HUMAN MOVEMENT RECOGNITION WITH DYNAMIC MOVEMENT PRIMITIVES

A Thesis

by

Alp Burak Pehlivan

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Masters of Science

in the
Computer Science Department

Özyeğin University
September 2015

HUMAN MOVEMENT RECOGNITION WITH DYNAMIC MOVEMENT PRIMITIVES

Approved by:

Assoc. Prof. Erhan Öztop, Advisor
Computer Science Department
Özyeğin University

Assistant Professor Hasan Sözer
Computer Science Department
Özyeğin University

Assoc. Prof. Nafiz Arıca
Department of Computer Engineering
Bahçeşehir University

Date Approved: 22 August 2015

To my wife who always supported me and encouraged me to finish my thesis.

ABSTRACT

Dynamic Movement Primitives (DMPs)-originally a method for movement trajectory generation has been also used for recognition tasks. However there has not been a systematic comparison between other recognition methods and DMPs using human movement data. We have implemented a movement recognition method based on DMPs with Gaussians centered equally spaced in phase variable and scaled one-nearest-neighbor weight comparison. Furthermore, in thesis, we presented a comparison of commonly used Hidden Markov Model (HMM) based recognition with our implementation of DMP based recognition using human generated letter trajectories. As the working principles of these two methods are very different, in addition to the performance, the numbers of adaptable parameters that are used in each method and, process time were compared. The results indicate that DMP gives better results than HMM in the tests with noiseless data, noisy data and derogated data with given human movement dataset.

ÖZETÇE

Dinamik Hareket Birimleri (DHB), ilk olarak hareket güzergahlarının üretilmesinde kullanılan bir yöntem olduğu halde hareket tanıma görevlerinde de kullanılmıştır. Fakat DHB'lerle yapılan tanıma ile diğer tanıma yöntemleri arasında sistematik bir karşılaştırma yapılmamıştır. Biz de faz değişkeninde eşit merkezlenmis Gaussian fonksiyonları kullandığımız DHB'lerde, boyutları değiştirilmiş ağırlık karşılaştırması ile hareket tanıma yöntemi gercekleştirdik. Ayrıca, bu tezde yaygın olarak kullanılan Saklı Markov Modeli (SMM) yöntemi ve DHB ile yapılan insan tarafından üretilmiş hareket güzergahları üzerinde tanıma işlemleri karşılaştırılmıştır. Bu iki yöntemin çalışma prensipleri çok farklı olduğu için, performansa ek olarak adapte edilebilir parametrelerin miktarı ve tanıma işleminin aldığı zaman karsılaştırılmıştır. Sonuçlar, DHB'nin insan hareketleri verisi üzerine gürültüsüz, gürültü eklenerek ve veriler azaltılarak yapılan testlerde SMM'den daha iyi sonuçlar verdiğini göstermektedir.

ACKNOWLEDGEMENTS

I would like to thank my advisor Erhan Öztop for the useful comments, directives and remarks through the learning process of this master thesis. Furthermore, his motivating talks and mentoring helped me a lot. I would like to thank Advanced Telecommunication Research (ATR) for giving me work opportunity to study some parts of this thesis. I would like to thank all Özyeğin University Robotics Laboratory members, friends whom I shared similar academic goals and studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In this thesis, we aimed to recognize human movement in order to achieve successful applications in Robotics and Human-Machine Interaction area. Firstly, we have implemented a practical approach -Autotag Software- for human movement detection in order to be used incorporation with human movement recognition and analysis algorithms later. Later, we have studied the properties of Dynamic Movement Primitives (DMP) framework which can be useful in human movement recognition. Finally, we have analyzed the performance of DMP in human movement recognition by several recognition tests including the ones with noisy data, lessened data and comparing the results with HMM recognition performance in a comparable learned parameter complexity.

1.1 Motivation

Human Motion Recognition and Analysis is important for many robotics applications. In order to implement a successful human-robot interaction understanding human movements is essential. If a robot was to react to a human action interactively, naturally a robot is expected to understand the action of human. Therefore the robot can take counter action or cooperative action.

DMP, a general way to learn and encode movements was studied in this thesis for the purpose of using as a method of human movement understanding in robotic studies. We wanted to see if DMP could be used for movement recognition and analysis. Human movements are dependent on newton dynamics and these movement have an acceleration limit. Since DMP takes the dynamic system into account, it may capture biological movements well too. We applied tests with a human movement dataset in order to recognize specific movements out of several. Furthermore, for the purpose of investigating the success of this method we compared the results with Hidden Markov Model (HMM) recognition on the same set of movement data. Since HMM is widely used in motion recognition, this study gave us information about how the DMP recognition results compare to widely used successful motion recognition method, namely HMM.

1.2 Related Literature

One of the works done about Action Recognition is "Action Recognition and Understanding Through Motor Primitives" by Vicente et al.[21]. This work is mainly about action recognition using HMM and understanding the motor primitives. In my thesis, I have used HMM classifier with movement coordinates as data sequence for recognition. However in Vicente et al.'s work, time series data used in HMM classifier is not the movement trajectory itself but a sequence of the motor primitives [21]. Each manipulation task of a complex movement was designated as one motor primitive which correspond to individual states of the HMM such as grasping and rotating objects. This implementation was based on the idea that actions can be represented as sequences of motion primitives [22]. The data of complex action movement was collected by a sensor glove that gives relative positions of hand, index, and thumb from 10 different demonstrator for 5 manipulation actions. A support vector machine (SVM) is used to model and recognize individual primitives, while the sequences of primitives are modeled using a Hidden Markov model (HMM). Vicente et al. obtained recognition rates ranging from 59.1% to 95.8% by different methods that applied using HMM [21]. In this work, the data were not diverse and the results were not as successful as one expect. However the study of several implementations of 'actions as primitives' and recognition using action primitives were useful for my thesis work. As future work, the authors mentioned that they will model large variety of actions in visual data context [21]. Another work that uses motor primitives for human activity recognition is by Zhang et al. In their study, Zhang et al. used inertial sensor was used to collect data during human activity and 'Bag Of Features' was used to recognize statistically obtained motion primitives [38].

Dynamic Movement Primitives (DMPs) are seen as units of action that are formalized as stable nonlinear attractor systems [28]. In his paper, Schaal has shown that DMP has applications in Humanoid Robotics and parallels in biological research. He has applied rhythmic DMP for making a humanoid robot play drums successfully. In another application, Schaal has used imitation learning to teach a robot how to swing tennis racket by demonstration. An interesting approach to the biological connections of DMPs were investigated in Schaal's paper too. An action like stable ball bouncing with a racket needs some balancing effort even though humans do it without realizing. How do humans successfully do this? When Schaal has tried stable ball bouncing, DMPs used for this movement introduced a different way of accomplishing this task then other alternatives such as accelerating at the time of impact. This way is characterized by a negative acceleration at the time of impact which is quite non-intuitive as told in the author. Such examining by Schaal revealed that DMPs capture human behavior very well [28]. Therefore, in this thesis, using DMPs for human movement recognition makes more sense in this context.

Dynamic Movement Primitives (DMPs) encode a desired movement trajectory in terms of the attractor dynamics of nonlinear differential equations with the help of adjustable weight parameters [9]. The main principle that allows DMPs to be used for recognition is the fact that two movements can be compared by using their DMP representations, i.e. weights. For example, the movements that need to be recognized can be stored in the form of weight vector templates for later recognition. A demonstrated movement then can be fit with a DMP, giving a weight vector which can be compared against the stored templates for finding the best matching movement. Using such a mechanism Ijspeert et al. has obtained 87% recognition rate out of recognition of 130 handwritten letters [2, 10, 3]. The exact recognition method used by Ijspeert was one-nearest-neighbor classifier in weight space. Furthermore, a baseline comparison like we did in this thesis has been done by Ijspeert too. Ijspeert compared Dynamic Time Warping (DTW) recognition on letter trajectories. They have got 79% recognition rate which lower results than 87% of DMP's recognition rate [10]. In addition to recognition tasks, placing cup by a robot arm in different places from one demonstration by goal change in DMP was applied on robot experiments [10].

In one of the works of Ijspeert et al., the use of Nonlinear Dynamical Systems for imitation with humanoid robots were studied. The control policies(CPs) were designed as DMPs. In this work, they have tested the CPs for a learning by imitation task in a humanoid robot with 30 DOFs [25].

DMPs were used in several applications including movement segmentation [4]. Meier et al. used DMP to solve the problem of segmenting complex movements into a sequence of primitives. Meier et al. has obtained a modified DMP formulation by reformulating it as a linear dynamical system with control inputs. Using this new formulation they were able to perform movement recognition and prediction of the goal position of a partially observed motion, and perform movement segmentation [4]. Another segmentation work was done by Schaal [39]. He focused on two such invariants in one part of the study, the 2/3 power law and piecewise planar movement segmentation, and how a parsimonious explanation of those effects can be obtained [31]. The systematic relationship between angular velocity and curvature of the end effector traces of human movement identified by Viviani and Terzuolo [31] has been applied to handwriting and 2D drawing movements in Schaal's work [39].

In his master thesis, Glatz presented a learning from demonstration method which is able to learn complex object manipulation tasks using Dynamic Movement Primitives. Additionally, a sensor data based segmentation method was applied to divide the complex tasks into simpler goal-directed subtasks. He also applied the experiments of learning from demonstration in robot simulation [26].

In the work of Forte et al., a humanoid robot was taught to do a reaching movement [23]. Then this movement were generalized in order to reach every possible point in the robot's operating range using Gaussian process regression. The trajectories for robot to learn were created by moving the robot arm by hand. Then each trajectory was encoded as DMPs. The generalization enabled robot to move its arm to a point which it haven't learned previously by DMP reproduction of the movement. Thus, DMP parametrization of the movement can be useful in such generalization tasks too [23]. Furthermore, DMPs can be used for dimensionality reduction as done by Colome et al. While using reinforcement learning to train DMPs, they proposed a strategy to reduce the number of Gaussians so that representing the learned trajectory will require low cost and low dimensionality [19].

In a work of Stulp et al. DMPs were trained using human motion data for reaching tasks in order to derive stereotypical reaching trajectories for variations of the task with obstacles present [32]. As claimed by the authors, DMP allowed compact representations, predictable motion, and legible motion as in human-like behavior in this case [32].

In a work of Akgun et al. titled as 'Action recognition through an action generation mechanism', the authors were capable of online action recognition within approximately the first one-third of the observed action with a success rate of over 90% using DMP. They have generated the trained trajectories in order to calculate error of Cartesian coordinates of observed trajectories for online recognition [33].

Miguel et al. has studied DMPs for Human Robot interaction. In their work, they have trained DMPs with Human movement data about object transfer between human and robot. They have modified DMP formula by decoupling weighting function and adding adaptation for goal change in order to obtain a better control of the transition in between the shape-attractor and the goal-attractor [24].

Reinforcement learning has been used in the obtaining the parameters in DMP training process by Kober et al. [34, 35]. In one of Kober et al.'s work they have studied the problem of meta-parameter learning for motor primitives [34]. It is as stated, an essential step towards applying motor primitives for learning complex motor skills in robotics more flexibly. They have showed an appropriate reinforcement learning algorithm for mapping situations to meta-parameters and applications in both simulations and several physical robots [34]. In another similar work of Kober et al., they have shown evaluations of adjustment of robot movements generated by DMP using reinforcement learning in dart throwing and table tennis strike tasks with robots [35].

One of the works by Nemec et al. presents some novel applications using DMP. In their paper Nemec et al. have addressed to the problem of joining movement primitives. In order to achieve affective sequencing with continuous desired acceleration signals, they have implemented a third-order DMP formulation [1].

Joining two or more DMPs is also possible as done in paper by Kulvicius et al. [37]. In order to achieve this, two simple DMP were joined by setting centers of later DMP to the continuing points after the end of first DMP. In this paper DMPs were trained by handwriting trajectories and the result was implemented on a robot [37].

One of the studies related to DMP parameter change is the paper titled as Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives by Ude et al. The task of generalization is to synthesize action toward any given target within the training space. The parameters specifying the goal of the task in this study are the target positions. Ude et al. have shown several applications of goal change in DMP reproduction of trajectories [30]. We have also shown the effect of goal change in DMP reproduction in this thesis.

HMM is one of most famous tools as a recognition method for time-series data. There are used in many applications including speech recognition, pattern recognition, gesture spotting, and molecular biology[18, 6, 36]. A detailed tutorial on HMM is done by Rabiner in 1989 explaining how HMM works and how to solve problems that will make HMM useful in real world problems [11]. There are many successful application studies that use HMMs for different types of recognition. One such an example is Yang et al.s [12], where 99.78% accuracy could be achieved for an isolated recognition task involving nine gestures [12]. Another example of HMMs use for movement recognition is the work of Starner and Pentland [7]. Starner and Pentland used HMM to model sign language gestures and they achieved 99.2% word accuracy . A work on sign language recognition using Kinect (Microsoft Corp.) was done by Godoy et al. [5]. Authors were able to get correct recognition rates ranging from 76.2% to 91.2% using 181 video collected from 23 different subjects [5]. There are also examples of HMM being used in handwritten text recognition, such as the work undertaken by Hu et al. [8]. Hu et al. were able to achieve writer independent recognition rate of 94.5% on 3,823 unconstrained handwritten word samples from 18 writers covering a 32 word vocabulary [8].

CHAPTER II

MOVEMENT DETECTION : A PRACTICAL APPROACH

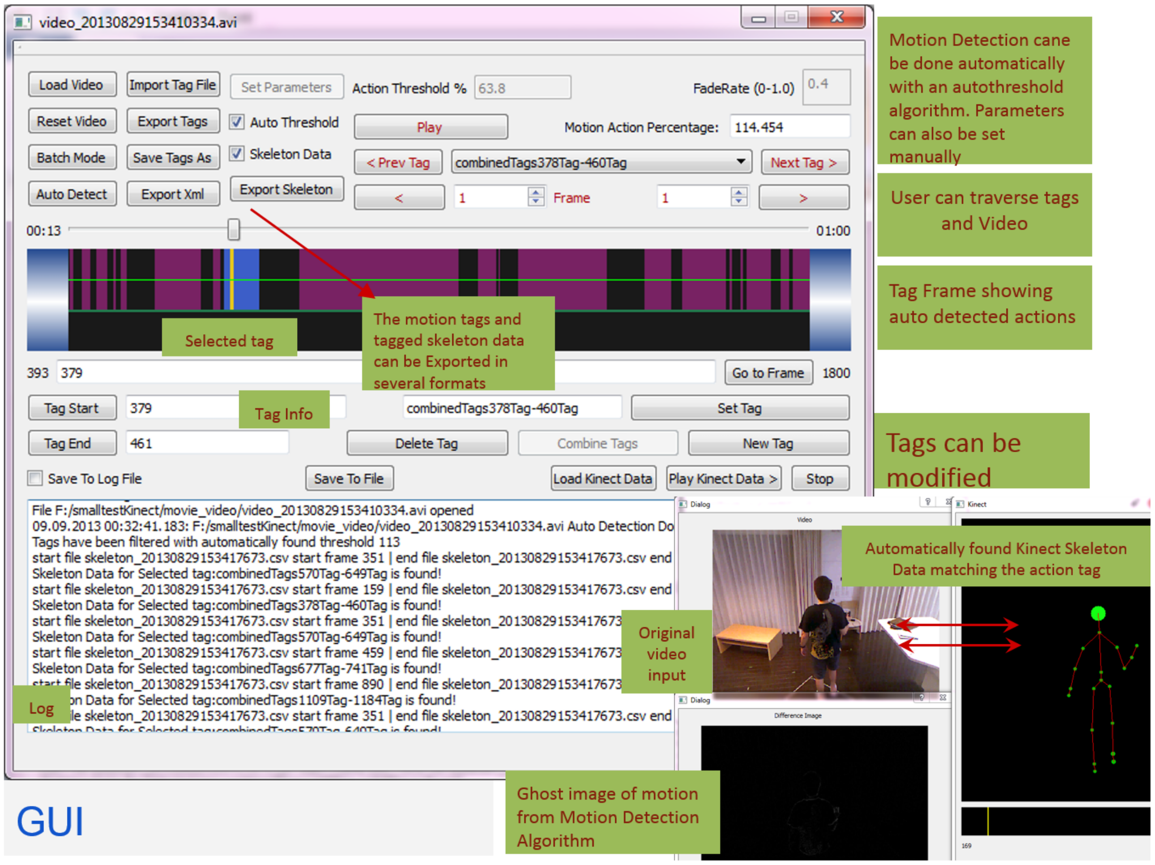
Recognizing human movements and understanding the actions in software environment consists of many subparts. One of the sub applications of this subject area is to spot a motion in videos and tagging of this motion for classification. This application has many use cases. One of them is to monitor a house for possible human movements and recognize the action that human does. Autotag application was implemented in order to help realization of such task in Advanced Telecommunications Research Company (ATR) in Japan in Summer 2013 as a part of Smart House Project of DBI laboratories and part of this Master thesis research.

The Smart House Project of DBI Laboratories ATR was about helping elderly or disabled people with smart electronic assistance. For this purpose several hardware and software solutions were proposed. One of the solutions included monitoring house with 13 Kinect cameras and recording the skeleton and RGB data streams. In addition to the Kinect camera data, during the experiments, subject person would wear a functional near-infrared spectroscopy (fNIRS) device. Synchronously with the Kinect data, fNIRS data would be recorded. These videos and skeleton data were to be analyzed further after experiments done in Smart House. Most parts in video and skeleton data were motionless video if there were no humans in that area of the house. Furthermore the skeleton data didn't start immediately after a person comes into the view of a Kinect camera since Kinect usually detects the skeleton after the person faces the camera. The sections of the video where a valuable action happens needed to be tagged by a human manually for the activities they represent so they can be used for several learning tasks and analysis of fNIRS data. Having so much video and skeleton data that are recorded during experiments posed a problem. How to find the action parts of video and tag these parts easily without spending too much time and effort? Autotag software was implemented in order to answer this question.

2.1 Autotag - A Software for Movement Detection

Autotag software is able to process a video or a batch of videos from a location and sub folders of that location. After the video processing and filtering, Autotag is able produce automatically detected parts of the videos where an action happens together with their corresponding skeleton data. These parts are registered to a file with the time stamps of start and stop of actions in video. Autotag also lets the user of program to tag each action manually through a user interface. In this user interface user can edit the action size, start and stop times or delete/add tags while viewing the video for confirmation. The tags in a video can be exported in a special form of XML. This XML file consists of the action tags given by the user and the action start and end times. The XML files were used for getting the corresponding fNIRS data that was continuously recorded during the experiments. Since the videos are collected from Kinect cameras together with skeleton data, if corresponding skeleton data exists, it also can be exported. The user can click on an action tag and play the corresponding skeleton data in another video window. After making sure that this data is needed in machine learning algorithms used in the project, the user can export this data in a format which is suitable for Matlab (MathWorks Inc.).

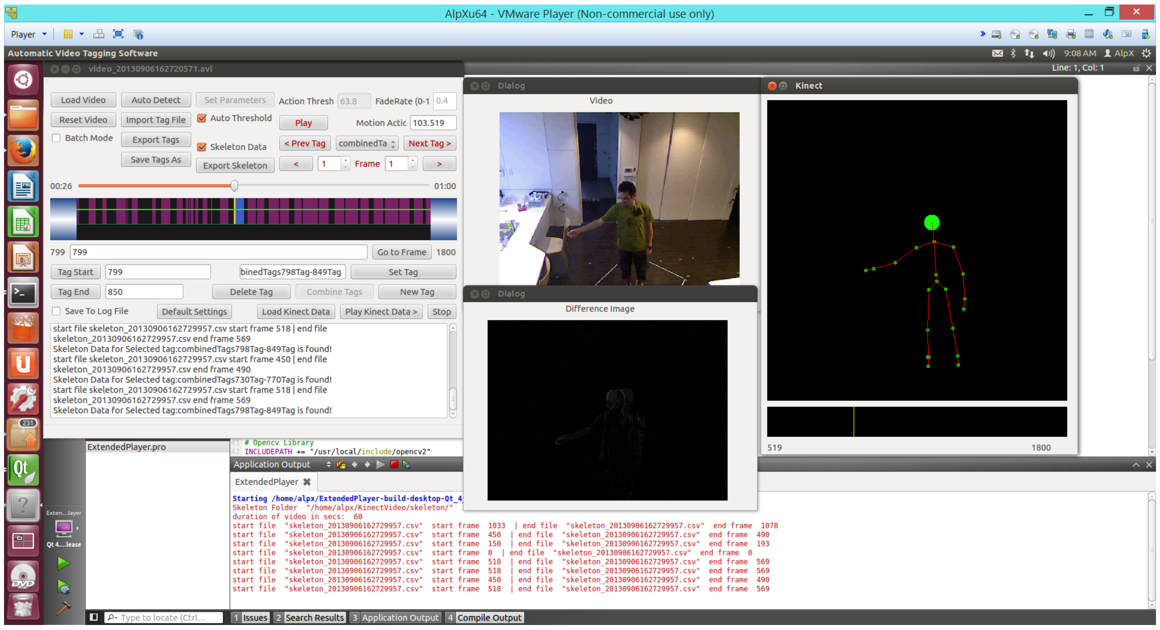
- The software can detect the time when an action happens and finishes in the video by motion analysis.
- These auto detected actions are shown in GUI for further editing, such as combining-deleting-modifying the duration, and tag description by human input.


video_20130829153410334.avi
Load Video
Import Tag File
Set Parameters
Reset Video
Export Tags
Batch Mode
Save Tags As
Auto Detect
Export Xml
Export Skeleton
Play
< Prev Tag
Next Tag >
<
Frame
>
00:13
01:00
Go to Frame
Tag Start
Tag End
Set Tag
Delete Tag
Combine Tags
New Tag
Save To File
Load Kinect Data
Play Kinect Data >
Stop
Dialog
Video
Difference Image
Kinect
Motion Detection cane be done automatically with an autothreshold algorithm. Parameters can also be set manually
User can traverse tags and Video
Tag Frame showing auto detected actions
The motion tags and tagged skeleton data can be Exported in several formats
Selected tag
Tag Info
Tags can be modified
Automatically found Kinect Skeleton Data matching the action tag
Original video input
Log
Ghost image of motion from Motion Detection Algorithm
GUI

Action Threshold % 63.8
FadeRate (0-1.0) 0.4
[✓] Auto Threshold
[✓] Skeleton Data
Motion Action Percentage: 114.454
combinedTags378Tag-460Tag
1 Frame 1
393 379 1800
Tag Start 379 combinedTags378Tag-460Tag
Tag End 461
[ ] Save To Log File

```
File F:/smalltestKinect/movie_video/video_20130829153410334.avi opened
09.09.2013 00:32:41.183: F:/smalltestKinect/movie_video/video_20130829153410334.avi Auto Detection Do
Tags have been filtered with automatically found threshold 113
start file skeleton_20130829153417673.csv start frame 351 | end file skeleton_20130829153417673.csv end
Skeleton Data for Selected tag:combinedTags570Tag-649Tag is found!
start file skeleton_20130829153417673.csv start frame 159 | end file skeleton_20130829153417673.csv end
Skeleton Data for Selected tag:combinedTags378Tag-460Tag is found!
start file skeleton_20130829153417673.csv start frame 351 | end file skeleton_20130829153417673
Skeleton Data for Selected tag:combinedTags570Tag-649Tag is found!
start file skeleton_20130829153417673.csv start frame 459 | end file skeleton_20130829153417673
Skeleton Data for Selected tag:combinedTags677Tag-741Tag is found!
start file skeleton_20130829153417673.csv start frame 890 | end file skeleton_20130829153417673
on Data for Selected tag:combinedTags1109Tag-1184Tag is found!
le skeleton_20130829153417673.csv start frame 351 | end file skeleton_20130829153417673.csv end
```

Figure 1: Explanation of the Graphical User Interface of Autotag



Action Thresh 63.8 FadeRate (0-1 0.4
[✓] Auto Threshold Motion Actic 103.519
[✓] Skeleton Data combinedTa
< 1 Frame 1 >
00:26 01:00
799 799 1800
Tag Start 799 binedTags798Tag-849Tag
Tag End 850

```
start file skeleton_20130906162729957.csv start frame 518 | end file
skeleton_20130906162729957.csv end frame 569
Skeleton Data for Selected tag:combinedTags798Tag-849Tag is found!
start file skeleton_20130906162729957.csv start frame 450 | end file
skeleton_20130906162729957.csv end frame 490
Skeleton Data for Selected tag:combinedTags730Tag-770Tag is found!
start file skeleton_20130906162729957.csv start frame 518 | end file
skeleton_20130906162729957.csv end frame 569
Skeleton Data for Selected tag:combinedTags798Tag-849Tag is found!
```

```
# Opencv Library
INCLUDEPATH += "/usr/local/include/opencv2"
```

```
Starting /home/alpx/ExtendedPlayer-build-desktop-Qt_4
Skeleton Folder  "/home/alpx/KinectVideo/skeleton/"
duration of video in secs:  60
start file  "skeleton_20130906162729957.csv"  start frame  1033  | end file  "skeleton_20130906162729957.csv"  end frame  1078
start file  "skeleton_20130906162729957.csv"  start frame  450  | end file  "skeleton_20130906162729957.csv"  end frame  490
start file  "skeleton_20130906162729957.csv"  start frame  150  | end file  "skeleton_20130906162729957.csv"  end frame  193
start file  "skeleton_20130906162729957.csv"  start frame  0  | end file  "skeleton_20130906162729957.csv"  end frame  0
start file  "skeleton_20130906162729957.csv"  start frame  518  | end file  "skeleton_20130906162729957.csv"  end frame  569
start file  "skeleton_20130906162729957.csv"  start frame  518  | end file  "skeleton_20130906162729957.csv"  end frame  569
start file  "skeleton_20130906162729957.csv"  start frame  450  | end file  "skeleton_20130906162729957.csv"  end frame  490
start file  "skeleton_20130906162729957.csv"  start frame  518  | end file  "skeleton_20130906162729957.csv"  end frame  569
```

Figure 2: GUI in Ubuntu and Auto Motion Detection

- A tag file, and also an Xml file in a nice format can be exported for later use with external programs, such as MATLAB.

- Changing the parameters for motion detection or auto threshold motion detection is available in program.

- Program can find the corresponding skeleton data for a video-action tag and display the data simultaneously.

- Program can export the skeleton data for each tagged action.

The motion detection in video is done by taking differences of the pixels in each consecutive frame and applying a sliding average filter on the pixel value difference. You can see the result after Auto Detect is done in Figure 2. There are purple block in video slider for representing detected movements in the video.

The auto detected movement are shown in purple block. You can see some consecutive frames and the video bar showing tagged area in Figure 2.1. Since the auto detected movement might not be the ones needed for further processing, we may

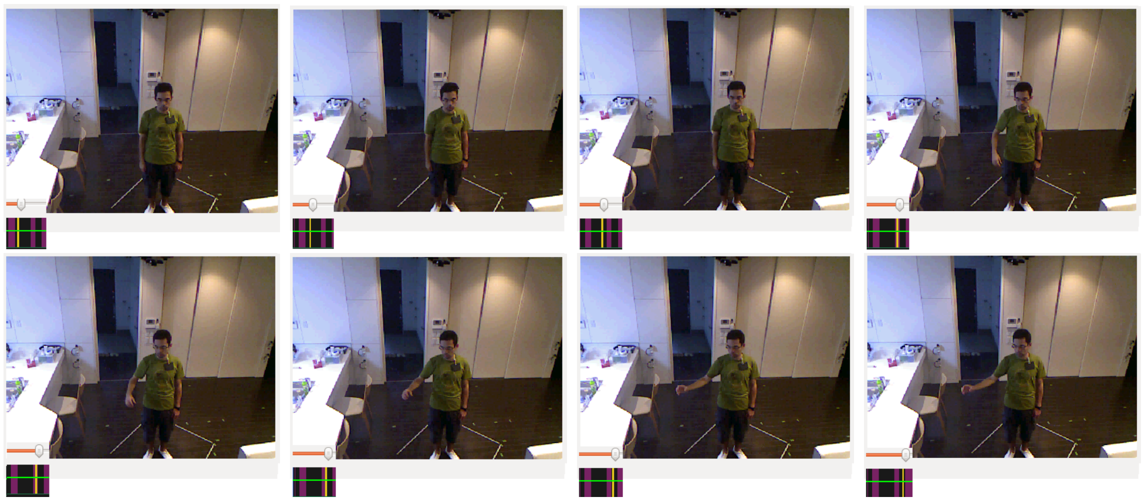
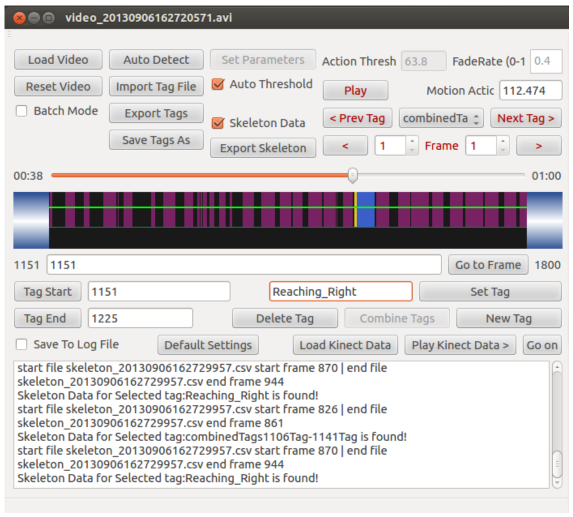


Figure 3: Motion Detection Detail

delete them. Using the lower bar in video slider, we can select the tag by dragging mouse. After selection we can delete the selected tags as shown in Figures 4, 5, 6, 7. Moreover it is possible to merge tag by selecting them from lower bar and clicking Combine Tags button as shown in Figures 8 and 9. Setting a Tag Name for a motion is done by writing tag in middle textbox and clickin Set Tag. The tag for the combined Tag in Figure 8 was set to Reaching_Right as shown in Figure 9.

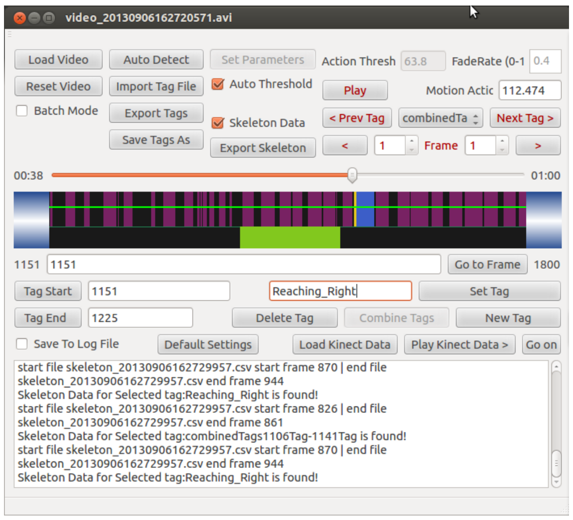
Furthermore, the user may want to edit or create a new motion tag other than auto detected motions in this GUI. New tag creation is done by clicking on New Tag button and clicking the first position of tag as in Figure 10. After clicking first position (blue area), we can change the start position by clicking Tag Start. Then we click to the slider position we want to be the start of the tag as shown in Figure 11. Same process in done for end position of the tag by Tag End button as shown in Figure 12. Lastly, we click New Tag Created button to finish new tag creation process(Figure 13). The new tag is shown as blue area in Figure 13. The processes for setting tag start and tag end is applicable to already created tags too.



Action Thresh 63.8 FadeRate (0-1 0.4
Motion Actic 112.474
combinedTa
1151 1151
Tag Start 1151 Reaching_Right
Tag End 1225

```
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
```

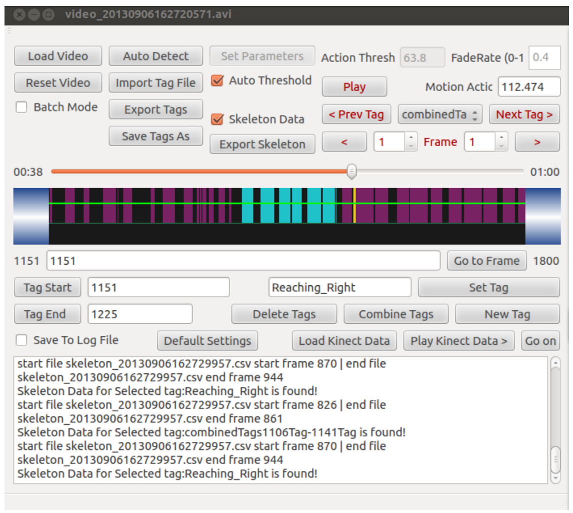
Figure 4: A motion tag can be selected by clicking on it



Action Thresh 63.8 FadeRate (0-1 0.4
Motion Actic 112.474
combinedTa
1151 1151
Tag Start 1151 Reaching_Right
Tag End 1225

```
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
```

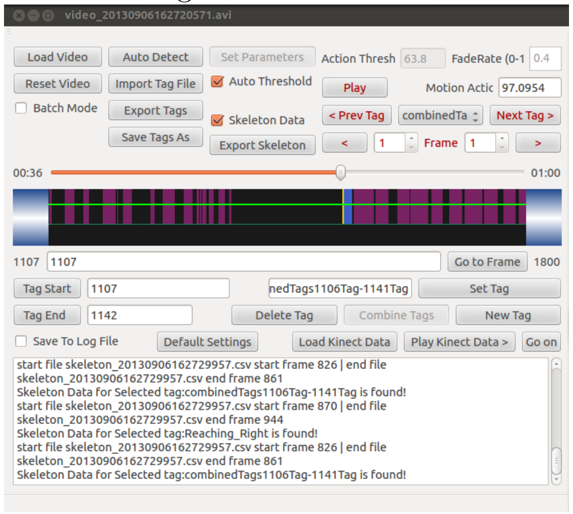
Figure 5: Dragging mouse in the lower bar creates green area



Action Thresh 63.8 FadeRate (0-1 0.4
Motion Actic 112.474
combinedTa
1151 1151
Tag Start 1151 Reaching_Right
Tag End 1225

```
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
```

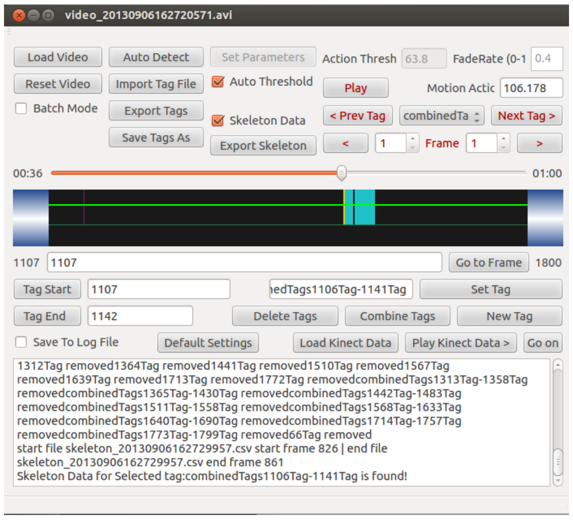
Figure 6: Tags that are left inside green area are selected



Action Thresh 63.8 FadeRate (0-1 0.4
Motion Actic 97.0954
combinedTa
1107 1107
Tag Start 1107 nedTags1106Tag-1141Tag
Tag End 1142

```
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
start file skeleton_20130906162729957.csv start frame 870 | end file
skeleton_20130906162729957.csv end frame 944
Skeleton Data for Selected tag:Reaching_Right is found!
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
```

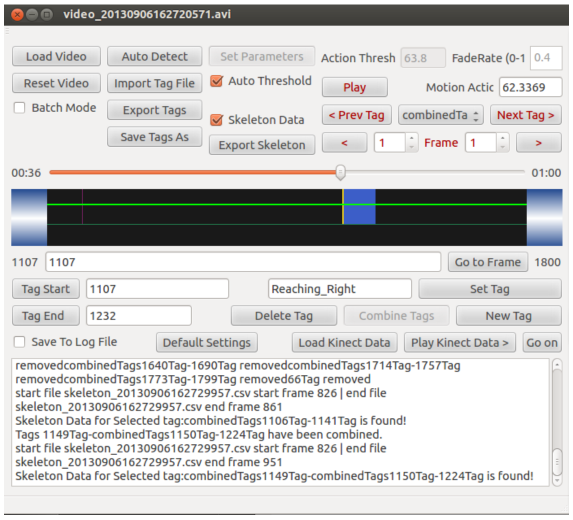
Figure 7: Delete Tags button deletes selected tags



```
Action Thresh 63.8    FadeRate (0-1 0.4
[✓] Auto Threshold    Motion Actic 106.178
[✓] Skeleton Data     combinedTa    Frame 1
[ ] Batch Mode
00:36  01:00
1107 1107    1800
Tag Start 1107    ıedTags1106Tag-1141Tag
Tag End 1142
[ ] Save To Log File
```

```
1312Tag removed1364Tag removed1441Tag removed1510Tag removed1567Tag
removed1639Tag removed1713Tag removed1772Tag removedcombinedTags1313Tag-1358Tag
removedcombinedTags1365Tag-1430Tag removedcombinedTags1442Tag-1483Tag
removedcombinedTags1511Tag-1558Tag removedcombinedTags1568Tag-1633Tag
removedcombinedTags1640Tag-1690Tag removedcombinedTags1714Tag-1757Tag
removedcombinedTags1773Tag-1799Tag removed66Tag removed
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
```

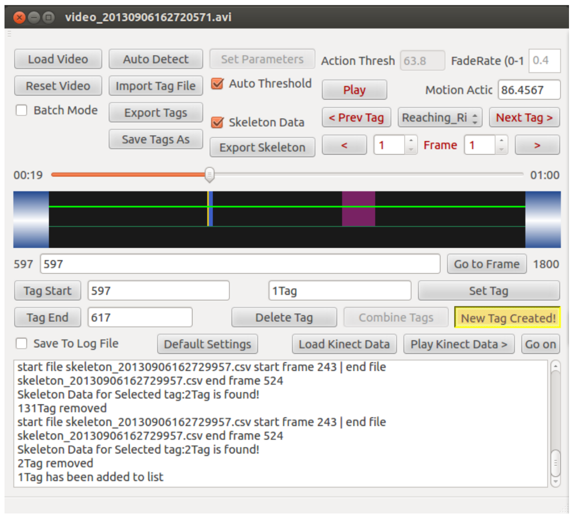
Figure 8: Several Tags are selected



```
Action Thresh 63.8    FadeRate (0-1 0.4
[✓] Auto Threshold    Motion Actic 62.3369
[✓] Skeleton Data     combinedTa    Frame 1
[ ] Batch Mode
00:36  01:00
1107 1107    1800
Tag Start 1107    Reaching_Right
Tag End 1232
[ ] Save To Log File
```

```
removedcombinedTags1640Tag-1690Tag removedcombinedTags1714Tag-1757Tag
removedcombinedTags1773Tag-1799Tag removed66Tag removed
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 861
Skeleton Data for Selected tag:combinedTags1106Tag-1141Tag is found!
Tags 1149Tag-combinedTags1150Tag-1224Tag have been combined.
start file skeleton_20130906162729957.csv start frame 826 | end file
skeleton_20130906162729957.csv end frame 951
Skeleton Data for Selected tag:combinedTags1149Tag-combinedTags1150Tag-1224Tag is found!
```

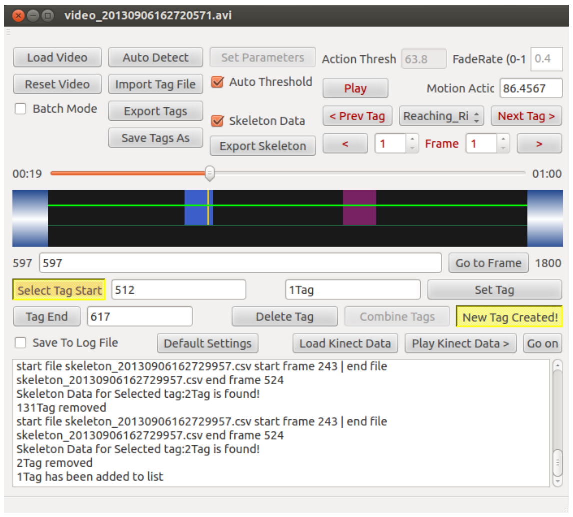
Figure 9: Tags are combined



Action Thresh 63.8 FadeRate (0-1 0.4 Motion Actic 86.4567 1 1
00:19 01:00
597 597 1800
Tag Start 597 1Tag
Tag End 617

```
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
131Tag removed
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
2Tag removed
1Tag has been added to list
```

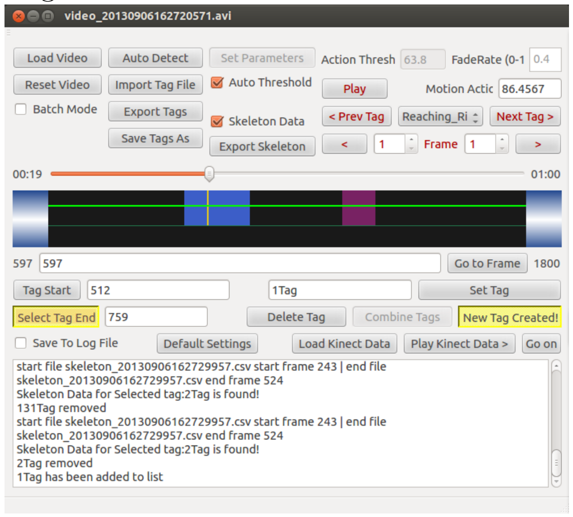
Figure 10: Clicking first position for new Tag



Action Thresh 63.8 FadeRate (0-1 0.4 Motion Actic 86.4567 1 1
00:19 01:00
597 597 1800
Select Tag Start 512 1Tag
Tag End 617

```
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
131Tag removed
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
2Tag removed
1Tag has been added to list
```

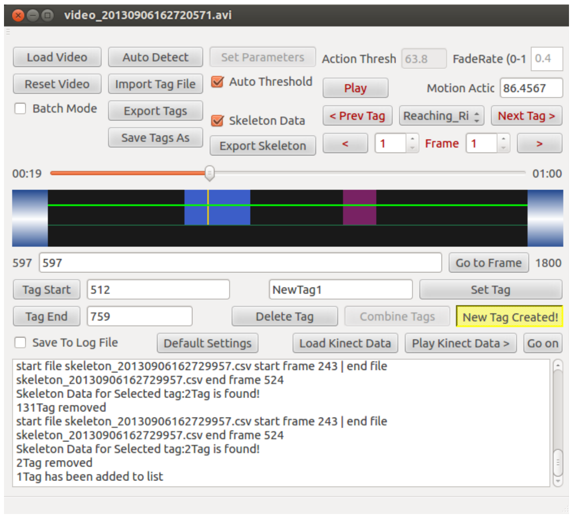
Figure 11: Changing start position of tag



Action Thresh 63.8 FadeRate (0-1 0.4 Motion Actic 86.4567 1 1
00:19 01:00
597 597 1800
Tag Start 512 1Tag
Select Tag End 759

```
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
131Tag removed
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
2Tag removed
1Tag has been added to list
```

Figure 12: Changing end position of tag



Action Thresh 63.8 FadeRate (0-1 0.4 Motion Actic 86.4567 1 1
00:19 01:00
597 597 1800
Tag Start 512 NewTag1
Tag End 759

```
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
131Tag removed
start file skeleton_20130906162729957.csv start frame 243 | end file
skeleton_20130906162729957.csv end frame 524
Skeleton Data for Selected tag:2Tag is found!
2Tag removed
1Tag has been added to list
```

Figure 13: Finishing New Tag creation

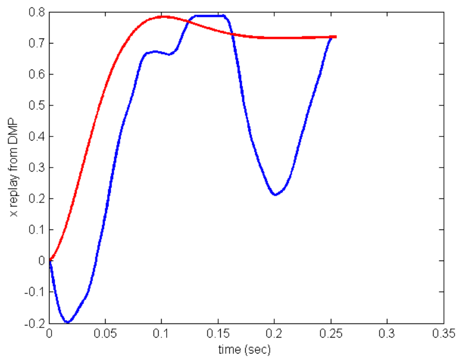
CHAPTER III

DYNAMIC MOVEMENT PRIMITIVES (DMP)

Dynamic Movement Primitives have been seen as building blocks of complex movements which can be used and modulated in real time [17, 10]. Initially introduced by Ijspeert et al. [25], DMPs are proposed as a possible mathematical formalization of such primitives. The DMP approach consists of a non-linear dynamical system which is forced to follow a desired trajectory by applying a parametric forcing term. A DMP model can be trained from one trajectory, or more by taking the mean of the weights as the representative for the trajectories belonging to a specific class. It is possible to reproduce a trajectory (or the mean trajectory) by numerical integration, and by using the learned weights [10]. There are two types of DMP described by Ijspeert et al., one is Rhythmic, and the orher one is Discrete DMPs [10]. Rhytmic DMPs are for cyclic movements. If we choose the forcing term f to be periodic, this will result in an oscillator. In this case we use Rhytmic DMP [29]. If we choose f to be phasic i.e. active in a finite time window, then we use Discrete DMP [10]. In my thesis, we have worked on only Discrete DMPs which are for discrete accelleration movements.

3.1 Representing Movements with Dynamical Systems

Dynamic Movement Primitives (DMPs) encode a desired movement trajectory in terms of the attractor dynamics of nonlinear differential equations with the help of adjustable weight parameters. DMP uses a second order linear dynamical system which is stimulated with a non-linear forcing term.



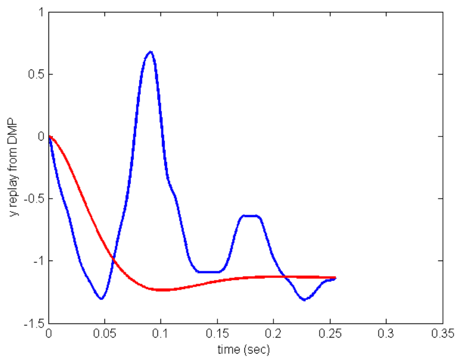


Figure 14: Replay of X from DMP for K=24 D=3

Figure 15: Replay of Y from DMP for K=24 D=3

Blue: Original Trajectory
Red: Replay by dynamical system without external force

3.1.1 Parameters of the Dynamical System

DMP uses a second order linear dynamical system which is stimulated with a non-linear forcing term. Let $x(t)$ denote a one-dimensional trajectory starting at $x(t_0) = x_0$ towards $x(t_f) = g$. 'Damped Spring'-like model which is used in many Robotics (PD control) applications is chosen as Dynamical System in DMP framework as described in the work of Ijspeert et al. as well as Miguel et al. [24, 10]. The equation is shown in Equation 1.

$$\tau\dot{v} = K(g - x) - Dv + (g - x_0)f(s) \tag{1}$$

$$\tau\dot{x} = v \tag{2}$$

$$\tau\dot{v} = a \tag{3}$$

The dynamical system in Equation 1 , named transformation system, has a global damping term $-Dv$, and $K(g - x)$ creates an attractor system towards the goal position [24]. The Transformation System's response with $f = 0$ is shown in Figures 14 and 15 for $K = 24$ $D = 3$.

The Transformation System's response with $f = 0$ is shown in Figures 16 and 17 for $K = 1$ $D = 1$. f is dependent on a phase variable s such that: [10]

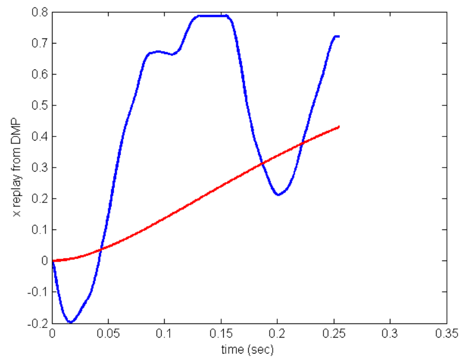


Figure 16: Replay of X from DMP for K=1 D=1

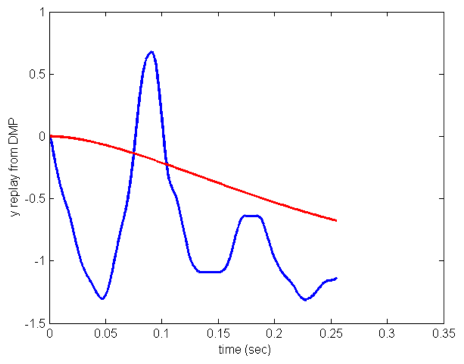


Figure 17: Replay of Y from DMP for K=1 D=1

Blue: Original Trajectory
Red: Replay by dynamical system without external force

$$\tau \dot{s} = -\alpha s \tag{4}$$

The canonical system is a first order dynamical system with the same temporal scaling factor τ, as in Equation 1 is a predefined constant. If no temporal scaling is wanted, τ is set to the duration of the movement [24].

K and D parameters can be selected to make the system critically damped as done by Glatz.[26]. However, other selections could result in more desired trajectories too. The phase variable s is initialized with one s(0)=1. The constant is chosen such that the phase variable s goes from one to zero as time passes $s \rightarrow 0$. The canonical system Equation 4 can be solved symbolically, its solution is given in Equation 5 [24].

$$s(t) = e^{-\frac{\alpha}{\tau}t} \tag{5}$$

As shown in Figure 18 with initial value of s being 1 and decreasing in time with $\alpha = 1$ and $\tau = 0.225$ seconds. The phase variable s on which the forcing term depends evolves exponentially from 1 to 0 as the time passes. The phase variable is used to remove the direct time dependency of the forcing term $f(s)$, and provides the complete system with a time scalability by adjusting the parameter τ. The phase variable is also used to weigh the forcing term, enabling this way to continuously shift towards a purely goal-attracted system.

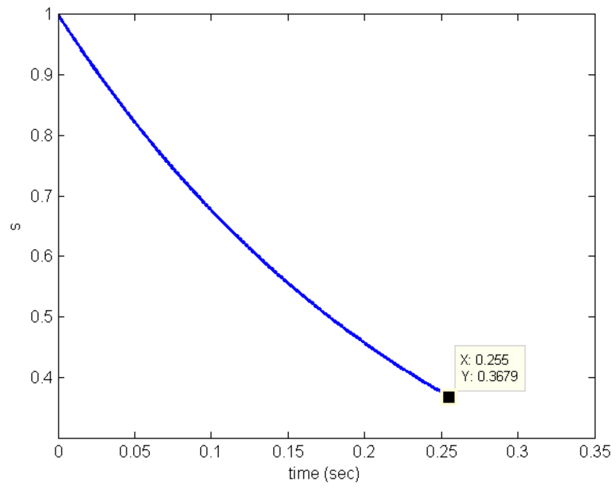


Figure 18: Phase variable s versus time

3.1.2 Forcing Function on the Dynamical System

The forcing term f representing is formulated as an arbitrary non-linear function as a sum of weighted exponential basis functions as seen in Equation 7 by Ijspeert et al. [10].

$$f(t) = \frac{\sum_{i=1}^{N} \Psi i\,(t)\, w_i}{\sum_{i=1}^{N} \Psi i\,(t)} \tag{6}$$

f(t) can be rewritten as f(s) by removing time dependency in Equation 8 [10].

$$f(s) = \frac{\sum_{i=1}^{N} \Psi i\,(s)\, w_i}{\sum_{i=1}^{N} \Psi i\,(s)} s(g - x_0) \tag{7}$$

The Basis functions formula which we used is shown below. The variables are width h_i, center c_i and adjustable weights w_i:

$$\Psi_i\,(s) = e^{-h_i (s - c_i)^2} \tag{8}$$

$$f_{target} = \frac{-K\,(g - x) + D\,\dot{x} + \tau\,x}{g - x_0} \tag{9}$$

f_{target} can be calculated with the Equation 9 for each trajectory [26]. In Figures 20 and 21, f_{target} for x and y coordinates of the example trajectory in Figure 19 is shown.

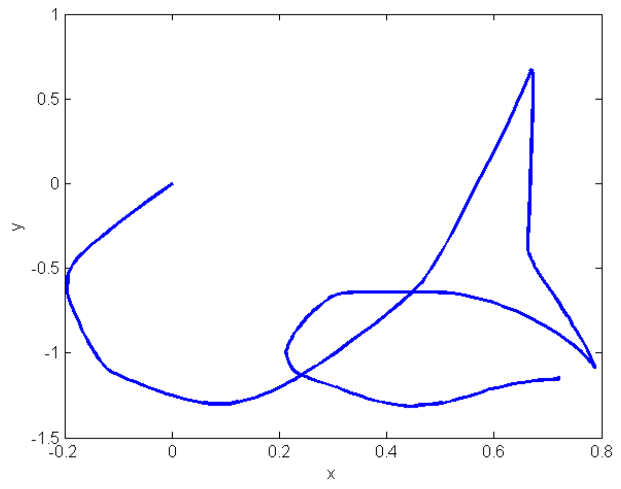


Figure 19: 2D trajectory of the example movement

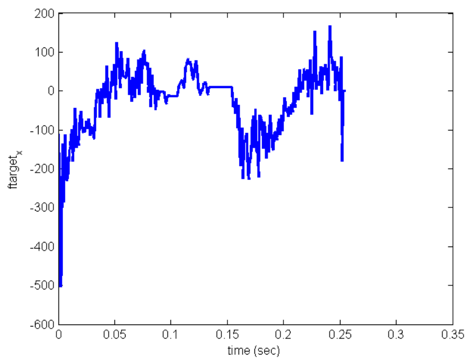


Figure 20: $f(s) = f_{targetx}$

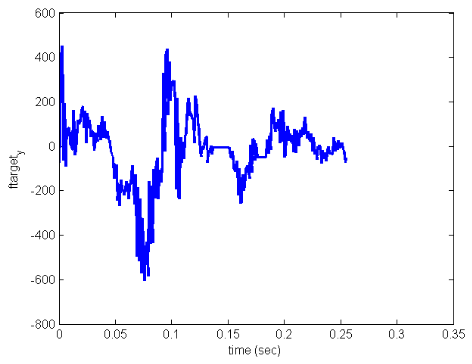


Figure 21: $f(s) = f_{targety}$

3.2 Learning and Reproduction in Dynamic Movement Primitives

3.2.1 Regression on Forcing Function

The non-linear forcing term $f_{target} = f(s)$ in the Equation 1 is where the learning takes place. We need to fit a sum of exponential basis functions to the calculated f_{target}. Parameters and Given Values:

1. We know the positions, velocities and accelerations from demonstration (training data). f_{target} is calculated from the Equation 9.

2. g=goal, the last position of the trajectory. x_0 is starting position.

3. K is spring constant and D is the damping term. We choose appropriate values for K and D such that the system converges to the goal before movement duration ends.

4. τ is temporal scaling factor. Without scaling, its value is the time duration of the movement. [24]

5. N is the number of Gaussians in the weighted sum of Gaussians that we fit to f_{target}.

6. h is the widths of Gaussians.

7. c is the centers of Gaussians.

Learning:

1. f_{fitted} is fitted on f_{target} by regression (locally weighted or linear regression).

2. We train one DMP for each dimension of data separately.

3. (for locally weighted regression) We minimize the locally weighted quadratic error criterion Equation 10 for locally weighted regression: [10]

$$J_i = \sum_{t=1}^{P} \psi_i(t) \left(f_{target}(t) - w_i \xi(t)\right)^2 \tag{10}$$

The calculation of weights by local regression is done by Equations 11, 12, 13, 14, 15 as decribed by Ijspeert et al. [10]

$$\xi(t) = s(t)(g - y_0) \tag{11}$$

$$\zeta = \begin{bmatrix} \xi(1) \\ \xi(2) \\ : \\ \xi(P) \end{bmatrix} \tag{12}$$

$$\Gamma_i = \begin{bmatrix} \psi_i(1) & .. & 0 & 0 \\ 0 & \psi_i(2) & .. & 0 \\ 0 & .. & .. & 0 \\ 0 & .. & 0 & \psi_i(p) \end{bmatrix} \tag{13}$$

$$f_{target} = \begin{bmatrix} f_{target}(1) \\ f_{target}(2) \\ : \\ f_{target}(P) \end{bmatrix} \tag{14}$$

$$w_i = \frac{\zeta^T \Gamma_i f_{target}}{\zeta^T \Gamma_i \zeta} \tag{15}$$

3.2.2 Reproduction of Learned Movement

We can calculate the weights for ‘Weighted Gaussian Sum’ as shown in Equation 15. After learning w_i, $f_{reproduced}$ can be reproduced by Equation 16 [10] Then the

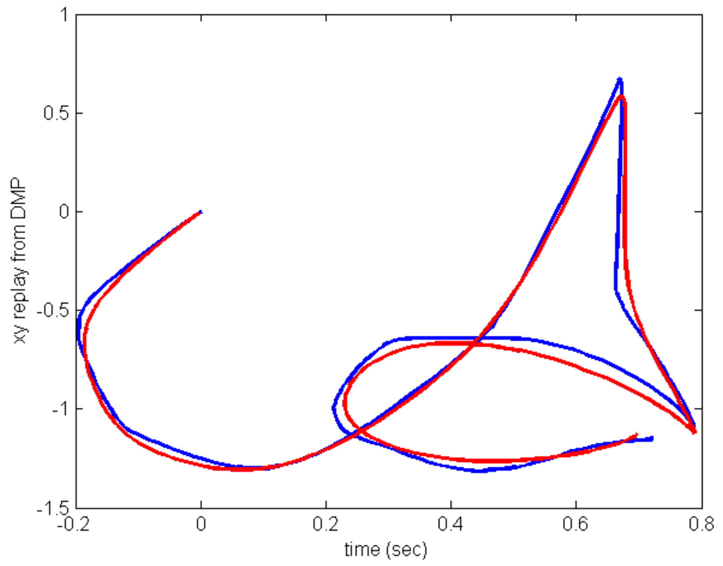


Figure 22: x-y trajectory replay with 40 Gaussians
Blue: Original Data
Red: Replay by DMP from learned parameters

trajectory or the 'mean' trajectory can be generated by 'Transformation System' shown in Equation 1 by numerical integration as shown in Equation 17 [26].

$$f_{reproduced}(s) = \frac{\sum_{i=1}^{N} \Psi i\,(s)\, w_i}{\sum_{i=1}^{N} \Psi i\,(s)} s(g - x_0) \tag{16}$$

$$\begin{aligned} \dot{v} &= (K(g-x) - Dv + (g-x_0) f_{reproduced}(s))/\tau \\ \dot{x} &= v/\tau \end{aligned} \tag{17}$$

The data of Figure 22 includes 256 points in x and y coordinates. The parameter values that are used in the Figures 23, 24, 25, and 26 are as follows:

Time duration of movement $= \tau = 0.2550$ $\alpha = 1$ $K = 24$ $D = 3$ Number Of Gaussians $= 40$ $h = 1$ for all Gaussians.

The Figures 23 and 24 shows the calculated f_{target} in Blue and the fitted f_{target} in Red for x and y coordinates respectively.

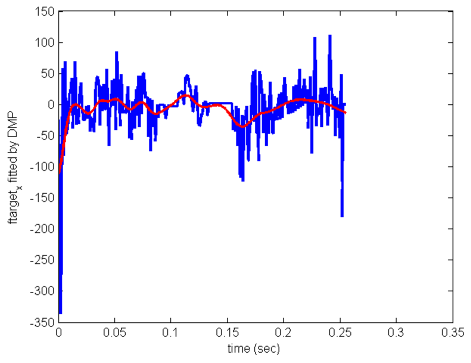


Figure 23: f_{target} for x trajectory

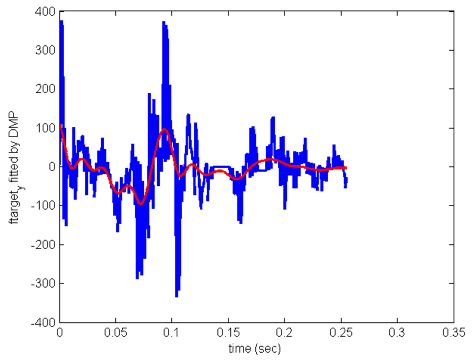


Figure 24: f_{target} for y trajectory

Blue: Original Data

Red: Replay by DMP from learned parameters

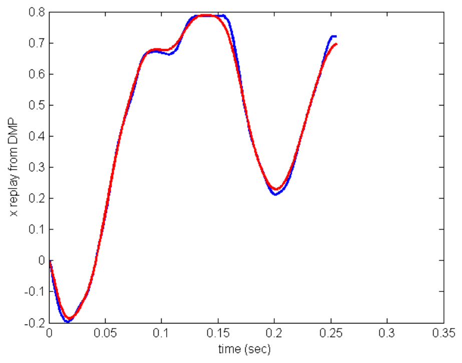


Figure 25: x trajectory replay

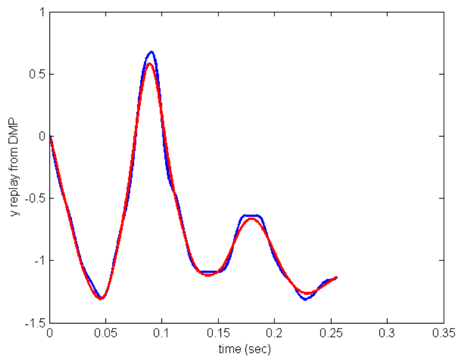


Figure 26: y trajectory replay

Blue: Original Data

Red: Replay by DMP from learned parameters

40 Number of Gaussians K=23 D=4

The Figures 25 and 26 shows the position coordinates for x and y respectively. Original position is shown in Blue and the reproduction of position calculated using Equation 17 is shown in Red.

Here are the errors we have in reproduction of the 'mean' trajectory from Figure 22 (40 number of Gaussians).

- Average position error in x : $error_x = 0.0100$
- Average position error in y : $error_y = 0.0336$
- Average velocity error in x : $error_{vx} = 0.4644$
- Average velocity error in y : $error_{vy} = 1.7774$
- Average acceleration error in x : $error_{ax} = 70.5403$
- Average acceleration error in y : $error_{ay} = 217.8957$
- Total average position error : $error_{xy}$= 0.0435

Average errors are calculated with 'mean of absolute difference of original value and reproduced value divided by the original value in each data point (of 256)' in trajectory duration.

Here is another example with less number of Gaussians in Figure 27 with the same data of Figure 22. The data of Figure 27 includes 256 points in x and y coordinates. The parameter values that are used in the Figures 28, 29, 30, and 31 are as follows: Time duration of movement $= \tau = 0.2550$ $\alpha = 1$ $K = 23$ $D = 4$ Number Of Gaussians $= 20$ $h = 1$ for all Gaussians.

The Figures 28 and 29 shows the calculated f_{target} in Blue and the fitted f_{target} in Red for x and y coordinates respectively.

The Figures 30 and 31 shows the position coordinates for x and y respectively. Original position is shown in Blue and the reproduction of position calculated using Equation 17 is shown in Red.

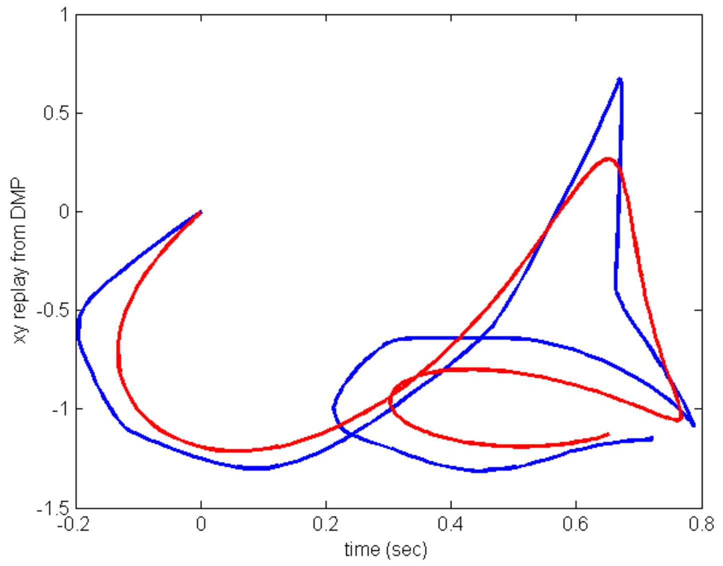


Figure 27: x-y trajectory replay with 20 Gaussians
Blue: Original Data
Red: Replay by DMP from learned parameters

Here are the errors we have in reproduction of the 'mean' trajectory from Figure 27 (20 number of Gaussians).

- Average position error in x : $error_x = 0.0394$
- Average position error in y : $error_y = 0.0982$
- Average velocity error in x : $error_{vx} = 0.8740$
- Average velocity error in y : $error_{vy} = 2.7714$
- Average accelleration error in x : $error_{ax} = 73.9018$
- Average accelleration error in y : $error_{ay} = 228.8778$
- Total average position error : $error_{xy}$= 0.1377

As one can see from this result the total position error has increased (from 0.0435 to 0.1377) with the decreased number of Gaussians.

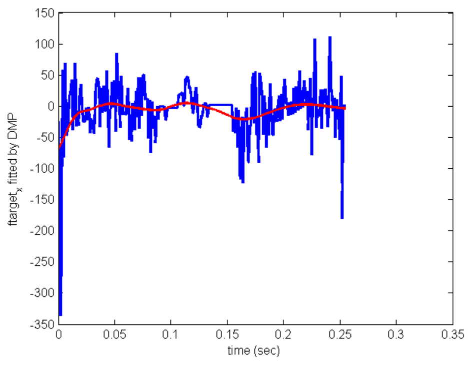


Figure 28: f_{target} for x trajectory

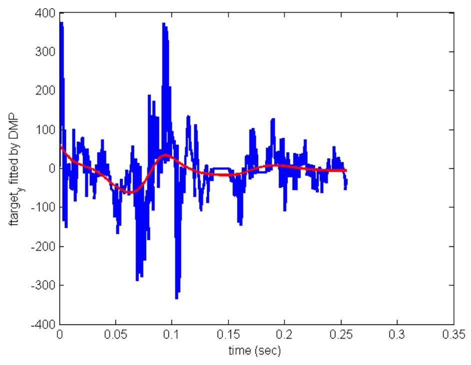


Figure 29: f_{target} for y trajectory

Blue: Original Data

Red: Replay by DMP from learned parameters

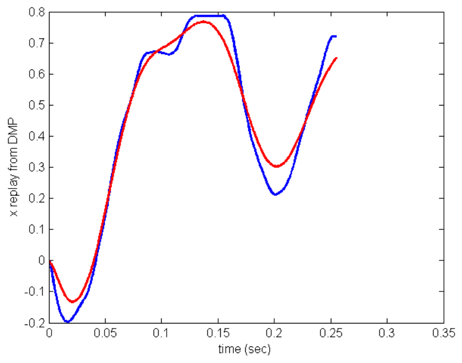


Figure 30: x trajectory replay

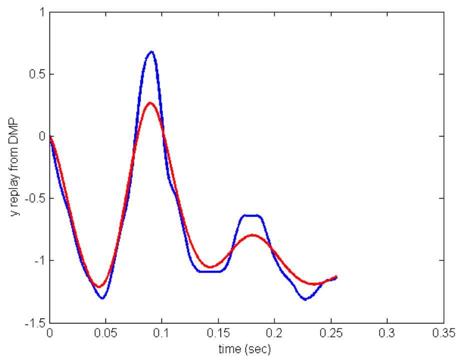


Figure 31: y trajectory replay

Blue: Original Data

Red: Replay by DMP from learned parameters

20 Number of Gaussians K=23 D=4

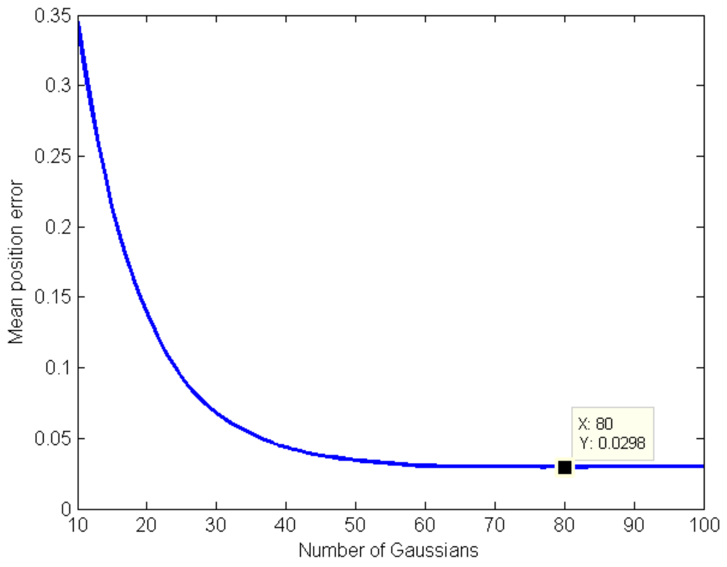


Figure 32: Mean Error versus number of Gaussians

Number of Gaussians clearly changes how well we can fit the DMP reproduced trajectory to the original data. Here is a graph of change of mean absolute error with number of Gaussians in Figure 32. The parameters in this DMP are K=24 D=3. The data in Figure 22 is used. The minimum error was obtained at 80 number of Gaussians. Here is another example with differen K and D values (K=1 D=1) for the dynamical system with the same data of Figure 22. In Chapter 3.1.1 'Parameters of Dynamical System', the difference that two sets of K and D values makes in Dynamical System response were shown in Figures 16, 17, 14, and 15. Figures 16, 17 show that with K=1 D=1 values dynamical system doesn't even reach the goal in movement duration. As a result of this difference the reproduction results vary as explained below.

The data of Figure 33 includes 256 points in x and y coordinates. The parameter values that are used in the Figures 34, 35, 36, and 37 are as follows:

Time duration of movement $= \tau = 0.2550$ $\alpha = 1$ $K = 1$ $D = 1$ Number Of Gaussians $= 40$ $h = 1$ for all Gaussians.

The Figures 34 and 35 shows the calculated f_{target} in Blue and the fitted f_{target} in

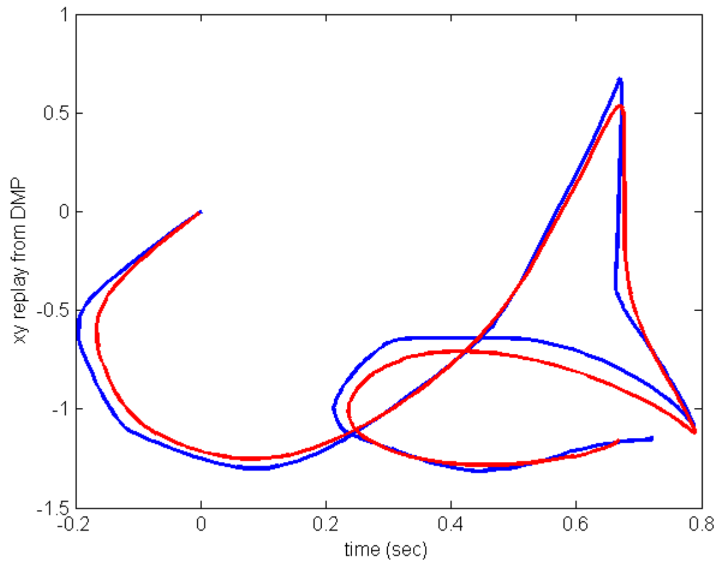


Figure 33: x-y trajectory replay with 40 Gaussians K=1 D=1
Blue: Original Data
Red: Replay by DMP from learned parameters

Red for x and y coordinates respectively.

The Figures 36 and 37 shows the position coordinates for x and y respectively. Original position is shown in Blue and the reproduction of position calculated using Equation 17 is shown in Red.

Here are the errors we have in reproduction of the 'mean' trajectory from Figure 33 (40 number of Gaussians K=1 D=1).

- Average position error in x : $error_x = 0.0160$
- Average position error in y : $error_y = 0.0422$
- Average velocity error in x : $error_{vx} = 0.4866$
- Average velocity error in y : $error_{vy} = 1.8194$
- Average accelleration error in x : $error_{ax} = 70.5643$
- Average accelleration error in y : $error_{ay} = 217.5266$

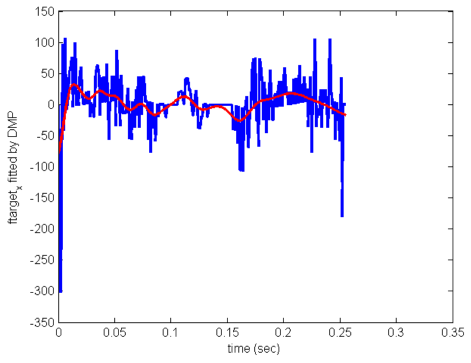


Figure 34: f_{target} for x trajectory

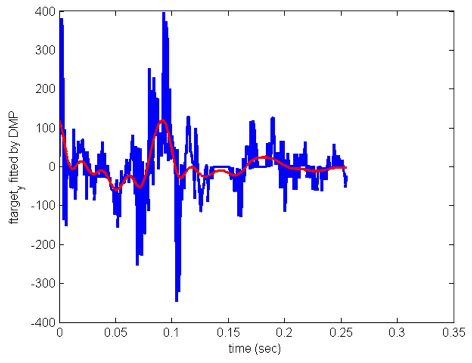


Figure 35: f_{target} for y trajectory

Blue: Original Data

Red: Replay by DMP from learned parameters

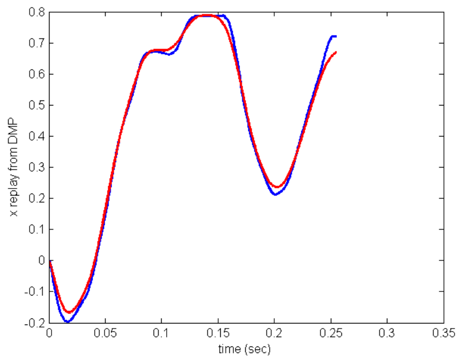


Figure 36: x trajectory replay

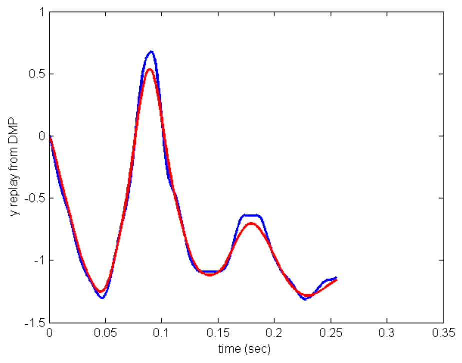


Figure 37: y trajectory replay

Blue: Original Data

Red: Replay by DMP from learned parameters

40 Number of Gaussians K=1 D=1

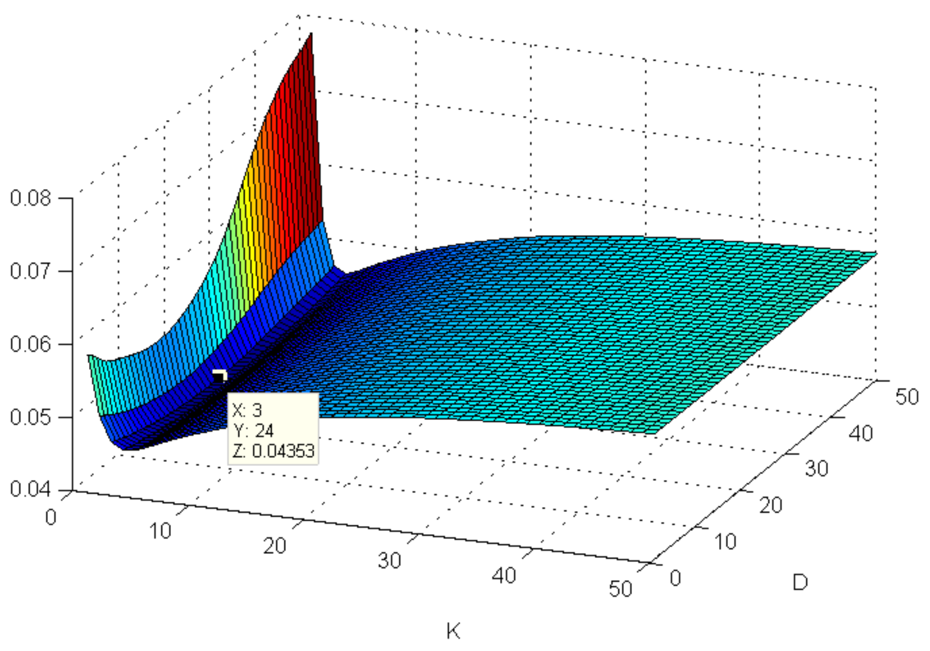


Figure 38: K-D vs Reproduction Error 3D View Plot
Blue: Lower Values
Red: Higher Values

- Total average position error : $error_{xy}$= 0.0582

As one can see from this result the total average position error has increased little (from 0.0435 to 0.0582) with the same number of Gaussians with Figure 22 but poor choise of K and D values.

For the trajectory data shown in Figure 22, 2500 DMPs were trained and the absolute error between original trajectory and the reproduction of the movement by DMPs were calculated. DMP models were trained with different values ranging from 1 to 50 for K and D variables with 40 Gaussian basis functions. The resulting error matrix for different K and D values was shown in Figure 38. The lowest error value was gained at the K=24 D=3 values in the given range of 1-to-50. The replay error is calculated by the sum of absolute position errors in x and y coordinate DMPs.

Here is the lowest error DMP reproduced trajectory in Figure 39. Here are the errors we have in reproduction of the 'mean' trajectory from Figure 39 (80 number

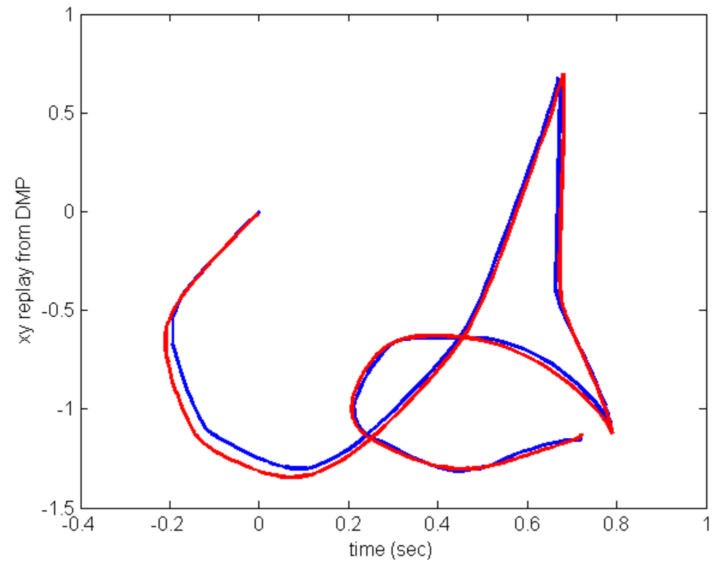


Figure 39: K=24 D=3 80 Number of Gaussians DMP Reproduction
Blue: Original Data
Red: Replay by DMP from learned parameters

of Gaussians). As we can see the total position error has decreased (from 0.0435 to 0.0298) with the increased number of Gaussians compared to Figure 22.

- Average position error in x : $error_x = 0.0077$
- Average position error in y : $error_y = 0.0221$
- Average velocity error in x : $error_{vx} = 0.2788$
- Average velocity error in y : $error_{vy} = 0.9041$
- Average accelleration error in x : $error_{ax} = 66.7336$
- Average accelleration error in y : $error_{ay} = 198.5779$
- Total average position error : $error_{xy}$= 0.0298

We get the minimum average error of 0.0056 using DMP with 80 number of Gaussians and K=12 D=3 $h = 1$ values. The absolute difference of max and min x values is

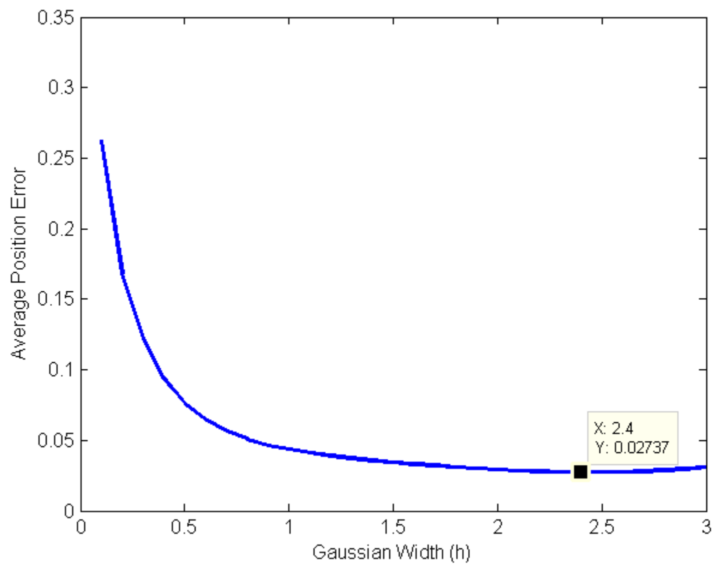


Figure 40: Width of Gaussians vs Average Position Error
K=24 D=3 40 Number of Gaussians

0.9833. If we divide the error in x axis trajectory with this value, we get:
0.0056/0.9833 = 0.0057 = 0.5695% average error in x trajectory. We get the minimum average error of 0.0207 using DMP with 80 number of Gaussians and K=17 D=2 values. The absolute difference of max and min y values is 1.9876. If we divide the error in y axis trajectory with this value, we get:
0.0207/1.9876 = 0.0104 = 1.0400% average error in y trajectory. We can decrease the errors with the cost of increased number of Gaussians and different K and D for x and y axis DMPs.

Widths (h) of Gaussians that are used in basis function Equation 8 is also one of the parameters that affect how well the DMP reproduction of the trajectory fits the original trajectory. Until this part we have taken widths of Gaussians as '1'. Now we investigate how the width of Gaussians' change position errors. The graph shows width of Gaussians vs average position error is given in Figure 40. As seen in Figure 40, 2.4 is a better choice for a DMP with 40 Number of Gaussians and K=24 D=3 values. That is because the average position error is decreased compared to DMP with h=1 (from 0.0435 to 0.0274).

Gaussians that we used as basis functions have one more parameter, centers of the Gaussians (c). As described in Equation 7, the external force on the sytem depends on phase variable s which is decreasing with time. So we have chosen to set the centers of Gaussians equally seperated in range of s during duration time of the movement. This results in unequal distribution of centers in time axis. Therefore, Time-vs-s plot doesn't result in equal changes in time. The absolute difference in consecutive values of s is changing with time as shown in Figure 42. Since the basis functions depend on s values which change exponentially as the time moves, the resultant Gaussian centers will be shaped differently when plotted on time axis. In addition to the 'equally distanced in s' approach I have also tried 'equally distanced in time' approach as done by Ijspeert et al. [10]. Resultant 10 Gaussians and their centers are shown in Figure $\dot{S}$ince the average position error of reproduction of the data in Figure 22 with 2.4 width (h) have resulted in as low as 0.0274 before using 'equally distanced in s' approach which is shown in Figure 43. However the result with the approach 'equally distanced in time' as shown in Figure 44 is not better than the former, as the minimum average position error with same parameters other than centers (c) and widths (h) is 0.0402 (compared to 0.0274) in width 3.1 as shown in Figure 41.

τ is the temporal scaling factor. If no time scaling is needed, τ should be the movement duration. If we want time scaling in reproduction of the trajectory, it is possible to change the movement duration by adjusting τ. The changes of duration will result will result differences in reproduced trajectory as seen in Figures 45 and 46.

It is possible to change the goals during reproduction of the trajectories. Here is an example with 2 sets of changed goals in DMP trajectory reproduction in Figures 47 and 48. As the goals are the last position in x and y axis' originally, here we set first 0.75×original goal for both x and y axis. Second, we set 1.5×original goal for

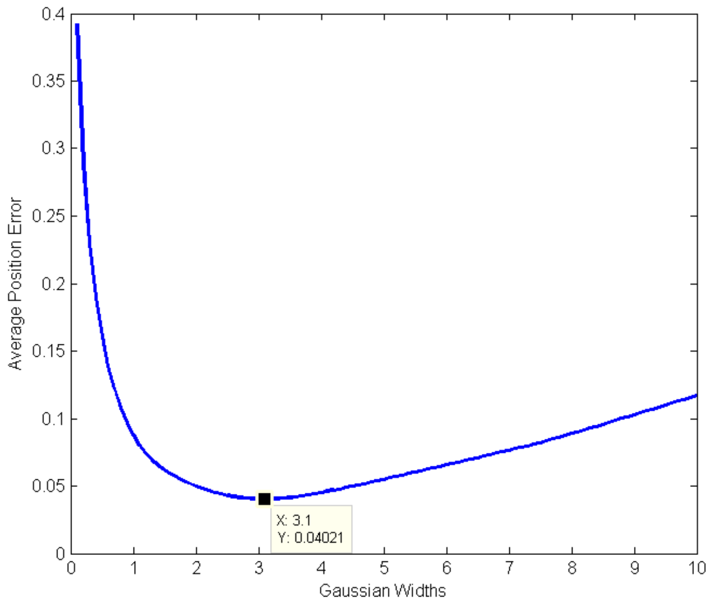


Figure 41: Width of Gaussians vs Average Position Error in 'equally distanced in time' approach

K=24 D=3 40 Number of Gaussians 'equally distanced in time'

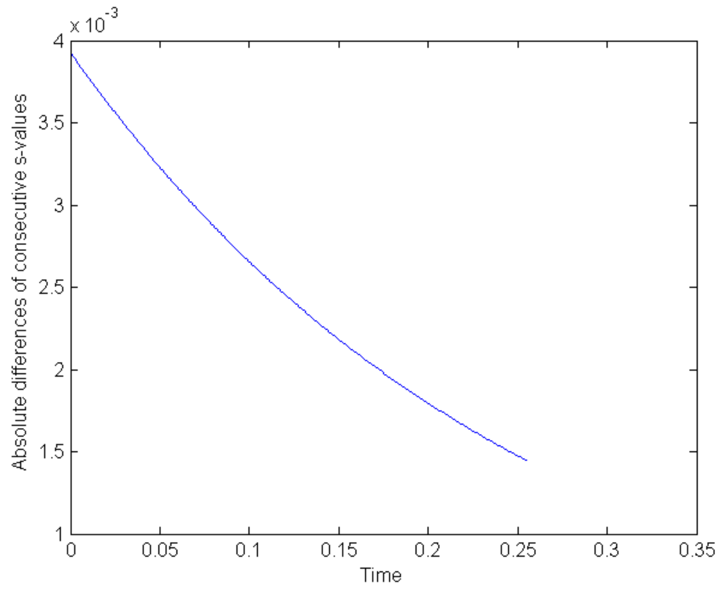


Figure 42: Absolute difference in consecutive values of s vs time

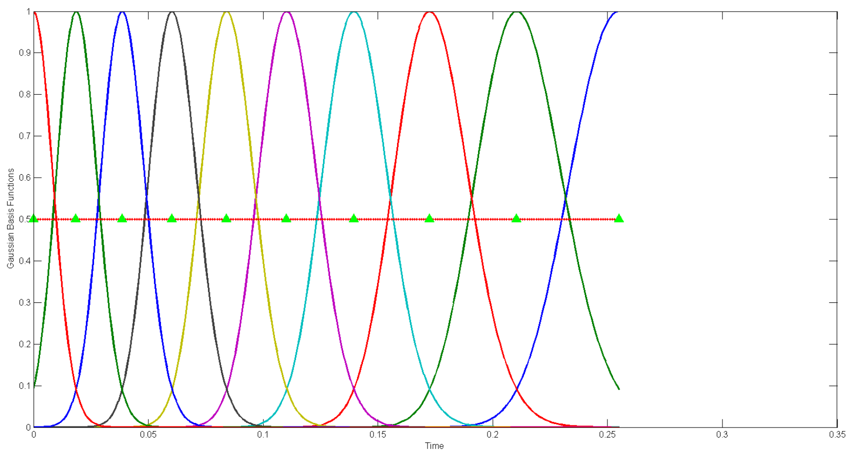


Figure 43: 10 Gaussians vs Time in ‘equally distanced in s’ approach
Green Triangles : Centers of Gaussians

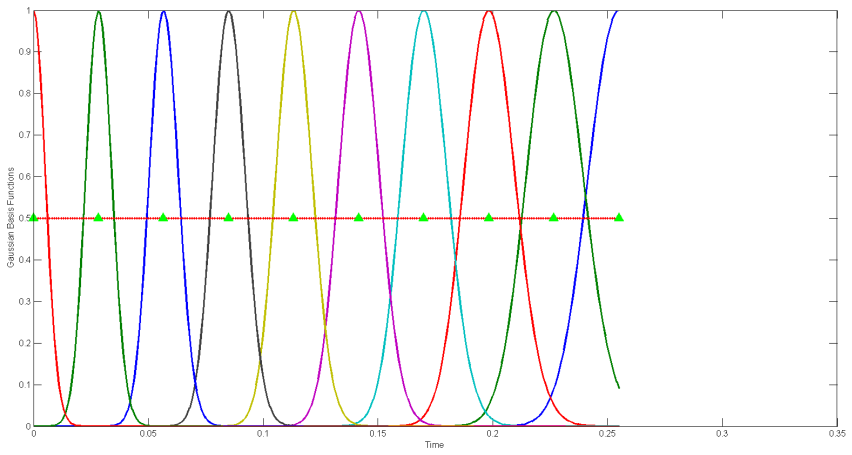
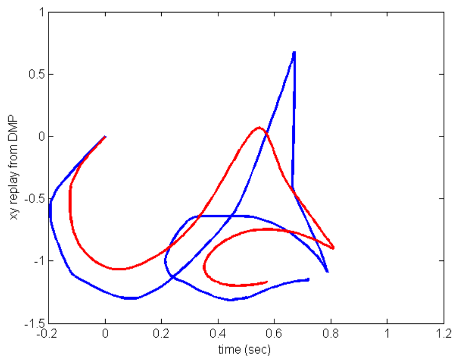


Figure 44: 10 Gaussians vs Time in ‘equally distanced in time’ approach
Green Triangles : Centers of Gaussians



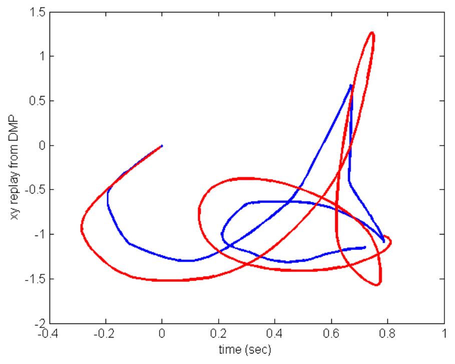
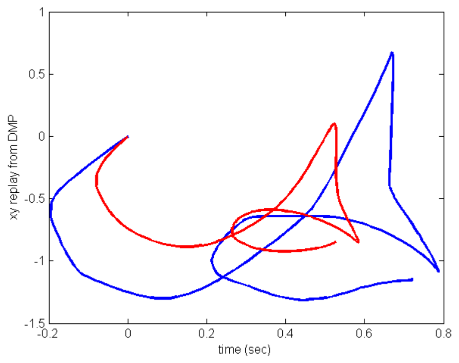


Figure 45: $\tau = 1.2 \times duration$ **Figure 46:** $\tau = 0.8 \times duration$
Blue: Original Data
Red: Replay by DMP from learned parameters
40 Number of Gaussians K=23 D=4



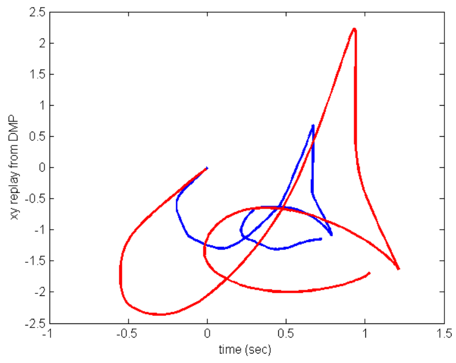


Figure 47: $g = 0.75 \times original$ **Figure 48:** $g = 1.5 \times original$
Blue: Original Data
Red: Replay by DMP from learned parameters
40 Number of Gaussians K=23 D=4

both x and y axis. This change also causes position scaling. In Ijspeert et al.'s work, a hand written letter was given as training data to a discrete DMP. Then the goal was changed by an amount. Due to the proper formulation of dynamical system with invariance properties and scaling term $(g - x_0)$ in Equation 1, the result of goal change in DMP was a properly uniformly zoomed version of the original letter [10].

Furthermore, changing the initial positions $(x_0\ y_0)$ during reproduction of the trajectories. Here is an example with changed initial positions in DMP trajectory reproduction in Figures 49 and 50. First only x_0 is changed by -0.2 in position, then only

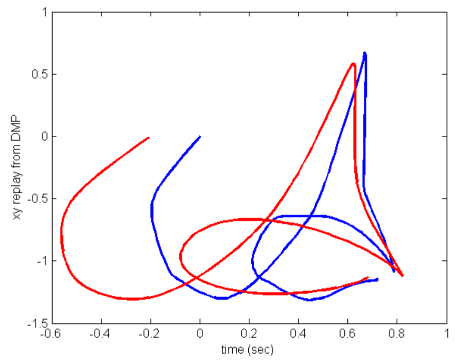


Figure 49: x_0 is set to $x_0 - 0.2$

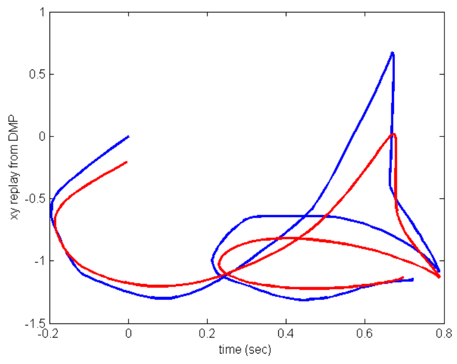


Figure 50: y_0 is set to $y_0 - 0.2$

Blue: Original Data
Red: Replay by DMP from learned parameters
40 Number of Gaussians K=23 D=4

y_0 is changed by -0.2 in position in Figures 49 and 50.

3.3 Movement Recognition with DMP

The main principle that allows DMPs to be used for recognition is the fact that two movements can be compared by using their DMP representations, i.e. weights. By nearest neighbor approach, we test the test data with mean of previously trained letters' weights. An example with 40 Gaussian DMPs shown in Figures 51, 52, and 53.

'Training A's data is 2x256 points.

'Test A's data is 2x251 points.

'Test U's data is 2x240 points.

Distances in w_x + Distances in w_y:

- 'Training A' vs 'Test A'= 7456
- 'Training A' vs 'Test U'= 9040

As expected 'Training A vs Test A' is smaller. Therefore 'Test A's weights are closer to 'Training A's weights. This property allows us to recognize different letters or movements.

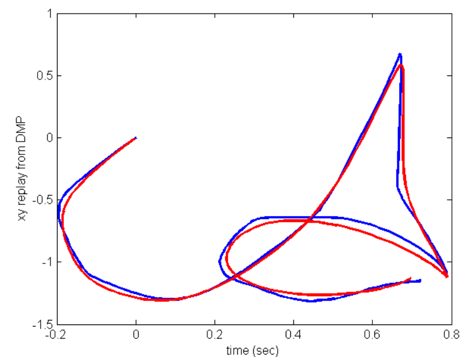


Figure 51: Training for letter A

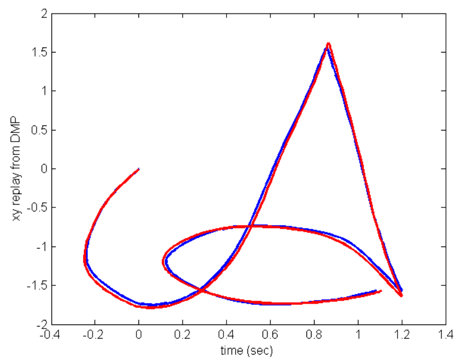


Figure 52: Testing letter A

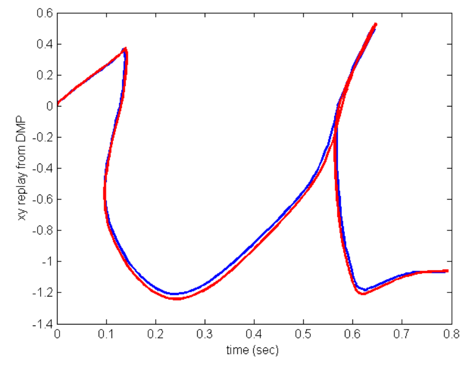


Figure 53: Testing letter U
Blue: Original Data
Red: Replay by DMP from learned parameters

CHAPTER IV

DMP BASED MOVEMENT RECOGNITION AND COMPARISONS WITH HMM

4.1 Setup and Implementation

For the tests we have collected coordinate data of hand written individual letters in the air by body movements. The data were recorded via Kinect for Windows v1.0 (Microsoft Corp.). Some frames from the recording scene can be seen in Figure 54. The right hand data were extracted from Kinect skeleton data. Trajectories, velocities, and acceleration data of the right hands movements during letter air writings are stored with corresponding time stamps. In Figure 55, a playback from recorded data is show.

4.1.1 Dataset and Modifications

Data is divided into training and test data. When the training and test sets are collected, the data were scaled and normalized. The starting point of trajectories were set to [0.0 0.0] coordinates and the total movement was scaled to a range between 0.0 to 1.0 in each dimension. The time between consecutive data were not equal. The time differences were made equal by linear interpolation. Then the time duration were scaled to a constant value of 10. Moreover, the number of data points were fixed at 150. Training set consists of scaled 15 set of 5 handwritten letters as seen in Figure 56. Test set consists of scaled 20 set of 5 handwritten letters as seen in Figure 57. The mean trajectories of the datasets are shown in the Figures 56 and 57 with red color. Only the training data is used when training both models. Only test data is used when testing the models.

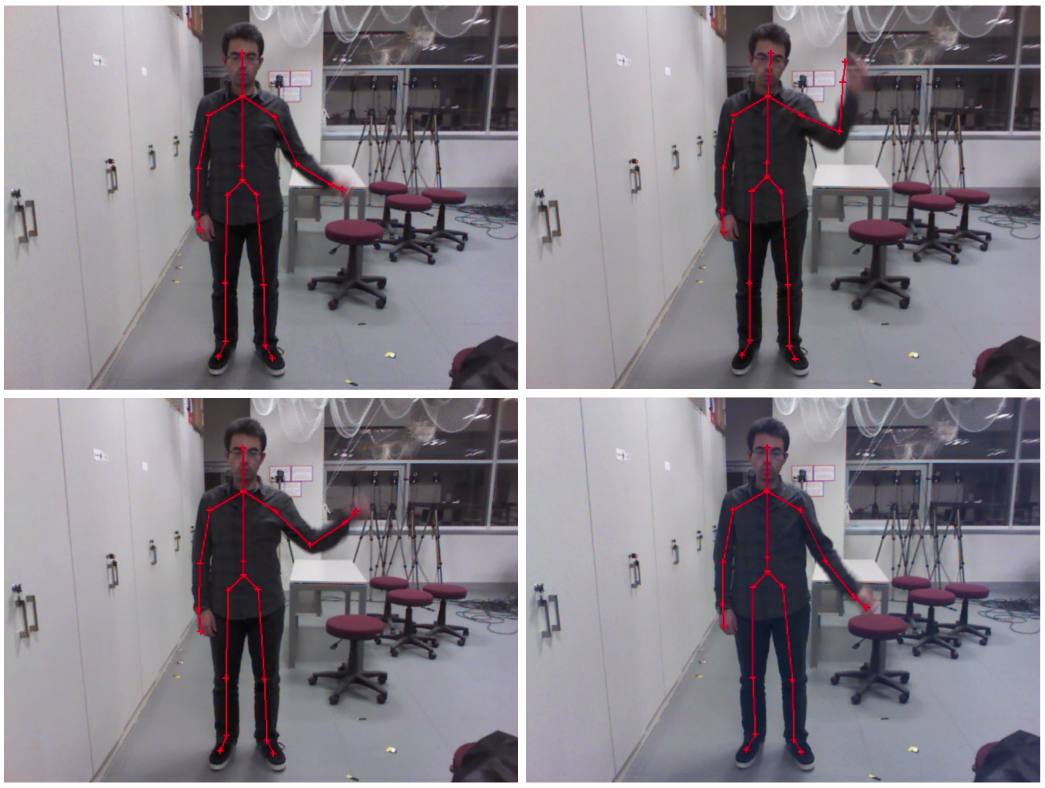


Figure 54: Some of the frames from a movie recorded via Kinect while data recording

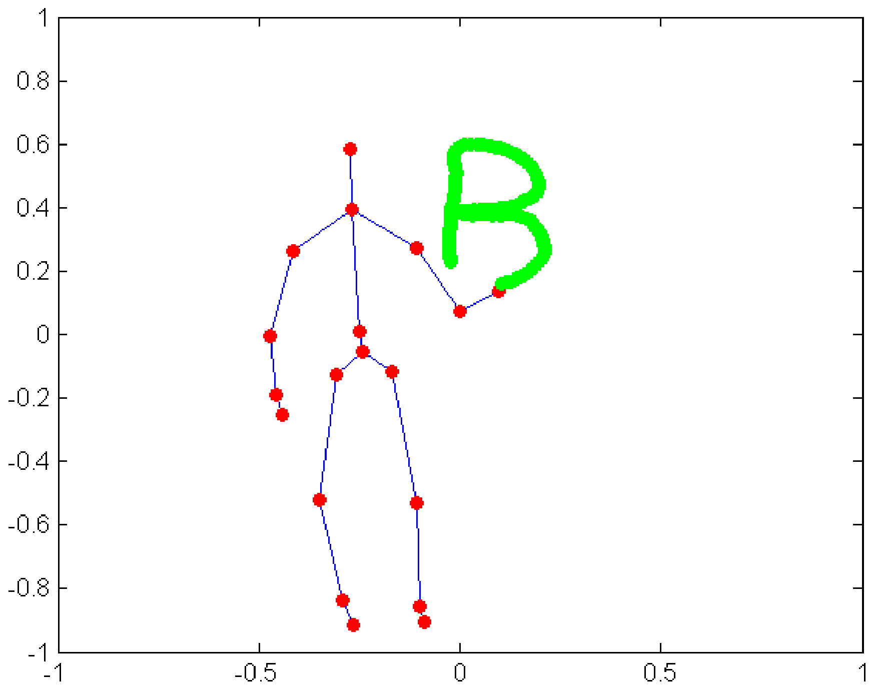


Figure 55: A snapshot from replay of a letter writing data recorded via Kinect.

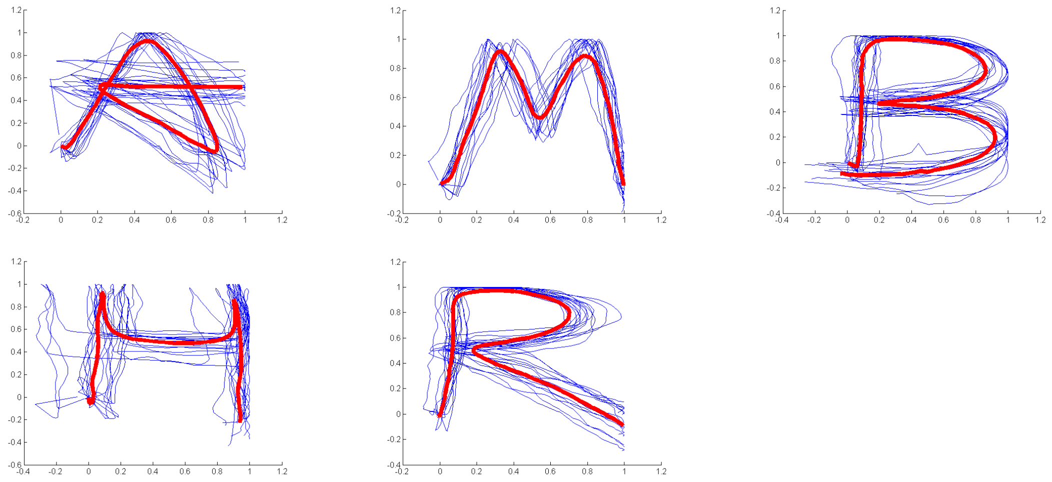


Figure 56: Scaled training set consisting of 15x5 handwritten letters by arm movements in the air
Blue: Training set
Red: Overlayed Mean Trajectory of 15 Training Data

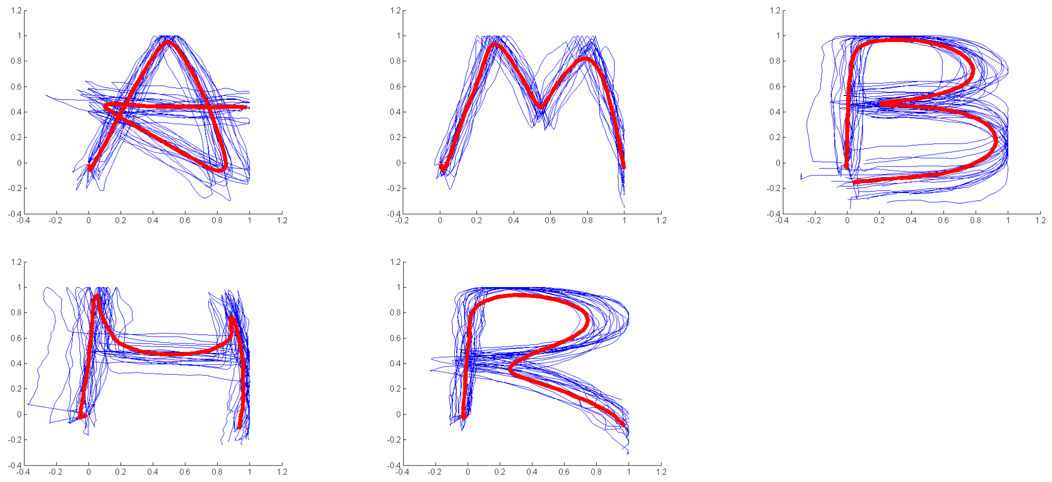


Figure 57: Scaled test set contains 20x5 hand written letters by arm movements in the air
Blue: Test set
Red: Overlayed Mean Trajectory of 20 Test Data

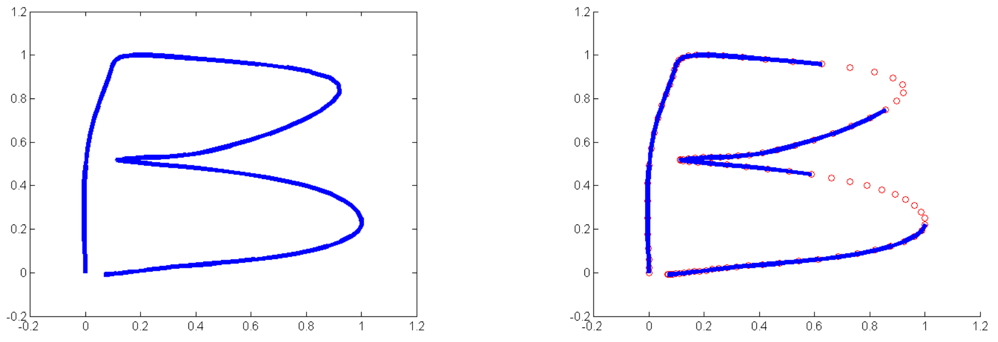


Figure 58: Original data and the data with broken pen modification

Test data was given to both recognizer with and without modifications. The modifications were made on the test data to test the noise robustness of the recognizers. First modification was Broken Pen modification, where two randomly selected portions (5% each) of the data was deleted. The deleted part is not interpolated, the data sequence jumps to the further data points in the next time step. The purpose of this modification was to simulate data loss during data records. Figure 58 demonstrates the Broken Pen modification in the data.

After the application of Broken Pen modification, the data were further modified with white noise at several signal to noise ratio (SNR) values. Figure 59 demonstrates the noise contaminated trajectory data used in the tests.

4.2 Recognition Task

Recognition task was done with two different methods. One is DMP other one is HMM. Models for both methods were trained with the same trainig data and tested with the same testing data. The correct recognition rate were compared between both methods.

4.2.1 Hidden Markov Model (HMM) for Recognition

HMM is a mathematical model of a stochastic process and includes three parameters $\lambda = (\Pi, A, B)$ where Π represents initial vector, A is the transition matrix and B

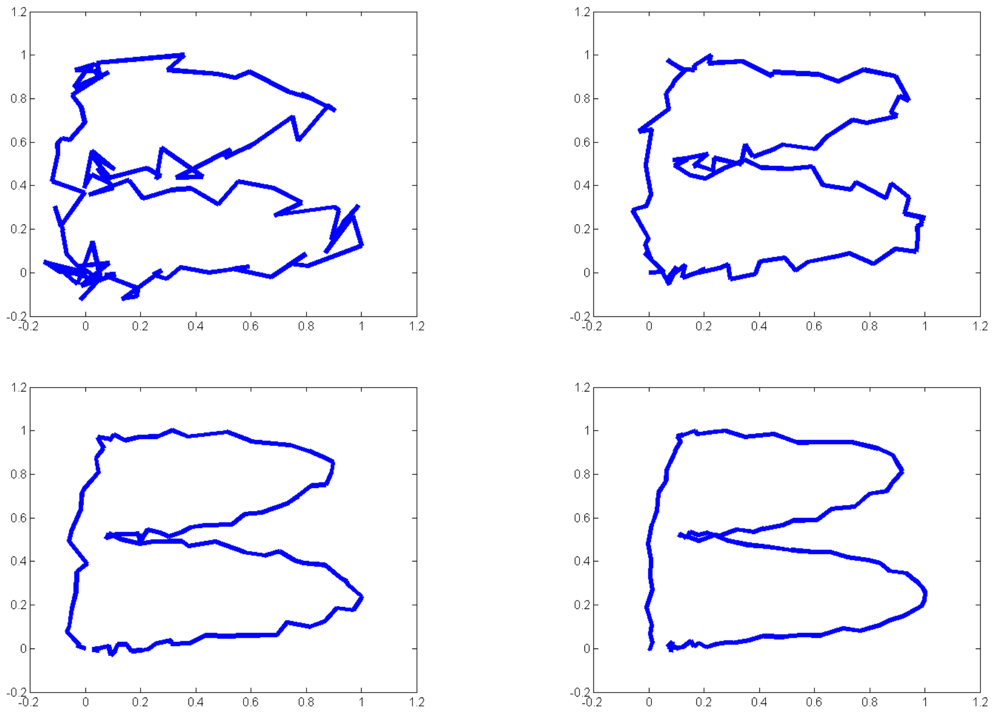


Figure 59: Test data with white noise. SNRs are as follows: 20, 25, 30, 35

refers to emission matrix. There 3 main problems in HMM: [11]

1. Evaluation Problem: Given a model and a sequence of observations, how do we compute the probability that observed sequence was produced by the model. In other words, scoring how well the model fits observed data.

2. Finding the optimal hidden state sequence for an observation.

3. Training the model: how do we optimize the parameters A,B, Π such that the probability calculated in problem 1 is maximum.

Problem 1 can be optimally solved by forward-backward procedure.
Problem 2 can be solved by Viterbi algorithm.
And problem 3 can be applied by Baum-Welch (or equivalently Expectation Maximization).
The solution for problem 1 can also be used for recognition. If we train a model for

each class of movements, then test an observed movement with each model. We can find the best fit model, therefore the most probable class which observed movement belongs to.

We have used Kevin Murphys pmtk3 toolkit in Matlab (Mathworks, Inc.) for HMM tests. We are using Gaussian function as the HMMs emission type. By default, pmtk3 toolkit lightly regularizes all parameters, so we are doing Maximum a Posteriori estimation, not Maximum Likelihood Estimation [16].

4.2.2 Parameter Selection of Models

4.2.2.1 Hidden Markov Model (HMM)

The parameters of the HMM that we used in our tests are explained as following:

N_{states} the number of hidden states is an open parameter that we chose. We have chosen this parameter according to the experiments. We have tested the range of 3 to 15 states with the test data. We get a local maxima at the 5 states which gave the maximum performance of 83 correct recognition over 100. The results can be seen at Figure 60.

The type of the emission was chosen as Gaussian. It represents the desired emission i.e. observation (local) distributions type. π_0 is initial value for the starting distribution. It is randomly initialized. It is a 1-by-N_{states} vector that sums to one. $trans_0$ is initial value for the state transition matrix. It is also randomly initialized. It is an N_{states}-by-N_{states} matrix whose rows sum to one.

$emission_0$ is initial value for the emission (local) distribution. It is also randomly initialized at the training of HMM. d is the dimension of the data, it is 2 in our case. The learned parameters in HMM are $\mu_{emission}$ mean of Gaussian emission. It is $d^2 \times N_{states}$ matrix. $\sigma_{emission}$ standard deviation of Gaussian emission. $2 \times N_{states}$ matrix A the transition matrix which is a $N_{states} \times N_{states}$ matrix is also one of the learned parameters. The parameters other then N_{states} are either randomly initialized, learned, or fixed parameters. Therefore we can approximate the number of learned

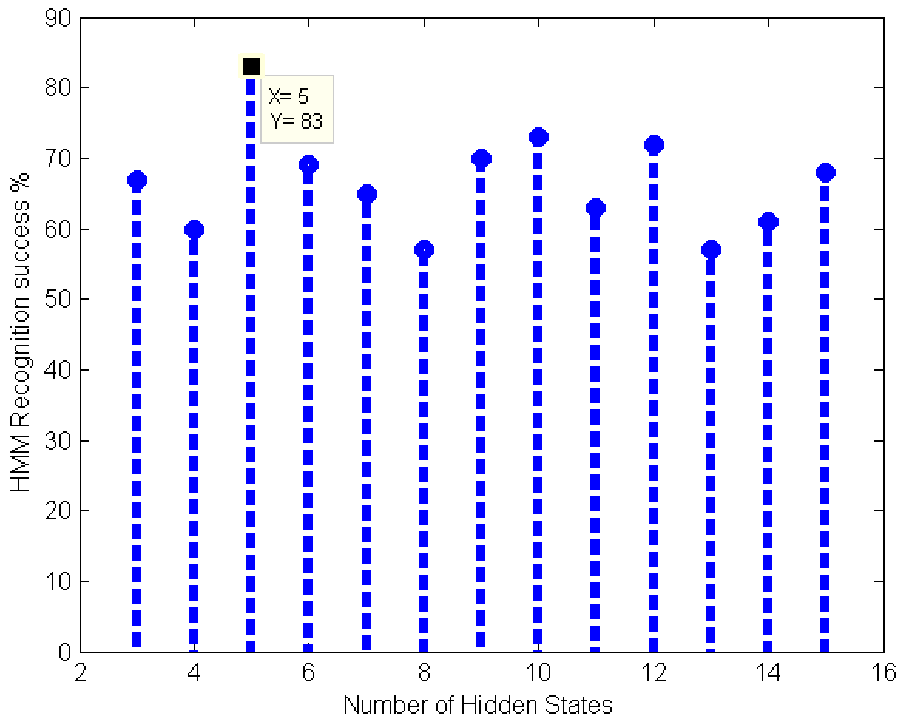


Figure 60: The recognition performance of HMM with different number of hidden states ranging from 3 to 15

parameters in order to fit the HMM model to training data as shown in Equation 18.

$$n_{HMM} \cong N_{states}^2 + d^2 \times N_{states} + d \times N_{states} \tag{18}$$

4.2.2.2 Dynamic Movement Primitives (DMP)

g is the goal state of point attractive system as seen in Equation 1 as described in Schaals paper [20]. g was taken as a constant the last position of trajectory in each trajectory. For the training data shown in Figure 56, 960 DMPs were trained and the absolute error between original trajectory and the reproduction of the movement by DMPs were calculated. DMP models were trained with different values ranging from 5000 to 100000 for K and 1 to 500 for D variable with 30 Gaussian basis functions. μ is mean (center) of the Gaussian basis functions. The centers of Gaussians were taken as equally seperated in values of phase variable s as described in Section 3.2.2 and Figure 43. The width of the Gaussians (h) were decided to be 5 at first by some experimentation. Such experimentation was necessary in order to find a general range of suitable variables at first. The resulting error matrix for different K and D values was shown in Figure 61. The lowest error value was gained at the K=51000 D=401 values in the given range with error of 0.01513. The replay error is calculated by the sum of absolute position errors in x and y coordinate DMPs. The number of Gaussians given intuitively at first. Since there are 150 data points, 30 Gaussians should be able to represent letter trajectories well enough. However this adjustable parameter is discussed further in next section 4.2.3.

After getting suitable K and D values from a local minimum as shown in Figure 61 for Gaussian width of 5, I tried different widths with the values K=51000 and D=401. The result for error vs width plot is shown in Figure 62.

We get h=3.5 as the newer lowest mean trajectory error point with the values

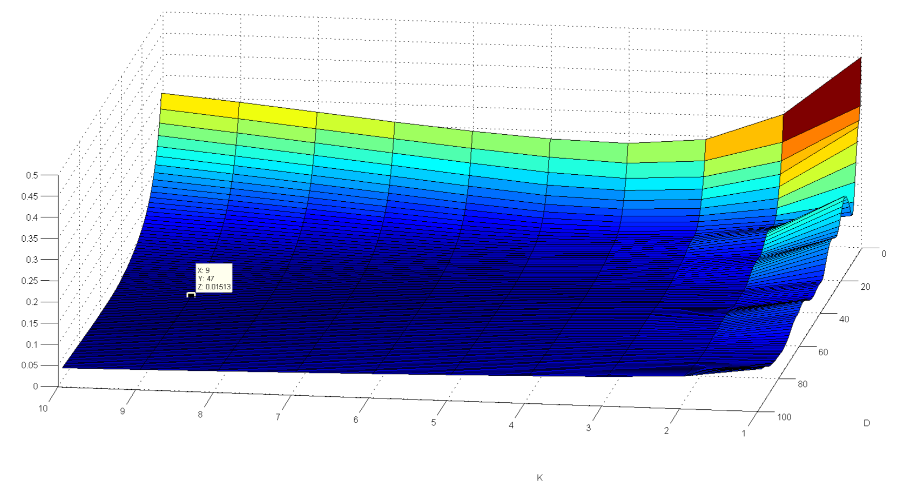


Figure 61: K-D vs Reproduction Error for Training Data h=5
30 Gaussians, Width of Gaussians=5

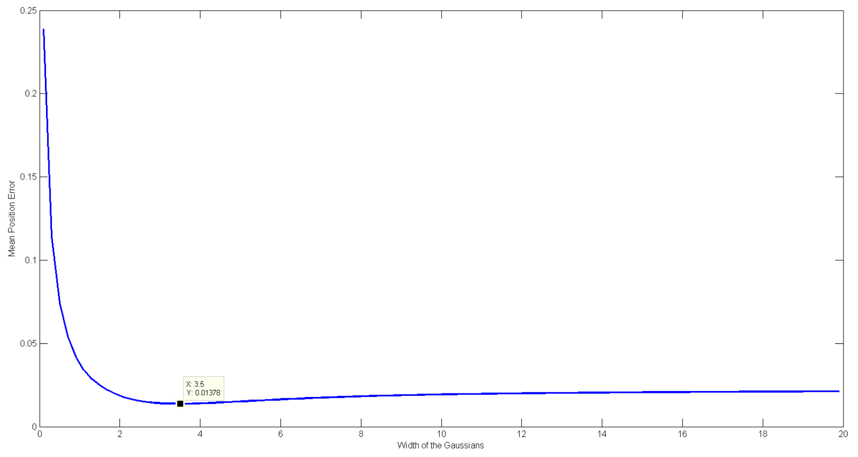


Figure 62: Mean Trajectory Error for Training Data h=0.1 to 20
30 Gaussians, K=51000 D=401

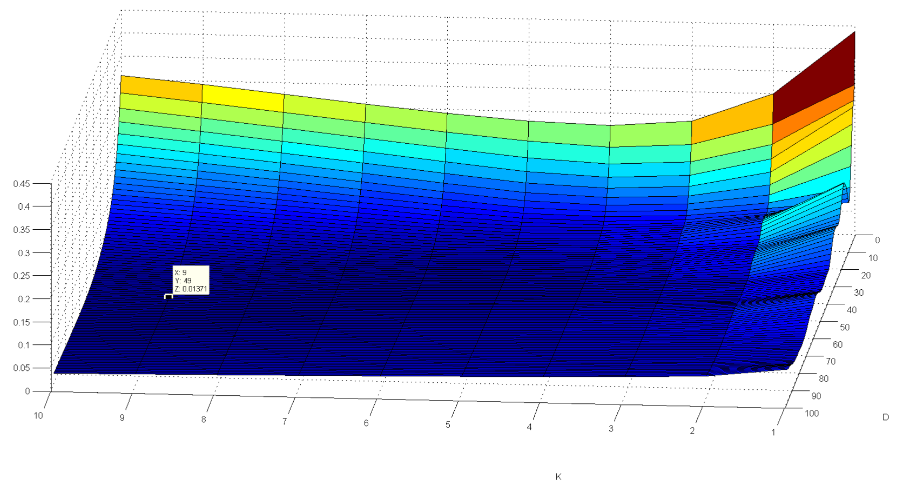


Figure 63: K-D vs Reproduction Error for Training Data h=3.5
30 Gaussians, Width of Gaussians=3.5

K=51000 and D=401. With the new width value, I tried different K and D values one more time to see if there is a better parameters to be selected then the current ones. The result of the K-D vs mean error plot is shown in Figure 63. Same value range was used as done in Figure 61.
The newer K-D vs mean error plot resulted in slightly different K and D values. The minimum error of 0.0137 which is slightly better than previous 0.01513 was obtained at the point K=53000 D=401. So I used K=53000 D=401 as the parameter for DMPs we used in the training and the tests for better represented training set trajectories.

The adjustable learned parameter $N_{Gaussians}$ can be used to calculate the learned parameter complexity approximately. The parameters other then $N_{Gaussians}$ are optimized once then fixed to a value. Therefore we can approximate the number of learned parameters in order to fit the DMP model to training data as shown in Equation 19. d (dimension) is 2 in our case.

$$n_{DMP} \cong N_{Gaussians} \times d \tag{19}$$

	DMP		HMM
$N_{gaussian}$	Complexity≅ Equation 19	N_{states}	Complexity≅ Equation 18
3	6	3	27
6	12	4	40
9	18	5	55
12	24	6	72
15	30	7	91
18	36	8	112
21	42	9	135
24	48	10	160
27	54	11	187
30	60	12	216
33	66	13	247
36	72	14	280
39	78	15	315

Table 1: Approximate learned parameter complexity

4.2.3 Learned Parameter Complexity

Using the approximation of learned parameter complexity of both models, we can derive learned parameter numbers as show in Table 1. As seen in Table 1, HMM have more learned parameter than DMP in general. DMP represents a recognition model in a smaller learned parameter memory namely only the weights of Gaussians in DMP.

4.3 Comparison of DMP and HMM in Recognition Tests

4.3.1 Training and Tests

4.3.1.1 DMP

Given the temporal and spatial invariance of our policy representation, trajectories that are topologically similar tend to be fit by similar weights, i.e., similar trajectories at different speeds and/or different amplitudes will result in similar weights. In our study we used a modified one nearest neighbor classifier on the weights space for movement recognition.

Two DMP models were trained for each trajectory. One for x coordinate, one for y coordinate of the writing trajectory. Each represents one dimensional equal-time-spaced data. By locally weighted regression we find the weights w_i of $N_{gaussian}$ Gaussians basis functions for each training data. In the current implementation, a modified one nearest-neighbor classifier was used. In a test instance we calculated the weight vector w_{test} for the test data. Then we find the nearest mean weight vector $w_{average}$ which were created during the training. $w_{average}$ is the weight vector that was calculated from training DMPs for the mean trajectories of training set. The closest match within scaled weights which is in nearest distance is selected as the recognized class and recorded. This scaling process in described in Section 4.3.1.2.

We can choose higher number of Gaussians in order to fit the DMPs to the training set. Thus, the DMPs for test reference will represent the training data better. The error in reproduction decreases as the number of Gaussians increase as expected. This is shown in Figure 64. The lowest error value was gained at 42 Gaussians in the given range of 10 to 50. Some training set reproduction examples to show how the reproduction fits better as the number of Gaussians increase is shown in Figure 65. We have tested how well the DMP fits training set until now. Since our aim in this chapter is to recognize test set using a training set with known classes, we should be looking at the test set recognition performances. In HMM parameter selection, we have done this using different number of states. Therefore a best performing number of states were chosen for the following test in next section with HMM. We can apply the same procedure for DMP. Here is a graph that shows the correct recognition performance of the DMP with the chosen parameters and different number of Gaussians in Figure 66. Other chosen parameters in the DMPs used in test are as follows: K=53000 D=401 h=3.5. As one can see from the Figure 66, DMP shows perfect recognition performance in recognition of the test set of 100 letters after 27 Gaussians. Even though more Gaussians result in better fit in training data, we have

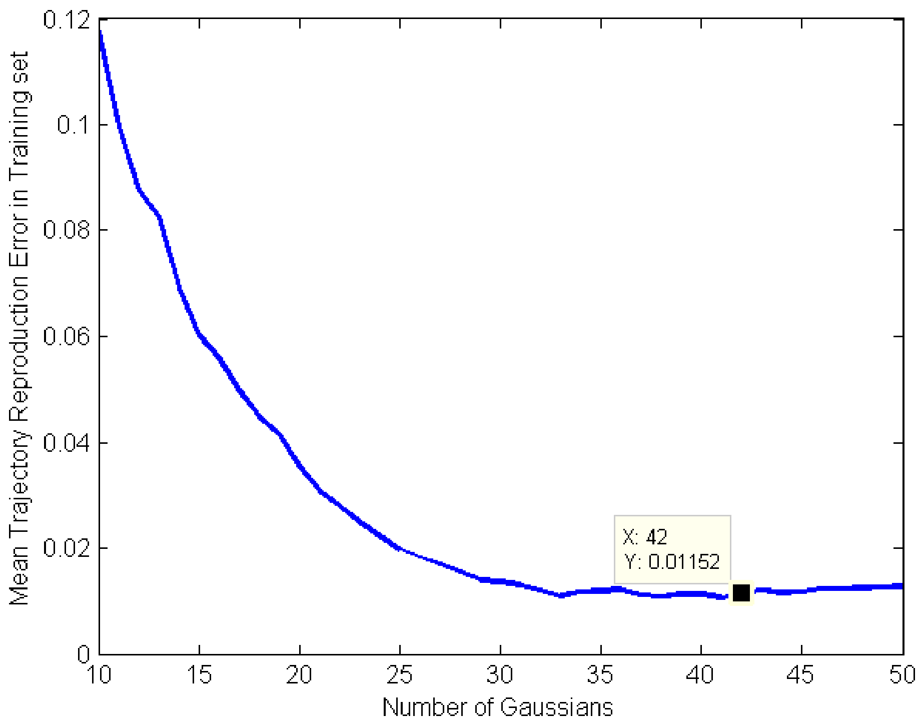


Figure 64: Mean Reproduction Error for Training Data
K=53000 D=401, Width of Gaussians=3.5

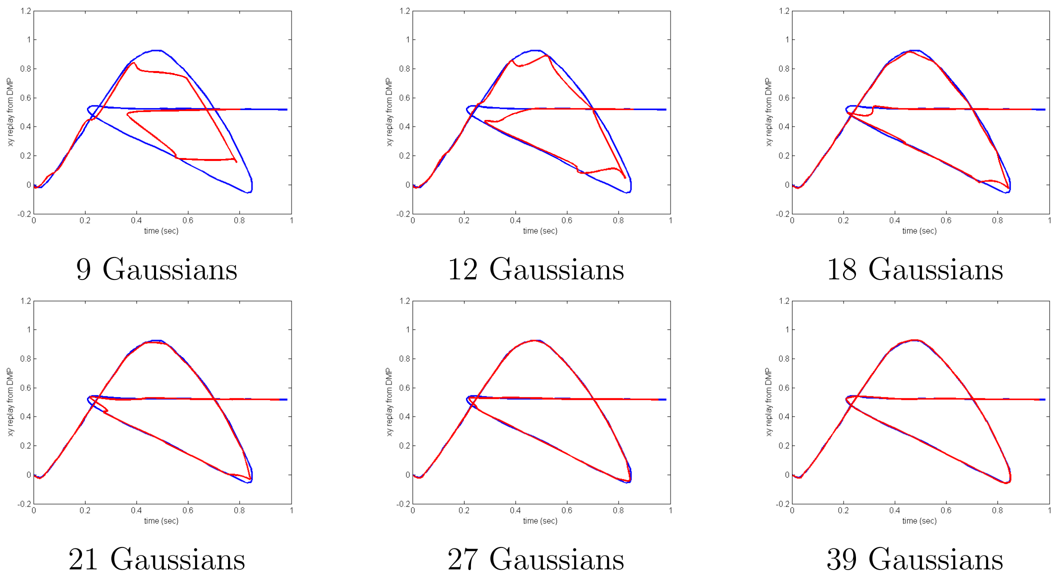


Figure 65: Reproduction Training DMPs with different number of Gaussians
Blue: Mean Training Trajectory
Red: Reproduction by DMP

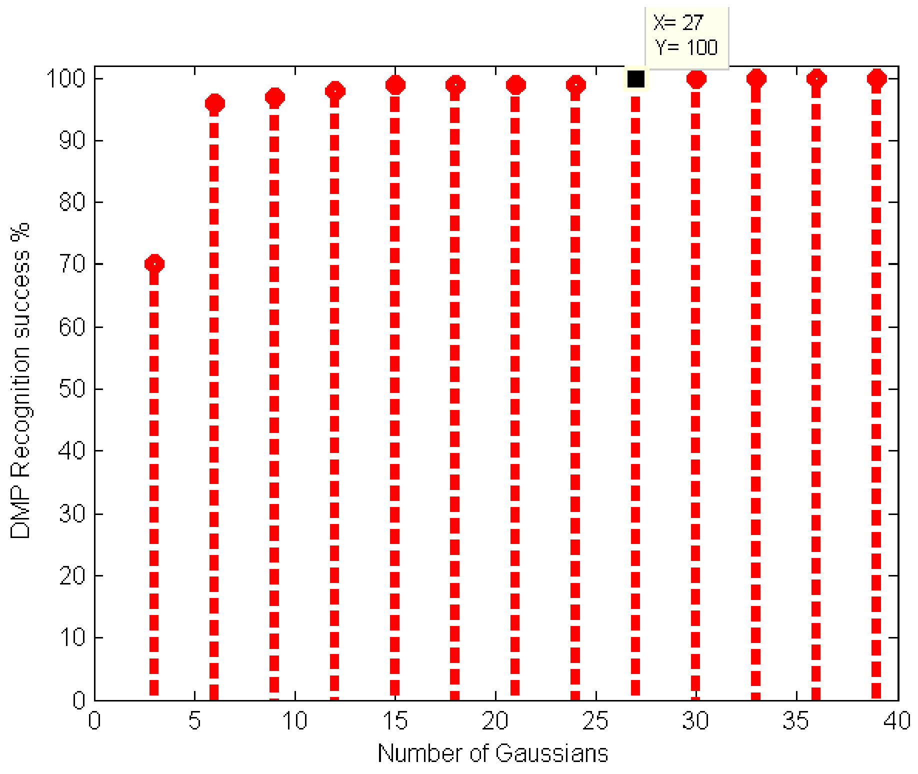


Figure 66: The recognition performance of DMP with different number of Gaussians ranging from 3 to 39

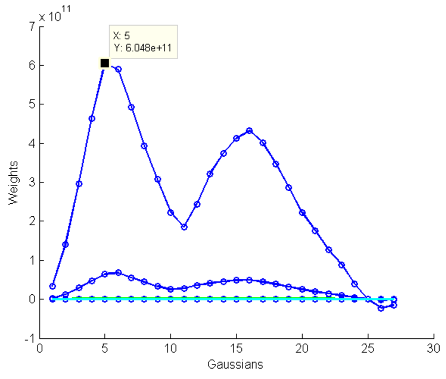


Figure 67: Weights of Gaussians of Test and Training set for letter M
Blue Dotted: Weights from 20 Test Data DMPs
Other Colors: Replay by training DMPs
27 Gaussians K=53000 D=401 h=3.5

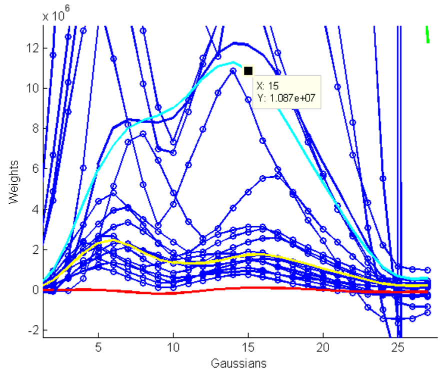


Figure 68: Zoomed in version of Figure 67

chosen 27 Gaussians for the tests in next section because it already gives 100% correct recognition with the original test data. Moreover the approximate learned parameter complexity of DMP (27 Gaussians) is closer the HMM chosen parameter (5 number of hidden states) as shown in Table 1. DMP has 54 approximate complexity with 27 Gaussians while HMM has 55 Complexity with 5 hidden states. Therefore, the tests will be done with a comparable learned parameter complexity.

4.3.1.2 Modified Distance Measure for Recognition

In DMP test, the recognition is done by comparing the weights. We have calculated distances between the Gaussian weights in the same order. The sum of total absolute distance between each Gaussian for 2 DMPs is used for as a distance measure. The closest distance is selected as recognized class. From 2 DMPs one is for x and one is for y axis of the trajectory. However, the comparison didn't result in good recognition performance. The weight that are compared with distances are shown in Figures 67 and 68. In Figure 67 we see that there are outliers in the test data weights. One example market in the figure has values as high as 6×10^{11} as weights. However if we get zoom in to the colored lines as shown in Figure 68, we can see that most of the

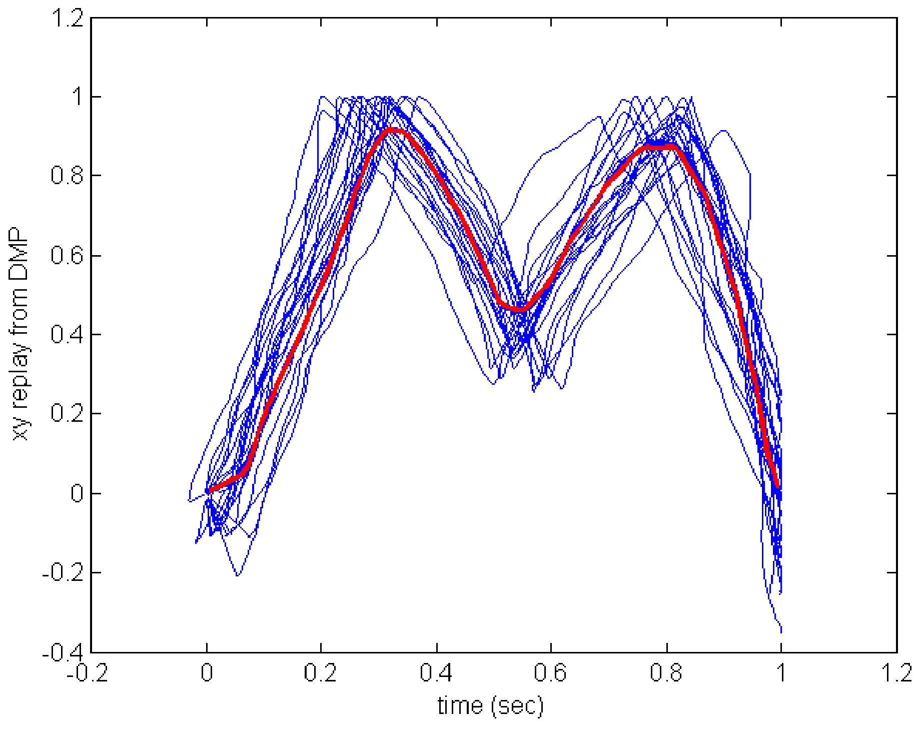


Figure 69: Training DMP trajectory reproduction over test data
K=53000 D=401, Width of Gaussians=3.5
Blue: Test Data
Red: Replay by training DMP for corresponding letter

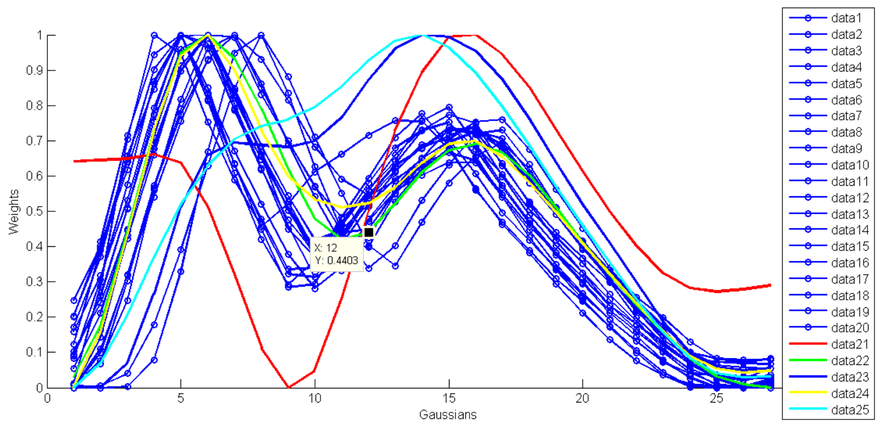


Figure 70: Scaled weights of Gaussians of Test and Training set for letter M
Blue Dotted: Weights from 20 Test Data DMPs
Other Colors: Replay by training DMPs
Data 1 to Data 20 in Legend is Test Weigths
Data 21 to Data 25 in Legend is Training Weigths
Data 22 Green Corresponds to the M letter
27 Gaussians K=53000 D=401 h=3.5

test weights and training weights have much lower values. Even though the training DMP's reproduction fits the test data well as seen in Figure 69, the differences in weights of test values are huge. So the distance comparison here only gives 2 correct recognition over 20 M letters.

In Figure 68, it can be observed that the plots of the consecutive Gaussian weights of test letters have similar shapes. This will allow us to recognize the letters better if we scale the weight and the huge number difference is gone. In order to achieve this we have scaled all the weights of test DMPs and the training DMPs to fit in the range of 0-to-1. The result can be seen in Figure 70. The test set's weights have similar shape to the correct training set weights (green). By this modified one nearest nearest-neighbor comparison we get 20/20 correct recognition performance.

4.3.1.3 HMM

The hidden Markov Model is a probabilistic model of the way in which the x_i and y_i sequences are generatedthat is, it is a representation of the joint distribution P(x,y).

It is defined by two probability distributions: the transition distribution $P(y_t \mid y_{t-1})$, which tells how adjacent y values are related, and the observation distribution P(x|y), which tells how the observed x values are related to the hidden y values. These distributions are assumed to be stationary (i.e., the same for all times t). We can get this distributions by using Expectation Maximization (EM) in several training data belonging to a single class. We can get the overall probability of a test data to be generated by this model using forward-backward algorithm. Then we can compare several models probabilities of generating the test data for classification. In our recognition task of HMM, we give 2 dimensional sequential training data as data to the pmtk3 tools model fit function which uses Expectation Maximization to find the model parameters. We are not using 2 models for 2 dimensions of data in HMM. We only use one model to represent a letter because HMM can map hidden states to multidimensional outputs. In recognition task, we use the models with these calculated parameters to guess most probable letter by testing each models log probability of the given observation sequence. The observation sequences probability is calculated by forward-backward procedure [11].

4.3.2 Results and Comparison

We have done tests in several data categories, original data, broken pen data, and noisy data. We applied our tests with the best performance models that are trained with the parameters chosen as described previous section 4.2.2. HMM models have 5 states, DMP models have variance of 0.125 and number of Gaussian of 9. Each test result consists of percentage of successful recognition of total of 100 handwritten letter movements. In addition to the recognition performance in original data tests, the process time of both algorithms in computer were measured. Time measurements were taken by Matlabs Run and Time function. The result are presented in the following Tables 2, 3, and 4.

	Successful recognition %	$N_{learned}$	Process Time (seconds)
DMP 27 Gaussians	100	54	8.096
HMM 5 States	83	55	4.996

Table 2: Original data test results over 100 test instances

DMP	HMM
Successful recognition %	Successful recognition %
99	82

Table 3: Broken Pen modification test results over 100 test instances

As seen in Table 2, the success rates in DMP higher than HMM on the noise-free the original data. DMP has 100% correct recognition rate whereas HMM has 83%. The number of learned parameters were approximately calculated as 54 in DMP and, 55 in HMM. Both methods use comparable learned parameter complexity, so that their comparison will be more fair. In a computer with 2.4 ghz i7 4700hq (Intel Corp.) processor with Windows 8.1 (Microsoft Corp.), the time it takes for DMP to finish recognition test of 100 trajectory data is 8.096 second, where as HMM takes only 4.996 seconds for the same task. This result can easily be understood since the DMP needs to do linear regression each time it processes a trajectory in order to recognize. Recognition by DMP is utilized by comparing the weights that are calculated by locally weighted regression procedure as described in Ijspeerts paper [10] as well as in Section 3.2.

However HMM is utilizing a more efficient way to recognize new data. The forward-backward algorithm is an inference algorithm for HMM which computes the posterior marginals of all hidden state variables given a sequence of observations/emissions.

	DMP		HMM
SNR	Successful recognition %	SNR	Successful recognition %
20	99	20	75
25	100	25	80
30	100	30	81
35	99	35	83

Table 4: Noisy data test results over 100 test instances

The brute-force procedure for generating all possible N^T hidden state sequences and calculating the joint probability of each state sequence with the observed data. This approach has time complexity $O(T.N^T)$, where T is the length of sequences and N is the number of symbols in the state alphabet. This time complexity becomes very high in most cases [11]. However, the forwardbackward algorithm has time complexity $O(N^2.T)$. In our HMM recognition task, we use forward-backward algorithm in order to calculate the observation sequences probability in every model then we compare these probabilities in order to categorize. Since the time complexity of forward-backward procedure is low we are able to get fast results compared to DMP which applies a linear regression for every test data during recognition task.

In Table 3, the success rates with the missing data tests are given. In this modification randomly selected 2 parts each 5% of the sequential data are deleted including the indexes they correspond to. Therefore, the sequential data continues in this cut regions with jumps meaning that the data continues from further data points in one time step later. Later the modified data is scaled to the original data length. Broken pen modifications intention was to see how the recognition is affected in case of some sequential data go missing. We see that HMM nearly same with 83% original data recognition success (82% in broken pen). DMP has dropped only 1% from the results of recognition of original data (100%). As in the original data results, DMP is giving much higher results than HMM. Comparing the two methods results, both methods are giving better results with missing data compared to their original data results.

Figure 59 shows an example noisy data used in these tests. Noisy data modification consists of applying white noise with 4 levels of SNR (signal to noise ratio) to the both coordinate axis of the 2D movement data. If SNR is lower the noise level is higher. From the data in Table 4, we can see that DMP didnt change much from original data recognition success. DMP has dropped 1% from the original data success in recognition of noisy data with 20 SNR and 35 SNR. In the same case (20 SNR) HMM

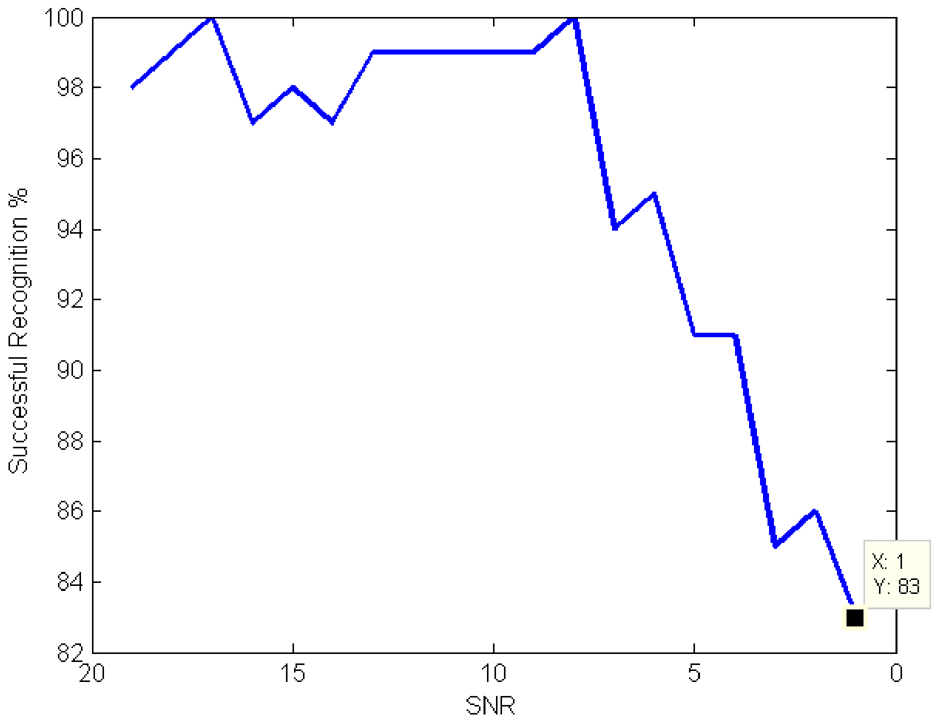


Figure 71: Noisy Data Test With DMP from 19 to 1 SNR
K=53000 D=401, Width of Gaussians=3.5 Number of Gaussians=27

has dropped 21% from original data recognition success. It can be said that DMP was not affected by noise in the data and had successful recognition results in this case. HMM catched up on the original data results in 35 SNR value which has less noise on the data.

Since DMP had very good results in noisy data, we did further test with lower SNR, meaning more noise. The results are plotted in Figure 71. As seen in Figure 71 the as low as 83 at 1 SNR.

CHAPTER V

CONCLUSION

5.1 Summary

We have implemented a movement recognition method based on DMPs with Gaussians centered equally spaced in phase variable and scaled one-nearest-neighbor weight comparison. With the series of tests we made, we have observed that DMP gives 100% success compared to 83% success of recognition with HMM in the test on the non-modified, noiseless human movement data. In general, HMM was much faster than DMP in same recognition task. Furthermore, in tests with human movement data which we modified to emulate data loss during recording, DMP and HMM gave similar results compared to their original data recognition performances. HMM changes little from noiseless original data recognition rate, DMP doesn't even drop more than 1% from previous performance. In noise contaminated data experiments with 4 levels of noise, we have seen that DMP based recognition is more robust and have a slight advantage over HMM in terms of catching up with noiseless data recognition success rate. Our work not only gives an initial comparison of these methods on the same task, but it also underlines the need for more critical tests to assess the performances of two recognition systems. In particular, it would be very valuable to know which tasks and what kind of noise distributions render one method superior to the other. The results indicate that, DMP gives superior correct recognition performance on the given human movement data even compared to generally accepted time series data recognition method HMM in the tests with noisy data and derogated data. Thus, using DMPs with Gaussians equally distanced in phase variable and scaled one-nearest-neighbor weight comparison is a successful method in tasks of human movement recognition.

APPENDIX A

LOCALLY WEIGHTED LEARNING

In most regression problems, a global model is fitted to the training data such that the model can predict the data. After training process, usually the training data is not stored. However, in memory based methods in other words lazy learning methods, we use the training data at the time of query in order to predict the values. So the training data needs to be stored and used whenever a prediction about a point is to be made [27].

Locally weighted regression (LWR) is a memory-based method that performs a regression around a point of interest using only training data that are local to that point. Global regression can be expressed as:

$$x_i^T \beta = y_i \tag{20}$$

with the vectors $x_i \in \mathbb{R}^{k+1}$, $x_i = (1, x_{i1} \cdots, x_{ik})^T$ and the parameters $\beta \in \mathbb{R}^{k+1}$, $\beta = (\beta_{i1} \cdots \beta_{ik})^T$ or in matrix form as:

$$X\beta = y \tag{21}$$

with the vectors x_i as rows of $X \in \mathbb{R}^{n \times (k+1)}$,

$$X = \begin{bmatrix} -x_1^T- \\ \vdots \\ \vdots \\ -x_n^T- \end{bmatrix} \tag{22}$$

Estimation of the parameters β can be done with the least squares method which minimize the unweighted error criterion:

$$E = \sum_i \left(x_i^T \beta - y_i\right)^2 \tag{23}$$

solving the normal equations for β leads to:

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y \tag{24}$$

The resulting β tries to fit all points in the data set by a linear function. In order to turn the global model into a local model, the distance between query point q and each point x_i is taken into account:

$$E(q) = \sum_i \left(x_i^T \beta - y_i\right)^2 \psi_i(q) \tag{25}$$

where ψ denotes a kernel function, for example a squared exponential function $\psi_i(q) = exp(-d(x_i, q)^2)$. This function weights each data point by the distance between the query point q and x_i. If the distance is small the value of ψ is close to 1 which results in a higher impact on the output values. A greater distance of the points leads to a weight close to 0 which minimizes the effect of far distant points. The parameters β which minimizes the error criterion E(q) are given by [26]:

$$\hat{\beta} = \left(X^T W X\right)^{-1} X^T W y \tag{26}$$

where $W \in \mathbb{R}^{n \times n}$ is a diagonal matrix with elements $W_{ii} = \psi_i(q)$. A prediction of $\hat{y}(q)$ is calculated by:

$$\hat{y}(q) = q^T (X^T W X)^{-1} X^T W_y \tag{27}$$

Note, many different distance functions d as well as kernel functions ψ can be used with this method. For prediction each parameter $\hat{\beta}^l$ of the linear model from Equation 26 is weighted with the kernel functions ψ_i. Therefore prediction of a query point q is given by:

$$\hat{y}(q) = \frac{\sum_i^r \psi_i(q) q^T \hat{\beta}^i}{\sum_i^r \psi_i(q)} \tag{28}$$

A survey of Locally Weighted Learning can be found in [13].

APPENDIX B

HIDDEN MARKOV MODEL

HMM is a mathematical model of a stochastic process and includes three parameters $\lambda = (\Pi, A, B)$ where Π represents initial state vector, A is the transition matrix and B refers to emission matrix.

HMMs can be used for representing probability distributions over sequences of observations. Let us denote the observation at time t by the variable Y_i. In HMMs, it is assumed that the observation at time t was generated by some process whose state S_t is hidden from the observer. HMM also assumes that the state of the hidden process satisfies the Markov Property, that is given the value of S_{t-1}, the current state S_t is independent of all states prior to t-1, In other words, the state at some time includes the history of process inside. So, in principle we can predict the next state given the current state. The output also satisfies the Markov Property.

Taken together these Markov Properties mean that the joint distribution of sequence of states and observations can be found in the following way:

$$P(S_{1:T}, Y_{1:T}) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t) \tag{29}$$

To define a probability distribution over sequences of observations, we need a probability distribution over initial states $P(S_t)$, in the $K \times K$ state transition matrix defining $P(S_t|S_{t-1})$ and the output model defining $P(Y_t|S_t)$. The initial probability distributions can be randomly initialized. The state transition and output matrix or in other words emission matrix are found from the training data with Expectation Maximization algorithm [14]. In its discrete form, a hidden Markov process can be visualized as a generalization of the Urn problem with replacement (where each item from the urn is returned to the original urn before the next step) [11]. Consider this

example: in a room that is not visible to an observer there is a genie. The room contains urns X1, X2, X3, ... each of which contains a known mix of balls, each ball labeled y1, y2, y3, ... The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn. The genie has some procedure to choose urns; the choice of the urn for the n-th ball depends only upon a random number and the choice of the urn for the (n1)-th ball. The choice of urn does not directly depend on the urns chosen before this single previous urn; therefore, this is called a Markov process.
The Markov process itself cannot be observed, and only the sequence of labelled balls can be observed, thus this arrangement is called a "hidden Markov process". Lets say only balls y1, y2, y3, y4 can be drawn at each state. Even if the observer knows the composition of the urns and has just observed a sequence of three balls, e.g. y1, y2 and y3 on the conveyor belt, the observer still cannot be sure which urn (i.e., at which state) the genie has drawn the third ball from. However, the observer can work out other information, such as the likelihood that the third ball came from each of the urns.
In HMM there are 3 main problems to be solved in order the model to be useful in real world applications. These problems are the following [11]:

1. Given the observation sequence $O = O_1 O_2 .. O_T$ and a model $\lambda = (\Pi, A, B)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

2. Given the observation sequence $O = O_1 O_2 .. O_T$ and a model λ, how do we choose a corresponding state sequence $Q = q_1 q_2 .. q_T$ which is optimal in some meaningful sense (i.e. best explains the observations)?

3. How do we adjust the model parameters $\lambda = (\Pi, A, B)$ to maximize $P(O|\lambda)$?

First problem is to calculate the probability of an observed sequence. The most straight forward way of doing this is through enumerating every possible state sequence of same length with data. However this approach is not efficient as described in Rabiners paper. This problem can be solved more efficiently by forward-backward procedure [11]. We use this probability to compare how well a model fits the given observation. Thus we decide class of that test data belongs [15].
Second problem is to find the correct state sequence. It should be clear that for all but the case of degenerate models, there is no correct state sequence to be found. However, for practical reasons we use an optimality criterion to solve this problem as best as possible. Viterbi algorithm is used in order to find the best possible state sequence in this case [11].
Third problem is about training the model. As described above, the parameters of the model are locally maximized using Expectation Maximization or Baum-Welch algorithms iteratively [11].

Bibliography

[1] Nemec, Bojan, and Ale Ude. "Action sequencing using dynamic movement primitives." Robotica 30.05 (2012): 837-846.

[2] Ijspeert, Auke Jan, Jun Nakanishi, and Stefan Schaal. "Movement imitation with nonlinear dynamical systems in humanoid robots." Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Vol. 2. IEEE, 2002.

[3] Ijspeert, Auke Jan, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. No. BIOROB-CONF-2002-004. 2002.

[4] Meier, Franziska, et al. "Movement segmentation using a primitive library."Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[5] Godoy, Vinicius, et al. "An HMM-based Gesture Recognition Method Trained on Few Samples." Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on. IEEE, 2014.

[6] Gales, Mark, and Steve Young. "The application of hidden Markov models in speech recognition." Foundations and Trends in Signal Processing 1.3 (2008): 195-304.

[7] Starner, Thad, and Alex Pentland. "Real-time american sign language recognition from video using hidden markov models." Motion-Based Recognition. Springer Netherlands, 1997. 227-243.

[8] Hu, Jianying, Michael K. Brown, and William Turin. “HMM based online handwriting recognition.” Pattern Analysis and Machine Intelligence, IEEE Transactions on 18.10 (1996): 1039-1045.

[9] A. Ijspeert, J. Nakanishi, and S. Schaal, “Learning attractor landscapes for learning motor primitives, Advances in neural information processing systems, pp. 15471554, 2003.

[10] Ijspeert, Auke Jan, et al. “Dynamical movement primitives: learning attractor models for motor behaviors.” Neural computation 25.2 (2013): 328-373.

[11] Rabiner, Lawrence. “A tutorial on hidden Markov models and selected applications in speech recognition.” Proceedings of the IEEE 77.2 (1989): 257-286.

[12] Yang, Jie, and Yangsheng Xu. Hidden markov model for gesture recognition. No. CMU-RI-TR-94-10. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1994.

[13] Atkeson, Christopher G., Andrew W. Moore, and Stefan Schaal. “Locally weighted learning for control.” Lazy learning. Springer Netherlands, 1997. 75-113.

[14] Maya R. Gupta and Yihua Chen (2011), “Theory and Use of the EM Algorithm”, Foundations and Trends in Signal Processing: Vol. 4: No. 3, pp 223-296. http://dx.doi.org/10.1561/2000000034

[15] Ghahramani, Zoubin. “An introduction to hidden Markov models and Bayesian networks.” International Journal of Pattern Recognition and Artificial Intelligence 15.01 (2001): 9-42.

[16] Dunham, M., and K. Murphy. “PMTK3: Probabilistic modeling toolkit for Matlab/Octave, version 3.” 2012 (2012).

[17] Hogan, Neville, and Dagmar Sternad. “Dynamic primitives of motor behavior.”Biological cybernetics 106.11-12 (2012): 727-739.

[18] Rouchka, Eric C. “Pattern Matching Techniques and Their Applications to Computational Molecular Biology-A Review.” (1999).

[19] Colom, Adria, and Carme Torras. “Dimensionality Reduction and Motion Coordination in Learning Trajectories with Dynamic Movement Primitives.” Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014.

[20] Schaal, S., et al. “Control, Planning, Learning, and Imitation with Dynamic Movement Primitives.” Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE International Conference on Intelligent Robots and Systems. 2003.

[21] Vicente, Isabel Serrano, et al. “Action recognition and understanding through motor primitives.” Advanced Robotics 21.15 (2007): 1687-1707.

[22] Newtson, Darren, Gretchen A. Engquist, and Joyce Bois. “The objective basis of behavior units.” Journal of Personality and social psychology 35.12 (1977): 847.

[23] Forte, Denis, Ale Ude, and Andrej Kos. “Robot learning by Gaussian process regression.” Robotics in Alpe-Adria-Danube Region (RAAD), 2010 IEEE 19th International Workshop on. IEEE, 2010.

[24] Prada, Miguel, and Anthony Remazeilles. “Dynamic Movement Primitives for Human Robot Interaction.” IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, workshop on Robot Motion Planning: online, reactive and in Real-time, Algarve, Portugal. 2012.

[25] Ijspeert, Auke Jan, et al. ‘Nonlinear dynamical systems for imitation with humanoid robots.” Proceedings of the IEEE/RAS International Conference on Humanoids Robots (Humanoids2001). No. BIOROB-CONF-2001-001. 2001.

[26] Glatz, Karl. “Adaptive Learning from Demonstration using Dynamic Movement Primitives.” (2012).

[27] Schaal, Stefan, and Christopher G. Atkeson. “Robot juggling: implementation of memory-based learning.” Control Systems, IEEE 14.1 (1994): 57-71.

[28] Schaal, Stefan. “Dynamic movement primitives-a framework for motor control in humans and humanoid robotics.” Adaptive Motion of Animals and Machines. Springer Tokyo, 2006. 261-280.

[29] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, “Learning Movement Primitives,” in International Symposium on Robotics Research (ISRR2003), Springer Tracts in Advanced Robotics. Ciena, Italy: Springer, 2004.

[30] Ude, Ale, et al. “Task-specific generalization of discrete and periodic dynamic movement primitives.” Robotics, IEEE Transactions on 26.5 (2010): 800-815.

[31] Viviani, Paolo, and Tamar Flash. “Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning.” Journal of Experimental Psychology: Human Perception and Performance 21.1 (1995): 32.

[32] Stulp, Freek, et al. “Compact models of motor primitive variations for predictable reaching and obstacle avoidance.” Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on. IEEE, 2009.

[33] Akgn, Baris, Doruk Tunaoglu, and Erol Sahin. “Action recognition through an action generation mechanism.” International Conference on Epigenetic Robotics (EPIROB). 2010.

[34] Kober, Jens, et al. “Reinforcement learning to adjust parametrized motor primitives to new situations.” Autonomous Robots 33.4 (2012): 361-379.

[35] Kober, Jens, Erhan Oztop, and Jan Peters. “Reinforcement learning to adjust robot movements to new situations.” IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No. 3. 2011.

[36] Lee, Hyeon-Kyu, and Jin-Hyung Kim. “Gesture spotting from continuous hand motion.” Pattern recognition letters 19.5 (1998): 513-520.

[37] Kulvicius, Tomas, et al. “Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting.” Robotics, IEEE Transactions on 28.1 (2012): 145-157.

[38] Zhang, Mi, and Alexander A. Sawchuk. “Motion primitive-based human activity recognition using a bag-of-features approach.” Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium. ACM, 2012.

[39] Schaal, Stefan. “Movement planning and imitation by shaping nonlinear attractors.” Proceedings of the 12th Yale workshop on adaptive and learning systems. 2003.