conference on Computer Vision*, pages 32–39. Springer, 2016.

Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In *Advances in Neural Information Processing Systems*, pages 3108–3116, 2016.

Ching-Hang Chen and Deva Ramanan. 3d human pose estimation= 2d pose estimation+ matching. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, volume 2, page 6, 2017.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from

a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, page 32–36. IEEE, 2004.

D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):97, 2008.

A. Phan. Mcgill-reparti artificial perception database. `http://www.cim.mcgill.ca/~apl/database/`, 2015.

Hashim Yasin, Umar Iqbal, Bjorn Kruger, Andreas Weber, and Juergen Gall. A dual-source approach for 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4948–4956, 2016.

Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994.

MakeHuman Team. Makehuman. `http://www.makehuman.org/`, 2001. Accessed: 2018-03-19.

Remco C Veltkamp. Shape matching: similarity measures and algorithms. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, page 188–197. IEEE, 2001.

Ronald Poppe and Mannes Poel. Comparison of silhouette shape descriptors for example-based human pose recovery. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, page 541–546. IEEE, 2006.

William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, page 73–78, 2003.

A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Karl Pearson. Correlation coefficient. In *Royal Society Proceedings*, volume 58, page 214, 1895.

Raphaël Canals, Ali Ganoun, and Rémy Leconge. Occlusion-handling for improved particle filtering-based tracking. In *Signal Processing Conference, 2009 17th European*, pages 1107–1111. IEEE, 2009.

Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 212–219. IEEE, 2005.

Carine Hue, J-P Le Cadre, and Patrick Pérez. Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):791–812, 2002.

Tiancheng Li, Miodrag Bolic, and Petar M Djuric. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal processing magazine*, 32(3): 70–86, 2015.

John Eaton and al. Gnu octave. `https://www.gnu.org/software/octave/`, 1988. Accessed: 2019-09-23.

SiliconGraphics. Opengl. `https://www.opengl.org/`, 1992. Accessed: 2019-07-11.

KhronosGroup. Opencl. `https://www.khronos.org/opencl/`, 2009. Accessed: 2019-07-11.

KhronosGroup. Vulkan. `https://www.khronos.org/vulkan/`, 2016. Accessed: 2019-07-11.

François Chollet. Keras. `https://keras.io/`, 2015. Accessed: 2019-07-11.

GoogleBrainTeam. Opencl. `https://www.tensorflow.org/`, 2015. Accessed: 2019-07-11.

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

D.D. Corkill, K.Q. Gallagher, and P.M. Johnson. Achieving flexibility, efficiency and generality in blackboard architectures. In *Proceedings of the 6th National Conference on Artificial Intelligence*, page 18–23, 1987.

Apache Software Foundation. Couchdb. `http://couchdb.apache.org/`, 2005.

Salvatore Sanfilippo and Pieter Noordhuis. Redis. `http://redis.io`, 2009.

Mongodb. `http://www.mongodb.org/`, 2009.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.