LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
FACULTY OF TECHNOLOGY MANAGEMENT
INFORMATION TECHNOLOGY

LEARNING ROBOT MOTIONS FROM VISUAL SENSING

Master's thesis

Supervisor: Professor Ville Kyrki

Examiners: Professor Ville Kyrki, Sergey Rybin D.Sc. (Tech.)

Lappeenranta, May 30th, 2010

Nataliya Strokina
Punkkerikatu 2 A 7
53850 Lappeenranta
Tel. 0465790841
nataliya.strokina@lut.fi

ABSTRACT

Lappeenranta University of Technology

Faculty of Technology Management

Information Technology

Nataliya Strokina

Learning Robot Motions from Visual Sensing

Master's thesis

2010

68 pages, 44 figures, 11 tables.

Examiners: Professor Ville Kyrki
Sergey Rybin D.Sc. (Tech.)

Keywords: Imitation Learning, Trajectory Replication, Motion Learning, Dynamic Movement Primitives, Visual Sensing, Movement Tracking, Machine Vision

Learning from demonstration becomes increasingly popular as an efficient way of robot programming. Not only a scientific interest acts as an inspiration in this case but also the possibility of producing the machines that would find application in different areas of life: robots helping with daily routine at home, high performance automata in industries or friendly toys for children. One way to teach a robot to fulfill complex tasks is to start with simple training exercises, combining them to form more difficult behavior.

The objective of the Master's thesis work was to study robot programming with visual input. Dynamic movement primitives (DMPs) were chosen as a tool for motion learning and generation. Assuming a movement to be a spring system influenced by an external force, making this system move, DMPs represent the motion as a set of non-linear differential equations.

During the experiments the properties of DMP, such as temporal and spacial invariance, were examined. The effect of the DMP parameters, including spring coefficient, damping factor, temporal scaling, on the trajectory generated were studied.

PREFACE

First of all, I would want to thank my supervisor, Professor Ville Kyrki, for giving me an opportunity to work on this topic, for his guidance, advice and encouragement.

Thanks a lot to Jarmo Ilonen who was helping me with the stereo camera.

Thanks to Ivan Martynov and Janne Laaksonen for their help and support.

Thank you!

Lappeenranta, May 30th, 2010

Nataliya Strokina

CONTENTS

ABBREVIATIONS AND SYMBOLS

DMP	dynamic movement primitive
K	spring coefficient for the equation of motion
D	damping factor for the equation of motion
τ	temporal scaling parameter
φ_i	Gaussian basis function
M	number of Gaussian basis functions
w	width of Gaussian basis functions

1 INTRODUCTION

The section provides the motivation and the main objectives of the work, as well as the structure of the thesis.

1.1 Motivation

Recent advances of technological progress have made it possible to replace people with robots in some areas of life, for example, in industry, services sector and medicine. This has enabled manufactures to reduce their staff and increase the speed of work; medical operations are now performed more accurately, and, on the whole, such a replacement diminished the influence of human factors on work. A person can get tired and loose concentration, which increases the number of mistakes. A robot has a predictable behavior, the errors might be either prevented or taken into account.

However, before a robot can be employed it should learn how to perform the operations. This challenging problem can be solved by learning from visual sensing. In this case a person has to demonstrate a motion to the robot, that later will be able to replicate it. The question is how to make a robot to see what should be done, to learn how to do it and to perform an operation.

1.2 Objectives and Restrictions

The first objective of the research is to study the way of programming a robot with visual input. A robot should be able to learn and memorize the motion, demonstrated with a pointer, and be able to replicate it. The questions are: how to choose a pointer that will be easy to distinguish, how to detect the pointer and to determine the demonstrated trajectory in three dimensions. Dynamic movement primitives (DMPs) model was chosen for motion learning and generation.

The second objective is to study the DMPs concept and examine how the DMP parameters affect the trajectory generated. It is necessary to determine what kind of trajectories could be learnt with this method and to find out whether it is possible to single out parameter values that would allow fitting for a large set of motions.

The third objective is to study how the properties of DMPs, such as temporal and space invariance, are fulfilled.

The trajectories will be considered in Cartesian system of coordinates, the robot should start from the initial position and its motion should converge to the goal position. The orientation and direction of the motion are not taken into account. The motion generated should not be scaled unless otherwise specified. During the research the following high level hardware should be used: stereo camera and robot arm. Programming languages are MatLab, for motion detection and learning, and C++, for motion generation and robot control.

1.3 Structure of the Thesis

The rest part of the thesis has the following structure: Section 2 considers the issues of the trajectory imitation learning paying special attention to the application of dynamic movement primitives. Section 3 represents the components of the system of motion learning and generation from visual sensing. Section 4 considers the importance of visual sensing, including the problems of pointer detection and 3D reconstruction. Section 5 describes the issues of the dynamic movement primitives implementation. Section 6 includes the basics of robot kinematics and introduces the way the robot is controlled. Section 7 is intended for presenting and discussing the results of the experiments conducted. The conclusion of the Master's thesis work is provided in Section 8.

2 TRAJECTORY IMITATION WITH DYNAMIC MOVEMENT PRIMITIVES

2.1 Imitation Learning for Robots

On one hand, robot learning is inspired by the scientific interest, researchers are trying to find out how far a human can go in the simulation of intelligence. On the other hand, researchers are interested in producing machines that would have practical use for ordinary people at home, at their working places, in industry, increasing the performance of the operations and decreasing the human factor.

Imitation is one of the most effective and simplest mechanism of learning for robots as well as for human beings. In imitation learning the robot restores the information about the motion directly from what a human has showed [1] and forms its behavior. In traditional robot programming a great attention was payed to the accuracy of manual modeling of the performance. In [2] the example of an animatronic device is showed; the machine was replaying the motions that were recorded by manually putting the device in the sequence of positions that could be combined as a motion. In addition to the time consuming teaching process, those machines were non-interactive [2] which means that they were not able to cope with the difficulties of the real world, such as obstacles or new conditions. Another approach, reinforcement learning [1], allows a robot not only to learn a motion but to improve performance using a reward function. The system learns a policy by interaction with the environment and correcting its own behavior. Such systems are autonomous, but time-consuming for a robot to learn.

Learning from demonstration technique has been presented in detail by Schaal in [3]. The main idea is to show a robot a motion, which it should learn and be able to replicate under new circumstances or with new goals. One of the key advantages of the approach is that only few demonstrations are needed for a robot to learn a motion successfully. In [1] the examples of areas, appropriate for imitation learning, are given, among which are learning racket sports, manipulation, drumming on anthropomorphic systems. Complex policies can be constructed out of primitive motions. In this case the system is much more predictable, since all the smaller motions can be tested and provide the expected result. Imitation learning from visual sensing creates a model of a motion, which might be used for motions recognition. Although it is not difficult to get kinematic information about the demonstrated motion, the replication with stable robot dynamics might be a complicated task [4]. Another shortcoming is that there is no evidence why the derived policy for behavior is a good one [1].

2.2 Dynamic Movement Primitives

There are three main challenges that imitation learning methods should take into account [5]. First of all, the assumption should be made that a human and a robot does not necessarily have the same links and joints that are used in a particular movement. Second, generalization to new types of movements should be handled, since it is impossible to demonstrate to the robot all the possible variants of movements. Finally, a method should be able to react adequately to changes in the environment that might affect the system.

The above mentioned issues are addressed in the model of dynamic movement primitives, that was proposed by Schaal et al. [5], [6], [7] and has been shown to be applicable in robotics for imitation and skill learning in, for example, gestural interaction with mobile computers [8], learning maneuvers in flight control [9] and humanoid robot control [10].

Assuming that motion is a spring system influenced by an external force making this system move, DMPs represent the motion as a set of non-linear differential equations. DMP consists of the canonical and transformation systems of equations [5]. Equations (1) and (2) introduce the transformation system:

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

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

where x and v are the position and velocity of the system correspondingly, x_0 is the starting position, g the goal position; τ a temporal constant, K a spring constant, and D a damping factor, which is used to reduce the amplitude of oscillations. Non-linear function

$$f(s) = \frac{\sum \omega_i \varphi_i(s)s}{\sum \varphi_i(s)} \tag{3}$$

can be thought of as a set of Gaussian basis functions and enables the system to describe the arbitrarily complex movements [10]. $\varphi_i(s) = exp(-h_i(s - c_i)^2)$ is a Gaussian basis function with center c_i and width h_i. The adjustable weights w_i determine the generated motion. The transformation system is a second order dynamic linear system, since the non-linear function f does not have any modulation effect on it. Thus the proper choice of spring and damping parameters can guarantee that the system will eventually converge to the goal position [10].

The canonical system, represented by the equation 4 , aims at generating a phase variable s, that the non-linear function depends on.

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

where α is a predefined constant. The reason for using the phase instead of time is that one can manipulate the time evolution of phase by additive coupling terms or phase resetting [10]. Phase is initially set to 1. With time s approaches zero, which decreases the influence of function f and the equation (1) becomes linear.

The equations introduced describe the motion only in one dimension. In multi-dimensional space motion in each dimension should be described by one-dimensional equation, and for each dimension a set of weights, describing a motion, is obtained. The phase variable used in several dimensions can be the same.

2.3 Properties and Restrictions

The following advantages of DMP are mentioned in the literature [11], [6], [10]:

1. Spatial invariance includes online adaptation to the new goal position, invariance to the change of the amplitude of the motion, translation invariance.
2. Temporal invariance implies that the duration of the motion can be scaled varying the temporal parameter τ.
3. Robustness against perturbations. In real world systems a robot may encounter unexpected obstacles on its route and DMP provides online modification of the

trajectory that allows robot to avoid obstacles. During the perturbation the run of the generated trajectory is paused and the system tries to avoid obstacle keeping relatively close to the initial direction, which is possible due to the feeding back of the error between the actual and planned trajectory. Once the perturbation is over the system continues to follow the generated trajectory.

4. Movement Recognition. Due to the fact that similar motions have the same weight coefficients although the speed, the duration and the amplitudes might differ, one can create a library of movements and recognize which of them was shown. In other words, DMPs can be used to solve the opposite problem, called motion recognition.

5. Superposition of DMPs. Combining in time DMPs to generate different movements, one could model not only the control policy for a single movement but also to create behavioral policies that make the robot to fulfill the sequence of the motions.

However, the DMP concept has several shortcomings [11], [5].

1. If the initial and the goal positions are too close to each other, the canonical equation is hardly able to drive a system.

2. If the goal position does not differ enough from the initial one, a small change in goal position can lead to extremely high acceleration and exceed the limits of the robot.

3. Whenever $(g - x_0)$, where g is a goal position and x_0 a start position, changes its sign the generalization is mirrored.

The next section describes the approach of overcoming the above mentioned drawbacks.

2.4 Modified DMP

The modified version of DMP, described in [11], aims at curing the drawbacks of the traditional method. The following neurophysiologycal findings [12] of spinal force fields in frog appeared to become a motivation for the new approach:

1. the force fields, measured at different leg positions of a frog after stimulation, are usually convergent;
2. bell-shaped time pulses modulates the magnitude of the force field;
3. simultaneously stimulated force fields add up linearly.

These facts imply that following a certain trajectory different force fields should be activated during the motion. However, in modified DMP approach an acceleration field rather than a force field is employed. In [11] the derivation of the modified equations of motion is shown.

Assuming that there are no external forces influencing the system, acceleration field space in three dimensions could be introduced as linear fields, equation (5) centered at w_i:

$$a_i(x, v) = K(w_i - x) - Dv, \tag{5}$$

where x and v are position and velocity correspondingly. Given that each field is modulated with a Gaussian basis function $\varphi_i(s) = exp(-h_i(s - c_i)^2)$, centered at c_i, more complex equation (6) of acceleration field can be obtained.

$$a_i(x, v, t) = \frac{\sum \varphi_i(t) a_i(x, v)}{\sum \varphi_i(t)}. \tag{6}$$

Substituting (5) in (6) and adding another linear field around g for convergence, the equation (7) can be obtained.

$$\dot{v} = sK(\frac{\sum \varphi_i(s) w_i}{\sum \varphi_i(s)} + x_0 - x) + (1 - s)K(g - x) - Dv, \tag{7}$$

where time is changed for phase which also plays role of weights. x_0 is added to make the motion translation invariant. The final set of equations describing the motion is introduced in the equations (8), (9)

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) \tag{8}$$

$$\tau \dot{x} = v \tag{9}$$

In this equation $f(s)$ is the same as in the original form, equation (3), and $K(g - x_0)s$ enables to avoid rapid increase of acceleration at the beginning of the movement.

In [11] the invariance under affine transformation of the new approach is proved, which means that the new form is not only translationally but also rotationally invariant. The improvement of the adaptation to the new goal is also represented in [11]. The goal parameter can be changed before the motion or even while the motion is being performed.

2.5 Obstacle Avoidance with DMP

One of the main properties of the DMP is the possibility of the trajectory correction if an obstacle is encountered. In [11] the problem of obstacle avoidance is given a special attention.

The steering angle describes the angle between the velocity $\mathbf{v}$ and the obstacle $\mathbf{o}$ directions, as it is presented in Figure 1, so that for large angles, $\dot{\varphi}$ approaches zero, and this means that if the obstacle is far enough it does not make any change to the trajectory.

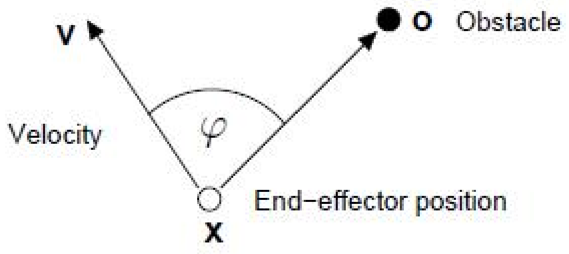

Figure 1. The Steering Angle [11]

The velocity vector is changed by a rotational matrix $\mathbf{R}$ with axis $\mathbf{r} = (\mathbf{o} - \mathbf{x}) \times \mathbf{v}$ and the angle of rotation $\pi/2$. The differential equation (10) describing the human steering angle is derived in [13]

$$\dot{\varphi} = \gamma \varphi exp(-\beta|\varphi|), \tag{10}$$

where γ and β are adjustable constants.

If the obstacle is static, the only change that is needed for the equation of motion is adding a term $p(\mathbf{x}, \mathbf{v})$, given in the equation (11).

$$p(\mathbf{x}, \mathbf{v}) = \gamma \mathbf{R} \mathbf{v} \varphi exp(-\beta|\varphi|), \tag{11}$$

where $\varphi = cos^{-1}((\mathbf{o} - \mathbf{x})^T \mathbf{v}/(|\mathbf{o} - \mathbf{x}| \cdot |\mathbf{v}|))$, which is always positive. As a result, the equation (12) introduces the transformation equation.

$$\tau \dot{v} = K(g - x) - Dv + K(g - x_0)s + Kf(s) + p(x, v) \tag{12}$$

Figure 2 shows the examples of trajectory correction with different distances to the obstacle

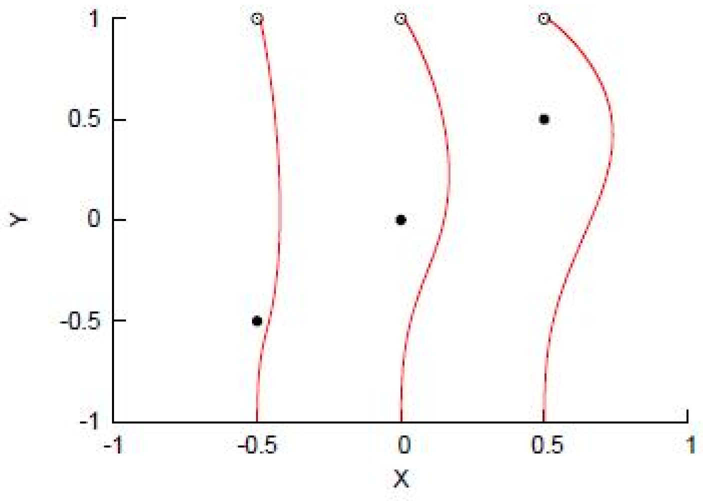

Figure 2. Obstacle avoidance example [11]

In a case with many obstacles, the term $p(\mathbf{x}, \mathbf{v})$ sums up the correction of the trajectory caused by each of the obstacles, forming a new path to the goal position around all the hindrances.

2.6 Imitation learning with DMPS

Once a motion has been demonstrated by a tutor, the system has the information about the trajectory of the movement. The way of how the imitation learning is performed can be found in [7], [5]. The positions of the pointer $x(t)$ at each moment of time, when the motion was recorded, are obtained. Then the first $\dot{x}(t)$ and the second $\ddot{x}(t)$ derivatives of the path, the velocity of the motion and the acceleration correspondingly, can be numerically calculated for each time-step. After that the phase variable $s(t)$ is calculated for an appropriate temporal scaling which is adjusted such that the nominal dynamics achieves 95% convergence at the end of a motion [6]. After that the target function $f_{target}(s)$, derived from the transformation equation (8), should be calculated:

$$f_{target}(s) = \frac{\tau \dot{v} + Dv}{K} - (g - x) + (g - x_0)s, \tag{13}$$

where x_0 and g correspond to the positions of the pointer in the beginning and the end of the motion.

At the next stage calculating the weights of Gaussian basis functions is performed so that the error between the target function and the non-linear function, modeling the motion, is minimum. The error criterion can be formulated as in the equation (14)

$$J = \sum_s (f_{target}(s) - f(s))^2, \tag{14}$$

Minimizing the error J is a linear regression problem, that delivers weights for Gaussian basis functions. A linear regression problem can be solved, for example, using the least squares method [14]. Learning part of the process ends with obtaining the weights for a non-linear function. After that the motion can be generated using the weights. The transformation system of equations are solved for velocity and motion generation for each single moment of time. For example, one can use the Runge-Kutta methods of numerical analysis to solve the system of differential equations.

2.7 Combination of Movement Primitives

Since the generation of each primitive motion starts with zero velocity and acceleration, the motions can be combined so that they are performed one directly after another. However, in [5] the approach to generate a sequence of motions without a pause is presented. According to the approach in [5] a motion starts before the preceding motion is finished. In this case the velocity and acceleration between the motions are not zero. A proper initialization needs to be performed for the succeeding velocities and positions $(v_{pred} \rightarrow v_{succ}, x_{pred} \rightarrow x_{succ})$.

3 SYSTEM DESCRIPTION

The section describes the components of the system of motion learning and generation from visual sensing. First, a large scale view of the system is given. After that the parts of it are introduced in more details. The system, represented in Figure 3, includes three components: stereo camera, robot arm and computational module.

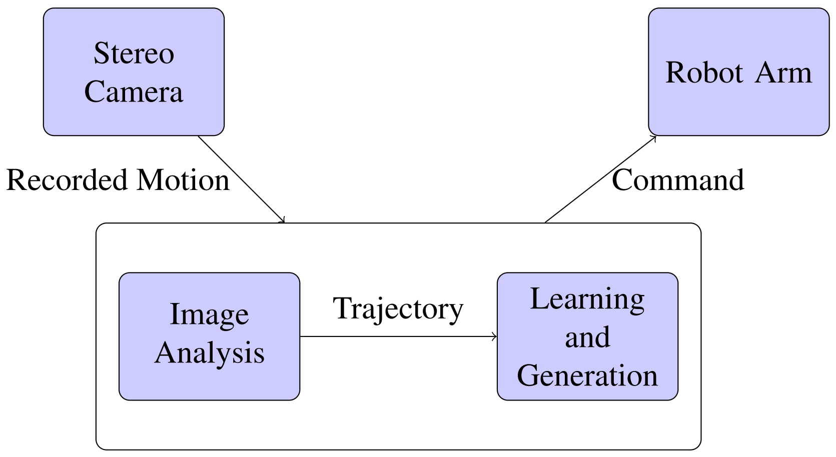

Figure 3. Large Scale Model of the System

3.1 Hardware Components

Stereo vision camera provides the system with a set of left and right images recorded during the motion demonstration. Intrinsic camera calibration parameters are known, including focal length, principal point, skew coefficient, defining the angle between the x and y pixel axes, and distortions, including radial and tangential distortions. Transformation matrix from camera to the global system of coordinates is provided as well.

Robot Arm Melfa RV-3SB is the 6-DOF robot arm at the laboratory, equipped with a JR3 force/torque sensor at the wrist, and a Weiss Robotics WRT-102 parallel gripper (Schunk PG-70 gripper equipped with Weiss tactile sensor arrays).

Computational Module consists of a personal computer, that is responsible for image analysis procedure, learning, generation process and sending commands to the robot.

3.2 Software

Image capturing program, which continuously saves images of the demonstration with the expected frame rate of 15Hz, is provided by the laboratory.

Camera calibration software,which enables a user to find the matrix of transformation between the camera and the world coordinate system, is provided by the laboratory.

Image analysis and learning program performs the tracking of the pointer, 3D reconstruction of the trajectory and, after learning procedure, returns the weight coefficients of the motion. The program was implemented in Matlab, as part of the thesis.

Motion generation and robot control software runs the velocity generation routine and sends commands to the robot. The software was implemented in C++ as part of the thesis work.

3.3 Image Analysis

Image analysis process, illustrated in Figure 4, need to be devoted much attention, since the results of it will affect the final result of the system. The image processing consists four steps.

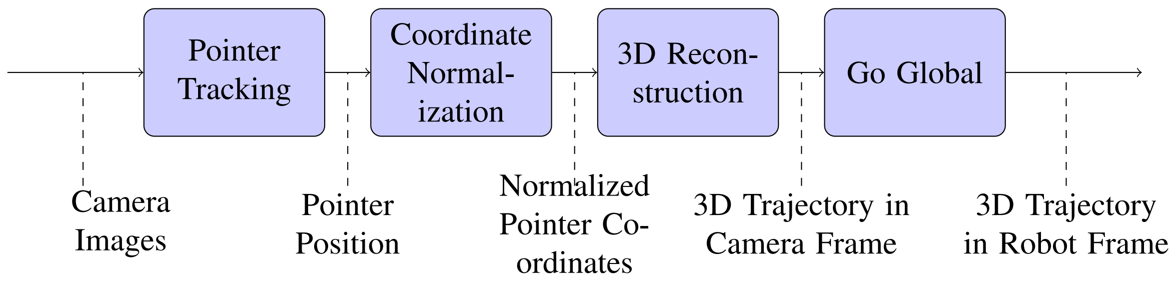
Pointer Tracking
Coordinate Normal-ization
3D Recon-struction
Go Global
Camera Images
Pointer Position
Normalized Pointer Co-ordinates
3D Trajectory in Camera Frame
3D Trajectory in Robot Frame

Figure 4. Image Analisys Routine

In the first block, pointer tracking, given the input images the program has to detect the pointer in all of them. After that the coordinates of the pointer are normalized according to the intrinsic calibration parameters. Then the trajectory is reconstructed with respect to the left camera frame. Finally, the trajectory is moved to the global coordinate system using the transformation matrix. Image analysis component is described in more detail in Section 4.

3.4 Trajectory Learning and Generation

Figure 5 presents the diagram of the learning process.

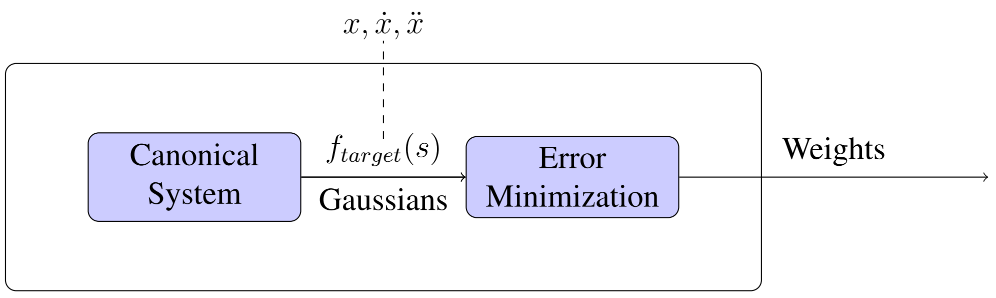

Figure 5. Learning Process

The canonical equation generates the phase parameters, for which the target non-linear function f_{target} and Gaussian basis functions are calculated, given the coordinates x, velocities $\dot{x}$ and accelerations $\ddot{x}$ of the trajectory demonstrated. As an output the learning block provides the set of weight coefficients, the result of the error criterion minimization box.

Figure 6 demonstrates the process of motion generation.

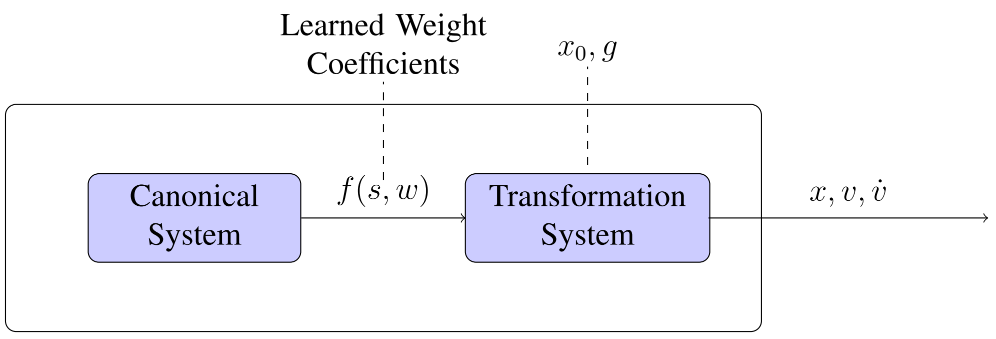

Figure 6. Motion Generation

Knowing the weights of the non-linear function describing the motion and given the task specific start and goal positions the program outputs the set of expected trajectory coordinates, velocities and accelerations.

Similarly to the learning stage the canonical system generates the phase variable for the whole duration of motion, using which the non-linear function is calculated. After that the transformation system is employed to compute the following position, velocity and acceleration. The implementation of learning and generating are discussed in Section 5.

4 VISUAL SENSING

The importance of visual sensing, the first stage in the system of learning robot motion, is difficult to overestimate, since the accuracy of trajectory replication strongly depends on the accuracy of the measured path. This section represents the fundamentals of stereo geometry as well as the description of pointer tracking, normalizing frame coordinates and 3D reconstruction.

4.1 Pointer Detection

4.1.1 Target Selection

One part of the problem was to choose the pointer for motion demonstration. A tutor was using a red ball to demonstrate the movement and a stereo camera with known calibration parameters is recording left and right images, that will make up the representation of the whole motion. The frame rate of the saved images might not be constant (i.e., the gaps between images might be the expected to be 15Hz seconds most of the time but sometimes more). The color of the pointer was intended to be easy detected on the background, so it is red. A round object was chosen to get almost the same view of it regardless of the projection, which can differ in different points of the trajectory. The size of the ball is convenient to demonstrate the motion and at the same time big enough to detect its center of mass correctly, so that the trajectory obtained is close to the original one. The material of the pointer should have as little specular reflection as possible not to get holes in the image of the ball.

4.1.2 Detection Algorithm

Pointer localization and tracking is the first part of the trajectory detection, which can not be much time and resource consuming. That is why color segmentation was employed to detect the pointer. Since the lighting conditions and the position of the camera might change from one experiment to another, a user is asked to show the position of the ball in the first image, and after that the program is using these samples of the color to localize the pointer in further images. Colors are considered in HSV color space. Maximum and minimum of HSV-values of the pixels specified by the user are utilized as the threshold for detecting the area of points of the ball. In addition, the system memorizes the previous location of the ball and later on takes into account not the whole image but the part that is expected to contain the pointer.

Blue points in Figure 7 represents the pixels assumed to be the area of the ball due to the thresholds:

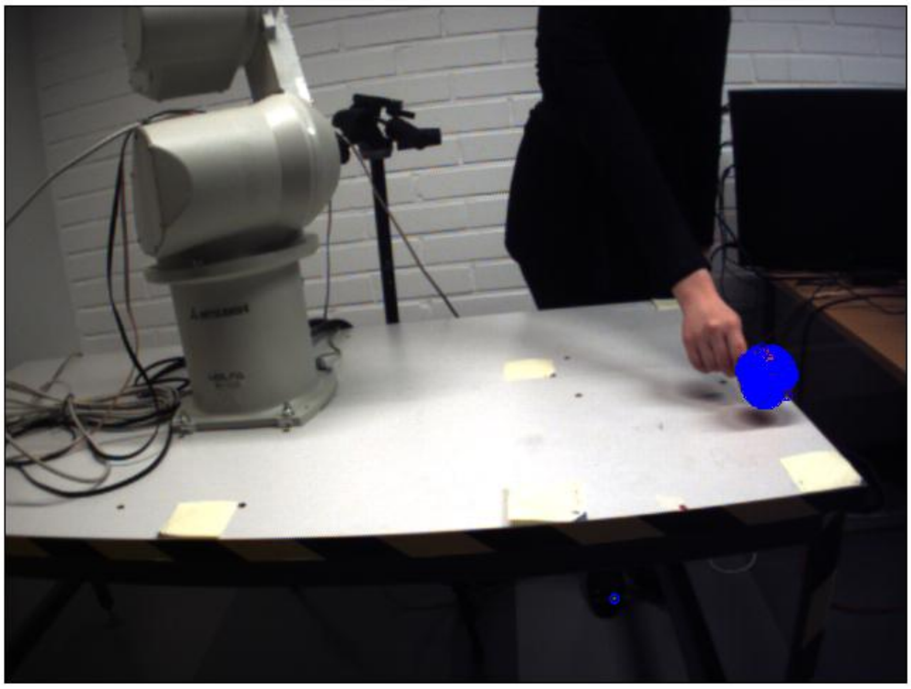

Figure 7. Localizing the Pointer, Color Segmentation

There might be several regions including pixels of the sample color, as Figure 7 shows there is also a part of a shoe detected as the pointer. In order to resolve this problem, first morphological closing is performed on the binary image to make the regions more homogeneous. A disk structuring element is used to preserve the circular nature of the object. After that blob detection reveals the areas remained and for the region with the largest area the center of mass is calculated. Figure 8 shows the region of the ball as a binary image and the detected center of mass.

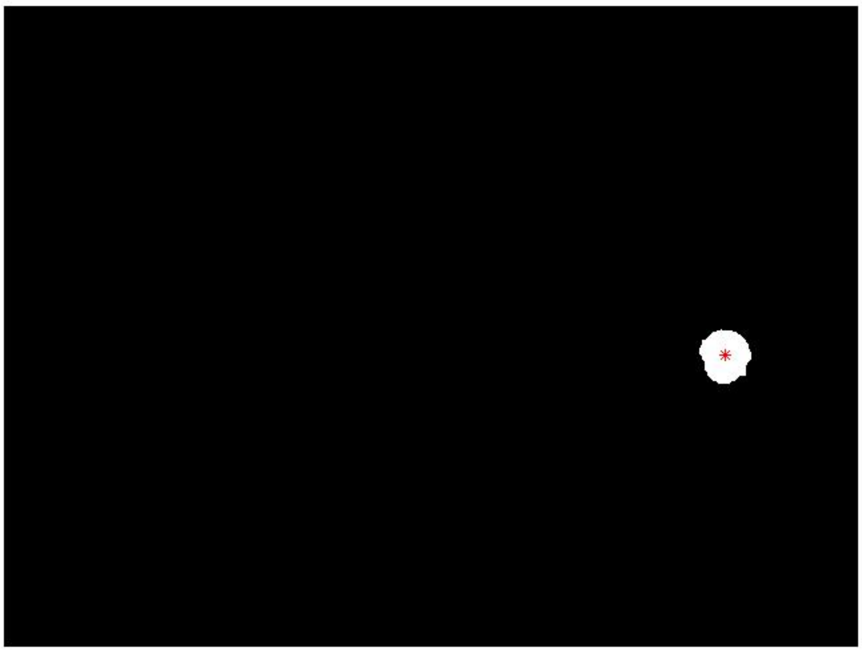

Figure 8. Pointer Localization

After the color threshold was obtained, the further algorithm of a pointer detection is as follows:

1. determine the window, containing the pointer, according to the previous center of mass detected;
2. detect the clouds of the red points according to the threshold;
3. get a binary image, where red areas are white and all the other points black;
4. perform image closing;
5. blob detection;
6. choose the blob with the largest area;
7. calculate the center of mass of the largest blob.

The pointer is localized in the same way in all of the images of the trajectory, including right and left views of the camera. After this stage the system obtained the trajectory in the left and the right frame of the camera, but the question is how to get the motion in 3D, and here stereo geometry needs to be employed.

4.2 Coordinate Normalization

Before implementing 3D reconstruction, the coordinates of the points localized in the right and left images should be normalized with respect to the intrinsic camera parameters that describe the internal properties of the device optics. The focal lengths f_l and f_r for both lenses, the principle points or the centers cc_l and cc_r of the left and the right camera frames, the skew coefficients s_l and s_r, the radial and tangential distortions are provided as the internal parameters of the camera [15].

In simple case, shown in Figure 9, if the projection plane $Z = 1$, the projections on X and Y planes can be obtained as $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$ correspondingly.

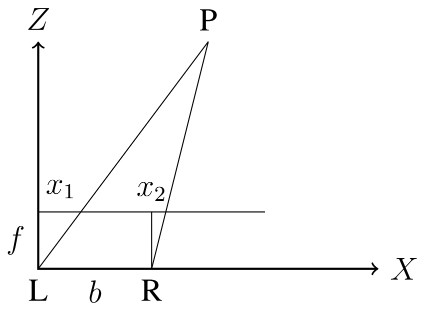

Figure 9. Perspective projection

In real cameras the focal length is not equal to 1 and therefore the z-coordinates of the points in the projections should be scaled to the focal length. The coordinates should be also normalized with respect to the principal points of each frame. The pointer coordinates in the left and right camera frames can be computed as

$$pl_{norm} = \frac{pl - cc_l}{f} \tag{15}$$

$$pr_{norm} = \frac{pr - cc_l}{f}, \tag{16}$$

where pl and pr are the detected coordinates for the left and right camera frames correspondingly.

The skew coefficients defining the angle between the x and y pixel axes should be taken into account both for right and left cameras as follows:

$$x_l = x_l - s_l * y_l \tag{17}$$

$$x_r = x_r - s_r * y_r \tag{18}$$

Finally the third step is to compensate for the lens distortion, which all optical devices are suffering from. The location of a point in the image can be larger or smaller from the principal point than can be obtained from the projection equations, this offset is increasing for the pixels that are far away from the center [15]. The problem of compensating lens distortion was solved using the function of Matlab toolbox provided by the University of Oulu [16].

4.3 Reconstruction of 3D Coordinates

Having obtained the normalized coordinates of the trajectory points in the left and right images of the camera one can move to the reconstruction problem.

4.3.1 Stereo Vision

A word 'stereopsis' comes from the Greek language, where it meant 'solid sight'. Nowadays this term is referred to as an important visual ability of people and animals to sense the depth according to the difference in points of view of two eyes [17]. As a result human is able to percept three-dimensional shapes and distances. The idea of stereopsis is employed in stereo cameras, which are able to capture three-dimensional images. Unlike a human, a stereo camera can have more than two lenses, but in this thesis a camera with two lenses is used.

Figure 10 represents a simple stereo system composed of two pinhole cameras

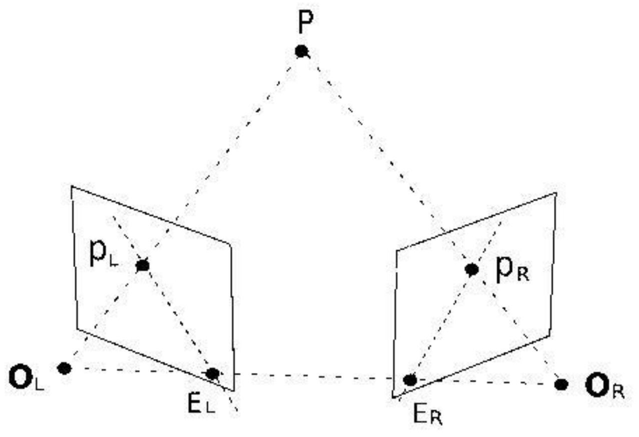

Figure 10. A Simple Stereo System [18].

There are two problems of stereo vision to be considered [18]: the correspondence problem and the reconstruction problem. The first stands for defining the corresponding points in left and right images, the main difficulty here is that some elements of the scene might be visible only in the left image, for example, and not visible in the right. The solution is to make the stereo system to detect the areas in two images that should not be matched and exclude them from the images.

The second problem, 3D reconstruction, in human vision is solved by brain that is aware of the differences in retinal position, disparity, between matched elements and thus reconstructs an image of the object seen by a person.

4.3.2 Reconstruction Algorithm

One of the methods used to reconstruct the 3D coordinates of the point is triangulation [18], [19]. In ideal conditions, when the matching problem is solved correctly, the point P in 3D can be obtained by intersecting the ray going through the optical centers O_l and O_r and the projections p_l and p_r of the points in the corresponding images. In most of the cases these rays do not intersect in one point in 3D space, which means that the system looks as in Figure 11

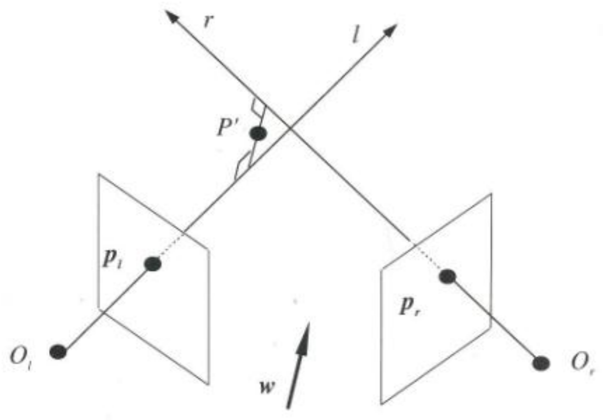

Figure 11. Stereo System with Non-intersecting Rays [18].

In this case the problem becomes more complicated but can be solved using a method called midpoint triangulation [20]. Assuming that the intrinsic and extrinsic camera parameters are known, it is possible to calculate the point in 3D as the point with the equal distance to the rays l and r going from the centers of lenses and through the projections p_l and p_r in both left and right images. If w is the vector perpendicular to both l and r and s is the part of it joining l and r together then the required point in 3D will be lying in the middle of s. The equations are provided in [18, p.162-163]

5 DYNAMIC MOVEMENT PRIMITIVES IMPLEMENTATION

The section presents the core part of the research, dealing with the mechanism of learning the motion from demonstration and generating a duplicate of it according to the weight coefficients obtained in the process of training. First, the way of motion representation and transformation between the robot and the camera system of coordinates is considered. Next, the issues of motion learning are presented: what was the target function, how the motion was described and what was the output of the learning procedure. The section of motion generation describes the method of obtaining the trajectory and velocity of a motion needed using the parameters provided by the learning process.

5.1 Motion Representation

After the first stage of Visual Sensing, Section 4, performing the transformation from the images of the stereo camera to 3D world, the motion demonstrated is represented as a set of points in 3D with the center of the coordinate system in the camera origin. In addition, the vector of velocities in X, Y and Z directions are provided.

In order to perform the motion on a robot, it is necessary to get the trajectory with respect to the robot platform origin. For that purposes the transformation matrix is used and the conversion itself is shown in the equation (19), where P^R is the desired trajectory in the robot frame, T_C^R the transformation matrix from the camera to the robot coordinate system and P^C the trajectory in the camera coordinate system:

$$P^R = T_C^R \cdot P^C \tag{19}$$

Using the function of camera calibration made it possible to get the matrix of transformation from the robot to the camera coordinate system. The matrix T_R^C consists of two elements: 3x3 rotational matrix R and 4x1 translational vector t:

$$T_R^C = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{21} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{pmatrix}$$

To make the above mentioned transformation the matrix should be inversed 20

$$P^R = (T_R^C)^{-1} \cdot P^C \tag{20}$$

To perform the transformation it is necessary to move to homogeneous system of coordinates, so that the matrix turns to be square:

$$T_R^C = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{21} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In homogeneous coordinate system the trajectory points and velocities get the fourth coordinate as follows: $P = [XYZ1]$, $V = [V_xV_yV_z0]$. The fourth coordinate of velocity is zero as the translation should not have an effect on the velocities.

5.2 Description of Parameters

The parameters of the system could be divided into several groups: time and phase parameters, Gaussian basis function parameters, spring and damping factors.

5.3 Time and Phase Parameters

Time vector t is determined according to the duration of the motion ($move\_time$), and to the approximate frame rate of the camera. The number of pictures (pic_num) taken is known and the time vector can be defined as follows 21:

$$t = [0...move\_time]. \tag{21}$$

The time step for the array t is equal to $\frac{move\_time}{pic\_num}$. Since all the calculations are performed in phase domain one should determine phase variable s with one-to-one mapping to the time variable. A phase variable s monotonically changes from 1 to 0 and can be found using the fundamental equation (4).

The parameters α and τ should be selected such that s covers the maximum interval from 1 down to 0 for the specific duration of motion. Figure 12 shows the dependence between s and t.

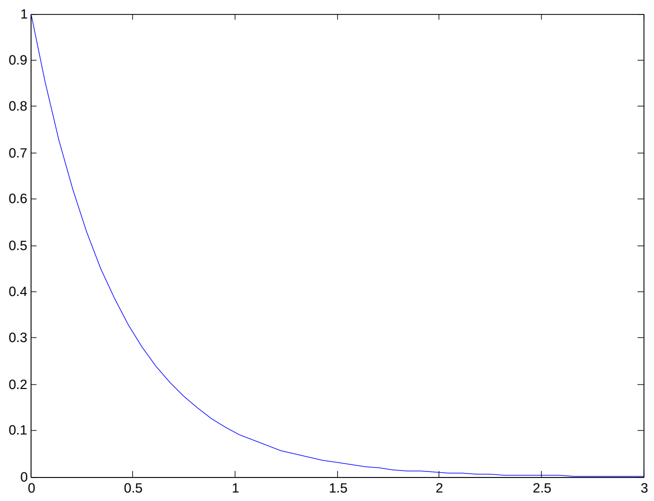

Figure 12. Phase Variable s

5.3.1 Gaussian Basis Function Parameters

The concept of DMP assumes that the motion is fully described with the set of weight coefficients w, which are obtained in the process of minimization the error criterion 14. In fact a motion is describe with M Gaussian basis function

$$\varphi_i(s) = exp^{(-h_i(s-c_i)^2)} \tag{22}$$

with centers c_i and width h_i.

The number of weights w is equal to the number of Gaussian basis functions M, which should be selected as small as possible but at the same time large enough not to lose essential information of the motion.

The centers of Gaussian basis functions should be equally distributed in time, and it is important to avoid unnecessary overlap of the functions, which is regulated by the width of the Gaussians, which should also be equal for all of them in time.

5.3.2 Spring and Damping Terms

The spring factor K and the damping term D contributes a lot in the behavior of the system. These terms are included in the system of differential equations describing the system (1), (2), that are used for motion generation, they also are considered in the process of learning as the terms with K and D coefficients enter the target function 13 of the movement.

According to the definitions K affects the speed of convergence to the goal position, and D should be chosen such that the system is critically damped. The effect of these parameters is studied in Section 7.

5.4 Time vs Phase Approaches

In this thesis a new approach for determining the Gaussian basis functions is proposed: to locate the centers of Gaussians in time domain. In the overviewed literature Gaussians are generated in phase domain. When moving to the time domain the necessary changes to the transformation function are described in the equations (23) and (24)

$$f(s) = \frac{\sum \omega_i \varphi_i(t) t(s))}{\sum \varphi_i(t)} \tag{23}$$

$$t = -\frac{\tau}{a} ln(s) \tag{24}$$

It is worth to mention that instead of multiplying the nominator of $f(s)$ by s , the transformation function $t(s)$ is employed. One of the targets of Section 7 is to show how the generation of the Gaussians in time and phase domain affects the resulting trajectory.

5.5 Learning Motion

In one dimension, a motion is represented by a modified set of differential equations describing a spring-like linear system. Previously the system has been described in Section 2. The target function was given in the equation (13). The main problem of the learning phase is to define the function that would match the target function in the best way.

A nonlinear function 25 can be represented as a set of Gaussian basis functions $\varphi_i = exp(-h_i(s-c_i)^2)$

$$f(s) = \frac{\sum \omega_i \varphi_i(s) s}{\sum \varphi_i(s)}, \tag{25}$$

If the centers c_i and the width h_i of Gaussians are predetermined then the weight coefficients w_i are responsible for similarity of the target and designed functions. The following steps should be taken in order to find the weight coefficients:

1. Calculating the first and the second derivatives of the trajectory using the numerical differentiation for each dimension separately. Central-difference formula was chosen for evaluating the derivatives of the position x and velocity v at each moment of time:

$$\dot{x_i} = \frac{(x_{i+h} - x_{i-h})}{2h} \tag{26}$$

$$\dot{v_i} = \frac{(v_{i+h} - v_{i-h})}{2h}, \tag{27}$$

   where step h is equal to the time step used in (21) for time array calculation.

2. Determining the target function for each phase variable and for each dimension using the obtained parameters and derivatives in the equation (13):

$$f_{target_i}(s) = \frac{\tau \dot{v_i} + D\dot{x_i}}{K} - (x_N - xi) + s(x_N - x1)), \tag{28}$$

where τ is a time scaling parameter, D damping term, K spring constant, x_N the last position in the array of the position, that is a goal position, x_0 the first position in the array of the positions, initial position. The derivatives were calculated for the moments of time and the target function depends on the phase variable. Such a substitution is possible due to one-to-one matching between time and phase variables.

3. Defining the weights of Gaussians.
   The weights calculation is aimed at maximizing the conformity between the target and the nonlinear function, representing the motion, or, in other words, the goal of this step is to minimize the error criterion 14, given in Section **??**

   One might consider this problem as a linear regression problem, the solution of which is

$$w = A^{+} f_{targ}(s), \tag{29}$$

   where A^{+} is a pseudoinverse matrix of the matrix A, that is not square and can be expressed as follows:

$$A_{ij} = \frac{\varphi_i(s_j)s_j}{\sum_i \varphi_i(s_j)} \tag{30}$$

The output of the learning procedure is the set of weight coefficients, defining the nonlinear function for motion replication.

5.6 Runge-Kutta Implementation for Movement Generation

In order to obtain the trajectory of the motion given the weight coefficient of the motion and the initial and goal positions the system of differential equations (**??**), (**??**) should be solved. The fourth-order Runge-Kutta method [21] for the approximation of solutions of ordinary differential equations is employed for that purpose with the chosen time step.

The equations to be solved are the following:

$$\dot{v} = \frac{K(g-x) - Dv + K(g-x_0)s + Kf(s)}{\tau}. \tag{31}$$

$$\dot{x} = v/\tau, \tag{32}$$

where x_0 and $v_0 = 0$ the initial values of the arguments.

The iterative formulas for coordinates x and velocities v are then:

$$x_{n+1} = x_n + 1/6h(K_1 + 2K_2 + 2K_3 + K_4) \tag{33}$$

$$v_{n+1} = v_n + 1/6h(K_1 + 2K_2 + 2K_3 + K_4) \tag{34}$$

where $t_{n+1} = t_n + h$, h is the time step for Runge-Kutta.

The terms K_i can be expressed as:

$$K_1 = f(s(t_n), v_n, x_n) \tag{35}$$

$$K_1 = f(s(t_n + 1/2h), v_n + 1/2hK_1, x_n + 1/2hK_1) \tag{36}$$

$$K_1 = f(s(t_n + 1/2h), v_n + 1/2hK_2, x_n + 1/2hK_2) \tag{37}$$

$$K_1 = f(s(t_n + h), v_n + hK_3, x_n + hK_3) \tag{38}$$

As a result Motion Generation provides a smoothed trajectory for the motion including both positions and velocities.

6 ROBOT CONTROL

In this work one of the assumptions was to use the available high level robot interface, but it is essential to understand what the employed functions perform. In order to do that it is necessary to understand the basics of robot kinematics and explore how the robot is controlled.

6.1 Robot Kinematics

Unlike the dynamics, which considers forces and torques making the robot move, the kinematics deals with the definition of the possible movements of the robot, considering aspects of redundancy, collision avoidance and singularity avoidance [22]. Robot kinematics can be divided into two types: forward kinematics and inverse kinematics. The former determines the position of the end-effector given the location of joints. The latter calculates joint positions given a position in 3D space.

6.1.1 Forward Kinematics

A robot arm, see Figure 13, is typically built as a set links connected with revolute joints, that rotate a robot around given axis allowing it to perform movements activating different joints.

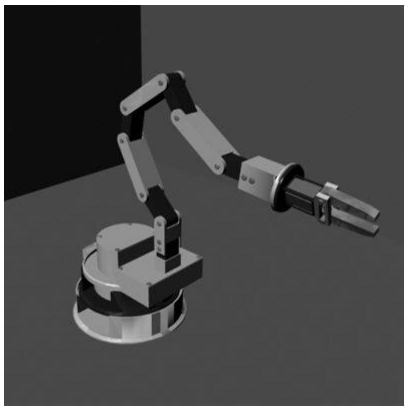

Figure 13. General Structure of a Robot Arm, [23]

Like in Figure 13, the arm used in this work ends with a sensor with an end-effector attached to it.

Forward kinematics solves a transformation equation to find the location of the end-effector using the joint angles. Each joint has its own coordinate frame, but the position of the end-effector is found in the base frame, so that the forward kinematics procedure performs transformations between all the neighboring joint frames, as shown in Figure 14.

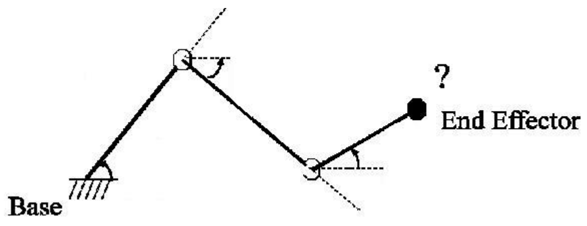

Figure 14. Forward Kinematics by Composing Transformations, [22]

The solution of the forward kinematics is always unique for special robots, one combination of joint positions corresponds only to one location of the end-effector in space.

6.1.2 Inverse Kinematics

As the name implies the inverse kinematics aims at determining the joints angles given the position of the end-effector. Therefore if x is the position of the end-effector and θ the joint angles, the task for the inverse kinematics is to solve:

$$x = f(\theta). \tag{39}$$

For θ,function f represents the forward kinematics transformation. As a rule [24] joint angles are found using numerical methods. Unlike the forward kinematics inverse kinematics does not necesseraly give a unique solution, as illustrated in Figure 15.

Thus, there might be multiple combinations of joint locations corresponding to one position of the end-effector in space.

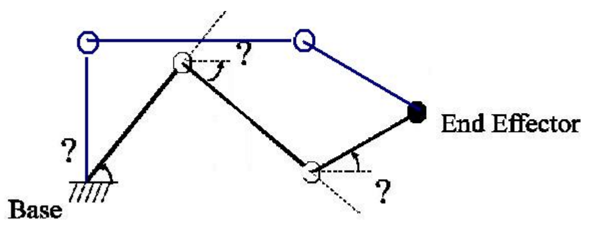

Figure 15. Multiple Solution for Inverse Kinematics Problem

6.2 Robot Control Implementation

The robot arm has two control modes:

1. Teaching mode: the robot is controlled using a control panel. One can change the robot position varying the coordinates of the end-effector in the Cartesian coordinate system or the position of joints.
2. Automatic mode is used to control the robot running programs on it.

In the thesis work the teaching mode was used to estimate the trajectory generated by the software, to see if the trajectory is reasonable. In the automatic mode the software was run on the robot.

There two stages in the robot control for performing trajectory replication:

1. moving the robot end-effector to the initial position;
2. running the trajectory.

In the former stage the robot is given the initial position coordinates in the Cartesian coordinate system, the center of which is situated in the robot base. The inverse kinematics procedure is performed to find one of the possible solutions for the joints. The function needs an initial guess for the inverse kinematics solution which was found experimentally. Knowing the location of joints the robot is able to go to the initial position.

Once the robot is at the start position it can be provided with the velocity vector for joints, containing velocities for three dimensions. Later on the robot is controlled by the velocities calculated by the generation procedure.

It is necessary to notice that the frequency, which the robot is able to get commands with, should correspond to the frequency of sending new velocities to it. This will affect the smoothness of the trajectory performed by the robot.

7 EXPERIMENTS AND DISCUSSION

7.1 Visual Sensing Experiments

The first stage of the system gets the visual information of the motion, detecting the trajectory. Since a motion is represented as a set of images from left and right cameras and considering the correspondence problem of 3D reconstruction, it would be useful to know how precise the detection of the pointer is and how precisely the 3D reconstruction is performed.

7.1.1 Pointer Detection Precision

This part of experiments measures the precision of the pointer detection. One should distinguish between the terms ’accuracy’ and ’precision’. Accuracy shows how close a measurement is to the true value, while precision shows how successfully one can get the same measurement under unchangeable conditions.

A series of images of the pointer was taken with pointer situated in the same position as in Figure 16

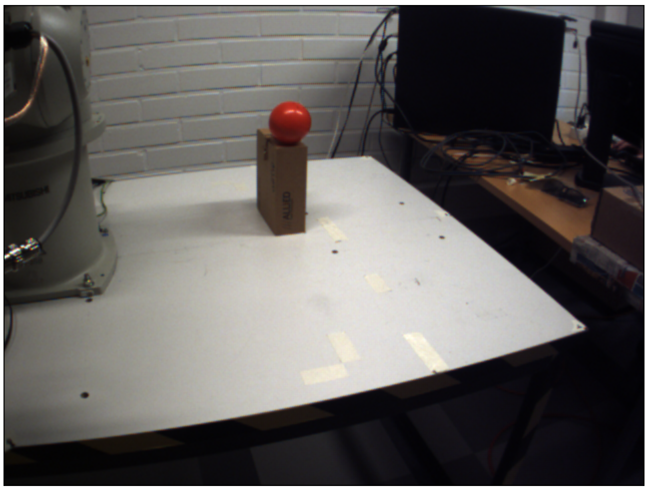

Figure 16. Pointer Precision Test

The set of images contains 25 images for each camera, left and right. The results of detection are represented in Table 1

Table 1. Pointer Coordinates In Left and Right Images

Image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
XL	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287	287
	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116	116
XR	215	215	215	215	215	215	216	215	215	215	215	215	215	215	215	215	215	215	215	215	215	216	215	215	215
	115	114	115	115	114	1165	114	115	115	115	115	115	115	115	114	115	115	115	115	115	115	115	114	115	115

XL and XR assign the coordinates of the pointer in the left and right camera images correspondingly. The means for each dimension of right and left image detection result were calculated according to:

$$XL_{mean} = \frac{1}{N} \cdot \sum_{i=1..N} XL_i \tag{40}$$

$$XR_{mean} = \frac{1}{N} \cdot \sum_{i=1..N} XR_i, \tag{41}$$

where $N = 25$ is the number of test images.

To find out the precision, the variance of the data was calculated using:

$$(\sigma_{XL})^2 = \frac{1}{N} \cdot \sum_{i=1..N} (XL_i - XL_{mean})^2 \tag{42}$$

$$(\sigma_{XR})^2 = \frac{1}{N} \cdot \sum_{i=1..N} (XR_i - XR_{mean})^2. \tag{43}$$

Standard deviation was computed as:

$$\sigma_{XL} = \sqrt{\frac{1}{N} \cdot \sum_{i=1..N} (XL_i - XL_{mean})^2} \tag{44}$$

$$\sigma_{XR} = \sqrt{\frac{1}{N} \cdot \sum_{i=1..N} (XR_i - XR_{mean})^2}. \tag{45}$$

The result of the calculation is shown in Table 2

Table 2. Means and variances for Left and Right Images Detection

	Mean	Variance	Standard deviation,%
XL	287	0	0
	116	0	0
XR	215.08	$2.75 \cdot 10^{-7}$	$2.44 \cdot 10^{-4}$
	114.80	$5.98 \cdot 10^{-7}$	$6.74 \cdot 10^{-4}$

In the worst case the standard deviation is less than 0.001%, which means that the detection has a good predictability. This precision is appropriate for using at the trajectory detection stage.

7.1.2 3D Reconstruction Precision

The same set of images was used to estimate the 3D reconstruction precision, the idea was to reconstruct the coordinate of the pointer center for all of the images and calculate the mean and variance for them. Table 3 contains the coordinates of the pointer center.

Table 3. 3D Coordinates of the Trajectory

Image	1	2	3	4	5	6	7	8	9	10	11
P	-0.0074	-0.0086	-0.0074	-0.0074	-0.0086	-0.0074	-0.0074	-0.0074	-0.0086	-0.0074	-0.0074
	0.5413	0.5417	0.5413	0.5413	0.5417	0.5413	0.5413	0.5413	0.5417	0.5413	0.5413
	0.1934	0.1940	0.1934	0.1934	0.1940	0.1934	0.1934	0.1934	0.1940	0.1934	0.1934

Table 4. 3D Coordinates of the Trajectory. Continuation

12	13	14	15	16	17	18	19	20	21	22	23	24	25
-0.0074	0.0046	-0.0074	-0.0086	-0.0086	-0.0074	-0.0074	-0.0074	-0.0074	-0.0086	-0.0074	-0.0074	-0.0086	-0.0086
0.5413	0.5387	0.5413	0.5417	0.5417	0.5413	0.5413	0.5413	0.5413	0.5417	0.5413	0.5413	0.5417	0.5417
0.1934	0.1967	0.1934	0.1940	0.1940	0.1934	0.1934	0.1934	0.1934	0.1940	0.1934	0.1934	0.1940	0.1940

Mean and variance were then calculated using:

$$P_{mean} = \frac{1}{N} \cdot \sum_{i=1..N} P_i \tag{46}$$

$$(\sigma_P)^2 = \frac{1}{N} \cdot \sum_{i=1..N} (P_i - P_{mean})^2. \tag{47}$$

Standard deviation was computed as:

$$\sigma_P = \sqrt{\frac{1}{N} \cdot \sum_{i=1..N} (P_i - P_{mean})^2} \tag{48}$$

Result of the calculation is represented in the table 5

Table 5. Means and Variances for Reconstructed Coordinates in 3D

	Mean	Variance	Standard deviation,%
P	0.0034	$0.010 \cdot 10^{-7}$	0.9301
	0.5391	$0.139 \cdot 10^{-7}$	0.0219
	0.1965	$7.085 \cdot 10^{-7}$	0.4284

The standard deviation of the 3D reconstruction is less than 1%, which shows that the method for 3D reconstruction can be used in the system and give a predictable result.

7.2 Dynamic Movement Primitives Tests

As it was shown in Section 5 the Dynamic Movement Primitives concept assumes using a number of parameters and it is difficult to find guidance how to set them. The DMP experiments are intended for clarifying what is the influence of the parameters and coefficients, how they might be chosen and is it possible to find a universal set of parameters for a large range of motion primitives.

In this subsection a single parabolic motion was used to examine the effect of the DMP parameters.

7.2.1 Spring and Damping factors Influence

In the series of experiments described in this subsection the parameters of the motion are as follows:

Table 6. Parameters of the Motion

Parameter	Value
Move Time (s)	3.5
Number of Gaussians M	9
Width of Gaussians w	10
Spring Constant K	varied
Damping Factor D	varied
Temporal Scaling τ	3
Runge-Kutta Step(s)	0.007

According to the definition the spring coefficient K corresponds to how fast the system converges to the goal position, and damping factor D should be chosen such that the system is critically damped, which means that the system returns to equilibrium as quickly as possible without oscillating.

The questions for this part of the experiments are:

1. What is the effect of the spring and the damping factors on the generated trajectory?
2. How should one choose the parameters?
3. Is it possible to single out unique values for a large set of motions?

The following examples will illustrate the effect of these parameters. The spring and damping factors were chosen experimentally. In the figures, provided below, the initial and generated trajectories are represented in the 3D space and along each coordinate axis.

Figures 17 and 18 show the trajectory for $K = 70$ and $D = 10$. The damping factor is tuned so that the trajectory is smooth, but the spring constant is obviously not large enough to let the z-coordinate grow high to reach the initial values.

Figures 20-21 represent what influence the decrease and the increase of D have on the trajectory. In the first set of figures, with smaller D, the system does not have enough time to converge to the goal position for the time given. For larger D it smoothes the trajectory but overfits still trying to converge to the target point.

According to the experiments conducted, the values of K and D should be chosen in a way allowing the system to replicate the motion but at the same time to get the smoother trajectory for the reasonable interval of time. In Figures 24 and 23 $K = 150$ and $D = 15$, and the system seem to be quite close to the original.

As it was illustrated by Figures, the increase of D helps to get the smoother trajectory and to make the system to converge, but what will happen if D will be increased further.

As it is represented in Figure 25 and 26 one will get more bent trajectory, but replicating the original path much closer. This fact might be explained by the local effect of D parameter which makes the system more sensitive to any oscillations.

Discussion of the questions:

1. The spring and damping parameters affect how fast the system converges to the goal position, how closely it follows the original trajectory and how smooth the trajectory is.
2. The parameters need to be chosen experimentally and considered together.
3. Is seems to be possible to find the values that will fit some set of motions if they have similar shape and duration.

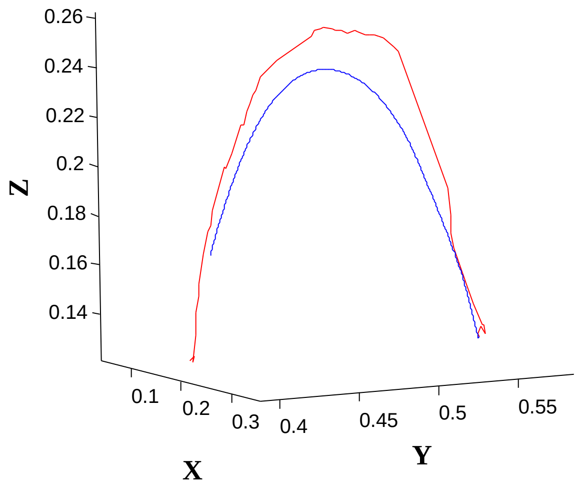

Figure 17. Initial and Generated Trajectories, $K = 70$ $D = 10$

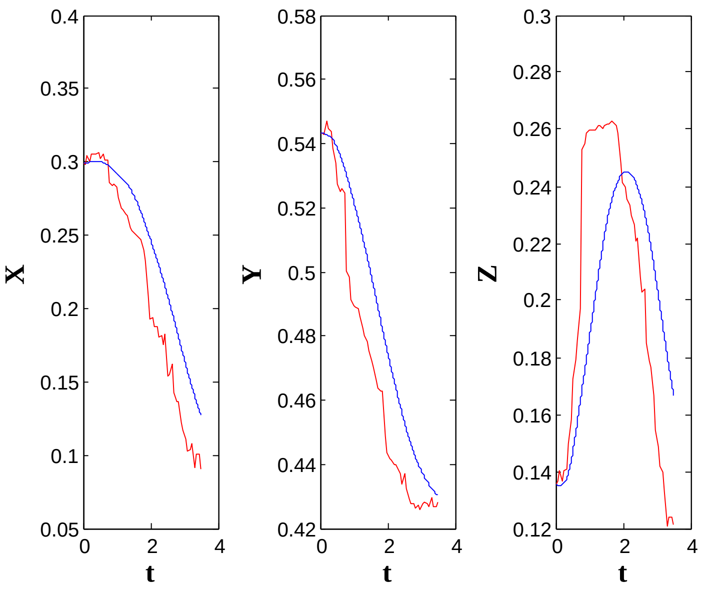

Figure 18. Initial and Generated Trajectories, $K = 70$ $D = 10$

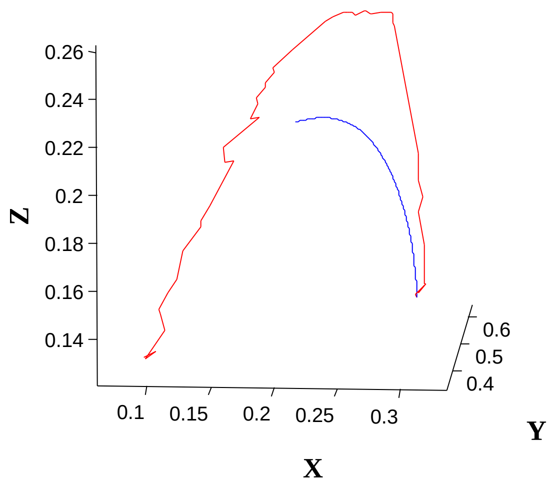

Figure 19. Initial and Generated Trajectories, $K = 70$ $D = 5$

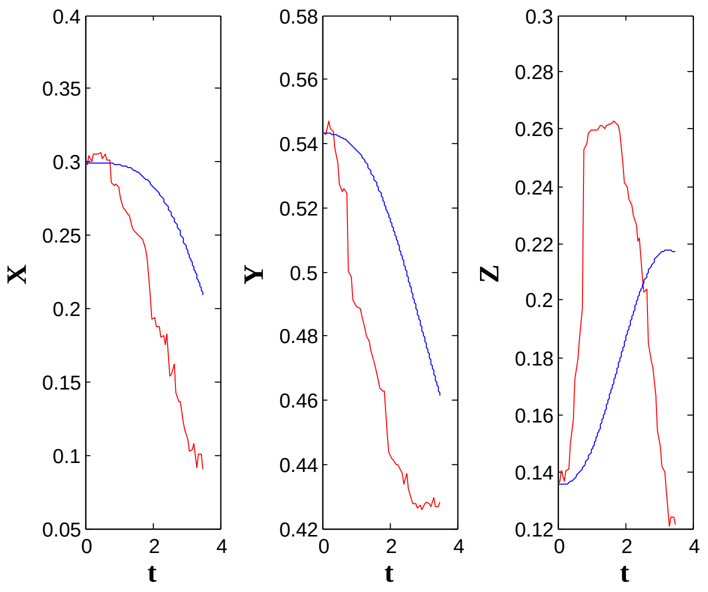

Figure 20. Initial and Generated Trajectories, $K = 70$ $D = 5$

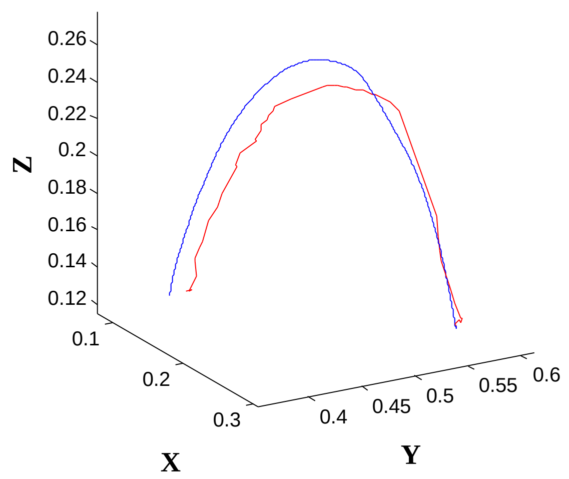

Figure 21. Initial and Generated Trajectories, $K = 70$ $D = 15$

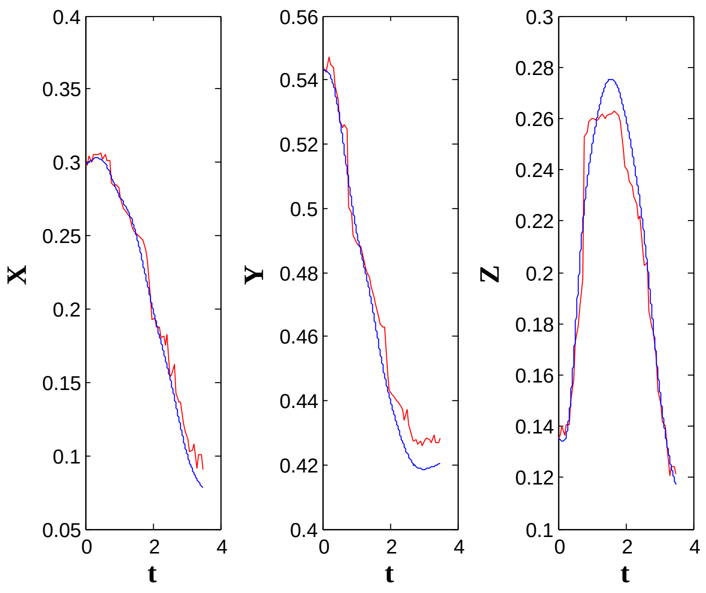

Figure 22. Initial and Generated Trajectories, $K = 70$ $D = 15$

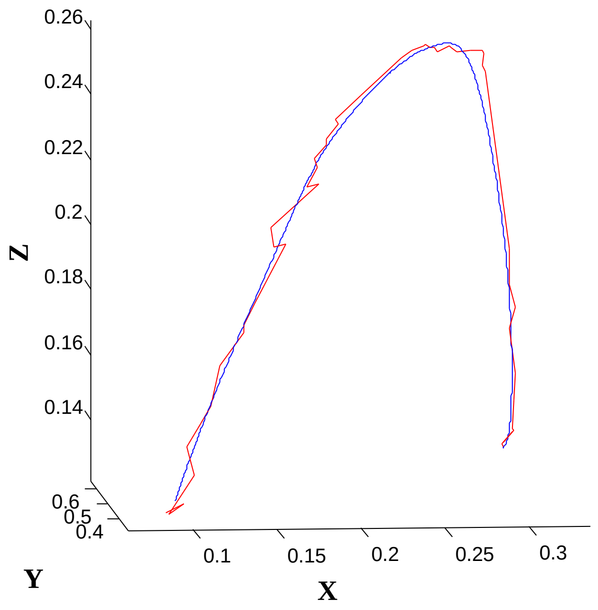

Figure 23. Initial and Generated Trajectories, $K = 150$ $D = 15$

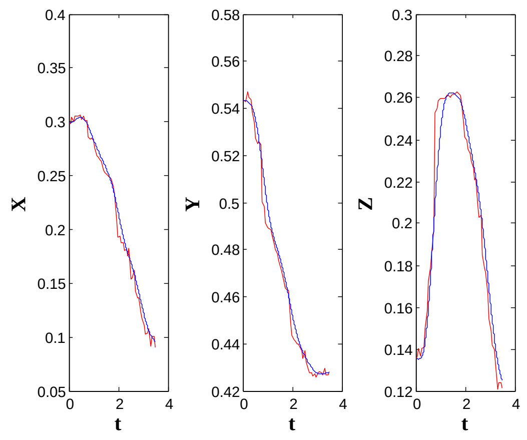

Figure 24. Initial and Generated Trajectories, $K = 150$ $D = 15$

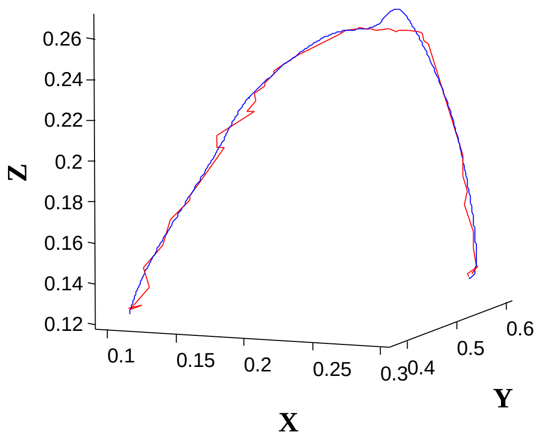

Figure 25. Initial and Generated Trajectories, $K = 150$ $D = 21$

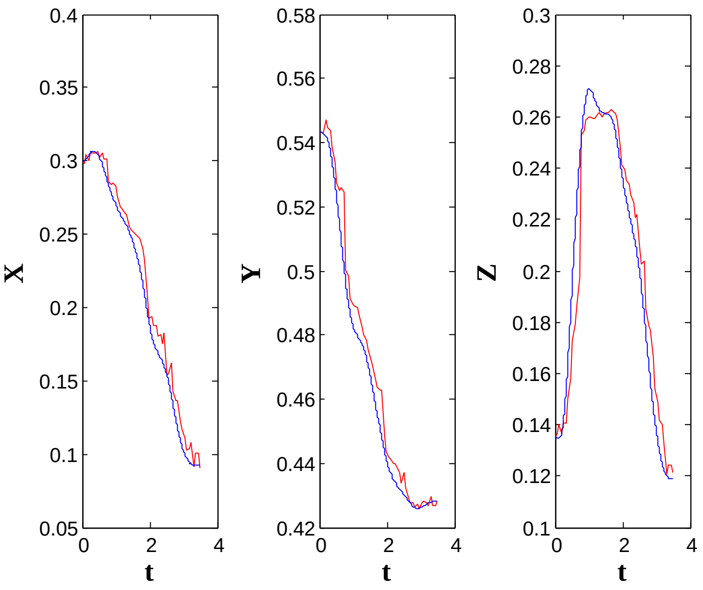

Figure 26. Initial and Generated Trajectories, $K = 150$ $D = 21$

7.2.2 The Effect of Gaussian Basis Function Parameters

The concept of DMP assumes that the motion is fully described with the set of weight coefficients w, which are obtained in the process of minimization of the error criterion. Therefore in fact a motion is described with M Gaussian basis function with centers c_i and width h_i.

The questions for this part of experiment are:

1. How to choose the number of Gaussian basis functions?
2. Is it possible to single out one number appropriate for a large range of motions?
3. How does the number of Gaussians depend on the duration of motion?

Motion parameters are as follows:

Table 7. Parameters of the Motion

Parameter	Value
Move Time (s)	3.5
Number of Gaussians M	varied
Width of Gaussians w	10
Spring Constant K	150
Damping Factor D	15
Temporal Scaling τ	3
Runge-Kutta Step(s)	0.007

In the first experiment the number of Gaussians was set to 3, $M = 3$, there was not enough information of the motion and therefore the generated curve does not match the initial, as it is shown in Figure 27.

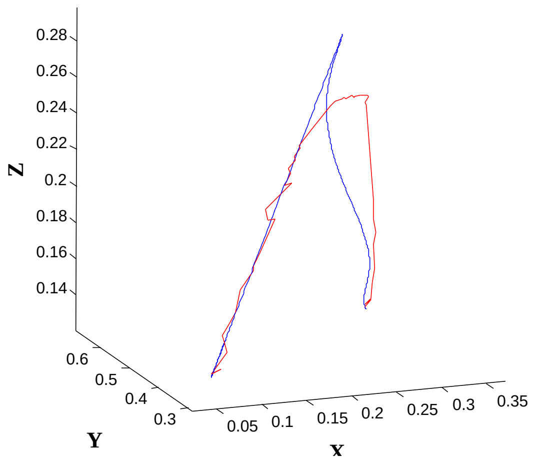

Figure 27. Initial and Generated Trajectories, $M = 3$

If $M = 20$, Figure 28, the generated curve is close to the original but there is no use in employing such a number of coefficients if the same effect could be obtained with smaller number of Gaussians.

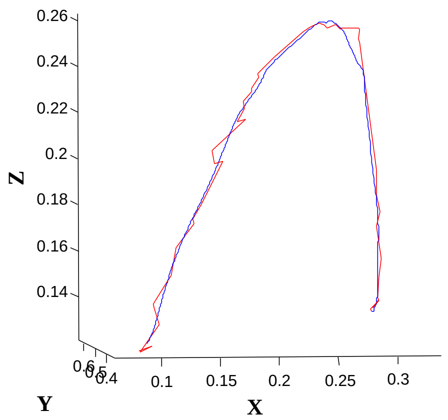

Figure 28. Initial and Generated Trajectories, $M = 20$

For example, $M = 9$, in Figure 29.

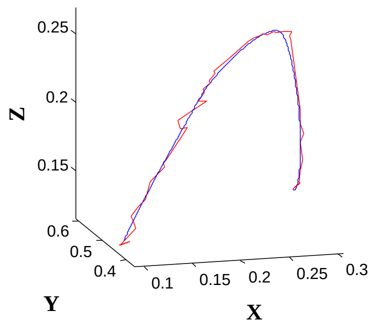

Figure 29. Initial and Generated Trajectories, $M = 9$

Questions discussion:

1. The number of Gaussian basis function should be chosen according to the duration of motion.
2. If the duration of motions from a large set is approximately the same and the shape of motions is similar, then the number of Gaussian can be found for all the motions from the set. If the shape of the motions is different, for example some motions are simple line motions, the number of Gaussians need not to be large.
3. The longer the motion is the larger number of Gaussians needed. If there are not enough Gaussian basis functions describing the motion on the whole duration, there will not be enough information to generate the motion close to the original one.

Another parameter if the width of Gaussian basis function.The questions for consideration are as follows:

1. How can the width of Gaussians be chosen?
2. Is it possible to choose one value for a large range of motions?

Motion parameters are as follows:

Table 8. Parameters of the Motion

Parameter	Value
Move Time (s)	3.5
Number of Gaussians M	9
Width of Gaussians w	varied
Spring Constant K	150
Damping Factor D	15
Temporal Scaling τ	3
Runge-Kutta Step(s)	0.007

Figure below shows the results of learning for different width h_i. The normalized Gaussians generated in time domain (left upper Figure), the same Gaussians in phase domain (right upper Figure) and Gaussians generated in phase domain (left lower Figure) are shown. The generated and initial trajectories are also shown in right lower Figure.

During the experiments more than three values for Gaussians were studied, but in the thesis work the most descriptive need to be shown.

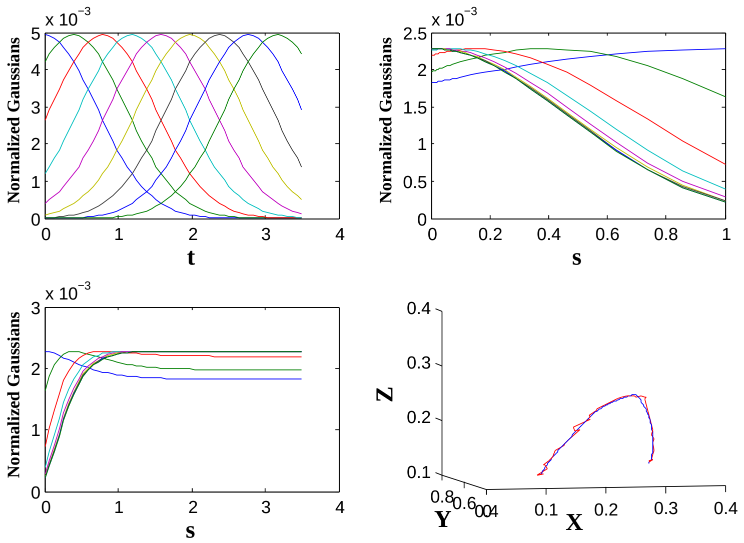

Figure 30. Initial and Generated Trajectories, $W = 1$

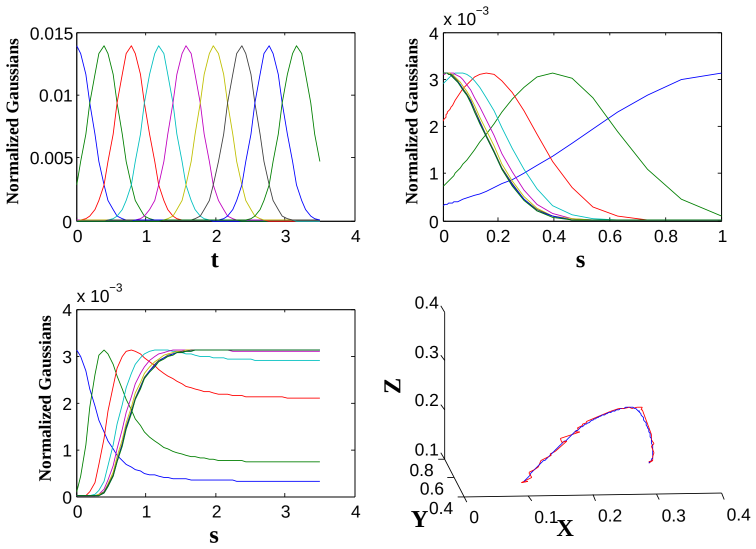

Figure 31. Initial and Generated Trajectories, $W = 10$

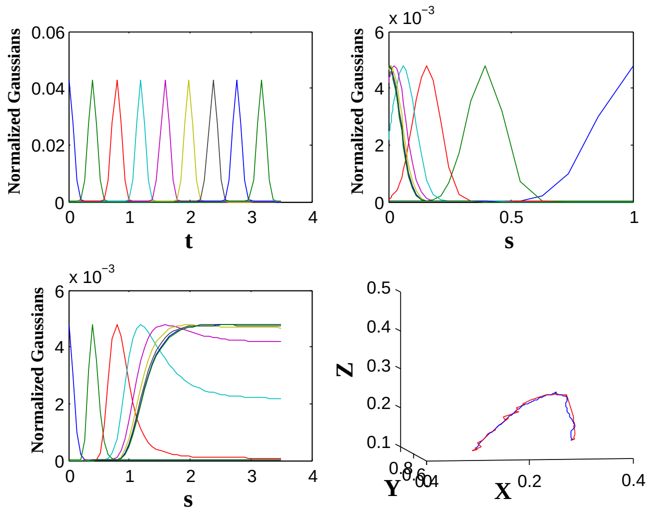

Figure 32. Initial and Generated Trajectories, $W = 100$

The next series of figures represents X, Y and Z coordinates of the initial and generated trajectories with respect to time.

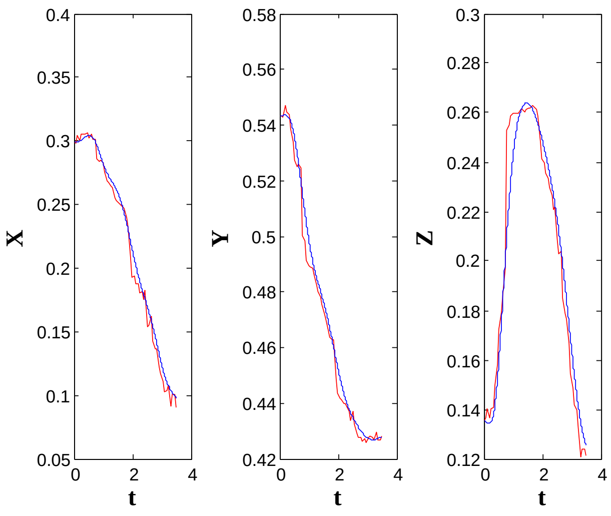

Figure 33. Initial and Generated Trajectories, $W = 1$

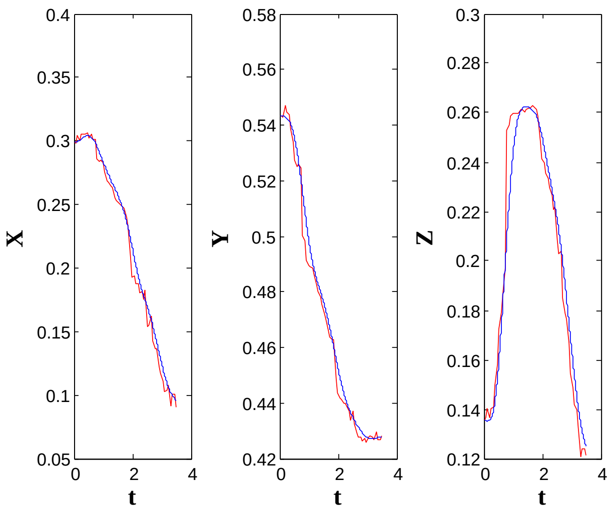

Figure 34. Initial and Generated Trajectories, $W = 10$

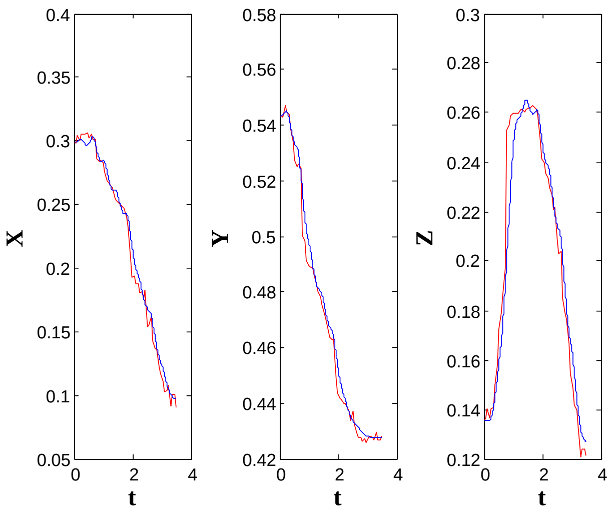

Figure 35. Initial and Generated Trajectories, $W = 100$

Questions discussion:

1. Since the Gaussians are generated in phase domain, it is preferable to have the width of Gaussians such that in phase domain they were not much overlapping. This will allow of getting the information about the motion for the whole duration of it. At the same time, the overlapping of Gaussians need to be sufficient to make the motion smooth, in other words, there should be some overlapping.
2. The width of Gaussians need to be chosen experimentally for a particular range of motions.

According to Figures it is reasonable to choose width equal to 10. The overlapping of Gaussians in time domain is adequate and the trajectory obtained is smooth and close to the initial.

7.2.3 Generation in Time and Phase Domains

The aim of this part of experiments is to show the performance of the system when the Gaussian basis functions were generated in time domain and how it differs from the case when phase dependent Gaussians were used.

Motion parameters are as follows:

Table 9. Parameters of the Motion

Parameter	Value
Move Time (s)	3.5
Number of Gaussians M	9
Width of Gaussians w	10
Spring Constant K	150
Damping Factor D	15
Temporal Scaling τ	3
Runge-Kutta Step(s)	0.007

Two experiments were made such that all parameters were kept equal, but in first case, illustrated by Figures 36 and 37, the Gaussians were described in phase domain and in the second case, Figures 38 and 39, in time domain.

When generated in the phase domain, the centers of Gaussians are not equally distributed in time domain, which means that there is not enough information about the rest of the motion. In the case when the Gaussians were generated equally distributed in time domain, they cover the whole duration of the motion, giving better fitting for the initial trajectory.

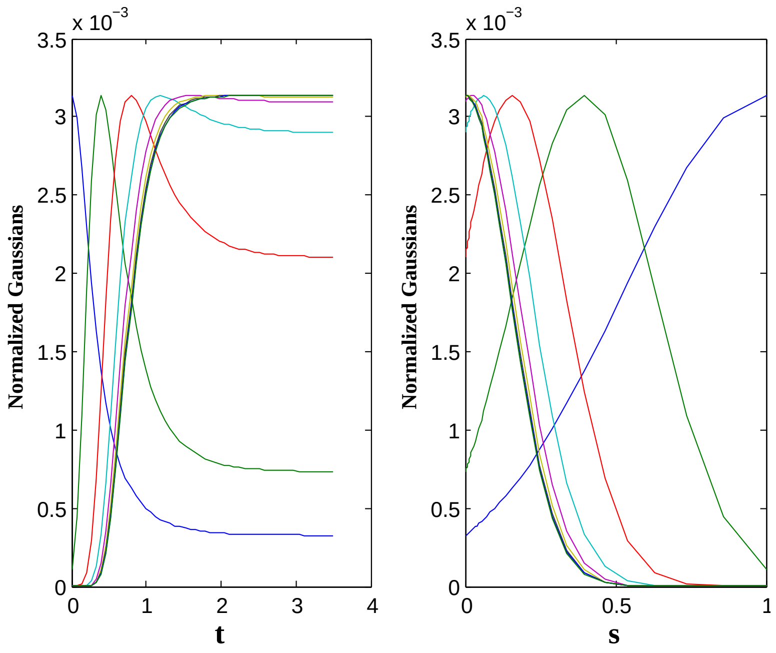

Figure 36. Gaussians in Time and Phase Domains, Generated in Phase Domain

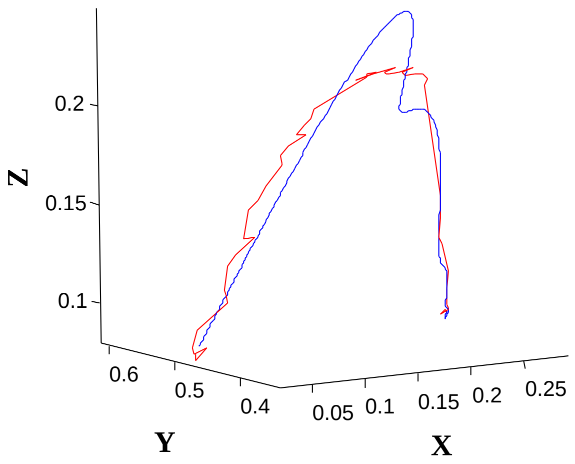

Figure 37. Initial and Generated Trajectories for Gaussians Generated in Phase Domain

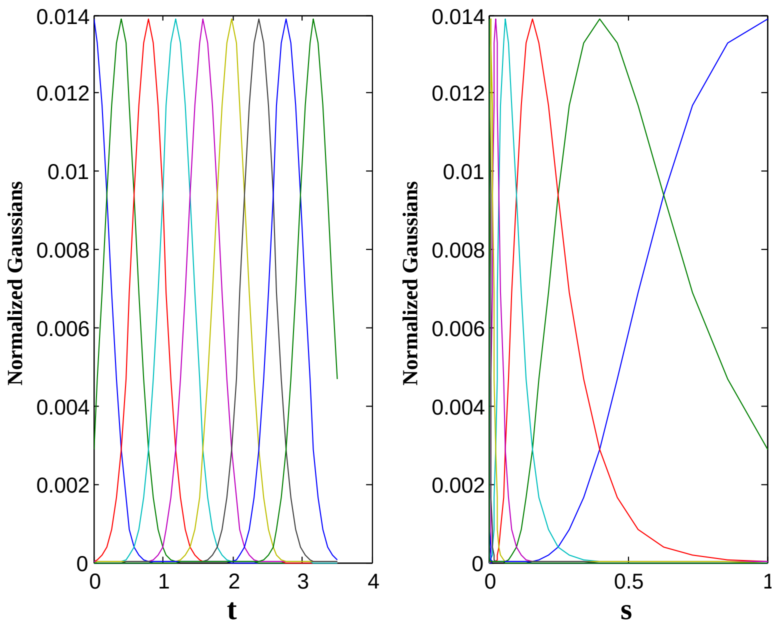

Figure 38. Gaussians in Time and Phase Domains, Generated in Time Domain

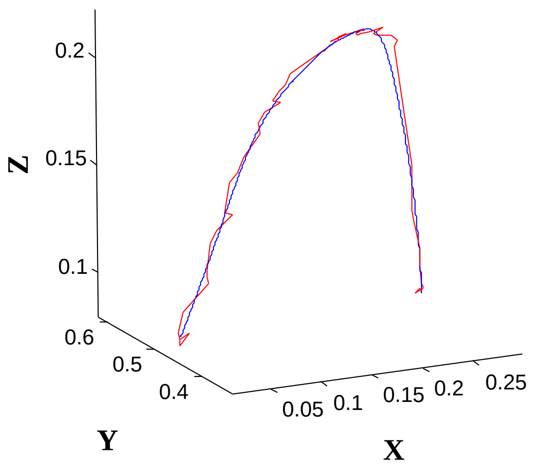

Figure 39. Initial and Generated Trajectories for Gaussians Generated in Time Domain

7.3 Trajectory Replication by a Robot

Before starting the experiment with the robot replication of the motion it is reasonable to define the set of trajectories that should be demonstrated and the possible application of those.

7.3.1 Trajectories under Consideration

First type of the trajectories includes straight line motions as shown in Figure 40

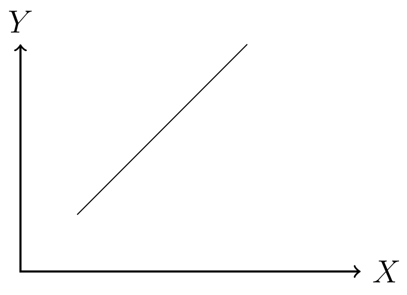

Figure 40. One-dimension Movement

which could be used as separate motions, for example, for lifting some objects or drawing lines, so that a robot could learn how to write. Besides that motions of that kind can be be used as components of a complex motion.

Second type includes parabolic motions, illustrated in Figure 41, for moving objects, putting them from one place for another, that could be used in industries.

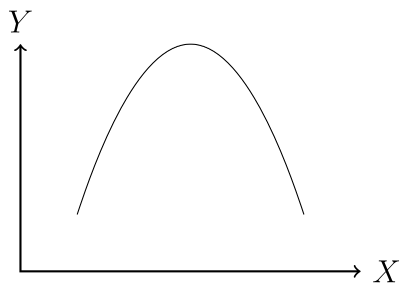

Figure 41. Parabolic Motion

If one considered medical area, then such kind of movement could be helpful in surgical operations.

7.3.2 Trajectory Test

Parabolic Motion proved to be possible to replicate. Given the initial and goal positions, the velocities for each point of the motion were generated.

Motion parameters are as follows:

Table 10. Parameters of the Motion

Parameter	Value
Move Time (s)	3.5
Number of Gaussians M	9
Width of Gaussians w	10
Spring Constant K	150
Damping Factor D	17
Temporal Scaling τ	3.3
Runge-Kutta Step(s)	0.007

In test 1, Figure 42, the duration of the generated motion and the goal position are the same as for the motion generated in Matlab.

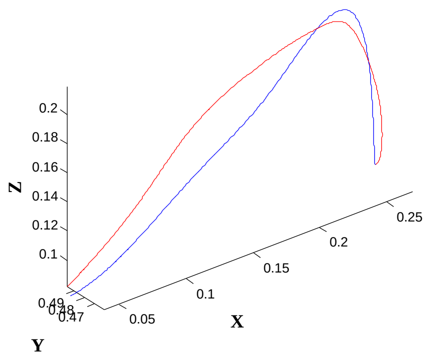

Figure 42. Generated Trajectories

In Figure 42 the motion generated in Matlab (red line) and the motion generated online for the robot (blue line) are represented. The difference between goal positions is not considerable, since the motion generation program does not take into account the performance of the robot, there might be some difference between the point where the robot was assumed to go and where it actually went.

Line Motion appeared to be difficult to replicate with the same setting of the DMP, it tends to take a shape of parabolic. When varying the damping and spring parameters, it occurred that if the motion was close to line then the trajectory was not able to converge to the goal position. But if the spring and damping factors were adjusted so that the motion converged to the needed position, the trajectory became curved. Since there are three DMPs, one for each dimension, a possible solution might be to use different damping and spring parameters for different dimensions.

7.3.3 Temporal and Space Invariance Tests

Spatial Invariance. In order to check the property of the Dynamic movement primitives, called spatial invariance, the series of tests was performed in which the goal position was changed. The system was expected to generate the trajectory with the shape close to the initial, but that would converge to the new goal position. The parameters of the motion are as shown in Table 11.

In test 2, Figure 43, time of the motion is the same as for the motion generated in Matlab, but goal position was changed. Figure 43 demonstrates how the change in the goal position affected the trajectory.

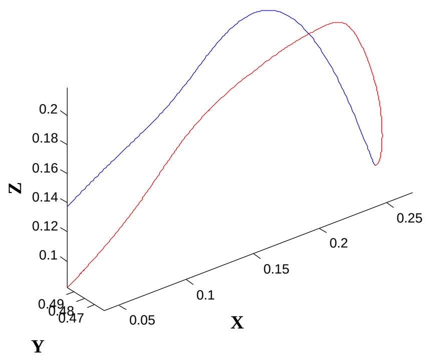

Figure 43. Generated Trajectories, Change of the Goal Position

In Figure 43 the motion, generated in Matlab (red line), and the motion, generated online for the robot with the changed goal position (blue line), are represented. As it is shown the generation program adjusted the Dynamic Motion Primitives to produce a trajectory that converged to the new goal position.

Temporal Invariance was checked by setting the recalculated parameter τ depending on the desired duration of the motion. Motion parameters are as follows:

Table 11. Parameters of the Motion

Parameter	Value
Move Time (s)	5
Number of Gaussians M	9
Width of Gaussians w	10
Spring Constant K	150
Damping Factor D	17
Temporal Scaling τ	4.75
Runge-Kutta Step(s)	0.007

Figure 44 demonstrates the trajectories with different durations.

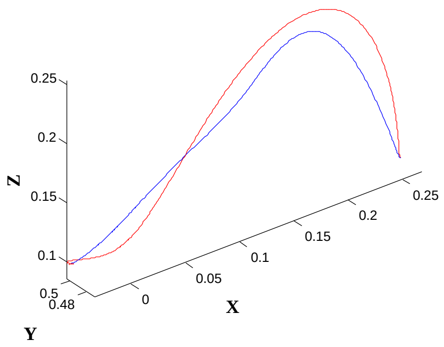

Figure 44. Generated Trajectories, Change of the Duration of the Motion

In Figure 44 red-line motion lasts for 5 seconds and blue-line one for 3.5 seconds. The shape of the motion remained the close to the original, although the trajectory was stretched in time. In order to get the motion close to the original one but with other duration, one would need to adjust also spring and damping factors.

8 CONCLUSION

During the research the first objective, studying the way of programming robot with visual input, was accomplished. A demonstrated motion was recorded by a stereo camera, the sequence of images were processed, the learning phase provided the weight coefficients of the motion and the generation procedure produced the velocities for robot joints.

Dynamic movement primitives concept proved to be an efficient tool for motion learning and generation. One of the advantages of the DMP is that the motion is stored as a set of weight coefficients, which allows to save memory. Another beneficial point is that the DMP makes it possible for a motion to be generated with different target positions and different duration, which are the space and temporal invariance properties of the DMPs.

The second objective was defined as the analysis of the influence of different parameters values on the trajectory generated was performed. The part of the experiments concerning the visual sensing proved that the methods for pointer detection and 3D reconstruction can be used in the system and give a predictable result.

During the second part of experiments the different parameters used in motion learning and generation were studied. It was shown that the number of Gaussians should be such that the whole motion was described without loss of information about the path and at the same time not to store excess weight coefficient that would not bring new knowledge. The centers of Gaussian basis functions should be equally distributed in time, and it is important to avoid unnecessary overlapping, which is regulated by the width parameter.

Damping and spring constants should be chosen for each application separately, since there seems not to be unique values that would satisfy all the types of motions. On the other hand, it might be possible to choose suitable parameters for the movements within one type. It is important to consider damping factor and spring coefficient together.

The parabolic trajectories were learnt and performed by the robot successfully, whereas for the line motions there were problems related to the difficulty in choosing one set of damping and spring parameters for all three dimensions. As a solution, different coefficients for different dimensions were suggested.

The third part of the experiments proved some of the properties of DMP, such as spatial and temporal invariance. The former implied the capability of the system to converge to the changed goal position. The latter implies that the duration of the motion can be varied changing the temporal coefficients.

The system designed includes the Matlab image analysis and learning program and the C++ motion generation software, which might be extended to add new functionality to the system.

As one of the improvements, obstacle avoidance could be implemented, which was not considered in the scope of the work. A good idea might be to build a library of motions, so that the robot obtained ability not only to perform the demonstrated motions but also to recognize the movements that were shown.

REFERENCES

[1] N. Roy J. Peters, R. Tedrake and J. Morimoto. Robot learning. *IEEE Robotics& Automation Magazine*, 16:19–20, 2009.

[2] C. Breazeal and B. Scassellati. Robots that imitate humans. *TRENDS in Cognitive Sciences*, 3:233–242, 2002.

[3] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 1999.

[4] K. Grochow R. Chalodhorn, D. B. Grimes and R. P. N. Rao. Learning to walk through imitation. *IJCAI'07: Proceedings of the 20th international joint conference on Artificial Intelligence*, pages 2084–2090, 2007.

[5] T. Asfour P. Pastor, H. Hoffmann and S. Schaal. Learning and generalization of motor skills by learning from demonstration. *IEEE International Conference on Robotics and Automation*, 2009.

[6] S. Schaal A. J. Ijspeert, J. Nakanishi. Learning attractor landscapes for learning motor primitives. *Advances in Neural Information Processing Systems 15 (NIPS2002)*, 9:1040–1046, 2002.

[7] S. Schaal A. J. Ijspeert, J. Nakanishi. Movement imitation with nonlinear dynamical systems in humanoid robots. *IEEE International Conference on Robotics and Automation (ICRA2002)*, 2002.

[8] I. Oakley S. Strachan, R. Murray-Smith and J. Angesleva. Dynamic primitives for gestural interaction. *Lecture Notes in Computer Science 3160*, pages 325–330, 2006.

[9] J. E. Slotine B. E. Perk. Motion primitives for robotic flight control. *ArXiv Computer Science e-prints*, 2006.

[10] J. Ijspeert S. Schaal, J. Nakanishi. Learning movement primitives. *International Symposium on Robotics Research (ISRR2003)*, 2004.

[11] D. Park H. Hoffmann, P. Pastor and S. Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. *2009 IEEE International Conference on Robotics and Automation*, pages 2587–2592, 2009.

[12] P. Pastor H. Hoffmann and S. Schaal. Dynamic movement primitives for movement generation motivated by convergent force fields in frog. *Adaptive Motion of Animals and Machines (AMAM)*, 2008.

[13] B. Fajen and W. Warren. Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology: Human Perception and Performance*, 29:343–362, 2003.

[14] A. S. Hadi S. Chatterjee. *Regression analysis by example*. Wiley-Interscience, 2006.

[15] O. Silven J. Heikkila. A four-step camera calibration procedure with implicit image correction. *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112, 1997.

[16] http://www.vision.caltech.edu/bouguetj/calib _doc/index.html.

[17] http://mysite.du.edu/ jcalvert/optics/stereops.htm.

[18] Emanuele. Trucco. *Introductory techniques for 3-D computer vision*. Upper Saddle River (NJ), 1998.

[19] P. F. Sturm R. I. Hartley. Triangulation. *Lecture Notes In Computer Science*, 970:190–197, 1995.

[20] P. Sturm R. I. Hartley. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.

[21] Bahvalov N. *Numerical Methods*. M. Nauka, 1975.

[22] Y. C. Sirma. Kinematic analysis for robot arm. Master's thesis, Yildiz Technical University, 2009.

[23] www.tirbosquid.com/3d-models/maya-robotic arm/281750.

[24] M. Wahde. Lecture notes, "humanoid robotics". *Chalmers University of Technology*, 2009.