technische universität
dortmund

Technische Universität München

TU Dortmund University and TU München University
Computational Intelligence Research Group
Dynamic Human Robot Interaction Group
Prof. Dr. Günter Rudolph
Prof. Dongheui Lee

Master Thesis

Robotic Calligraphy: Learning From Character Images

by

Omair Ali

Supervisors:	Affan Pervez, M.Sc.
Internal Examiner:	Prof. Dr. Günter Rudolph
External Examiner:	Prof. Dongheui Lee

Dortmund, October 06, 2015

DECLARATION

I hereby declare and confirm that the master thesis

Robotic Calligraphy: Learning From Character Images

is entirely the result of my own work except where otherwise indicated.

On the next page of this document, I have acknowledged the supervision, guidance and help I received from Prof. Dongheui Lee, Ph.D, M.Sc. Affan Pervez and Prof. Dr. Günter Rudolph.

Dortmund, October 06, 2015

Omair Ali

ACKNOWLEDGMENTS

This work was done in the department of Dynamische Mensch-Roboter-Interaktion für Automatisierungstechnik, Technische Universität München and Computational Intelligence Research Group, Technische Universität Dortmund
.

I would like to thank Prof. Dongheui Lee, Ph.D. for giving me the opportunity to work in this field of research. A special thanks to M.Sc. Affan Pervez, without whom, this work would not have been possible. His constant help and guidance was vital in completion of this work. I would further like to thank Prof. Dr. Günter Rudolph for his constant support and suggestions of improvement. Their recommendations and ideas helped me to compile the work in the best possible way.

In the end, I would like to thank my family for their constant support and encouragement.

Dortmund, October 06, 2015

Omair Ali

ABSTRACT

Known as the ”art of combining strokes to form complex letters”, Chinese and Korean Calligraphy has always been of vital significance throughout the history of these ancient civilizations. Its exquisite epitome of elegance is depicted through the strokes. A calligrapher assimilates the skill of drawing these strokes. Once this dexterity is mastered, it can be used to compose a calligraphic letter afterwards. The very notion of calligraphy is endeavored to be implemented on robots in this research. Korean calligraphy is the nucleus of this research work and the image of the calligraphic character is used as an input. Unlike humans or calligraphers, the robots are unacquainted with combination of various strokes used to draw a calligraphic letter. Hence it is quite cardinal and arduous to fragment the calligraphic letter into diverse strokes used to draw it. A novel approach ensuing the concept of Gaussian Mixture Model (GMM) is proposed to segregate the input image into assorted strokes. Once the image is partitioned into different strokes, the character is reproduced by blending the strokes in right order and right position using Gaussian Mixture Regression (GMR). When character is redrawn, Dynamic Movement Primitive (DMP) is applied to acquire the scale and temporal in-variance in the stroke trajectory. The parameters learned through DMP are iteratively streamlined using Reinforcement Learning until they converge or don’t revamp any further.

CONTENTS

LIST OF FIGURES

CHAPTER

ONE

INTRODUCTION

In the contemporary era, extensive research work in field of robotics is materializing to give birth to an artificial being, intelligent enough to compete with human counterparts. Without the aptitude of learning and transforming, no artificial entity is entitled to be intelligent. To inculcate these two basic traits of intelligent agent in artificial body, the outlook must be inspired from the most intelligent network on the face of the earth, the human's. Studies reveal that three strategies opted in Organic world to learn and transform are; Unsupervised Learning, Supervised Learning and Reinforcement Learning. The abstract ideas of all these three schemes are exploited in this research work for it is intrinsically quite captivating.

Unsupervised Learning pivots on retrieving some useful information from the data under observation rather than us explicitly stipulating the source or use of the data. It strives for finding pattern and structure in the incomplete data. Where as in Supervised Learning, labeled data is employed for learning. Reinforcement learning can be defined as optimizing the performance of a system by trial and error

The cornerstone of this research work is to traverse the idea of Korean Calligraphy on robots. Korean Calligraphy is an art of combining strokes to form complex letters. A calligrapher learns the skill of drawing these strokes rather than explicitly drawing the complete letter itself. The learned strokes can be used to draw complex letters afterwards.

The idea is to use the image of the calligraphic character and segregate it into different strokes used to form the letter. As stroke is the basic unit of a symbolic language, thus the basis of character analysis is stroke extrac-

tion. But unlike humans or calligraphers, the robots are unacquainted with combination of various strokes used to draw a calligraphic letter. As it is quite predominant, fundamental and strenuous in calligraphy to identify the strokes, it has been the area of interest for many researchers in recent time.

Existing stroke extraction approaches can be classified into two categories. Bottom Up Approach and Top Down Approach. Former extracts the strokes directly from binary image while the latter one exploits the skeleton of the image to extract the strokes. In bottom up approach Y.Sun et al. [17] used geometric properties of the contour of the character to extract the stroke. It extracts sub-contours first and then combine them to retrieve the strokes. This approach is dependent on the definition of basic strokes in Chinese Calligraphy. Considering the existing five basic strokes, this approach produces many failed cases of wrong stroke extraction which is shown in Figure 1.1. Y. M. Su et al. [15] [16] implemented Gabor filters and SOGD filters respectively to extract strokes in four directions (i.e. 0°, 45°, 90° and 135°). However, this approach is unable to extract the strokes beyond these directions.

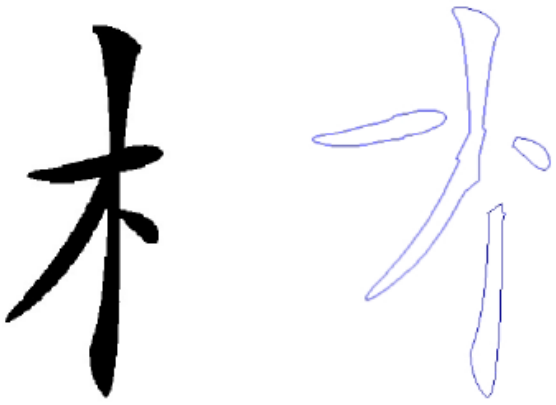


Figure 1.1.: (a)Input character and (b)Extraction result [17]

L. Wang et al. [20] used grey scale image to extract the strokes based on grey-scale surface features like peak, pit, saddle and ridge etc. There implies quite a high computational cost for this method. Another contour based approach is applied by K. YU et al. [22] in which they used fast-CPDA to identify the corners from the contour. Afterwards the half-edge coloring graph is introduced to connect the contours that reside on the same side of a

stroke. Matching contours on both sides of stroke finally results in complete stroke.

In this research work we have proposed a novel idea of extracting strokes from Korean Calligraphic letters based on the Gaussian Mixture Model (GMM). Given the image of calligraphic letter as an input, it is first thinned and then converted into data points. Afterwards, GMM is applied to cluster the data points. Expectation Maximization algorithm is then implemented to learn the parameters of GMM. Bayesian information criteria (BIC) is employed to determine the number of components required by GMM to represent the data. Once GMM is learned, the strokes are retrieved by combining respective components(gaussians) based on proposed algorithm that represent the stroke and then Gaussian Mixture Regression (GMR) is applied to get the smooth trajectory for the extracted strokes. As soon as strokes are retrieved, the trajectories encoded by GMM are re-encoded using Dynamic movement primitives (DMPs) to reduce the number of parameters to learn during reinforcement learning. The calligraphic brush is modeled as droplet to redraw the character as a proof of concept. Then the learned dexterity is implemented on KUKA LWR robot. The reinforcement learning is brought into service to improve iteratively the reproduction of the drawn letter.

1.1. Highlights and Layout of the Report

Chapter 2 spans the theoretical prerequisites required to follow the proceedings of the report. In Chapter 3 the proposed idea for extracting strokes is briefly described. Chapter 4 elaborates the complete work with help of examples and results. The conclusion is presented in last chapter 5.

CHAPTER

TWO

THEORETICAL PREREQUISITES

In this chapter, the theoretical framework needed to fully understand this research work are elaborated. First, a general summary of basic concepts coupled with the modeling of Gaussian Mixture Model (GMM) is provided. The abstraction which entails to fully comprehend the modeling of GMM are:

1. *Gaussian Mixture Model* (GMM)
2. *Expectation Maximization Algorithm* (EM Algorithm)
3. *Gaussian Mixture Regression* (GMR)

Afterwards, an insight regarding Dynamic Movement Primitives is given. Then basics of Reinforcement Learning are briefly touched at the end of this chapter.

2.1. Gaussian Mixture Model (GMM)

A *Gaussian Mixture Model* [12], [4] is considered to be a simple linear superposition of K Gaussian components, which are combined to provide a multimodal density. It is quite beneficial for modeling data that is obtained from one of several groups e.g. they can be utilized to implement a color based segmentation of an image or for clustering purpose. It is assumed to be like kernel density estimates with components much less than total number of data points

A GMM is a probability based model. It presumes that all the data points are originated from a mixture of a finite number of Gaussian distributions with known parameters. A GMM can be written as a linear superposition of these finite number of Gaussian distributions as:

$$p(\mathbf{y}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{y}|\mu_k, \Sigma_k) \tag{2.1}$$

Where:
K = fixed number of components
π_k = probability of k^{th} component
μ_k = mean of the k^{th} component
Σ_k = variance of the k^{th} component

The parameters π_k, μ_k and Σ_k of the Equation (2.1) are approximated by maximizing the expected value of incomplete data log-likelihood. To acquire the estimation of these parameters, first Latent Variables Z are presented which is quite significant in providing the deeper insight into the distribution and EM algorithm. Assume Z to be a K dimensional variable with one of the K representations in which a particular element z_k is one and rest of the elements are equal to zero. The value of z_k therefore fulfills $\sum_{k=1}^{K} z_k = 1$ and $z_k \in \{0, 1\}$. Hence, the total number of representations of the latent variable are K where each representation relates to a certain class or basis function in the GMM. Mixing probabilities π_k specify the marginal distribution over Z such that:

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$$

The conditional probability of data point x provided a particular value of Z is Gaussian distributed and is given by:

$$p(y|z) = \prod_{k=1}^{K} (\mathcal{N}(y|\mu_k, \Sigma_k))^{z_k} \tag{2.2}$$

The joint probability distribution is obtained by $p(Z)p(y|Z)$, and the marginal distribution of x is then evaluated by adding the joint probability distribution over all possible states of Z which is written as:

$$p(y) = \sum_{Z} p(Z)p(y|Z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mu_k, \Sigma_k) \tag{2.3}$$

The equation achieved in (2.3) is the equivalent representation of a GMM by explicitly involving a latent variable Z. Let $\mathbf{y} = \{y_1, y_2, ...y_n\}$ be the given set of observations and data is to be modeled using a mixture of K Gaussian components. The log-likelihood function of the data on the approximated model is acquired as:

$$ln\mathcal{L}(\pi, \mu, \Sigma|\mathbf{y}) = lnp(\mathbf{y}|\pi, \mu, \Sigma) = \sum_{i=1}^{n} ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(y_i|\mu_k, \Sigma_k) \right\} \tag{2.4}$$

The model corresponding to the values of parameters π,μ and Σ that maximize the function (2.4) for i.i.d dataset **y** best represents the data. The maximization of (2.4) is very arduous because of the involvement of addition terms inside the function of log. In fact, no closed form solution of the above equation can be found if its gradient is put equal to zero. Hence a substitute view of the likelihood, known as the completed log likelihood of the data is introduced. Assume that in addition to the observed data y, latent variables also known as hidden variables Z or the class labels are also given. Considering the likelihood of the complete dataset $\{y, Z\}$, the likelihood function can be written as:

$$lnp(\mathbf{y}, Z|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ki} \left\{ ln\pi_k + ln\mathcal{N}(y_i|\mu_k, \Sigma_k) \right\} \tag{2.5}$$

Where, z_{ki} shows the responsibility of the k^{th}component in producing the i^{th}data. Its value is known before hand in case of completed data. While comparing the Equation (2.5) with Equation (2.4), it can be seen that summation over K term is interchanged with logarithmic term, making the equation a viable closed form solution. The completed likelihood and the incomplete data likelihood can be related using the following identity:

$$\mathcal{L}(\pi, \mu, \Sigma|\mathbf{y}, Z) = \mathcal{L}(\pi, \mu, \Sigma|\mathbf{y}) + \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ki} log \frac{w_{ki}}{\pi_k} \tag{2.6}$$

2.2. Expectation Maximization (EM)

Expectation Maximization (EM) [7] is a strong tool that, with the help of latent variables, is employed to find the maximum likelihood solution for models. It is purely an iteration based method for finding the maximum likelihood, which implies that the means, co-variances and mixing coefficients of the model are initialized with some appropriate initial values and then iteratively refined through two steps namely:

1. *Expectation Step* (E-Step)
2. *Maximization Step* (M-Step)

E Step
In this step, the most recent values of parameters are used to calculate the Posterior probabilities for the data. Posterior probability or responsibility can be considered as the probability of i^{th} data produced by k^{th} component i.e. $p(z_{ki} = 1)$. It is mathematically formulated as:

$$w_{ki} = \frac{\pi_k \mathcal{N}(y_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(y_i|\mu_j, \Sigma_j)} \tag{2.7}$$

M Step
In M step, the parameters that include mean, mixing coefficient and co-variance are updated or re-estimated based on Posterior probabilities calculated in E-Step by using:

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} w_{ki} y_i \tag{2.8}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} w_{ki} (y_i - \mu_k)(y_i - \mu_k)^T \tag{2.9}$$

$$\pi_k = \frac{N_k}{n} \tag{2.10}$$

where

$$N_k = \sum_{i=1}^{n} w_{ki}$$

The above mentioned two steps are repeated until the Likelihood function converges or there is not further significant change.

2.3. Learning GMM Using EM

Approximation and estimation of unknown parameters and the number of components that best fit the data are required in unsupervised learning of mixture models for a given set of observations. Expectation Maximization is an iteration based algorithm which ensures to reach local maximum for Maximum likelihood estimate of the parameters. To estimate the optimal number of components K in mixture model that best represent the data is of great significance due to following reasons. Number of components greater than the optimal number results into over-fitting of the data which leads to poor performance of the model. To get the low bias, accurate estimation and low variance, optimal number of components are important to be known as shown in Figure 2.1

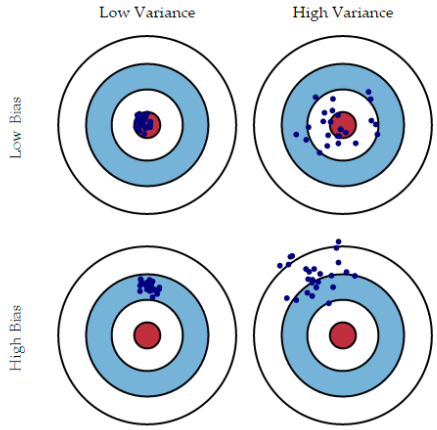


Figure 2.1.: Exemplification of the Bias and Variance. The higher the Bias is, the lower the accuracy from target is. With high variance, the target might be acquired but it can increase the variance in the data resulting in over-fitting of the model. The best model has low unexplained variance and the bias. [onl]

Components lower than the optimal number results in less accuracy which can be improved by enhancing the number of components. Increasing the number of components increases the variance in the estimated model which should also be circumvented. Hence optimal number of components are required to get this trade-off which is shown in Figure 2.2

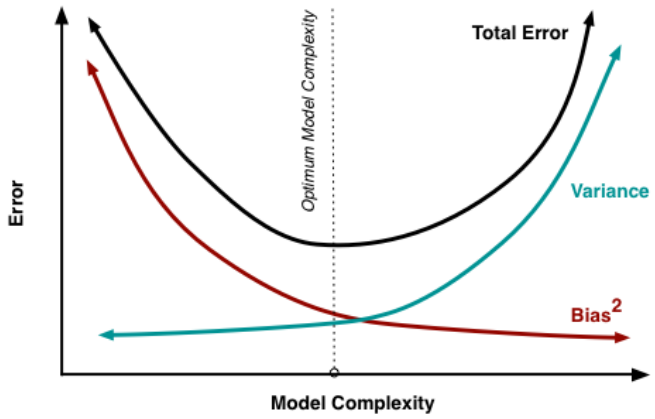


Figure 2.2.: Trade-off between bias and variance [onl]

2.3.1. Estimating the Optimal Number of Components

The information regarding the number of clusters and the clustering method [9] is always an important concern, for it plays a vital role in estimating the performance of a model. There are numerous methods for estimating the performance of the model against the number of components used to estimate the data. Few are mentioned as:

1. *Silhouette score*
2. *Akaike information criterion* (AIC)
3. *Bayesian information criterion* (BIC)
4. *Minimum Description Length and Minimum Message Length* (MML/MDL) [8], [19]

Silhouette This technique first explained by Peter J. Rousseeuw in 1986 [10], [13?] uses the average Silhouette to interpret and validate the consistency of the clusters of the data. It represents graphically that how perfect the data in a cluster is clustered.

In this research work, BIC is employed to estimate the optimal number of components in a Gaussian Mixture Model

Bayesian Information Criterion (BIC) The **BIC** [14] selects the model among the finite set of models. Like Akaike Information Criterion (AIC) [3], it is dependent on the likelihood of the data and possess a term that penalizes the estimate in case of unwanted components which causes over-fitting of the

data. The penalty term is non-linear and take into consideration the sample size and total number of observations that estimated the model. It is mathematically written as:

$$BIC = D_k ln(n) - 2ln\mathcal{L} \tag{2.11}$$

where,

D_k = number of parameters in K-component model $\mathcal{L}$ = likelihood of the data on estimated model

In other words, it is a trade-off between complexity and the likelihood of the model. For number of components K for which BIC is lowest, are the optimal number of components.

2.4. Gaussian Mixture Regression (GMR)

Linear combination of Gaussian and conditional Gaussian densities are the basis of GMR [5]. Once, the input data is modeled by GMM, Gaussian Mixture Regression gives a quick and analytic way to get smooth output trajectories in addition to covariance matrices that explain the change and correlation among various variables. At the time of fitting GMM on dataset, there is no difference between input and output variables. In a special case of trajectory learning, time can be referred to as an Input variable $\mathbf{y}^I$ and position can be considered as output variable $\mathbf{y}^O$ and the goal is to reproduce the trajectory ($\mathbf{y}^O$) or part of it provided a time step $\mathbf{y}^I$. The data $\mathbf{y}$ modeled using GMM can be segregated into input $\mathbf{y}^I$ and output variables $\mathbf{y}^O$.

The probability of a data point y = $[y^I, y^O]$ that corresponds to GMM distribution is shown as.

$$p(\mathbf{y}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{y}|\mu_k, \Sigma_k) \tag{2.12}$$

where the parameters: means μ_k and co-variance matrices Σ_k are composed of input and output components and are represented as:

$$\mu_k = \begin{bmatrix} \mu_k^I \\ \hline \mu_k^O \end{bmatrix}, \begin{bmatrix} \Sigma_k^I & \Sigma_k^{IO} \\ \Sigma_k^{OI} & \Sigma_k^I \end{bmatrix} \tag{2.13}$$

The distribution of output variable $\mathbf{y}^O$ given the corresponding input variable $\mathbf{y}^I$ and Component k is written as:

$$p\left(y^O|y^I,k\right)=\mathcal{N}(\zeta_k,\varsigma_k)$$

where

$$\begin{aligned}\zeta_k &= \mu_k^O+\Sigma_k^{OI}\left(\Sigma_k^I\right)^{-1}\left(y^I-\mu_k^I\right)\\ \varsigma &= \Sigma_k^O-\Sigma_k^{OI}\left(\Sigma_k^I\right)^{-1}\Sigma_k^{IO}\end{aligned} \tag{2.14}$$

Taking into consideration the whole Gaussian Mixture model, the probability distribution of y^O given y^I is written as:

$$\begin{aligned}p\left(y^O|y^I\right) &\approx \Sigma_{k=1}^K h_k\mathcal{N}(\zeta_k,\varsigma_k)\\ where:&\\ h_k &= \frac{\pi_k p(y^I|k)}{\Sigma_{j=1}^K\pi_j p(y^I|j)}\end{aligned} \tag{2.15}$$

Hence, Gaussian Mixture Regression can be used to regenerate the skill, once an optimal Gaussian Mixture Model is at hand.

2.5. Dynamic Moment Primitives (DMPs)

Dynamic Movement Primitives [11] are utilized to encode a trajectory. They were first proposed by Stefan Schaal's Lab in 2002. Trajectories encoded using GMM have high number of free parameters to update in reinforcement learning. So to reduce the number of free parameters while applying reinforcement learning, DMPs are introduced and the trajectories encoded using GMM are now encoded using DMPs. The motion generated using them are highly stable and it can be used to encode both discrete and rhythmic motions. In this research work, DMPs were used only for discrete movements.

It has a hidden system known as canonical system

$$\dot{z}=u(z)$$

which drives the transformed system

$$\dot{x}=v(x,z,g,\theta)$$

for all degrees of freedom (DoFs) i, where θ is the parameter to be learned. Desired positions x_1 and the velocities $\dot{q} = \tau x_2$ of the robot either in joint space or in the task space are used as the states x = [x_1,x_2] of the transformed system. Accelerations are calculated by $\ddot{q} = \tau \dot{x_2}$. These are converted into torques or forces required by the robot using a controller. It ensures the stability of the movement. As only discrete movements are considered in this research work hence only the corresponding policies of DMPs are provided. In case of only single discrete movement the canonical form u can be represented as: $\dot{z} = -\tau \alpha_z z$. Here τ = 1/T time constant where T is the total duration of a MP. The value of the parameter α_z is selected such that it ensures the safe termination of the movement. It is chosen such that value of z approaches zero at T. The transformed function v is shown in the form:

$$\dot{x_2} = \tau \alpha_x (\beta_x (g - x_1) - x_2) + \tau \mathbf{A} f(z)$$

$$\dot{x_1} = \tau x_2$$

Here the value of τ is same as that of the canonical system. The values of the parameters α_x and β_x are chosen so that the system is critically damped at A = 0. g is goal and f is a transformation function and A is a diagonal matrix that modify the amplitude. The matrix A ensures linear re-scaling when the the goal is varied. The transformation function varies the output of the transformed system and is given by:

$$f(z) = \sum_{n=1}^{N} \psi^n(z) \theta^n z$$

Here θ^n contains the nth set of adjustable parameters for all DoFs. N is the number of parameters per DoF. z is the value of canonical system. $\psi^n(z)$ are the corresponding weights which are given by:

$$f(z) = \frac{exp(-h_n^{-2}(z - c_n)^2)}{\sum_{m=1}^{N} exp(-h_m^{-2}(z - c_m)^2)}$$

Each DoF has a separate DMP and the parameters for each DoF are learned separately. The cost function that needs to be minimized for all parameters θ^n is given in form of weighted squared error: $e_n^2 = \sum_{t=1}^{T} \psi_t^n (f_t^{ref} - z_t \theta^n)^2$.
The error written in matrix form is given as:

$$e_n^2 = (f^{ref} - Z\theta^n)^T \psi (f^{ref} - Z\theta^n)$$

Here θ^n is given by:

$$\theta^n = (Z^T \psi Z)^{-1} Z^T \psi f^{ref}$$

Here f^{ref} contains the values of f_t^{ref} for all samples.

2.6. Reinforcement Learning (RL)

Reinforcement learning can be defined as optimizing the performance of a system by trial and error. In this research EM based RL mentioned in [6] is employed. In this RL the update policy is given by:

$$\theta^{(n)} = \theta^{(n-1)} + \frac{\sum_m^M r(\theta_m)[\theta_m - \theta^{(n-1)}]}{\sum_m^M r(\theta_m)} \tag{2.16}$$

where:
$\theta^{(n)}$ = Learned parameters of DMP.
$\theta^{(n-1)}$ = Old parameters of DMP.
r(θ_m) = Reward.
θ_m = Perturbed parameters of DMP.
The difference between perturbed and old parameters of DMP [11] $\theta_m - \theta^{(n-1)}$ are Exploration terms and give relative exploration between parameters in mth iteration and current parameters. In this research, reward (r(θ_m)) is the correlation between original image and image produced by robot.
The exploration terms are weighted by the respective correlation r(θ_m) and normalized by the sum of other correlations. They are sampled from normal distribution. It is expressed as co-variance matrix and updated as:

$$\sum{}^n = \frac{\sum_m^M r(\theta_m)[\theta_m - \theta^{(n-1)}][\theta_m - \theta^{(n-1)}]^T}{\sum_m^M r(\theta_m)} + \sum\nolimits_0 \tag{2.17}$$

where:
$\sum_0$ = is a regularization term corresponding to a minimum exploration noise used to avoid premature convergence to poor local optima.
The equations mentioned form a multivariate normal distribution at each iteration. The covariance can guide the exploration and it also provides information about the neighborhood of the policy solutions.

CHAPTER

THREE

STROKE EXTRACTION

This chapter is the nucleus of this research, for it endeavors to develop an approach to segment the input calligraphic image into the strokes that are required to draw that image. The focus of the work is primarily on partitioning the Korean Calligraphic Letters into respective strokes.

Segregating the image into correct strokes is the most important and the arduous part of Robotic calligraphy. Without accomplishing it, robot is unaware of the stroke combinations used for a particular character. Retrieving correct strokes from character images have following different phases:

1. Thinning
2. Image to Data Points
3. Gaussian Mixture Model on data set
4. Extracting End points of the Stroke
5. Stroke Retrieval

3.1. Thinning

Thinning is a morphological operation. It is usually applied on the image to thin it by reducing the number of pixels in it.

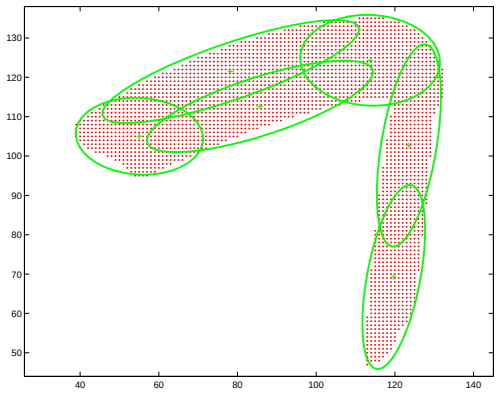


Figure 3.1.: It has two parallel overlapping Gaussian Components on extreme left side of the character.

In this phase, the image is thinned or in other words the data points are reduced to get a thinned image. This is quite an important phase in the process because it ensures in the later stage that when Gaussian Mixture Model is applied on the data points, it does not model the thickness of the character and there are no parallel or random Gaussian Components. Presence of parallel Gaussian components as shown in Figure 3.1 make it quite difficult to extract the strokes and also the thinning is helpful in reducing the computation time because it does not contain all the data points.

Hence thinning is required to extract the strokes with less computational power. After thinning the image obtained is as shown in fig 3.2

There is an established function for thinning operation in Matlab. It is available in its image processing toolbox therefore it is not required to write a new algorithm for thinning operation. The new data set obtained after thinning is used in processes to follow.

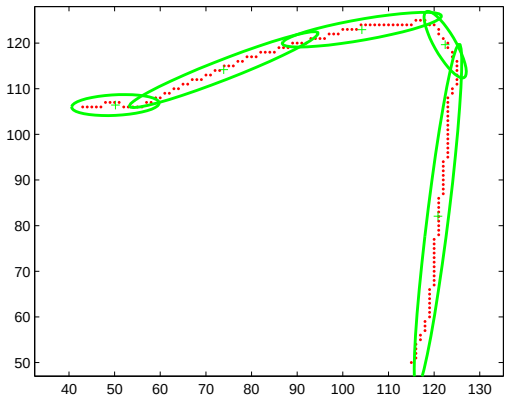


Figure 3.2.: It has no parallel Gaussian Components

3.2. Image to Data Points

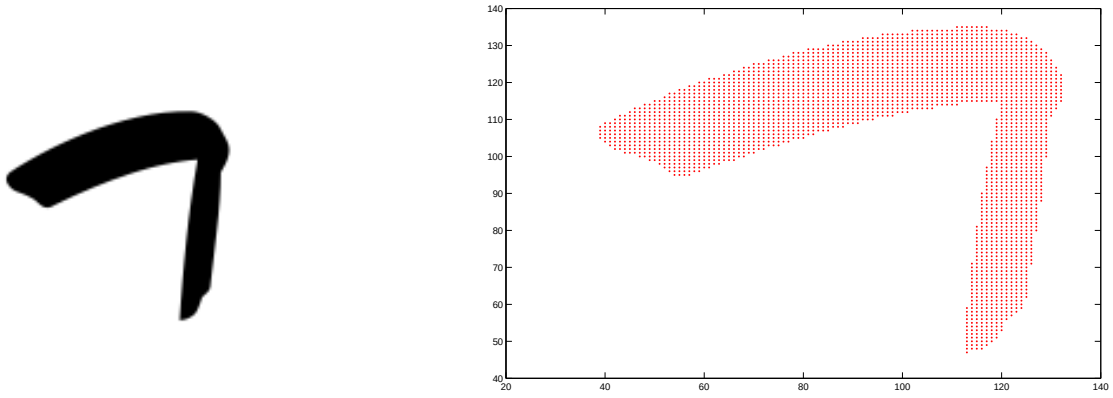
This is the second phase in stroke extraction process. In this phase the input image is converted into data points so as to make the image compatible for applying Gaussian Mixture Model later on.

It is the most basic step in the process and is shown in fig3.3

The algorithm for converting image to data points is given in 1

Algorithm 1 Image to Data Points

```
1: procedure IMAGE 2 DATAPOINTS
2:     Input Image
3:     for Each row of the image  do
4:         for Each column of the image do
5:             if The value of pixel is <threshold then
6:                 Store its coordinates in matrix
7:             else
8:                 Ignore the pixel
9:             end if
10:        end for
11:    end for
12: end procedure
```



(a) Input Image **(b)** Data Points

Figure 3.3.: It shows the conversion of input image into data points

3.3. Gaussian Mixture Model on Data set

Once the input image is under gone the first two phases of the process, it is all set to apply Gaussian Mixture Model to represent the complete data set. Then GMM is applied on the data points by using Equation (2.1). While applying GMM on the data set, its parameters are learned using E and M-steps of EM-Algorithm as shown in Equations (2.7),(2.8),(2.9),(2.10).

The optimal number of components that best represent the data are calculated using Bayesian Information Criteria which is shown in Equation (2.11). The algorithm starts with some fixed high number of Gaussian components K representing the data as shown in Figure 3.4 and runs for K times.

In each iteration, it calculates the BIC number using Equation (2.11) and merges one Gaussian component to other to reduce the number of components by 1. By the end of last iteration there is only one Gaussian Component left which is shown in Figure 3.5.

BICs of all these iterations are compared and the one with least value is selected to represent the data, for it has the optimal number of components to best represent the data which is shown in Figure 3.6

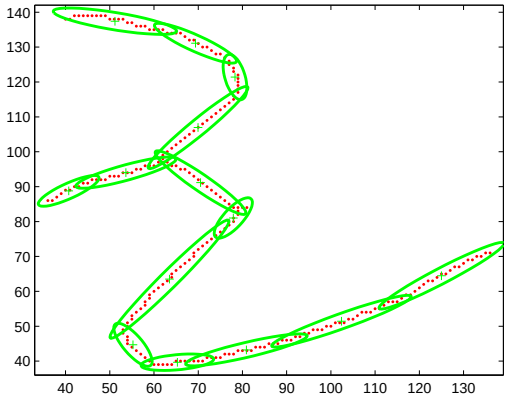


Figure 3.4.: GMM starting with high number of components

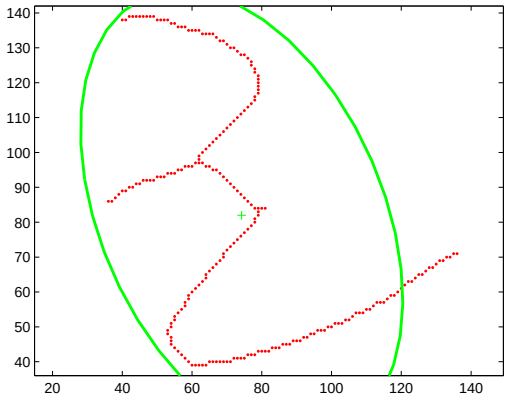


Figure 3.5.: GMM with only one component

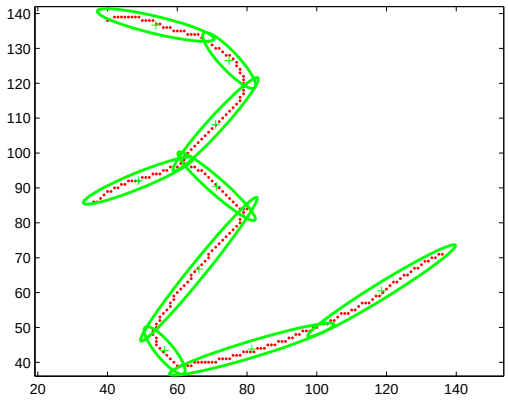


Figure 3.6.: GMM with optimal number of components. For this character it is 9.

After the GMM is employed and learned, the image is ready for the main and the most important phase of this process, the stroke extraction phase

3.4. Extracting End points of Stroke

This phase deals with extracting the end points of the stroke. Once the GMM is learned for a character, next phase is to connect the respective components that represent a stroke. This is the most significant phase of the process because once the end points of a stroke are retrieved, then its complete trajectory can be extracted using GMR.

To continue with the procedure, first the major-axes of the Gaussian components are calculated using the co-variance matrix of the components. The co-variance matrix Σ_k of the k^{th} component whose major-axis is to be determined can be shown using Singular Value Decomposition (SVD) as:

$$\Sigma_k = V D V^T$$

where,

V = $[v_1, v_2]$

Matrix whose columns are eigenvectors of Σ_k and it satisfies Σ*V = D*V

D = Diagonal matrix that contains the eigenvalues of Σ_k along the main diagonal and for 2x2 square matrix, it is written as:

$$\begin{bmatrix} \lambda_{11}^2 & 0 \\ 0 & \lambda_{22}^2 \end{bmatrix}$$

Here values of λ are calculated by solving $det(\Sigma_k - \lambda I) = 0$
For each eigenvalue, its corresponding eigenvector is calculated by solving the following equation:

$$\Sigma_k * v_1 = \lambda_{11} * v_1, \Sigma_k * v_2 = \lambda_{22} * v_2$$

Once the eigenvalues matrix D and eigenvectors matrix V are calculated, arrange them in such an order that vector v_1 is major eigenvector and λ_{11} is the corresponding biggest eigenvalue. Then the two end points of the major axes of the Gaussian components are calculated by using the Equation (3.1)

$$\begin{aligned} endpoint1 &= \mu_k + 2 * \lambda_{11} * v_1 \\ endpoint2 &= \mu_k - 2 * \lambda_{11} * v_1 \end{aligned} \tag{3.1}$$

Here μ_k is the mean of gaussian component k whose major axis is being determined. As most of the points lie in 2σ standard deviation, therefore factor of 2 is multiplied with eigenvalues and eigenvectors in the Equation (3.1). Now by joining the two endpoints calculated above, major axis of the gaussian component is obtained. This procedure is done as many time as number of components K in GMM are and store the endpoints of all the major-axes . Once the major axes of all the components are found, they are drawn on the GMM as shown in Figure 3.7

After the major axes have been retrieved, their slopes are calculated using the basic slope formula. Intersection of major axes of two components (say major-axes 1 and 2) are calculated using the Equation 3.2. After calculating all the intersection points, they are also drawn on the GMM as shown

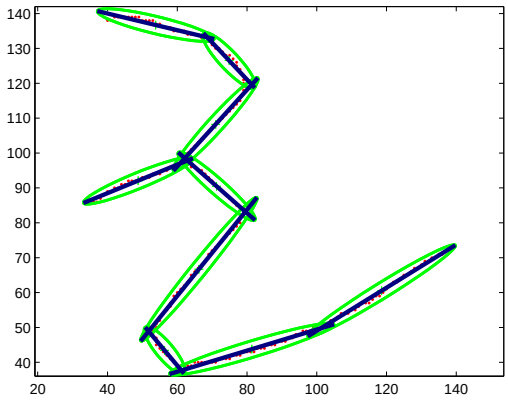


Figure 3.7.: GMM with major axes of all the Gaussian Components, shown in blue color

in the image 3.8

$$\begin{aligned} x_{intersect} &= \frac{b_2 - b_1}{m_2 - m_1} \\ y_{intersect} &= (m_1) * (x_{intersect}) + b_1 \end{aligned} \tag{3.2}$$

where:

b_1 = y-intercept for major-axis 1
b_2 = y-intercept for major-axis 2
m_1 = slope of major-axis 1
m_2 = slope of major-axis 2

Till this point, the major-axes of all Gaussian Components, their end points, their slopes and their intersections with major-axes of other gaussian components have been determined. From now on-wards, ”algorithm” is referred to as an ”agent” that is searching for a path or trajectory to travel from a start point to the end point that in the end represent a stroke. Now the agent is fully prepared to begin its search process for the end points of the stroke.

To begin the search process, there must a starting point for an agent to start its search. So in this case the start point is the point among all the end

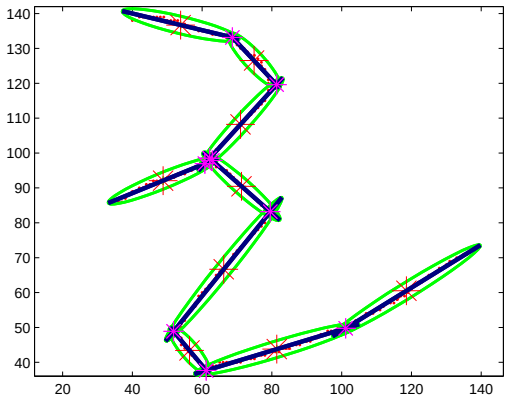


Figure 3.8.: GMM with major axes of all the Gaussian Components, their intersections and their mean points

points of major-axes that has the biggest y-coordinate. Hence it starts by looking for the point with largest y-coordinate among all end points of the major-axes as shown in Figure 3.9

After the starting point is determined, it is considered as the current-point (the point where the agent is) and the axis is considered as the current-axis (the axis where the agent is) and now the agent must find the next-point to jump or to reach to in order to carry on its search process. When the agent reaches one of the end points of a major-axis whose flag is not high, it must jump to its other end point as next search step or next current-point. Once both the end points of a major-axis are visited, its flag is turned high indicating that this major-axis has been visited and not be included in further search process. As for now the agent is at one of the end points of the current-axis and as it is the beginning of the search process, the flag of that major-axis is not high, so the agent must jump to the its other end point making it the new current-point as shown in Figure 3.10 and turn its flag high.

Now for the next current-point or step, the agent considers all the major-axes whose flags are not high or which are not visited before. To jump from one major-axis to the next in search of new current-point, the agent takes into consideration a lot of aspects. It considers all the major-axes that intersect with the current-axis and all those major-axes which do not intersect but one of the their end points lie in the Gaussian distribution of current-

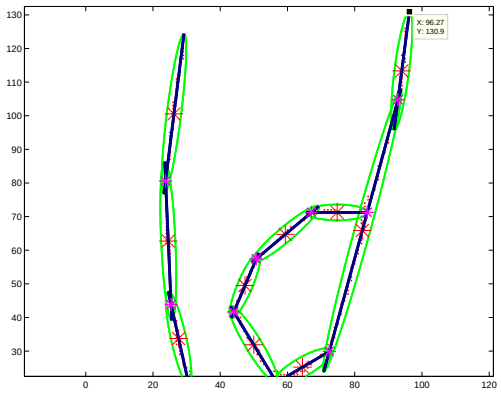


Figure 3.9.: Starting point of the agent is point with biggest y-coordinate. In this case it is the point with value of y = 130.9

component (Gaussian component corresponding to current-axis) as the potential candidate for jumping to get new current-point. These aspects are explained in detail in form of the following cases:

Case 1: One Intersecting Major-axis and no end-points in Gaussian Distribution
Consider a case when there is only one major axis that is intersecting with the current-axis and there are no other major-axes whose end points lie in the Gaussian Distribution of the current-component. In this case the agent first inspects whether the flag of the intersecting axes is high or low. If it is high then the current-point is **one end-point of the stroke**.

If the flag is low then the agent calculates the distance between intersection point of both axes and the current-point and also the difference of their slopes. If both the entities lie below some threshold (1/4)*length of current-axis and 60degrees respectively, then one of the end points of intersecting axis is to be chosen as new current-point. The nearest end point to the current-point is considered to be the new current-point. If the distance is greater than the threshold but below (1/2)*length of current-axis and the slope difference is below the 30degree then the new current-point is the nearest end point of the intersecting axis to the current-point. If even this level of threshold is not met then the current-point is **one end-point of the stroke**.

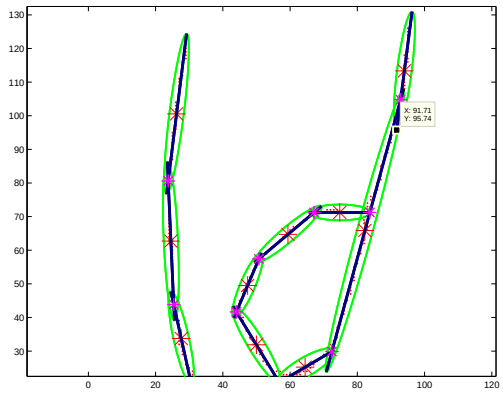


Figure 3.10.: Agent jumps to the other end point of same major-axis whose flag is not high. In this case the next current-point is one with values X = 91.71 and Y = 95.74

Case 2: More Intersecting Major-axes and no end-points in Gaussian Distribution
Consider a case where there are more than one intersecting major-axes and no end points in the Gaussian Distribution of current-component like shown in Figure 3.11

The agent scans the flags of all the major-axes intersecting with current-axis. If flags of all the intersecting axes are high, then the current-point is **one end-point of the stroke**.

If all the flags of intersecting-axes are high except one, it means that only one axis is of interest to be checked for new current-point, then solve it using case 1 as it now is similar to case 1.

If there are more than one axes whose flags are not high, and distance between their intersecting point and current-point and their slope difference with slope of current-axis are less than the thresholds mentioned in **Case 1** respectively, then the agent chooses the axes with least difference between its slope and slope of the current-axis. Then the end point of the intersecting axis which is nearest to the current-point is chosen as the new current-point. If the thresholds are not fulfilled, then the current-point is **one end-point of the stroke**.

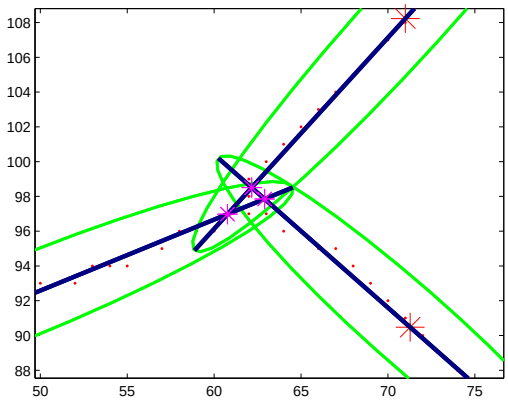


Figure 3.11.: A figure indicating more than one intersecting major-axes

Case 3: Intersecting Major-axes and end-points in Gaussian Distribution
Consider the case where there are major-axes that are intersecting with the current-axis and there are few major-axes which are not intersecting but have one of their end points in the Gaussian distribution of current-component.

In this case the agent scans the flags of all major-axes that intersect with current-point and all those whose end points are in Gaussian distribution current-component. If the flags are all high, then the current-point is **one end-point of the stroke**.

If all the flags are high except one, it means that only one axis is of interest to check for new current-point. In case that the axis of interest is the one whose one of the end points is in the Gaussian distribution of current-axis, the agent checks if its slope difference with the slope of current-axis is less than 60degrees and also checks if its end point lies in the (1/4)*length of current-axis from the current-point. If both the thresholds are satisfied, the new current-point is the end point that lies in the Gaussian distribution of current-component. If the above thresholds are not satisfied, both these parameters are checked with new threshold values as explained in **Case 1**. If even these threshold values are not fulfilled, then the current-point is **one end-point of the stroke**.

If there are more than one axes whose flags are not high, use the concept mentioned in **Case 2** to determine the new current-point or **one end-point of the stroke**. The agent repeats the above method based on the case encountered till one of the end points of the stroke are determined.

By now one of the end points of the stroke has been found by the agent. Agent uses this determined end point of the stroke as a starting point or starting current-point to find the other end point of the same stroke by employing the same method (the flags are reset) as mentioned above based on the case encountered by it during its search policy. It also stores the order in which the major-axes are visited to get the trajectory of the stroke.

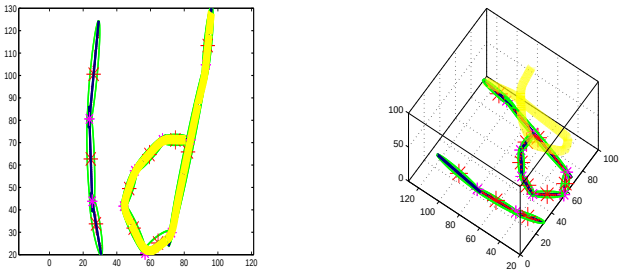
Once both end points of the same stroke are determined, these are stored and the flags of major-axes used in this process are permanently turned high and not used in finding the end points of other stroke. This process is repeated till end points of all the strokes are determined.

3.5. Stroke Retrieval

In this phase complete stroke is retrieved by getting its trajectory.

After getting the end points of the stroke, each of the Gaussian components involved in representing the stroke are linearly divided into samples and an increasing variable is attached with them having value in this case between 0 and 1. Then GMM is applied on these linearly divided samples along with their increasing variable. Once the GMM is learned, the increasing variable is used as an input to the Gaussian Mixture Regression(GMR) which in return generates the trajectory as output based on the linearly divided samples which is shown in the Figure 3.12

Above mentioned procedure for generating the complete trajectory is employed till all the strokes are full retrieved which is represented in Figure 3.13



(a) **(b)**

Figure 3.12.: (a) A fully extracted Stroke in two strokes Character , (b) Trajectory generated by GMR

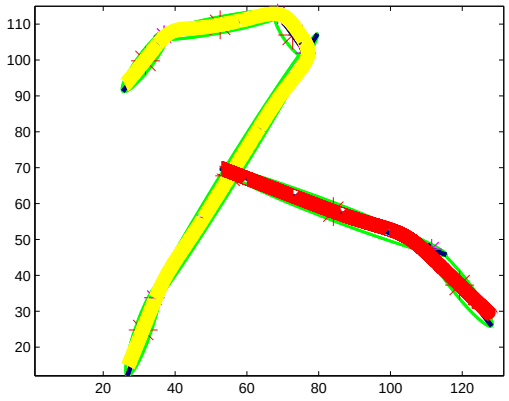


Figure 3.13.: Extracted Strokes of two stroke character

Once all the strokes are extracted, the character can be reproduced by the robot by drawing the extracted strokes in order. A stroke encoded using GMM has a high number of free parameters which become problematic when applying RL. Hence, to reduce the number of parameters, the trajectories extracted using GMR are re-encoded using DMPs before reproduction. The reproduction phase is discussed in following chapter.

CHAPTER

FOUR

CHARACTER REPRODUCTION

This chapter describes the reproduction phase of the character. Accurate reproduction of the character is possible only when its thickness information is known before hand.

4.1. Thickness of Character

In this step the thickness information of the character is restored using the original image. Once the stroke trajectory is retrieved, it is sub-sampled and circles [21] are drawn on the original image with center at each sampled point and the radius equal to the thickness of the original character corresponding to that point. Interpolation is done to get the thickness at points in between sub-sampled points. It is well represented in Figure 4.1

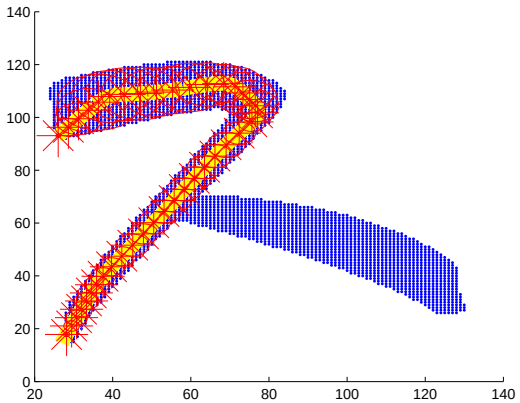


Figure 4.1.: Thickness of the stroke by drawing circles.

The thickness information is quite crucial in correct reproduction of the character.

The reproduction phase can further be partitioned into following parts namely:

1. Trajectory encoding by DMPs
2. Reproduction of Trajectory
3. Iterative Improvement of trajectory using Reinforcement Learning

Trajectory encoding by DMPs In order to improve the learned skill, the parameters of the extracted trajectory must be updated iteratively. A stroke encoded using GMM has high number of free parameters which become problematic when applying reinforcement learning because the number of free parameters to update increase tremendously which results in high computational cost and time to converge to better result. Hence, to reduce the number of parameters, the trajectories extracted using GMR are re-encoded using DMPs. Each stroke in the character has an individual DMP.

Reproduction of Trajectory The DMP encoded trajectory is then implemented on KUKA robot. As a proof of concept, the learned skill is verified in simulation as well. In simulation the paint brush is modeled as droplet. The drawn character using droplet model [dro] is shown in Figure 4.2.

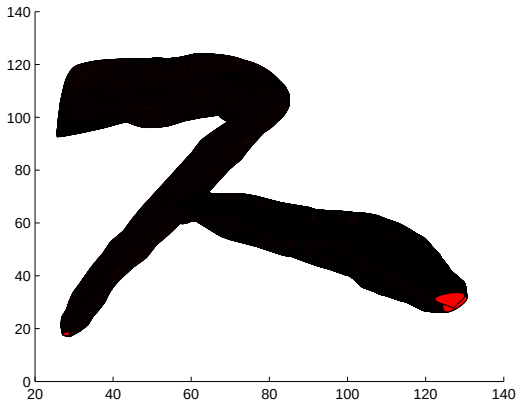


Figure 4.2.: Calligraphy letter drawn using drop model.

Iterative Improvement of trajectory using Reinforcement Learning After the character has been reproduced by the robot, its correlation with the original image is used as reward for RL and employed to update the parameters of DMP individually for each stroke. The update policy is shown by the Equation 4.1

$$\theta_{new} = \theta_{old} + \frac{\sum_{j=1}^{K} \gamma_j \epsilon_j}{\sum_{j=1}^{K} \gamma_j} \tag{4.1}$$

where:

θ_{new} = learned parameters of DMP
θ_{old} = old parameters of DMP
γ_j = Correlation between original image and image reproduced by robot
ϵ_j = exploration terms

The writing sequence of the strokes [18] is also very important in writing a character of Korean or Chinese language.

CHAPTER

FIVE

RESULTS

This chapter includes all the results of the research. The hardware used to acquire the results is KUKA Light weight robot which is operated using Robot Operating System. The software used in this research is MATLAB.

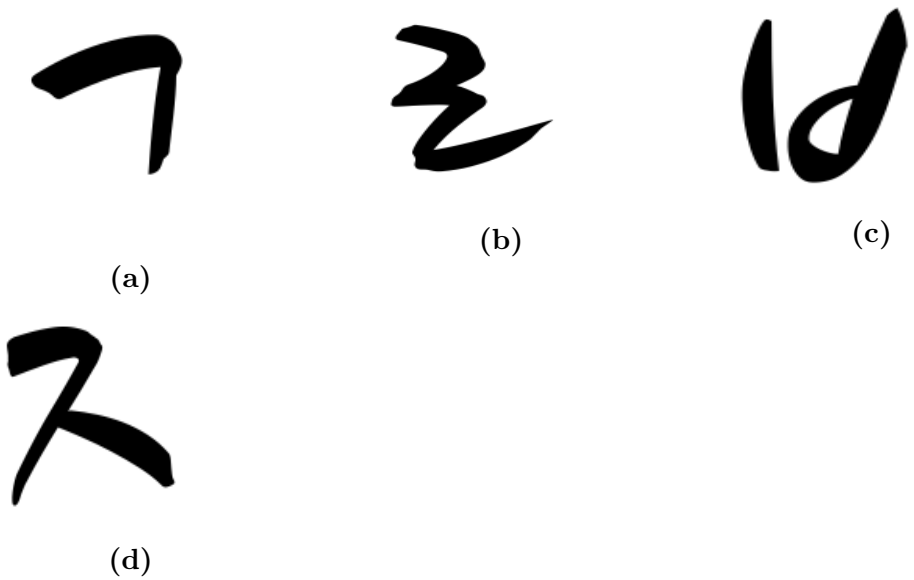
As a proof of concept the extracted strokes are also drawn in simulation. The paint brush is modeled as droplet [dro] which is given as:

$$x = thickness_i(1 - cos(2\pi t))sin(2\pi t)$$
$$y = thickness_i cos(2\pi t)$$

where:
$0 \leq t \leq 1$
and employed to draw the character. The original images of the characters and example of same characters drawn using drop model in simulation are shown in Figure 5.1 and 5.2

Later to verify the result, the learned skill is then implemented on KUKA robot. The result of KUKA robot drawing the characters are shown in Figures 5.3 and 5.7.

These drawn characters are then used to find the correlation with their respective original images and the calculated correlation is employed as a reward in Equation 4.1 to update the parameters of DMP which encodes the trajectory of these characters. The Figure 5.5 shows the graph of correlation between drawn character and its original image. With each increasing



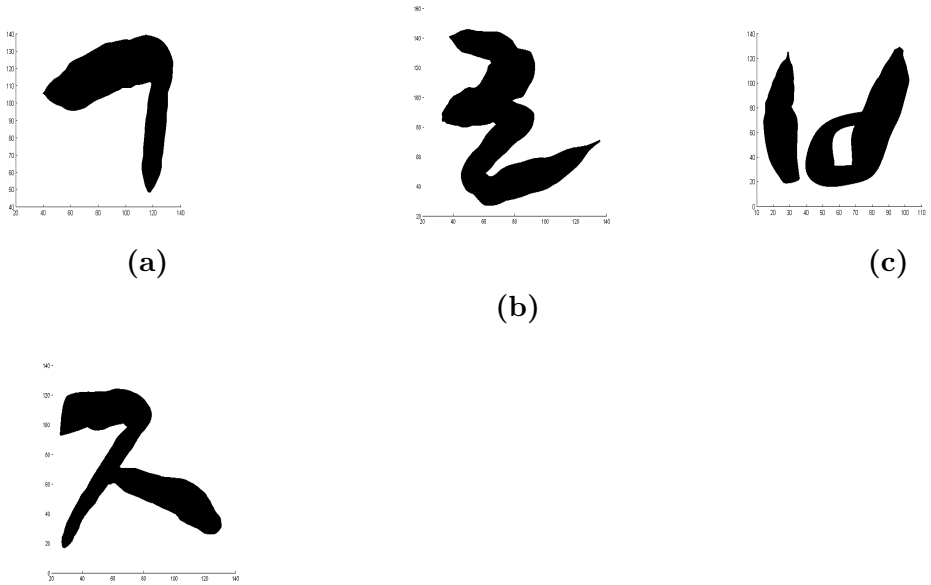
(a) **(b)** **(c)** **(d)**

Figure 5.1.: (a),(b),(c),(d) Show the original images of the calligraphic characters.

number of iteration of reinforcement learning, the correlation is increasing which shows the improvement in learned dexterity by the robot.

The Figure 5.6 gives the complete overview of the process.

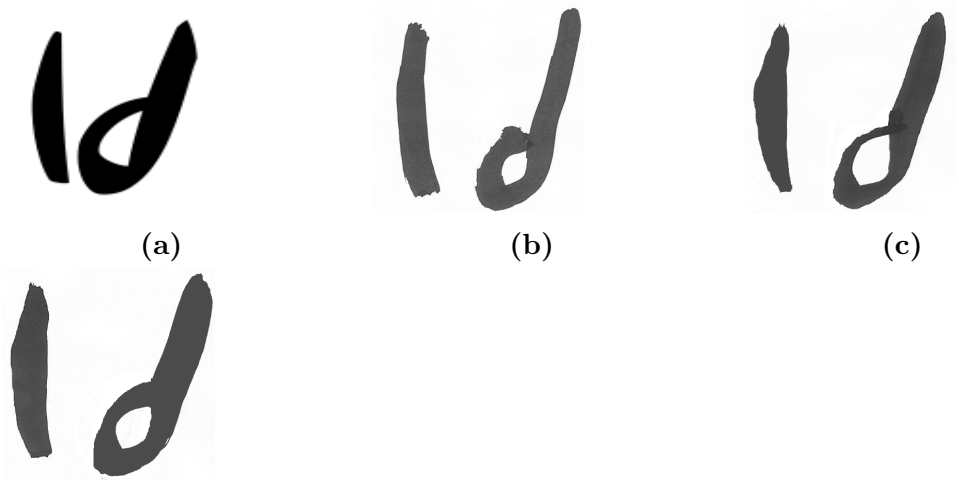
The whole procedure of extracting strokes and reproduction of the character is shown in Figure 5.7



(a) **(b)** **(c)**

(d)

Figure 5.2.: (a) Shows the character before reinforcement learning, (b) Shows the character after 3 iterations, (c) Shows the character after 7 iterations



(a) **(b)** **(c)**

(d)

Figure 5.3.: (a) Shows the original Character, (b) Shows the character before reinforcement learning, (c) Shows the character after 3 iterations, (d) Shows the character after 7 iterations

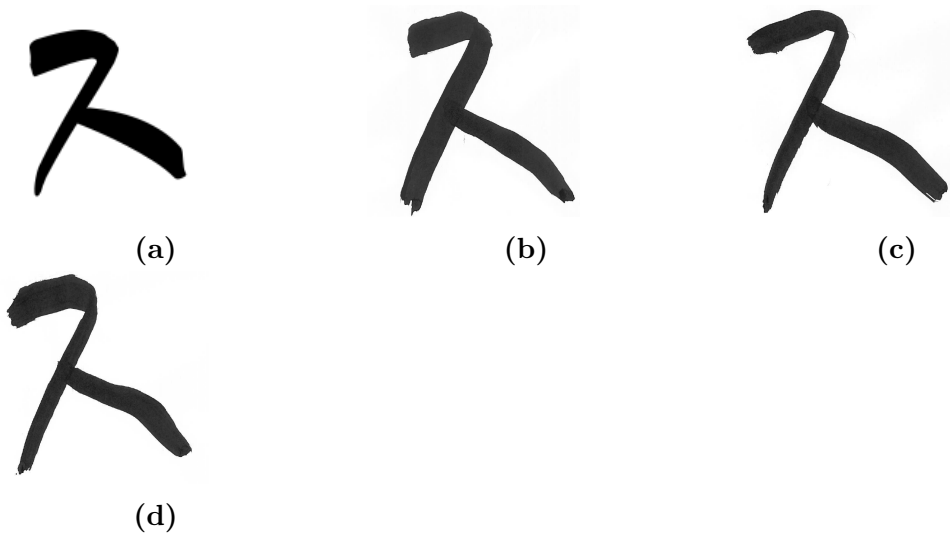


Figure 5.4.: (a) Shows the original Character, (b) Shows the character before reinforcement learning, (c) Shows the character after 3 iterations, (d) Shows the character after 7 iterations.

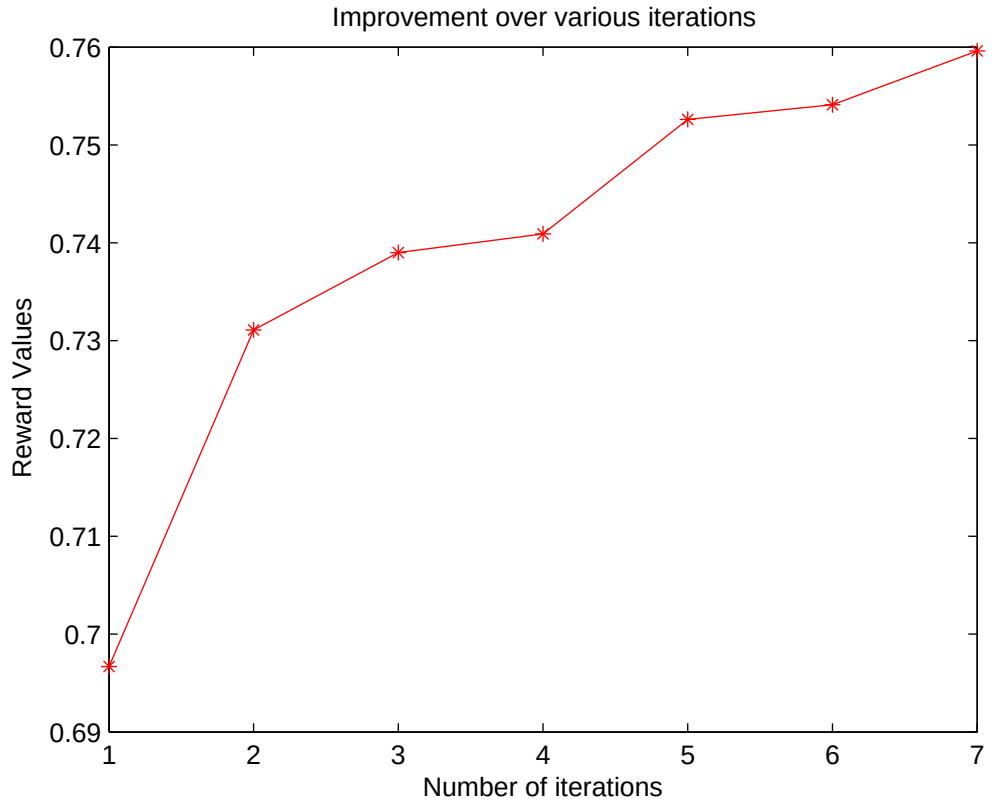


Figure 5.5.: Reward against number of iterations. It improves with each iteration of reinforcement learning.

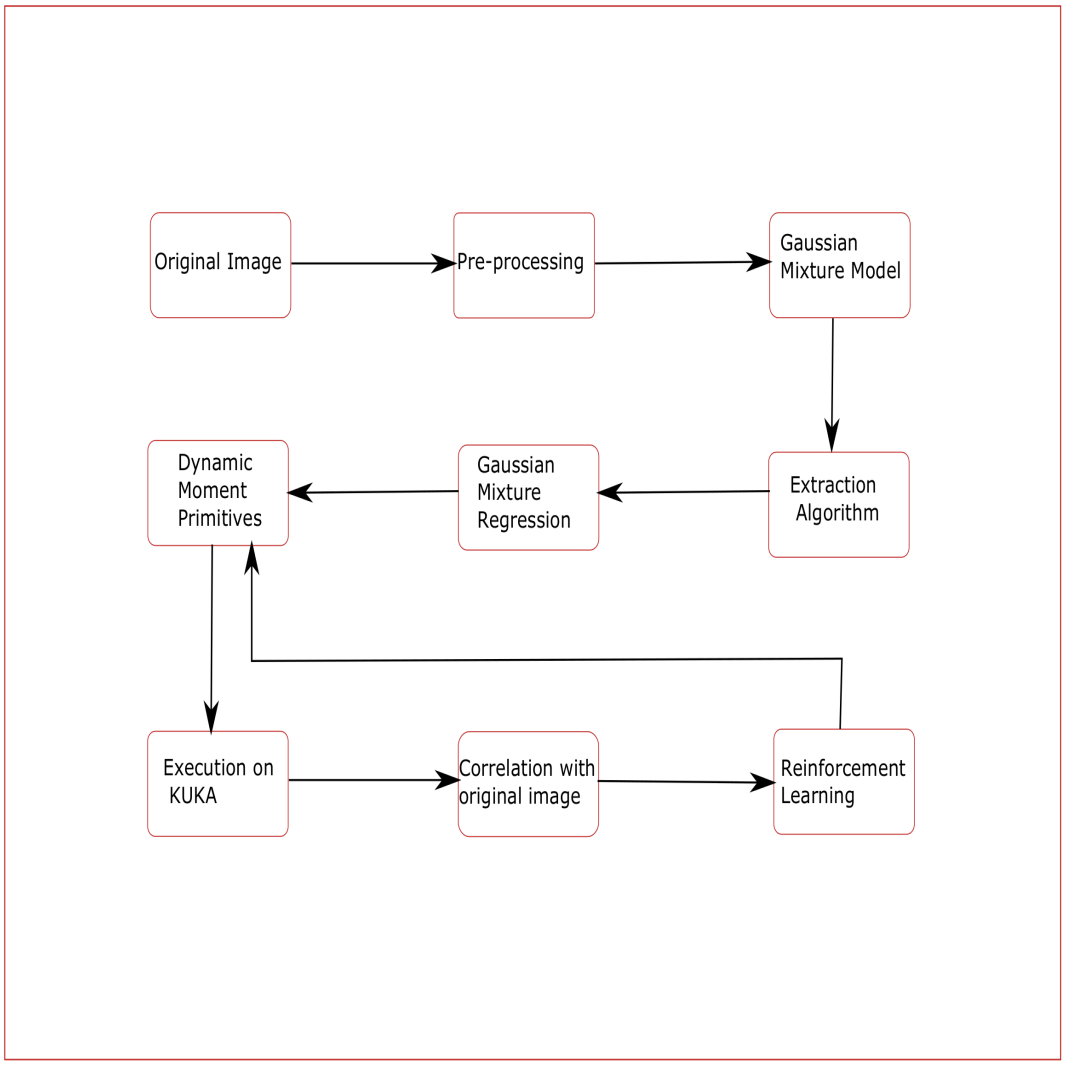


Figure 5.6.: Flow chart giving brief overview of the complete process.

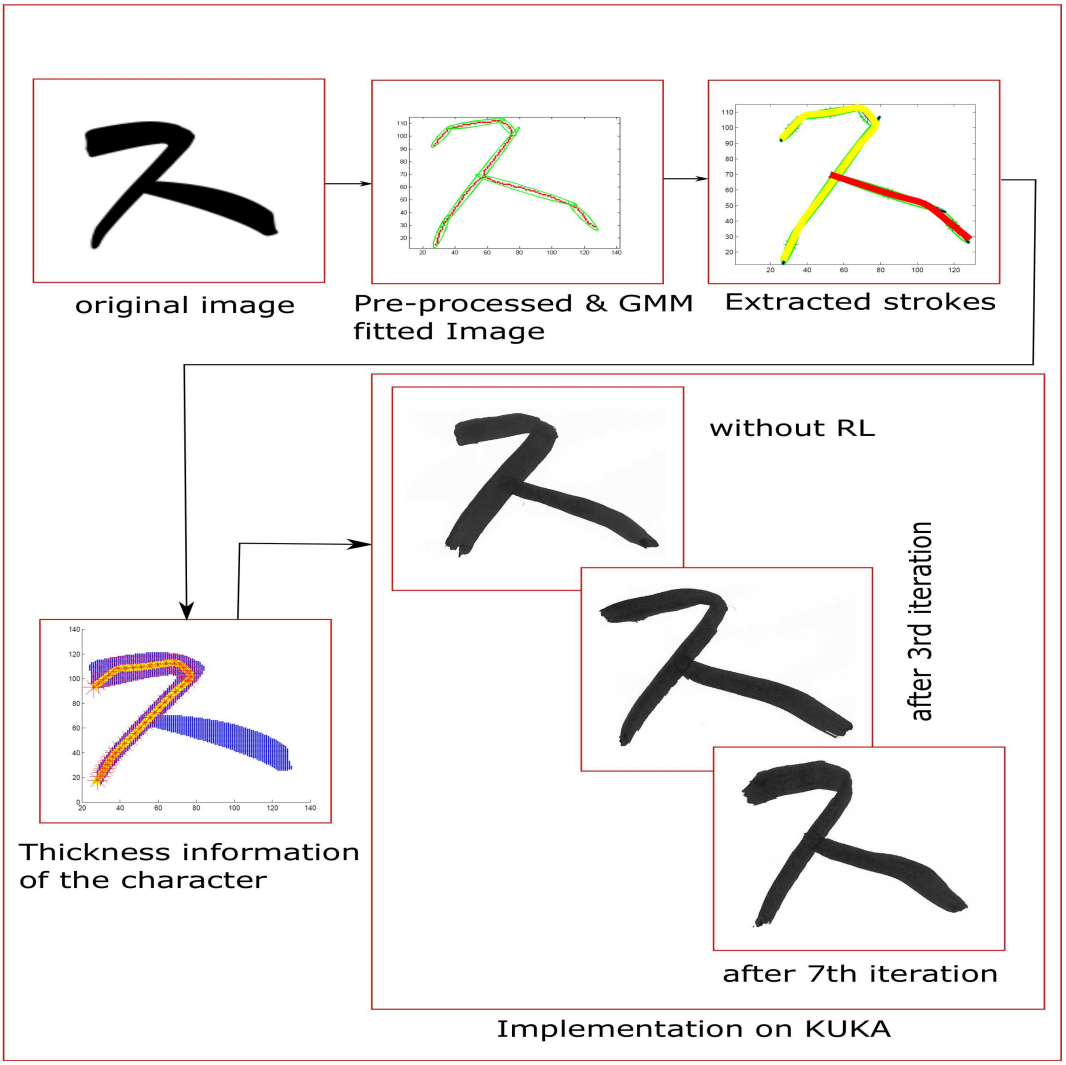


Figure 5.7.: Complete overview of the stroke extraction and reproduction of a character.

BIBLIOGRAPHY

[dro] A math function that draws water droplet shape? `http://math.stackexchange.com/questions/51539/a-math-function-that-draws-water-droplet-shapel`.

[onl] Understanding the bias-variance tradeoff. `http://scott.fortmann-roe.com/docs/BiasVariance.html`.

[3] Akaike, H. (1974). A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723.

[4] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[5] Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298.

[6] Calinon, S., Pervez, A., and Caldwell, D. G. (2012). Multi-optima exploration with adaptive gaussian mixture model. In *Proc. Intl Conf. on Development and Learning (ICDL-EpiRob)*, pages 1–6, San Diego, USA.

[7] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[8] Fitzgibbon, L. J., Allison, L., and Dowe, D. L. (2000). Minimum message length grouping of ordered data. In *Algorithmic Learning Theory*, pages 56–70. Springer.

[9] Fraley, C. and Raftery, A. E. (1998). How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588.

[10] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

[11] Kober, J. and Peters, J. (2010). Imitation and reinforcement learning. *Robotics & Automation Magazine, IEEE*, 17(2):55–62.

[12] Pervez, A. and Lee, D. (2015). A componentwise simulated annealing EM algorithm for mixtures. In *KI 2015: Advances in Artificial Intelligence*, pages 287–294. Springer.

[13] Pollard, K. S. and Van Der Laan, M. J. (2002). A method to identify significant clusters in gene expression data.

[14] Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

[15] Su, Y.-M. and Wang, J.-F. (2003). A novel stroke extraction method for chinese characters using gabor filters. *Pattern Recognition*, 36(3):635–647.

[16] Su, Y.-M. and Wang, J.-F. (2004). Decomposing chinese characters into stroke segments using sogd filters and orientation normalization. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 351–354. IEEE.

[17] Sun, Y., Qian, H., and Xu, Y. (2014). A geometric approach to stroke extraction for the chinese calligraphy robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3207–3212. IEEE.

[18] Tang, K.-t. and Leung, H. (2007). Reconstructing strokes and writing sequences from chinese character images. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 1, pages 160–165. IEEE.

[19] Wallace, C. S. and Dowe, D. L. (1999). Refinements of mdl and mml coding. *The Computer Journal*, 42(4):330–337.

[20] Wang, L. and Pavlidis, T. (1993). Direct gray-scale extraction of features for character recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(10):1053–1067.

[21] Wong, S. T., Leung, H., and Ip, H.-S. (2006). Fitting ellipses to a region with application in calligraphic stroke reconstruction. In *Image Processing, 2006 IEEE International Conference on*, pages 397–400. IEEE.

[22] YU, K., WU, J., and YUAN, Z. (2012). Stroke extraction for chinese calligraphy characters. *Journal of Computational Information Systems*.

APPENDIX

A

EIDESSTATTLICHE VERSICHERUNG

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem Titel

_____

_____

_____

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift