UNIVERSITY OF CALIFORNIA SAN DIEGO

Continuous Human Pose Estimation Using Long Short-Term Memory and Particle Filter

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Electrical Engineering
(Machine Learning and Data Science)

by

Chenghao Gong

Committee in charge:

Professor Vikash Gilja, Chair
Professor Truong Nguyen
Professor Nikolay Atanasov

2019

The Thesis of Chenghao Gong is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

TABLE OF CONTENTS

Signature Page . . . iii

Table of Contents . . . iv

List of Figures . . . vi

List of Tables . . . vii

Acknowledgements . . . viii

Abstract of the Thesis . . . x

Chapter 1 Introduction . . . 1

1.1 Motivation . . . 1

1.2 Thesis Overview . . . 2

Chapter 2 Existing Benchmark Human Pose Estimation Post Processing Methods . . . 3

2.1 Caffe Heatmap . . . 3

2.2 PatientPose . . . 4

2.2.1 Kalman Filter in PatientPose . . . 4

Chapter 3 Particle Filter . . . 7

3.1 Motivation . . . 7

3.2 Assumptions of Particle Filter . . . 8

3.3 Implementation Details . . . 9

3.4 Experiment of Particle Filter . . . 10

3.4.1 Experiment Setup . . . 10

3.4.2 Experiment Result . . . 11

Chapter 4 Particle Filter with Long Short-Term Memory . . . 14

4.1 Motivation . . . 14

4.2 LSTM details . . . 15

4.2.1 LSTM Cell . . . 15

4.2.2 LSTM Network . . . 16

4.2.3 Experiment of LSTM . . . 17

4.3 Our Proposed Algorithm . . . 19

4.3.1 Algorithm Detail . . . 19

4.3.2 Noise Estimation . . . 20

4.4 Experiment of Particle Filter with LSTM . . . 21

4.4.1 Experiment Setup . . . 21

4.4.2 Experiment Result . . . 21

Chapter 5 Conclusion and Future Work . . . 27

5.1 General Improvements . . . 27

5.1.1 LSTM Training Data Generation . . . 27

5.1.2 Using Camera Model 28
5.2 Model Improvements 28
5.2.1 Non-Uniformed State Prior 28
5.2.2 Heatmap as Observation Vector 29
5.2.3 More Complex State Transitional Probability 29
5.2.4 Better Noise Parameter Estimation 30
Appendix A Handling Numerical Issues When Multiplying the Probabilities in the Heatmaps 32
Appendix B Human Pose Data Augmentation 33
References 34

LIST OF FIGURES

Figure 2.1: Markov Model 4

Figure 3.1: Counter Example Heatmap 8
Figure 3.2: Result of Particle Filter Experiment 12
Figure 3.3: Example of Heatmap for all the joints 13

Figure 4.1: Details of LSTM Cell 16
Figure 4.2: Network Structure 17
Figure 4.3: Result of LSTM Experiment 23
Figure 4.4: Comparison of LSTM and Linear Kinematics Covariance Matrix 24
Figure 4.5: Example of Right Hand Trajectory 25
Figure 4.6: Algorithm Pipeline 25
Figure 4.7: LSTM State Inheritance 26
Figure 4.8: Result of Particle Filter with LSTM Experiment 26

Figure 5.1: Add Heatmap as LSTM Input 29
Figure 5.2: 2D Histogram of the Right Hand Error 31

LIST OF TABLES

Table 3.1: Pose estimation accuracy rates at 10 pixels (Particle Filter) 11

Table 4.1: Pose estimation accuracy rates at 10 pixels (LSTM + Particle Filter) 21

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Vikash Gilja for his effort in guiding me on how to be a researcher. Vikash is the one who was there for me in the first place when I started doing research in the first time of my whole life, and he has guided me with great patience and thoughtfulness. I will never be able to make it on my master thesis without Vikash. Then I would like to thank my committee members professor Truong Nguyen and professor Nikolay Atanasov. Professor Nguyen really thinks for his student, and he is always supportive and responsive when I needed help. I have gained so much help from professor Nguyen not only in my thesis but also on my long term developments. I gained the inspiration for my master thesis from professor Atanasovs ECE276A: Sensing and Estimation in Robotics. His instruction on Bayesian filtering and profound insight into the field of robotic helps me a lot to shape the topic and implementation of my master thesis. I would like to thank my current and previous labmates including Paolo Gabriel, Kenny Chen, Abdulwahab Alasfour, Haotian Zhang, Nick Rogers, Tejaswy Pailla, John Hermiz, Daril Brown, Pablo Tostado, and Abdullah Alothman for their companions in my time spending in UC San Diego Transitional Neuroengineering Lab and helpful discussion on my thesis. I would like to special thanks to Kenny Chen and Paolo Gabriel for their unique contribution to my master thesis. My project is a follow-up of Kennys master thesis, and Kenny was always helpful when I have trouble understanding and implementing his work. Paolo is like a peer mentor to me, and he has taught me a lot on research methods and engineering philosophy.

I also want to thank my parents for their unconditional love and support in my entire 5 years BSMS program, and I know they will always be there for me. This makes my life much easier when the difficulties come to me. Finally, I would like to thank my friends who always be on my side and give me companions. I will not be successful in my study at UC San Diego without them.

This thesis, in full is currently being prepared for submission for publication of the material. Gong, Chenghao; Gabriel, Paolo; Gilja, Vikash. The dissertation/thesis author was the primary author of this

material.

ABSTRACT OF THE THESIS

Continuous Human Pose Estimation Using Long Short-Term Memory and Particle Filter

by

Chenghao Gong

Master of Science in Electrical Engineering

(Machine Learning and Data Science)

University of California San Diego, 2019

Professor Vikash Gilja, Chair

Estimating human pose in a continuous time series has many practical applications. For example, imagine that some time in the future robot would like to interact with human beings, for that robot to meaningfully interact with a human it needs to interpret and anticipate human movements and gestures. Acquiring continuous human pose estimates can also inform specific applications like brain-machine interface; specifically, we can use accounts of human pose data across time to study the relationship between neural signals and human pose. In this thesis, we will focus our work on the continuous human pose estimation in the clinical environment.

There are many existing methods for estimating human pose from camera image, and many of

them employ deep learning and convolutional neural network (CNN) architecture, which are widely used in computer vision. However, after estimating possible human poses from a single image frame, might we be able to use the statistical regularity of human movement to improve pose estimation? In this work we demonstrate that by modeling this regularity across time pose estimation can be improved. We demonstrate this by applying a post-processing method to confidence maps of pose generated using existing computer vision methods applied to each frame. Our post-processing method models movement using a long short-term memory (LSTM) network and a particle filter based framework for estimation.

Chapter 1

Introduction

1.1 Motivation

The primary motivation of this work is to improve human pose and movement estimation in clinical settings. Human pose estimation in general is an important research area with applications on neuroscience and robotics. Recently, deep-learning based methods like DeepPose [1] and OpenPose [2], have achieved excellent performance on this task in uncluttered environments. However, in cluttered environments with high lighting variability these methods are less reliable. In general, these methods do not model human movement dynamics and thus one strategy for improving estimation is to integrate information across time.

There are previous studies that take the temporal dynamics into consideration when performing the human pose estimation. For example, Pfister [3] proposes to use optical flow to warp the current frame information onto the next frame to provide a regulation that constrains the predictions with natural human kinematics. However, this work did not fully utilize the temporal information, nor did it build a accurate model for the human motion. Therefore, in the current work, we propose a post-process method, applied to single image estimates of human pose. We introduce a particle filter [4] that integrates single image confidence maps across time and we utilize a Long Short Term Memory (LSTM) network architecture [5] to constrain pose estimates to naturalistic human kinematics. In addition to integrating confidence across time,

the proposed method utilizes soft estimates of human joint positions, with the ability to maintain multiple hypotheses, as opposed to making a hard choice at every time-step. In order to validate our method, we apply the method to two video-based human pose datasets, PatientPose [6] and HumanEva [7], and demonstrate the proposed method results in substantial improvement compared to existing post-process method in human pose estimation.

1.2 Thesis Overview

The remainder of this thesis is organized into four chapters, organized as follows:

In chapter 2, we will present existing algorithms that we will use as benchmarks to evaluate our proposed algorithm.

In chapter 3, we will illustrate how particle filtering can provide us with a better state-observation model compared to existing methods, and we detail our implementation. We will then benchmark the result of this stage.

In chapter 4, we describe why the LSTM can provide a better model of natural human movement kinematics when compared to the linear kinematics that was used previously, and we will integrate the LSTM into our particle filtering algorithms and will benchmark the results.

In chapter 5, we conclude with a description of the contributions of this work and we discuss the potential applications for which these methods are suitable. We also summarize the directions that this work could go further, and specifically what future work can be done to improve this method.

Chapter 2

Existing Benchmark Human Pose Estimation Post Processing Methods

We build off of PatientPose which is prior work that adds a linear kinematic model to the Caff Heatmap pose estimator. Thus, we briefly describe Caffe Heatmap and how PatientPose adds a post processing step to model kinematics.

2.1 Caffe Heatmap

Pfister [3] uses the convolutional neural network(CNN) [8] to regress heatmaps for the possible location of each joint of human pose. The method then aligns the heatmaps between consecutive frames with optical flow as a basic method to provide a temporal constraint between successive frames. This method, named Caffe Heatmap, achieves impressive results on uncluttered images. However, Chen [6] found that Caffe Heatmap performance drops in noiser and more cluttered environments. The Caffe Heatmap is very general of the model and has a very limited model of temporal dynamics. Therefore, Chen [6] proposed PatientPose as a the builds off of Caffe Heatmap and adapts it specifically to the patient monitoring scenario.

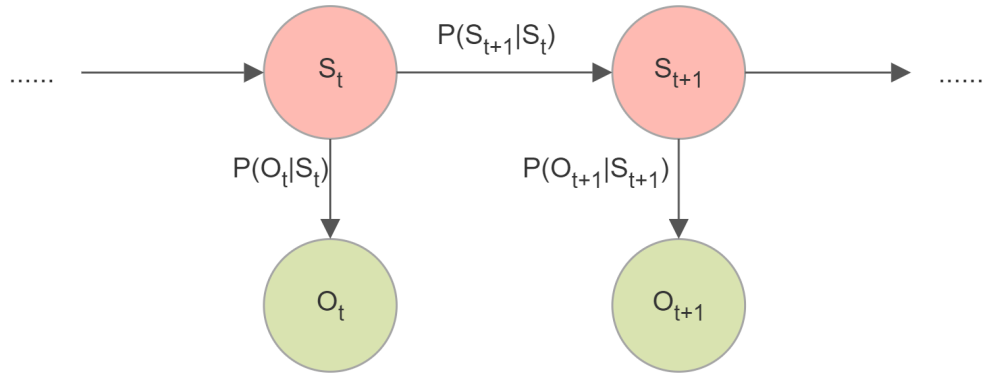

Figure 2.1: **Markov Model.** Hidden Markov model assumes that the current state is d-separated to other node by the previous state, while current observation is d-separated to other node by the current state.

2.2 PatientPose

In order to achieve better performance for subject-specific estimation in the clustered environment of the clinic, Chen [6] introduced a methodology that is built upon the Caffe Heatmap architecture. In contrast to the traditional human pose estimation, Chen utilizes 1. Image pre-process including cropping and lighting normalization, 2 subject-specific training with high variance and high-quality data, and 3, post-processing using standard Kalman filter [9] with statistically learned noise parameter individually for each human joints. In our work, we will benchmark the impact of replacing the Kalman filter with our proposed post-processing algorithm.

2.2.1 Kalman Filter in PatientPose

Markov Model

The Kalman filter is a Markov model. In a Markov model, as shown on figure 2.1, we assume the current state is conditionally independent to other nodes by given the previous state (Markov Assumption), and current observation is conditional independent to other nodes by given the current state. Therefore, there are two very important probabilities in Markov Models: state transitional probability: $P(S_{t+1}|S_t)$ and

state to observation probability: $P(O_t|S_t)$.

Kalman Filter Assumptions

As a Markov Model, the Kalman filter is also built upon the Markov assumption, and Kalman filter assumes that the world is built with states and observations. The Kalman filter can be used to estimate the state of the world given a set of the observations. This estimation or filtering has two steps: 1. **Prediction:** this is a forward prediction of the state and its error covariance based upon the current state estimate, 2. **Update:** then we revise the prediction based a new observation made at this next time step.

In PatientPose [6], the Kalman Filter assumes that the true joint locations and velocities of human poses are states, S_t, and the highest confident location of the joints from Caffe Heatmap are the observations, O_t:

$$S_t = \begin{bmatrix} Location_x\, at\, time\, t \\ Location_y\, at\, time\, t \\ Velocity_x\, at\, time\, t \\ Velocity_y\, at\, time\, t \end{bmatrix}$$

and

$$O_t = \begin{bmatrix} Observed\, location_x\, at\, time\, t \\ Observed\, Location_y\, at\, time\, t \end{bmatrix}$$

PatientPose assumes that the state transitional probability $P(S_{t+1}|S_t)$ is a Gaussian distribution with a mean that is a linear transformation of S_t (as in [10]) with a variance of Q. In patient pose, the state to observation probability $P(O_t|S_t)$, on the other hand, is a Gaussian distribution with a mean that is assumed to be the actual joint locations. This observation model has a variance parameter R.

Kalman Filter Parameter Estimation

While the means of both $P(S_{t+1}|S_t)$ and $P(O_t|S_t)$ can be specified directly from the definition of the states and observations, we need to lean the variance parameters Q and R. In PatientPose, Chen utilized Maximize Likelihood Estimation inspired by Abbeel et al. [11] with the close-form solution for both Q and R as following:

$$Q_{MLE} = \frac{1}{T}\sum_{t=1}^{T}(S_t - AS_{t-1})(S_t - AS_{t-1})^\top, \tag{2.1}$$

$$R_{MLE} = \frac{1}{T+1}\sum_{t=0}^{T}(O_t - HS_t)(O_t - HS_t)^\top. \tag{2.2}$$

with

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is the linear state transformation that AS_t will give us the rough prediction of the S_{t+1}, in the prediction step of the Kalman filter. This transformation assumes that the next position is simply determined by integrating velocity over the time step duration. This model also assumes velocity is constant.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

is the state to observation transformation that HS_t will yield the expected observation as a read out of the position terms from the state vector.

Chapter 3

Particle Filter

3.1 Motivation

In chapter 2 we described the assumptions made by the Kalman filter made as applied by PatientPose. Recall that the model assumes that both the state transitional probability $P(S_{t+1}|S_t)$ and the state to observation probability $P(O_t|S_t)$ are Gaussian distribution. In this section, we will focus on the definition of $P(O_t|S_t)$.

As stated on chapter 2, Kalman filter in PatientPose treat $P(O_t|S_t)$ as Gaussian distribution with a mean equals the position terms of the S_t. When applying the filter, PatientPose assumes the observations are the peak values in the heatmap. Intuitively speaking, assuming that the peak is the observation will result in a loss of all other information contained in the heatmap. For example, if we look at the example heatmap of a joint calculated from a real image frame shown in figure 3.1, we see that there are multiple possible peaks inside the heatmap. It is likely that the true joint position is near one of these peaks. If the most possible location inside this heatmap is not the ground true, the observation provided to the filter will be far from the true joint location.

Ideally, we would like to model the possibility of the joint being observed at any of the high-confidence areas of the heatmap. Unfortunately, the standard Kalman filter is not easily adapted to such a

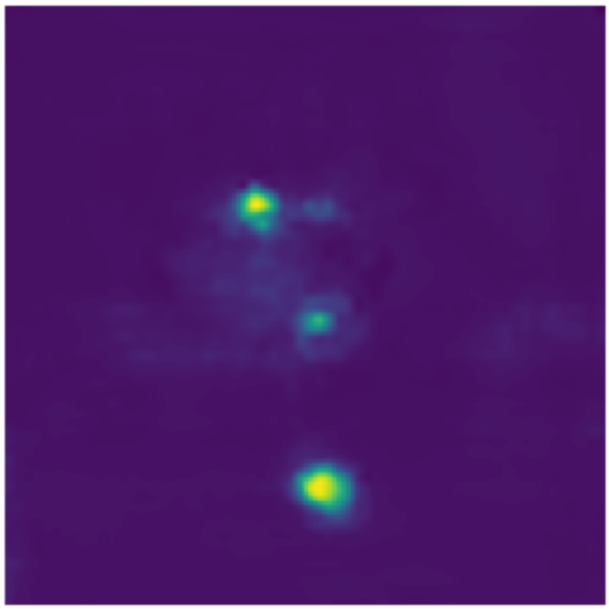

Figure 3.1: **Counter Example Heatmap.** This is a counter example of why only treat the peak of the heatmap as observation will cause the loss of information.

complex distribution, as it only tracks a state mean and covariance over time. Because of this shortcoming of the Kalman filter approach used in PatientPose, we decided to use a particle filter [4] to replace the Kalman filter as a post-processing methodology in continuous human pose estimation task. The particle filter, like the Kalman filter, is also a method to estimate the internal state when only partial observations are given. In contrast to the Kalman filter, the particle filter uses a collection of the "particles" to represent the state probability instead of assuming it is a Gaussian distributed. By having a set of discrete particles, the particle filter, unlike the Kalman filter, can maintain multiple hypnosis.

3.2 Assumptions of Particle Filter

In our initial implementation of the particle filter, like the Kalman Filter, we assume that this continuous human pose estimation task has linear kinematics. Thus, $P(S_{t+1}|S_t)$ is unchanged from the Kalman Filter.

The only change is in our model of the observations, $P(O_t|S_t)$. Instead of treating it as a Gaussian

distribution, for the particle filter we will use the heatmap to estimate this distribution. We normalize the heatmap to sum to 1 and assume that this normalized heat map is $P(S_t|O_t)$. Recall that the original heatmap values establish a confidence. According to Bayesian rule, our state to observation probability is $P(O_t|S_t) = \frac{P(S_t|O_t)P(O_t)}{P(S_t)}$. In our implementation we will assume that $P(O_t)$ and $P(S_t)$, the observation and state priors, are uniform distributions. If these distributions are uniform, their values are constant and so $P(O_t|S_t) = c * P(S_t|O_t)P(O_t)$, where c is a constant.

3.3 Implementation Details

We define a set of particles for each time t: $\hat{S}_t^i$ for i = 1,2,3,...N, where N is the number of particles. To find the true state S_t, we calculate the average of the particles at time t: $S_t(estimated) = \frac{1}{N}\sum_i \hat{S}_t^i$. To initialize the particles for $t = 0$, we sample N samples of $\hat{S}_0^i$ for i=1,2,...,N from the prior probability of $P(S_0)$, which is the first heatmap in time, and initialize the weight of each particle by $w_0^i = \frac{P(O_0|\hat{S}_0^i)}{\sum_j P(O_0|\hat{S}_0^j)}$, where the subscript 0 represents time = 0, and the superscript i refers to this being the i^{th} particle. Therefore, every w_0^i is proportional to the state to observation probability at time = 0 and all the weights sum up to 1. Below is the algorithm:

- **Sample N particles from** $P(S_0)$ get $\hat{S}_0^i$ for i=1,2,...,N
- **Weight each particles** $w_0^i = \frac{P(O_0|\hat{S}_0^i)}{\sum_j P(O_0|\hat{S}_0^j)}$
- **Looping through all the time t=1,2,...,T** :
    - **Looping through all the particles i=1,2,...,N** :
        - **Importance Sampling** Sample a particle $\hat{S}_{t-1}^k$ from t-1 using weight w_{t-1}, i.e the bigger the weight w_{t-1}^a, the more likely that $\hat{S}_{t-1}^a$ is sampled as $\hat{S}_{t-1}^k$
        - **Sequential Update** Sample $\hat{S}_t^i$ from $P(S_t|\hat{S}_{t-1}^k)$

· **Weighting** Weight $\hat{S}_t^i$ by $w_t^i = P(O_t|\hat{S}_t^i)$

∗ **Output the mean of all the particles**

∗ **Sum the weight** $WeightSum_t = \sum_i w_t^i$

∗ **For i=1,2,...,N** :

· **re-weight** $w_t^i = \frac{w_t^i}{WeightSum_t}$

One important note is that we don't actually calculate $P(O_t|\hat{S}_t^i)$ in the implementation. Instead, for each particle, we use the value of $P(\hat{S}_t^i|O_t)$. As described above, we assume uniform priors for the observation and the state, so $P(O_t|\hat{S}_t^i)$ and $P(\hat{S}_t^i|O_t)$ only differ by a constant scale term c. During the re-weighting step, this scale term cancels out.

As was done with the Kalman filter in PatientPose [6], we run this particle filter algorithm individually for each human joints that we are trying to track.

3.4 Experiment of Particle Filter

3.4.1 Experiment Setup

In order to test whether replacing the Kalman filter with particle filter improves the continuous human pose estimation result, we choose to use the same testing data for the same one subject with PatientPose [6], and we also use the same Caffe Heatmap model [3] to extract the confidence maps. For the particle filter, the number of particles is 500.

The selected joints to perform the experiment are head, both hands, both elbows, and both shoulders. We evaluate the result by calculating the Euclidean distance in image pixels from the estimated position to the ground truth for each human joint.

Table 3.1: **Pose estimation accuracy rates at 10 pixels (Particle Filter).** Continuous pose estimation accuracy [%] at a 10-pixel tolerance for head, hands, elbows, and shoulders with average the accuracy of left and right side of the body.

Method	Head	Hands	Elbows	Shoulders
Heatmap + Kalman Filter	**100.0**	45.2	69.5	**97.1**
Heatmap + Particle Filter	99.7	**50.0**	**82.9**	94.8

3.4.2 Experiment Result

Figure 3.2 compares Caffe Heatmap + Kalman Filter versus the Caffe Heatmap + particle filter. Please note the the y-axis of the plot is the accuracy if we consider a prediction to be correct when the error in distance between the prediction and ground truth is smaller than the x-axis value. We also provide Table 3.1 in which we note the accuracy when a prediction within 10 pixels is considered to be correct (this distance is visually indistinguishable by human annotators in a 256X256 image). From the results, we can see that the hands and elbows tends to have a much better result for Caffe Heatmap + particle filter comparing with Caffe Heatmap + Kalman filter (+4.8% for hands and +13.4% for elbows), with a slight under-performance for the particle filter in estimating the head and shoulders (-0.3% for head and -2.3%). Note that hands and elbows are typically more difficult to track because like in figure 3.3, hands and elbows tend to have much more noisy heatmaps and move more quickly. This suggests that particle filtering can out-perform the Kalman filter as a post-processing method in continuous human pose estimation task when heatmaps are noisy.

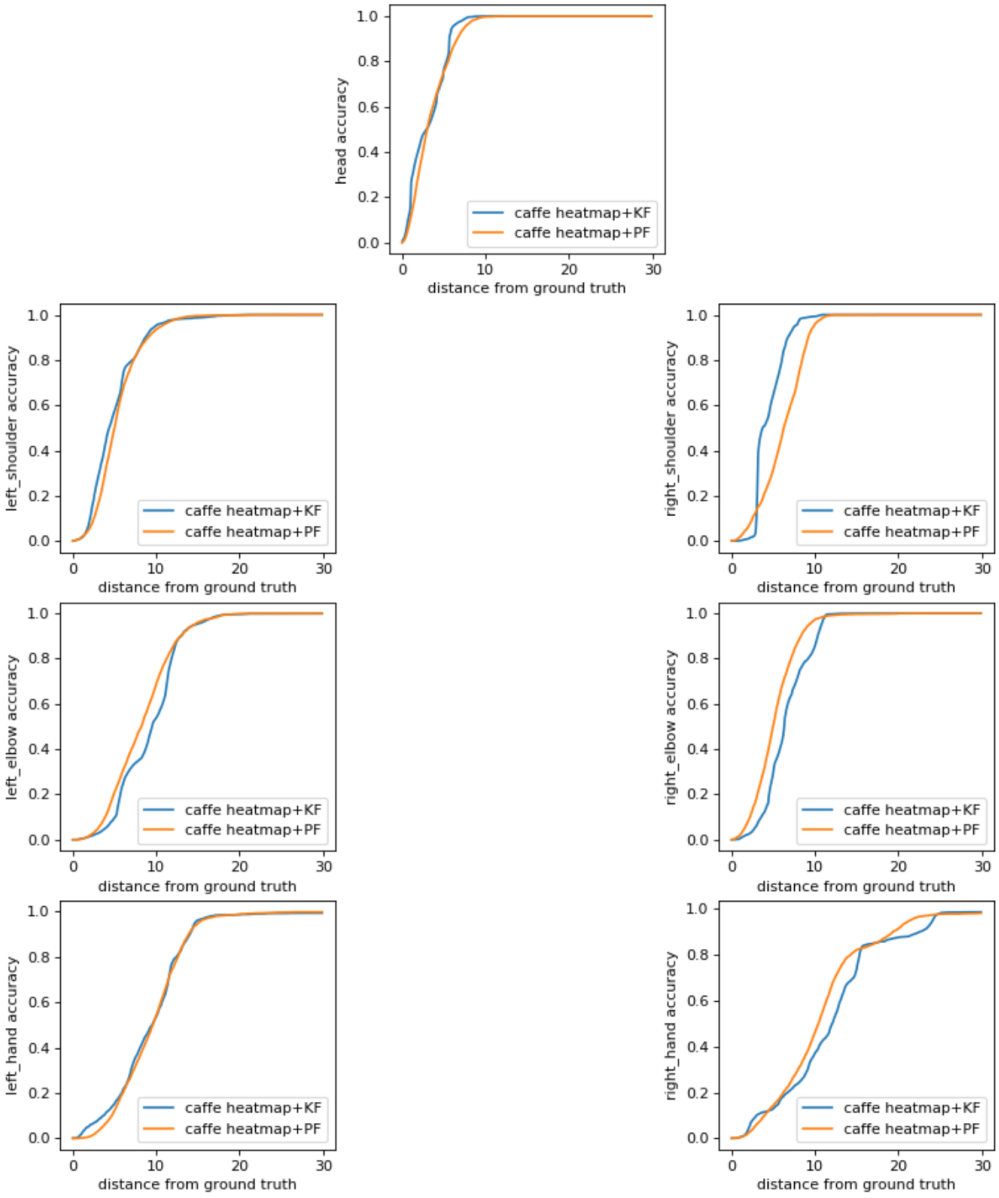

Figure 3.2: **Result of Particle Filter Experiment.** This is the result of Caffe heatmap + Kalman filter versus Caffe Heatmap + particle filter. The accuracy is plotted as a function with respect to a threshold distance from the ground truth under which the estimate is deemed correct.

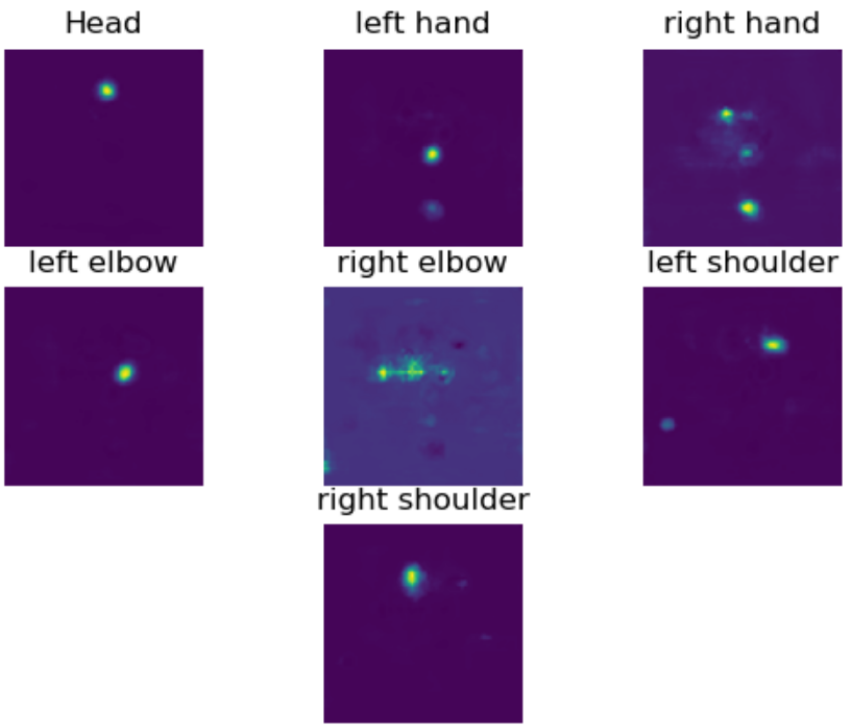

Figure 3.3: **Example of Heatmap for all the joints.** This is a example of heatmaps for all the targeted human joints from which we can see that the hands and elbows tend to have a more complicated heatmap comparing to head and shoulders.

Chapter 4

Particle Filter with Long Short-Term Memory

4.1 Motivation

In Chapter 3, we motivated and demonstrated an improvement over the Kalman filter approach used in PatientPose by changing the assumptions of state to observation probability $P(O_t|S_t)$. In this Chapter, we will address the limitations imposed by modeling kinematics are Markovian and limiting the state transitional probability $P(S_{t+1}|S_t)$ to linear kinematics.

In Chapter 2, we mentioned that in the Kalman filter, the state transition probability is a Gaussian distribution whose mean is a linear transformation of the previous state S_t. However, we know that measurements of human motion in the 2D image plane is better described by a nonlinear dynamical process because the movement is constrained by the length of the arms and the angle between arms. Another aspect of human movement is that pose can have long term dependencies in time because there are some common movements like scratching the head and thus movement trajectories can be highly predictable. Therefore, we were interested to see what would happen if the model of kinematics was no longer constrained by the Markovian assumption, by effectively changing the modeled state transitional probability from $P(S_{t+1}|S_t)$

to $P(S_{t+1}|S_{1:t})$ with mean equals to a nonlinear function $f(S_{1:t})$ that takes the input from time 1 to t.

We model this function using the Long Short-Term Memory (LSTM) [5] neural network architecture. LSTM is a special type of recurrent neural network (RNN) RNNs, in general, are designed to model sequence data, including time series. The LSTM is a specific type of RNN that was designed in part to overcome a degeneracy present in many simplete RNN structures, which tend to lose long term information due to the exponential decay of the gradient of the loss function over time. The LSTM is designed with the potential to save the information from the beginning of the time series into its cell state, and thus the output of LSTM can effectively consider all the information before the current state. Moreover, the activation functions like sigmoid function allow the LSTM to model non-linearities. Therefore, the LSTM can be applied to model state transitions as a nonlinear function that takes long term relation into account. We will use the LSTM as an estimator of the the mean of $P(S_{t+1}|S_{1:t})$ and for simplicity we will assume that this distribution is Gaussian with a variance term that we will infer from error in the LSTM estimator.

4.2 LSTM details

4.2.1 LSTM Cell

The LSTM has three different gates: forget gate, input gate, and output gate and is illustrated in figure 4.1. The LSTM is driven by the following equations:

$$f_t = \sigma(W_f[h_t, x_t] + b_f) \tag{4.1}$$

$$i_t = \sigma(W_i[h_t, x_t] + b_i) \tag{4.2}$$

$$C_t^{\cdot} = tanh(W_C[h_t, x_t] + b_C) \tag{4.3}$$

$$c_{t+1} = f_t * c_t + i_t * C_t^{\cdot} \tag{4.4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4.5}$$

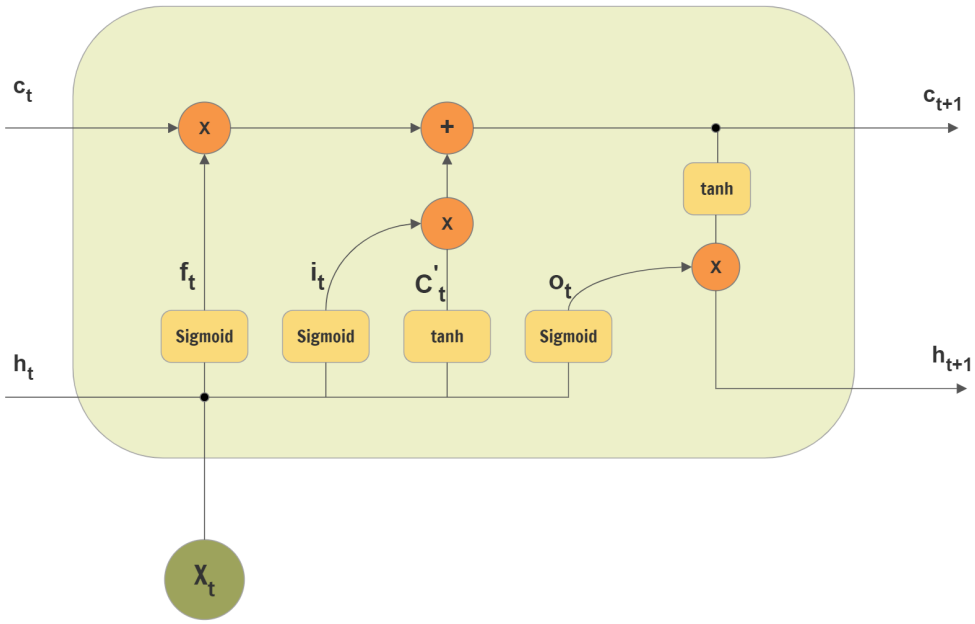

Figure 4.1: **Details of LSTM Cell.** A LSTM cell is consisted of a forget gate, a input gate and a output gate.

$$h_{t+1} = o_t * tanh(c_{t+1}) \tag{4.6}$$

4.2.2 LSTM Network

Our goal is to estimate the mean of $P(S_{t+1}|S_{0:t})$ and using the LSTM to estimate this value allows us to make a one step prediction of the human pose that can maintain long term information in the LSTM's hidden state (h_t) and cell state (c_t) is desired. Therefore beside the LSTM cell itself, we will also need a dense layer to output the prediction of the next human pose. The basic structure of the network can be seen in figure 4.2, from which we can see for each time step t, we will input the current human pose S_t and we will expect the network to predict the next step $\hat{S}_{t+1}$.

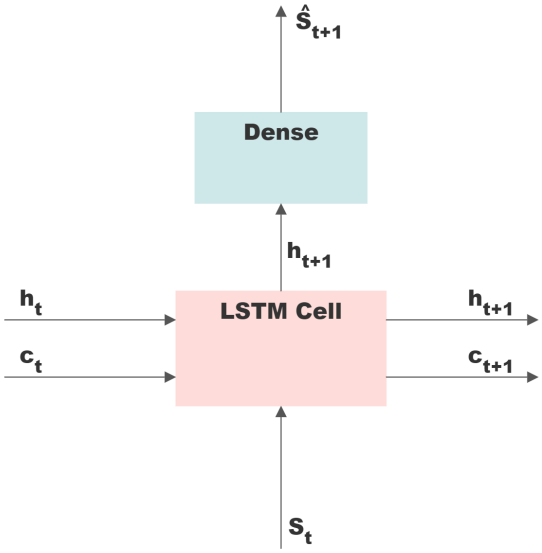

Figure 4.2: **Network Structure.** We use LSTM to perform one step prediction with the inputs are last hidden state (h_t), last cell state (c_t), and current human pose (S_t), while the outputs are the prediction of the next human pose (S'_{t+1}).

4.2.3 Experiment of LSTM

Since the purpose of using LSTM in combining with particle is to replace the mean of the state transitional probability from the linear kinematics AS_t with

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

to the nonlinear long term relationships $f(S_{0:t})$, it is significant to show that LSTM is better than linear kinematics in hidden state estimation problem. Therefore, we design a experiment that can illustrate that LSTM can out-perform linear kinematics when we have limited information about the true state. In this experiment, we will use the "Boxing", "Gesturing", and "Throwing and Catching" behaviors in HumanEva [7] data-set.

LSTM Training

The first step is training the LSTM. LSTM training requires the continuity inside each batch. However, the number of segments that are continuous inside the date-set is limited. Therefore, we use some data augmentation techniques (See Appendix B) to increase the number of continuous batch of size 32 frames (0.5 second) from 2000 to 8000. Using mean square error as the loss function, Stochastic gradient descent as optimizer with 0.01 as learning rate. We trained the network with all the joint locations is normalized so that the range is from -1 to 1.

Experiment Setup

We will divide the testing data into 500 trials, with each trial contains 40 frames (0.66s) of continuous human joints locations. We will use the first 10 frames of each trial to "warm-up" LSTM to get its h_t and c_t to memorize the trajectory. Then for the next 30 frames, we will let LSTM and linear kinematics "know" the true locations and velocities of the first frame (of course as our design of the LSTM does not take in velocity, the LSTM will only receive position as an input). After that, the LSTM and linear kinematics will have a "race" to keep predicting what will happen without any additional information, and we will compare the accuracy of the last frame in all trials.

Experiment Result

As shown in figure 4.3, in all the targeted joints, the LSTM wins the "race", and this suggests that once LSTM is warmed up with data, it has greater predictive power than simple linear kinematics, which implies that when we are totally blind without observations, the LSTM is more accurate forward estimator than linear kinematics. To better visualize the result, we also plot figure 4.5, from which we can see that in this specific trial, the LSTM has a much better right hand prediction than kinematics when both of them do not have the access to the true state.

Another interesting finding is that when we use maximized likelihood estimation to estimation the error of one step prediction using both LSTM and linear kinematics, as shown in figure 4.4, we see that linear kinematics will have a much larger error comparing with LSTM. This result allows us to make justification of replacing the mean of the state transitional probability from linear kinematics to LSTM one step predictor.

4.3 Our Proposed Algorithm

4.3.1 Algorithm Detail

Now we put LSTM and particle filter together. As in figure 4.6, the input of our algorithm are the pre-obtained confidence like heatmaps to the locations of the human joints and the trained LSTM model that does one-step prediction of the next pose, and the output is the predicted pose estimation for each time step from particle filter. The details of the algorithm are shown below.

- **Step 1** Initialize h_0^i and c_0^i to be the zeros vectors for each particle where i is the index of the particle
- **Step 2** In current step t, for each particle, use current prediction $\hat{S}_t^i$, and h_t^i, c_t^i as input to LSTM to make predictions of $\hat{S}_{t+1}^i$.
- **Step 3** with the predictions of each particle, we are able to get the state transitional probability, thereby able to perform the particle filter update step.
- **Step 4** In the importance re-sampling step, each particle will inherit the LSTM state from the previous particle that it is sampled from like in figure 4.7.
- **Go back to Step 2**

One thing to note here is instead of preforming particle filter for each human joint, we will combine

the x positions and y positions of all the joint together, thereby the state vector in this algorithm will become

$$S_t = \begin{bmatrix} Location_x\, for\, joint\, 0\, at\, time\, t \\ Location_y\, for\, joint\, 0\, at\, time\, t \\ Location_x\, for\, joint\, 1\, at\, time\, t \\ Location_y\, for\, joint\, 1\, at\, time\, t \\ \vdots \\ Location_x\, for\, joint\, k\, at\, time\, t \\ Location_y\, for\, joint\, k\, at\, time\, t \end{bmatrix}$$

where k is the number of the joints that we are targeting on. With this state vector, we also need to change the definition of the state to observation probability $P(O_t|S_t)$. Let $HM_t^i(x,y)$ be the heatmap function of the i^{th} joint at time t. We will assume that the heatmap for each joints are independent. Therefore, $P(O_t|S_t) = \prod_{i=1}^{k} HM_t^i(S_t[2*i-1], S_t[2*i])$. However, because the Heatmap probability tends to be really small, this product may cause numerical issues, and we will talk about how to handle it in Appendix A.

4.3.2 Noise Estimation

As with Chen's work [6], we applied maximize likelihood estimation [11] to fit the noise parameter Q. Estimation is setup as follows: suppose we are given a series of ground truth state S_0, S_1, ..., S_T and a series corresponding results drawn from one step predictions from LSTM: $\hat{S}_1$, $\hat{S}_2$, ..., $\hat{S}_T$. The variance parameter of $P(S_{t+1}|S_{1:t})$ that can maximize the joint likelihood $P(S_{0:T}, \hat{S}_{1:T})$ will be

$$Q_{MLE} = \arg\max_{Q} \Big[-T\log|2\pi Q| - \sum_{t=1}^{T} (S_t - \hat{S}_t)^\top Q^{-1} (S_t - \hat{S}_t) \Big] \tag{4.7}$$

$$= \frac{1}{T} \sum_{t=1}^{T} (S_t - \hat{S}_t)(S_t - \hat{S}_t)^\top \tag{4.8}$$

This noisy estimation may not be correct all the time, because of the error of the LSTM on step prediction may not be Gaussian distributed all the time (the same is likely true of linear kinematics estimator as well).

Table 4.1: **Pose estimation accuracy rates at 10 pixels (LSTM + Particle Filter).** Continuous pose estimation accuracy [%] at a 10-pixel tolerance for head, hands, elbows, and shoulders with average the accuracy of left and right side of the body.

Method	Head	Hands	Elbows	Shoulders
Heatmap + Kalman Filter	100.0	86.0	85.5	**96.4**
Heatmap + original Particle Filter	99.1	88.8	83.3	96.0
Heatmap + LSTM Particle Filter	**100.0**	**94.2**	**87.9**	95.5

Therefore, estimating its Gaussian variance may be a over-simplified approach. In the Chapter 5, we will talk about what future work can further improve this part.

4.4 Experiment of Particle Filter with LSTM

4.4.1 Experiment Setup

The setup of this experiment is similar to the setup of the experiment in Chapter 3. Except this time we will compare result of Heatmap + Kalman Filter, Heatmap + original particle filter, and Heatmap + particle filter with LSTM. We will use HumanEva [7] data-set to train and testing our result. Also, the LSTM model is the same one that we tested in chapter 4.2.

Noise Parameter Training

We used another 500 frames of the consecutive frames to train the variance of the distribution of state transitional probability in equation 4.8

4.4.2 Experiment Result

From figure 4.8 and table 4.1, we see that by adding LSTM into the particle filter algorithm, the resulting post processing methodology tends to have a better result for both hands and elbows comparing to both Kalman filter and original particle filter, although we do see an increase in error in the right shoulder estimate for this subject. This result suggests that by taking the long term nonlinear dependence of human

pose motion into the particle filter, we can better estimate those human joints that have closer relationship with human movements.

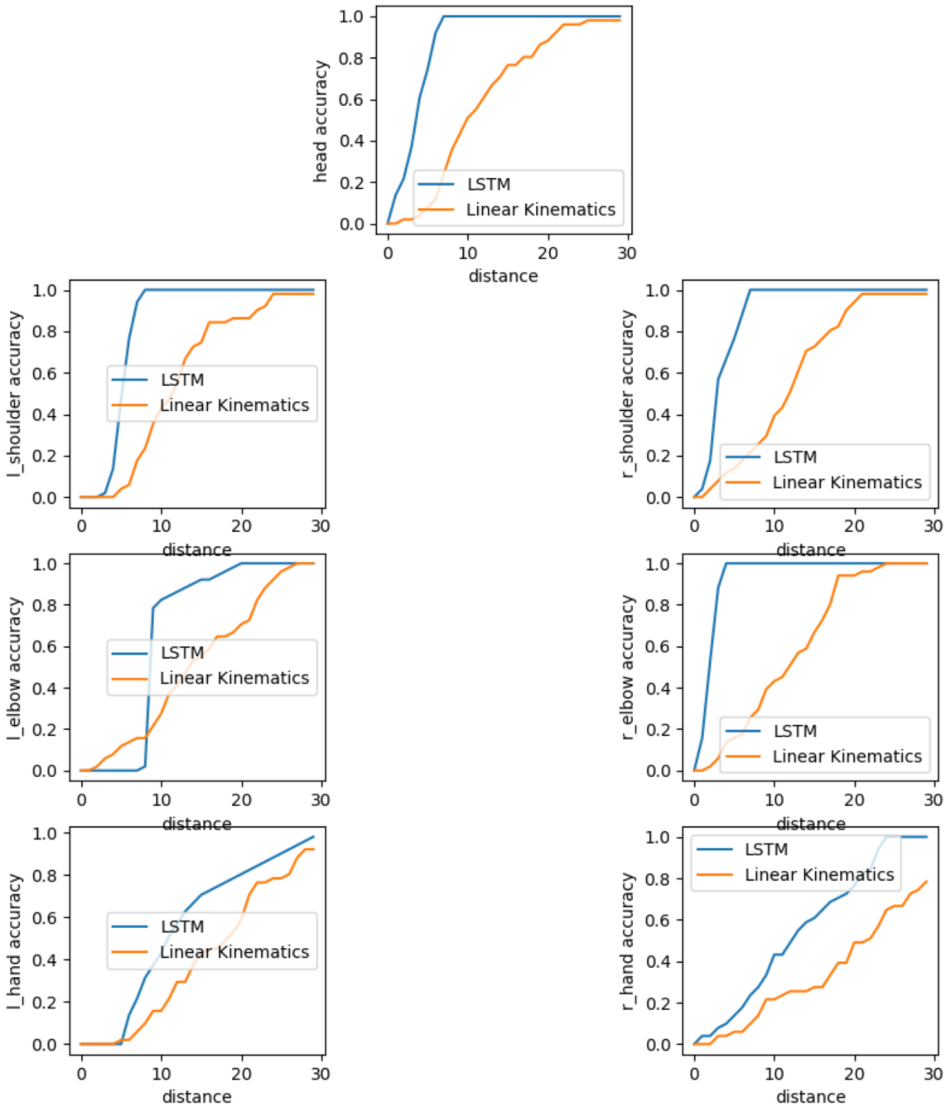

Figure 4.3: **Result of LSTM Experiment.** This is the resulting accuracy in the predicted position 30 frames after the first frame for both LSTM and linear kinematics. The accuracy is with respect to the distance from the ground truth that we still think it is a hit.

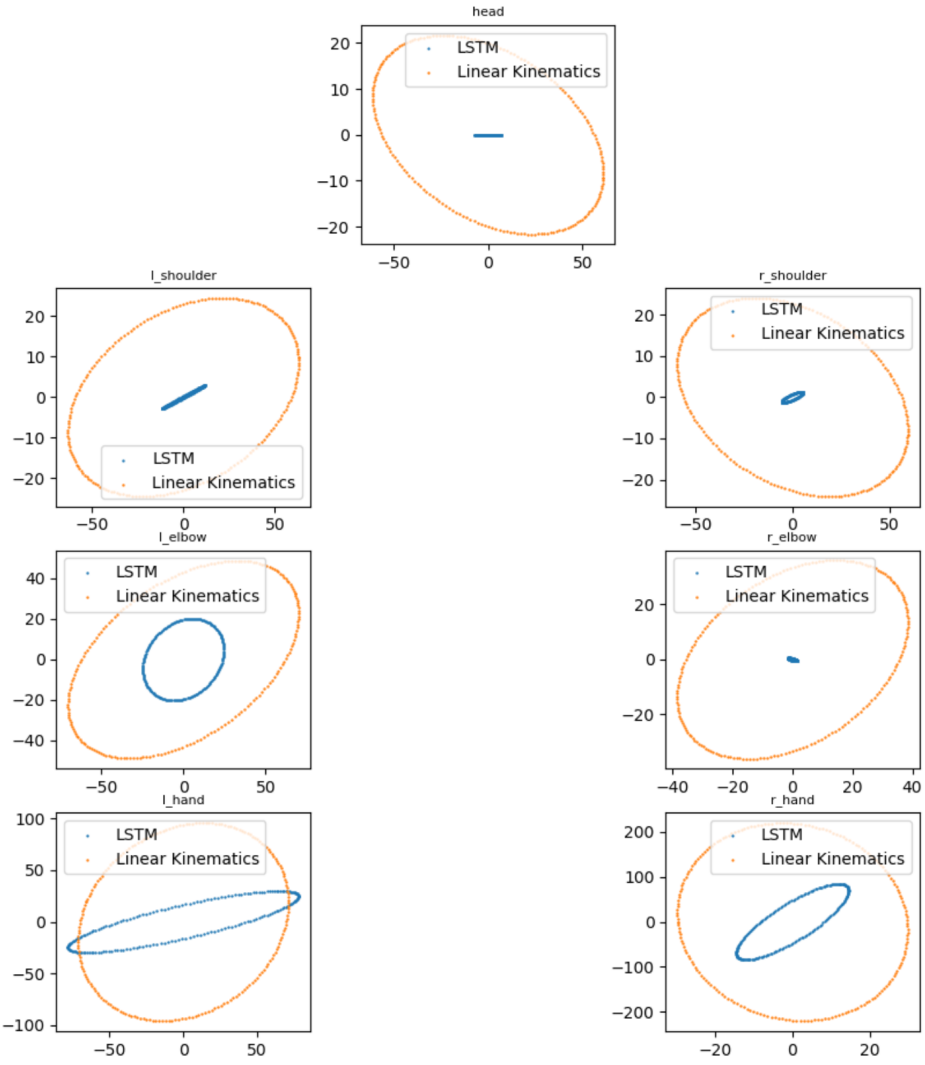

Figure 4.4: **Comparison of LSTM and Linear Kinematics Covariance Matrix.** This is a visualization of the covariance matrix that is calculated by maximized likelihood estimation using the result from LSTM and linear kinematics comparing to the ground truth. See equation 4.8. The ellipses in the plot encircle the areas that 80% of the predictions will have a error in this range.

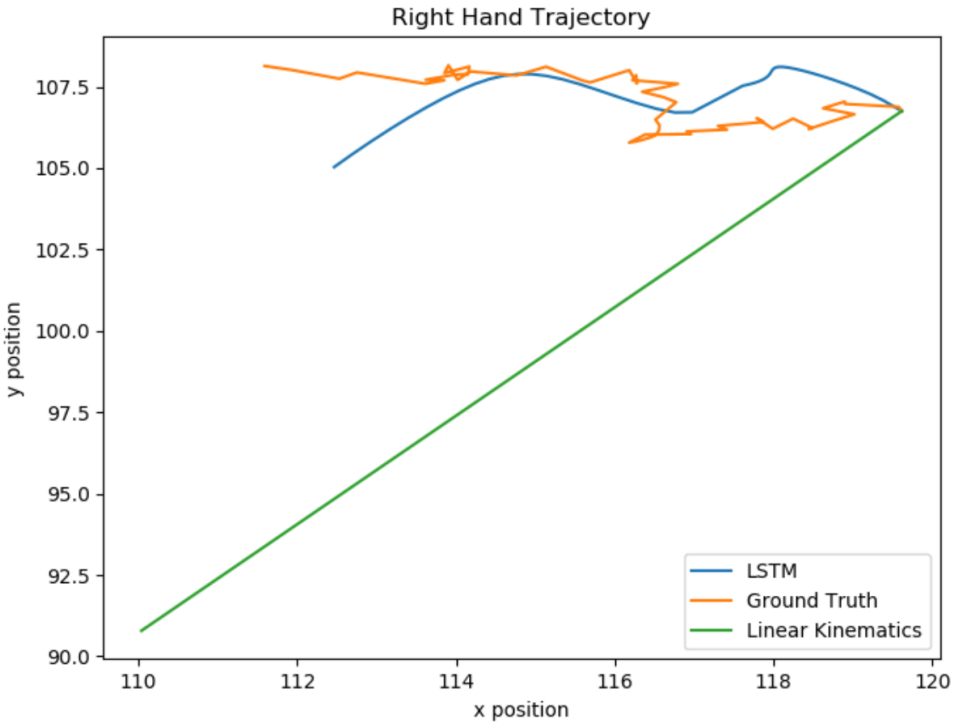

Figure 4.5: **Example of Right Hand Trajectory.** This is an example of the trajectory in a trail from which we can see when we put LSTM and linear kinematics into a "blind race", LSTM tends to have a much better sense of the moving tendency comparing with linear kinematics.

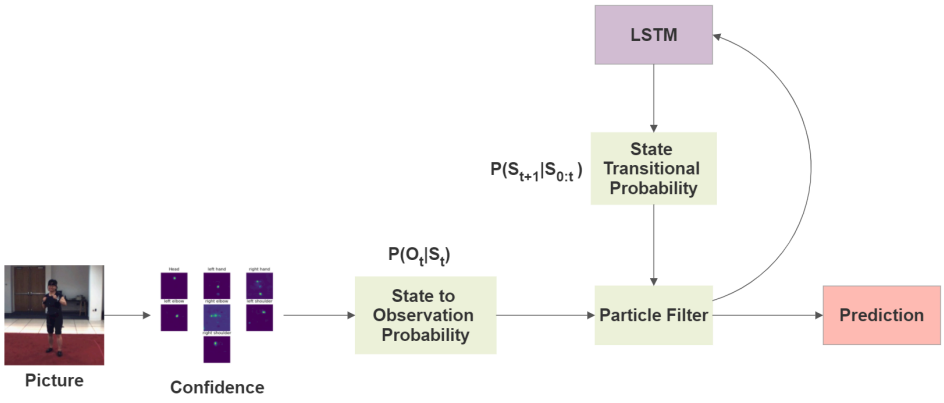

Figure 4.6: **Algorithm Pipeline.** Our Proposed algorithm takes in the training data for LSTM (or pre-trained model) and the pre-obtained human pose confidence as input and output the refined prediction. In the part of particle filter, LSTM will provide the state transitional probability, and pre-obtained confidence will provide the state to observation probability.

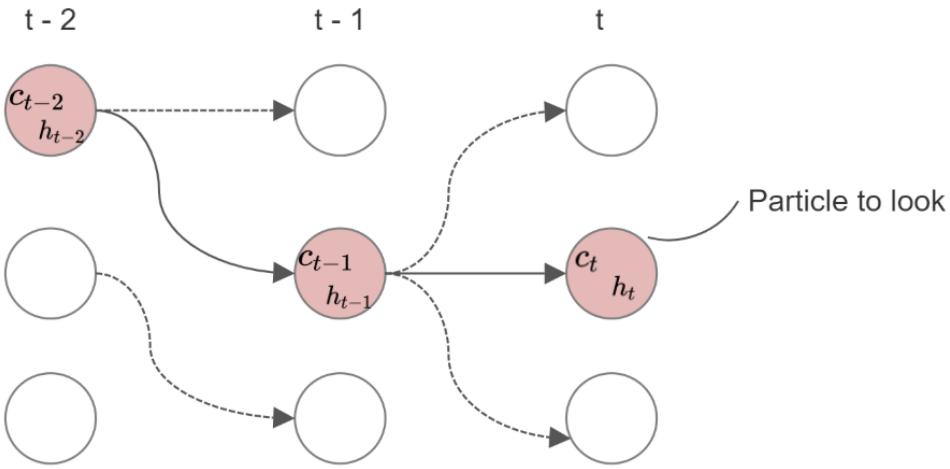

Figure 4.7: **LSTM State Inheritance.** As in the figure, the particle will inherit the LSTM state from the particle that it was re-sampled from.

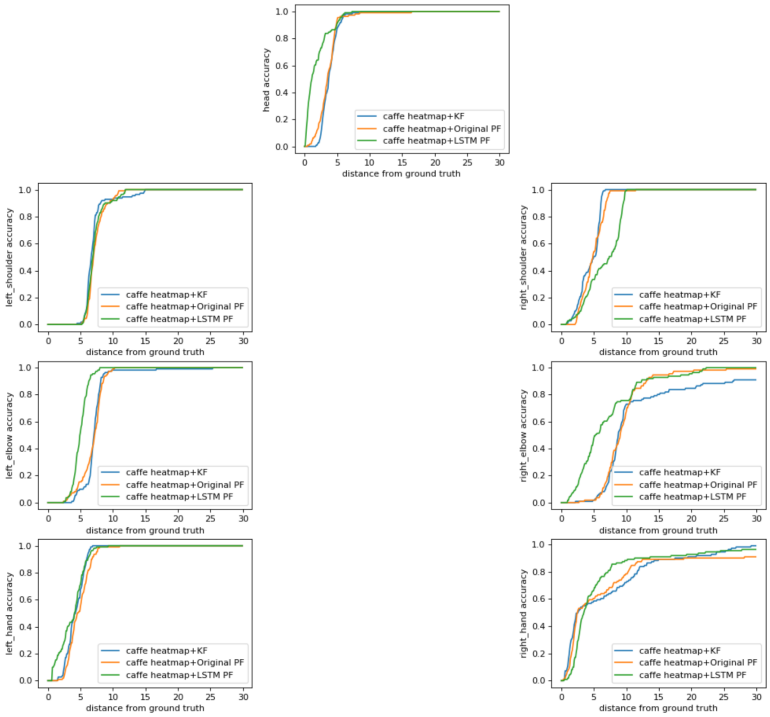

Figure 4.8: **Result of Particle Filter with LSTM Experiment.** This is the result of Caffe heatmap + Kalman filter versus Caffe Heatmap + original particle filter versus Caffe Heatmap + LSTM particle filter. The accuracy is with respect to the distance from the ground truth that we still think it is a hit.

Chapter 5

Conclusion and Future Work

Our work proposes a post processing algorithm for continuous human pose estimation after front end computer vision calculates a confidence map. This confidence can have multiple "hotspots" or regions of high probability, and as long as we can draw a probability distribution directly from the heatmap, we can fit it into our estimation framework. Specifically, we have shown that compared to Kalman filter, the combination of the LSTM and particle filtering can model long term nonlinear relation kinematic state and can maintain multiple hypotheses across time. In the experiments, our algorithm has a better result in tracking both human hands and elbows, which are not only the human joints that are hard to accurately find in static images, but are also the ones that move the most during a human movement. Although our model may improve continuous human pose estimation, there are a few ways in which we might improve estimation further.

5.1 General Improvements

5.1.1 LSTM Training Data Generation

In the training part of the LSTM, due to the limit number of valid continuous training data, we tried some data augmentation techniques (See Appendix B), however, there are other methods that may be

able to generate more continuous human pose data. [12] and [13] provide us with some novel ideas of the simulating the human movement using physical constraints and dynamic equations. In the future, if we implement these approaches, we will can generate an infinite number of training examples for the LSTM, and we may able to achieve better results.

Another aspect that will benefited by having more training data is we did not apply our complete framework in the clinical environment data even though our initial goal was to improve continuous human pose estimation in the clinical environment, and that is because we do not have enough training data to train the LSTM network. By having more generated data we will be able to finish this work.

5.1.2 Using Camera Model

Another shortcoming is we did not consider the camera parameter at all in this work. In the future, if we can consider both the intrinsic and extrinsic parameters of the camera we are using. In doing so, we may be able to generalize this algorithm by setting the camera parameter as the input to the algorithm, so that we can use the same trained algorithm on data collected with different cameras. Additionally, we can also extend this work into 3D by having multiple cameras.

5.2 Model Improvements

5.2.1 Non-Uniformed State Prior

In chapter 3, we mention that we are assuming that the state prior probability $P(S_t)$ is a uniform distribution, so that when weighting each particle, this probability is easily canceled out. However, we do know that there are some locations in the image that certain human joints are more likely to appear than other locations. We can address this by 1. statistically learn this prior probability by the data, 2. combine this with the LSTM and let the LSTM provide us with the prior probability of state at time t.

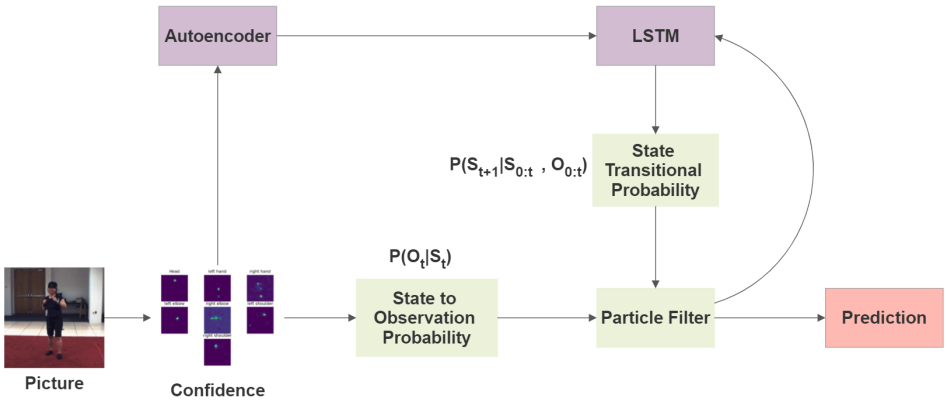

Figure 5.1: **Add Heatmap as LSTM Input.** This is an example the new pipeline of our algorithm if we choose to use pre-trained autoencoder as feature extractor and input the information of the heatmap into LSTM.

5.2.2 Heatmap as Observation Vector

In this work, even though we have fully utilized the heatmap information by setting the heatmap itself as the state to observation probability $P(O_t|S_t)$, the observation vectors O_t are still human joints locations. An interesting approach to take is if we were to treat the heatmap itself as the observation vector, and we learn the state to observation probability from the true distribution of the heatmap, which we can obtain from human annotations of the confidence heatmap.

5.2.3 More Complex State Transitional Probability

We have also successfully extended the state transitional probability from $P(S_{t+1}|S_t)$ to $P(S_{t+1}|S_{0:t})$ by using the LSTM. It is curious to see that if we further extend this probability and make it also depends on the observations. If we can find a way to express $P(S_{t+1}|S_{0:t}, O_{0:t})$, the update step of the particle filter will have all the information before time t including not only the states, but also all the observations to help it make better predictions. An idea, like shown in figure 5.1, is to add the heatmap as the input to the LSTM after we use some feature extract strategies like pre-trained CNN based autoencoder [14] to reduce the dimensionality of the image.

Another idea is to change this filtering problem into a smoothing problem. This means we will then use the information in the future to make our state transitional probability: $P(S_{t+1}|S_{0:t}, S_{t+2:t+1+d})$, where d is the number of frames that we are looking forward. This may be accomplished by the bidirectional RNN [15]. Also note that even we would like an online algorithm in the end, we can still using this model by setting d very small to only add a small lagging.

5.2.4 Better Noise Parameter Estimation

We can also work on the way we estimate the variance of state transitional probability $P(S_{t+1}|S_{1:t})$. Intuitively speaking, this variance parameter represents the error of our state transition function. Utilizing maximized likelihood estimation and assuming a Gaussian distribution may be an oversimplification of the problem. If we look at figure 5.2, we can see that the histogram here represents the true error of the state transitional function. It suggests that the error of state transitional function is something like an Gaussian mixture instead of single Gaussian distribution. Therefore, if we can model state transitional probability as a Gaussian mixture, we may get a better estimate of the position from the particle filter.

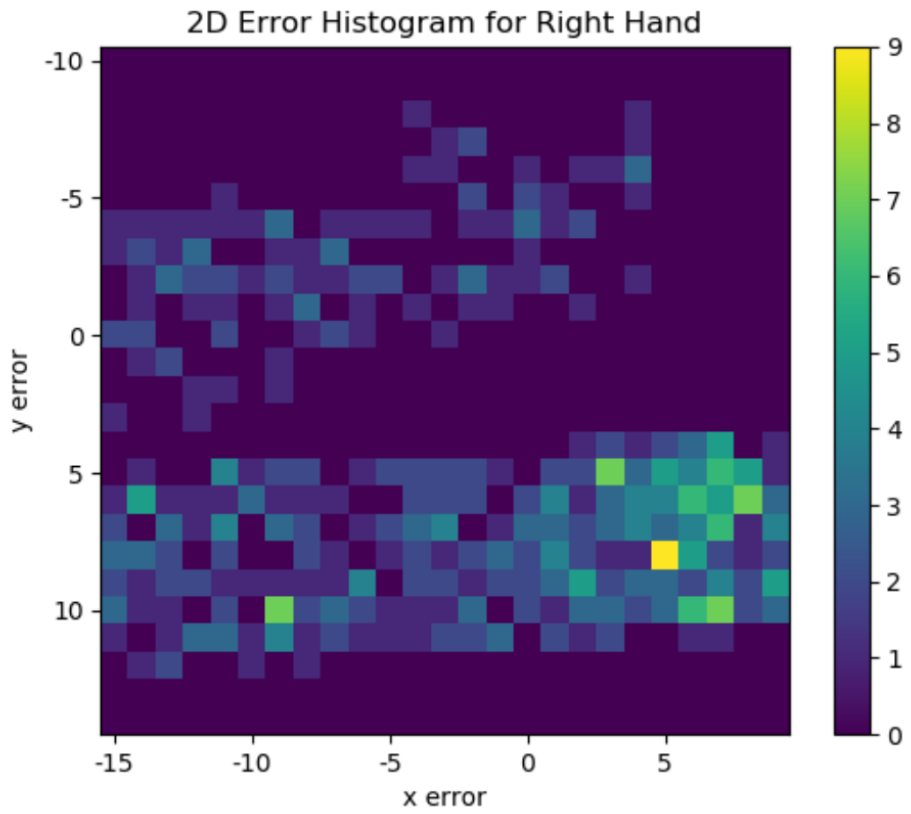

Figure 5.2: **2D Histogram of the Right Hand Error.** This is an example of the true distribution of the error between the one step prediction of LSTM and the ground truth.

Appendix A

Handling Numerical Issues When Multiplying the Probabilities in the Heatmaps

Calculating $P(O_t|S_t) = \prod_{i=1}^{k} HM_t^i(S_t[2*i-1], S_t[2*i])$ directly may cause numerical issues because $HM_t^i(x,y)$ is a normalized heatmap, and a point value inside this heatmap can be very small. In this case, if we multiply k of this small number together, we may not be able to get a valid number in most of the number type in different programming language. Therefore, we can use a trick to do not actually compute this number. Instead, since $P(O_t|S_t)$ is used to calculate the weight of each particle, and it will be normalized in the end anyway, it is good enough that we keep the weights for each particle proportional to each other as they should be in real value. Therefore, for each $P(O_t|S_t)$, all we need is actually $A*P(O_t|S_t)$ where A is a large enough number to avoid the numerical issues.

We define L as $log(A*P(O_t|S_t)) = log(A) + \sum_{i=1}^{k} log(HM_t^i(S_t[2*i-1], S_t[2*i]))$. When calculate L, we do not need to multiply any small numbers. When L is calculated, we can simply do e^L, and that will give us the proportional weights for each particle.

Appendix B

Human Pose Data Augmentation

Because of the lack of continuous training data, some data augmentation techniques have to be used in order for us to have enough training data for LSTM to learn the trajectory of human motion. Since training LSTM only need the continuous human pose data, and no images is needed, we can augment the data size by perform some transformation that will still yield valid human motion to the existing data.

We first horizontally flip the existing data and then perform the linear interpolation between every two frames of the data we have, and in the end we flip the interpolated data again. After these operations, we are able to get 4 times the original size of the training data, and those additional parts are still valid human motions.

This thesis, in full is currently being prepared for submission for publication of the material. Gong, Chenghao; Gabriel, Paolo; Gilja, Vikash. The dissertation/thesis author was the primary author of this material.

References

[1] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[4] P. Del Moral. Nonlinear filtering: Interacting particle solution. *Markov Processes and Related Fields*, 2(4):555–580, 1996.

[5] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[6] Kenny Chen, Paolo Gabriel, Abdulwahab Alasfour, Chenghao Gong, Werner K. Doyle, Orrin Devinsky, Daniel Friedman, Patricia Dugan, Lucia Melloni, Thomas Thesen, David Gonda, Shifteh Sattar, Sonya Wang, and Vikash Gilja. Patient-specific pose estimation in clinical environments. *IEEE Journal of Translational Engineering in Health and Medicine*, PP:1–1, 10 2018.

[7] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):427, 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):8490, 2017.

[9] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35, 1960.

[10] X. Rong Li and Vesselin P. Jilkov. Survey of maneuvering target tracking. Part I: Dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 2003.

[11] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y. Ng, and Sebastian Thrun. Discriminative training of Kalman filters. In *Proceedings of Robotics: Science and Systems I*, 2005.

[12] H. Hatze. A comprehensive model for human motion simulation and its application to the take-off phase of the long jump. *Journal of Biomechanics*, 14(3):135142, 1981.

[13] Yujiang Xiang, Hyun-Joon Chung, Joo H. Kim, Rajankumar Bhatt, Salam Rahmatalla, Jingzhou Yang, Timothy Marler, Jasbir S. Arora, and Karim Abdel-Malek. Predictive dynamics: an optimization-based novel approach for human motion simulation. *Structural and Multidisciplinary Optimization*, 41(3):465479, 2009.

[14] Jonathan Masci, Ueli Meier, Dan Cirean, and Jrgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Lecture Notes in Computer Science Artificial Neural Networks and Machine Learning ICANN 2011*, page 5259, 2011.

[15] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.