Low-Complexity Supervised Learning for Gesture and Shape Recognition

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Sait Celebi

in partial fulfillment for the
degree of Master of Science

in

Electronics and Computer Engineering

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electronics and Computer Engineering.

APPROVED BY:

Assist. Prof. Tarık Arıcı (Thesis Advisor)	[illegible]
Assist. Prof. Ahmet Bulut	[illegible]
Assist. Prof. Bugra Gedik	[illegible]

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 20 February 2014

SEAL/SIGNATURE: [illegible]

T.C. İSTANBUL ŞEHİR ÜNİVERSİTESİ BİLİMLERİ ENSTİTÜSÜ

Declaration of Authorship

I, Sait Celebi, declare that this thesis titled, 'Low-Complexity Supervised Learning for Gesture and Shape Recognition' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 20 February 2014

"Science is what we understand well enough to explain to a computer. Art is everything else we do."

Donald Knuth

Low-Complexity Supervised Learning for Gesture and Shape Recognition

Sait Celebi

Abstract

Classification is a machine learning task in which the objective is to categorize given samples according to their attributes. Gesture Recognition (GR) and Shape Recognition (SR) are two classification examples. Some daily-life applications of these include Hand Gesture Recognition (HGR) and Optical Character Recognition (OCR).

GR is a challenging classification problem often used in human-computer interaction applications to provide a natural interface between user and computer. Since the same gesture might be performed with different speeds, Dynamic Time Warping (DTW) is needed to find the optimal alignment between two time sequences. Oftentimes a pre-processing of sequences is required to remove variations between the reference gestures and the test gestures. We discuss a set of pre-processing methods to make the gesture recognition mechanism robust to these variations. DTW computes a dissimilarity measure by time-warping the sequences on a per sample basis by using the distance between the current reference and test sequences. However, all body joints involved in a gesture are not equally important in computing the distance between two sequence samples. We propose a weighted DTW method that weights joints by optimizing a discriminant ratio.

SR is another classification problem with increasing number of applications from OCR to pedestrian detection. Decision tree is a good choice of classifier for shape recognition because it is easy to implement and visualize and has lower computational complexity. Bagging randomized decision trees as random forests increases the accuracy rates if the trees are weakly correlated. We propose using random rectangles in combination with random forests and test our method on OCR and GR datasets. We show that the accuracy of our method is similar to the OCR state-of-the-art and better than the GR state-of-the-art, while executing significantly faster, which makes our proposed method a good fit for real-time object/shape recognition. Then discuss how a simple feature such as a random rectangle can perform similar to the complex statistical and structural features designed for shape recognition. Finally we analyze the effect of our parameters.

Keywords: Gesture Recognition, Dynamic Time Warping, Kinect, Shape Recognition, Random Forests, Decisin Trees

Hareket ve Şekil Tanıma için Az Karmaşıklıklı Gözetimli Öğrenme

Sait CELEBI

Öz

Sınıflandırma, verilen örnekleri özelliklerini kullanarak kategorize etme işini yapan makine öğrenmesi görevidir. Hareket Algılama (HA) ve Şekil Algılama (ŞA) iki adet sınıflandırma örneğidir. El Hareketlerini Algılama (EHA) ve Optik Karakter Tanıma (OKT) bu alanlardaki günlük hayatta karşılaşılan bazı uygulamalardır.

EHA, genellikle insan-bilgisayar etkileşimi uygulamalarında kullanılan, insan ve bilgisayar arasında doğal bir arayüz sunan zor bir sınıflandırma problemidir. Aynı el hareketi farklı hızlarda uygulanabileceği için, Dinamik Zaman Bükmesi (DZB) iki tane zaman dizisi arasındaki en iyi uyuşmayı bulmak için kullanılır. Çoğu zaman referans ve test örneklerindeki farklılıklardan dolayı bir ön-işleme mekanizması gereklidir. Hareket tanımanın bu tip farklılıklardan bağımsız olarak iyi çalışabilmesi için birkaç ön-işleme metodu gereklidir. DZB, hali hazırda bulunan test örneğiyle tüm referans örneklerini tek tek tüm parçalarını uyuşturmaya çalışarak bir farklılık ölçütü hesaplar. Fakat bir el hareketini algılarken vücudun tüm parçalarının ağırlığı eşit değildir. Bu çalışmada vücut parçalarını bir farklılık oranını optimize ederek ağırlaklandırmayı öneriyoruz. Son olarak, ön-işleme ve ağırlıklandırma yöntemlerimizi klasik DZB ve tekniğin bilinen en iyi durumu ile kıyaslıyoruz.

ŞA, OKT'den yaya algılamaya kadar uzanan artan sayıda uygulamalara sahip diğer bir sınıflandırma problemidir. Karar ağaçları uygulaması kolay olduğu için, görselleştirilebilmesi mümkün olduğu için ve hesaplama karışıklığı az olduğu ŞA için uygun bir sınıflandırıcı seçimidir. Eğer sınıflandırma için birden fazla birbiriyle az ilişkili karar ağacı beraber kullanılıyorsa (rastgele orman) sınıflandırma kalitesi artar. Bu çalışmada rastgele orman sınıflandırıcılarını resimlerden rastgele seçtiğimiz dikdörtgen özellikleriyle kullanıyoruz. Metodumuzu karakter tanıma ve hareket tanıma datasetleriyle test ediyoruz. Görülüyor ki bu yöntem şuana kadar bilinen en iyi yöntemlerle yaklaşık doğrulukta çalışmaktadır. Bunun yanında bunlara kıyasla çok daha hızlı çalışmaktadır ki bu özelliği bu yöntemi gerçek zamanlı nesne ve şekil tanıma uygulamalarına uygun kılmaktadır. Rastgele dikdörtgenler gibi basit tanımlayıcıların karışık istatistiksel ve yapısal tanımlayıcılara göre ne kadar da şaşırtıcı şekilde iyi çalıştığı üzerine tartışıyoruz. Son olarak da sistemde kullandığımız parametreleri analiz ediyoruz.

Anahtar Sözcükler: Hareket Tanıma, Dinamik Zaman Bükmesi, Kinect, Şekil Tanıma, Rastgele Orman, Karar Ağaçları

This thesis is lovingly dedicated to my mother for her constant love to me throughout my life.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Tarik Arici for his continuous support, motivation, enthusiasm, and time. His guidance helped me throughout the duration of the research conducted and the writing of this thesis.

Besides my advisor, I would like to thank to the rest of my thesis committee: Prof. Ahmet Bulut and Prof. Bugra Gedik for their helpful and constructive comments.

I thank my fellow lab-mates in Istanbul Sehir University, Data Science Group: Ali Selman Aydin, Erkan Bilmez and Talha Tarik Temiz for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last two years.

Also, I would like to thank all Sehir University members who participated in our gesture database recordings and patiently performed all the gestures that helped in our experiments.

Most importantly, I would like to thank my family, for their constant support without knowing a bit what I was doing on my thesis at the time.

Contents

Declaration of Authorship **ii**

Abstract **iv**

Öz **v**

Acknowledgments **vii**

List of Figures **ix**

List of Tables **xi**

1 Robust Gesture Recognition Using Feature Pre-Processing and Weighted Dynamic Time Warping **1**

1.1 Introduction 1
1.2 Related Work 3
1.3 Data Acquisition and Feature Pre-processing 3
1.4 Dynamic Time Warping for Gesture Recognition 5
1.4.1 Boosting The Reliability of DTW 8
1.4.2 Weighted DTW 9
1.5 Results 11
1.6 Conclusion 14

2 Low-Complexity Shape Recognition Using Random Forest Classifiers with Random Rectangle Features **16**

2.1 Introduction 16
2.2 Shape Features for Recognition 17
2.3 Decision Tree Based Classification 19
2.3.1 Decision Trees and Random Forests 19
2.3.2 Random Forest Classifiers For Recognition 22
2.3.3 Viola-Jones Object Detection Framework 24
2.4 Random Forest Classifiers with Random Rectangle Features 25
2.4.1 Rectangle Feature As A Linear Combination 27
2.5 Experiments 28
2.5.1 Optical Character Recognition Results 29
2.5.2 Gesture Recognition 34
2.6 Conclusion 35

Bibliography **36**

List of Figures

1.1 Kinect joints. 4

1.2 Camera A is used to record the ground-truth reference gestures with perpendicular angles, Camera B is used to record a rotationally distorted test sequence. β is the desired angle to rotate the skeleton in Y axis. After this rotation, the skeleton will be rotated in other axes if needed until it will be perpendicular to all axes. 4

1.3 Two skeletons with different orientations (Left: Ground-truth reference frame, Right: Rotationally distorted test frame due to improper body orientation) 5

1.4 DTW used to match two sequences, reference sequence and test sequence. 5

1.5 Accumulated cost matrix of two sequences $\mathbf{R}$ and $\mathbf{T}$ with sizes N and M, respectively. Global constraint region, R, *Sakoe-Chiba band* [49], is shown with gray color. 7

1.6 Predecessor nodes used in Bellman's principle where $n_l \in [1 : N]$, $m_l \in [1 : M]$ and $l \in [2 : L]$. Note that $(n_{l-1}, m_{l-1}) \in \{(n_l - 1, m_l), (n_l, m_l - 1), (n_l - 1, m_l - 1)\}$. 8

1.7 Two sample reference gestures in the gesture database: Right Hand Push Up and Left Hand Wave. 10

1.8 Discriminant ratios for with and without pre-processed gesture samples using the rotationally distorted gesture database. Note that the discrimant ratios are increased, on average, 42% with the proposed pre-processing method. There are 21 gesture samples in each gesture class. The gesture classes are, namely, Both Hands Pull Down, Both Hands Push Up, Left Hand Pull Down, Left Hand Push Up, Left Hand Swipe Left, Left Hand Swipe Right, Left Hand Wave, Right Hand Pull Down, Right Hand Push Up, Right Hand Swipe Left, Right Hand Swipe Right, Right Hand Wave, respectively. 13

2.1 An example data instance 19

2.2 Univariate decision tree 20

2.3 Multivariate decision tree 20

2.4 Example Viola-Jones features relative to the detection window. The sum of the pixel intensities in the grey rectangular regions are subtracted from the white rectangular regions. 24

2.5 Cascaded application of classifiers enable focusing on object-like regions. . 24

2.6 Visualization of a decision tree up to the fourth depth level trained on OCR data. The thickness of the edges connecting nodes is proportional to the number of images associated with the receiving node. The average of all training images at a node is displayed. The brightness of the circle below the average image is inversely proportional to the entropy at that node: the higher the entropy the darker the color of the circle. 26
2.7 Sample digits from the MNIST database 29
2.8 Accuracy as the number of trees is increased 31
2.9 The effect of the number of candidate rectangles per each split node . . . 31
2.10 Accuracy versus maximum tree depth using a random forest of five randomized trees. 32
2.11 Average per-tree accuracy versus maximum tree depth 33
2.12 Accuracy versus number of trees using different features 33
2.13 The characters in order b,e,t,y 35

List of Tables

1.1 Confusion matrix for the conventional DTW. 13
1.2 Confusion matrix for the weighted DTW in [47]. 14
1.3 Confusion matrix for our proposed weighted DTW. 14
1.4 Accuracies of the three methods. Note that not only six gesture classes given in Table 1.1, 1.2, and 1.3 are used, but all eight gesture classes are taken into consideration. 14
1.5 Overall performance comparison using the rotationally distorted and relaxed gesture database. 15

2.1 Classification times (milliseconds) on an example image of size 28×28. . . 30
2.2 Classification accuracies on the MNIST test set. 31
2.3 Gesture recognition accuracies 35

Chapter 1

Robust Gesture Recognition Using Feature Pre-Processing and Weighted Dynamic Time Warping

1.1 Introduction

Interacting with computers using human motion is commonly employed in human-computer interaction (HCI) applications. One way to incorporate human motion into HCI applications is to use a predefined set of human joint motions *i.e.*, gestures. Gesture recognition has been an active research area [20, 32, 47, 66], and involves state-of-the-art machine learning techniques in order to work reliably in different environments. A variety of methods have been proposed for gesture recognition including Dynamic Time Warping [47], Hidden Markov Models [20], Finite State Machines [23], hidden Conditional Random Fields (CRFs) [61] and orientation histograms [19]. In addition to these, there are methods employed in gesture recognition that are not view-based. Examples of these are the use of Wii controller (Wiimote) [50] and DataGlove [44].

DTW measures similarity between two time sequences which might be obtained by sampling a source with varying sampling rates or by recording the same phenomenon occurring with varying speeds [64]. After DTW was introduced in 1960s [10], it has been used in solving different problems such as speech recognition to warp speech in time to be able to cope with different speaking speeds [2, 41, 49], data mining and information retrieval to deal with time-dependent data [1, 45], curve matching [18], online handwriting recognition [59], hand shape classification [28]. In gesture recognition, DTW time-warps an observed motion sequence of body joints to pre-stored gesture sequences [16, 28, 46, 62]. Although we present the theory of the general DTW and its implementation issues, in this paper we focus more on its application to gesture recognition. Comprehensive surveys about the general DTW algorithm can be found in [40, 52].

The conventional DTW algorithm is basically a dynamic programming algorithm, which uses an iterative update of DTW cost by adding the distance between mapped elements of the two sequences at each iteration step. The distance between two elements is oftentimes the Euclidean distance, which gives equal weights to all dimensions of a sequence sample. However, depending on the problem a weighted distance might perform better in assessing the similarity between a test sequence and a reference sequence. For example in a typical gesture recognition problem, body joints used in a gesture can vary from gesture class to gesture class. Hence, not all joints are equally important in recognizing a gesture.

With Microsoft's launch of Kinect in 2010, and release of Kinect SDK in 2011, numerous applications and research projects exploring new ways in human-computer interaction have been enabled. Some examples are gesture recognition [47], touch detection using depth data [65], human pose estimation [25], implementation of real-time virtual fixtures [48], real-time robotics control applications [58] and the physical rehabilitation of young adults with motor disabilities [15].

We propose a weighted DTW algorithm that uses a weighted distance in the cost computation. The weights are chosen so as to maximize a discriminant ratio based on DTW costs. The weights are obtained from a parametric model which depends on how active a joint is in a gesture class. The model parameter is optimized by maximizing the discriminant ratio. By doing so, some joints will be weighted up and some joints will be weighted down to maximize between-class variance and minimize within-class variance. As a result, irrelevant joints of a gesture class (*i.e.*, parts that are not involved in a gesture class) will contribute to the DTW cost to a lesser extent, while keeping the between-class variances large.

Our system first extracts body-joint features from a set of skeleton data that consists of six joint positions, which are left and right hands, wrists and elbows. We have observed that the gestures in our training set, which have quite different motion patterns, require the use of all or a subset of these six joints only. These obtained skeleton features are used to recognize gestures by matching them with pre-stored reference sequences. Pre-processing is needed to suppress the noise due to different body and camera orientations, and different body sizes. After pre-processing is done, the matching is performed by assigning a test sequence to a reference sequence with the minimum DTW cost. By removing the variations in the data, the DTW cost becomes more reliable in classification as demonstrated by the increase in the discriminant ratio values.

1.2 Related Work

One commonly used technique for gesture recognition is using HMMs for modeling gesture sequences. HMMs are especially known for their application to speech recognition, gesture recognition, bioinformatics, etc. HMMs are statistical models for sequential data [7, 8], and therefore can be used in gesture recognition [20, 30, 57]. The states of an HMM are hidden and state transition probabilities are to be learned from the training data. However, defining states for gestures is not an easy task since gestures can be formed by a complex interaction of different joints. Also, learning the model parameters *i.e.*, transition probabilities, requires large training sets, which may not always be available. On the other hand, DTW does not require training but needs good reference sequences to align with.

Using a weighting scheme in DTW cost computation has been proposed for gesture recognition [47]. The method proposed in [47] uses DTW costs to compute between and within class variations to find a weight for each body joints. These weights are global weights in the sense that there is only one weight computed for a body joint. However, our proposed method computes a weight for each body joint and for each gesture class. This boosts the discriminative power of DTW costs since a joint that is active in one gesture class may not be active in another gesture class. Hence weights has to be adjusted accordingly. This helps especially dealing with within-class variation. To avoid reducing the between-class variance, we compute weights by optimizing a discriminant ratio using a parametric model that depends on body joint activity. In the next section we discuss data acquisition and feature pre-processing.

1.3 Data Acquisition and Feature Pre-processing

We use Microsoft Kinect sensor [53] to obtain joint positions. Kinect SDK tracks 3D coordinates of 20 body joints given in Figure 1.1 in real time (30 frames per second). Since the machine learning algorithm uses depth images to predict joint positions, the skeleton model is quite robust to color, texture, and background.

We have observed that only six out of the 20 joints contribute in identifying a hand gesture: left hand, right hand, left wrist, right wrist, left elbow and right elbow. A feature vector consists of 3D coordinates of these six joints and is of dimension of 18 as given below

$$\mathbf{f}_n = [X_1, Y_1, Z_1, X_2, Y_2, Z_2, \ldots X_6, Y_6, Z_6], \tag{1.1}$$

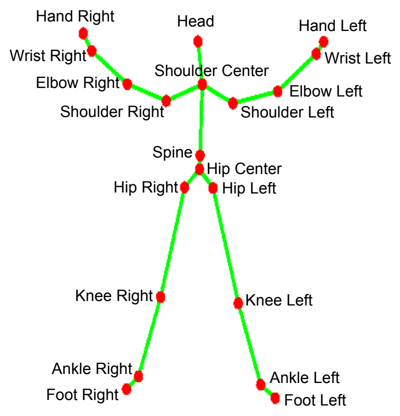

FIGURE 1.1: Kinect joints.

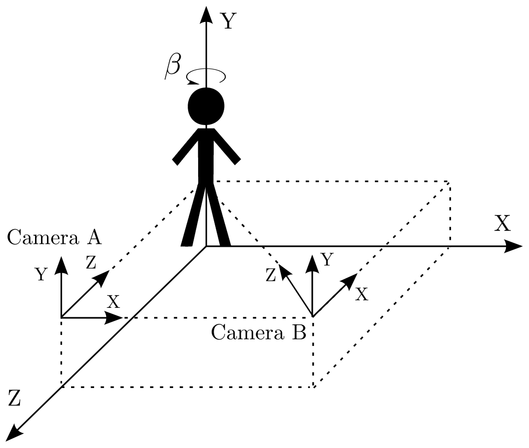

FIGURE 1.2: Camera A is used to record the ground-truth reference gestures with perpendicular angles, Camera B is used to record a rotationally distorted test sequence. β is the desired angle to rotate the skeleton in Y axis. After this rotation, the skeleton will be rotated in other axes if needed until it will be perpendicular to all axes.

where n is the index of the skeleton frame at time t_n. A gesture sequence is the concatenation of N such feature vectors.

After N feature vectors are concatenated to create the gesture sequence, they are pre-processed before the DTW cost computation. The pre-processing consists of three stages. First stage is the normalization stage which translates all skeletons to the center of the field of view. This could be done by subtracting the hip center joint position from the other joint positions. Note that the reference frames are already recorded at the center of the field of view. The second pre-processing stage removes the rotational distortion caused by different orientations of human bodies. Contrary to the reference gestures, where trained performers are used, it is highly possible to have different orientations or positionings of users with respect to camera in real-life cases. Such occasions are problematic for gesture recognition since they will result in rotationally distorted skeleton frames. To cope with these occasions, our pre-processing system rotates the skeleton frames if necessary, such that the skeleton frames will be orthogonal to the principal axis

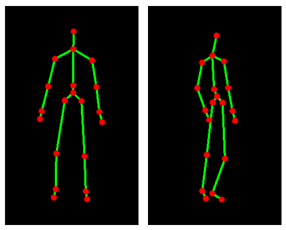

Figure 1.3: Two skeletons with different orientations (Left: Ground-truth reference frame, Right: Rotationally distorted test frame due to improper body orientation)

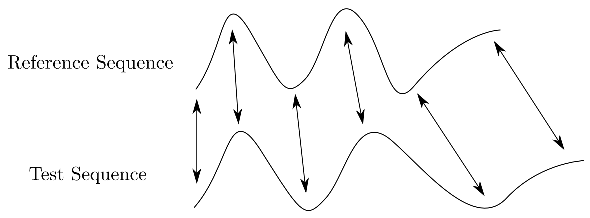

Figure 1.4: DTW used to match two sequences, reference sequence and test sequence.

of the camera. To this end, we define two vectors by using spatial coordinates of the right shoulder, left shoulder and hip center which are obtained from Kinect sensor. Using these two vectors, we calculate the three angles, α, β, θ, of the skeleton with respect to the camera's coordinate system, and compute the rotation matrices $\mathbf{R}_{\mathbf{x}}^{\alpha}, \mathbf{R}_{\mathbf{y}}^{\beta}, \mathbf{R}_{\mathbf{z}}^{\theta}$, respectively. The rotation is then applied using these angles with the appropriate order. See an example rotation in Y axis with $\mathbf{R}_{\mathbf{y}}^{\beta}$ in Figure 1.2. The third and the last pre-processing stage is the elimination of variations in the feature vectors due to different skeleton ratios (broad-shouldered, narrow-shouldered). All feature vectors are normalized with the distance between the left and the right shoulders to account for the variations due to a person's *size*. Note that the reference sequences are recorded with people who has average skeleton ratios. Next, we present a more detailed discussion on DTW.

1.4 Dynamic Time Warping for Gesture Recognition

DTW is a template matching algorithm to find the best match for a test pattern out of the reference patterns, where the patterns are represented as a time sequence of features. In Figure 1.4 we show an example matching of two sequences.

Let $\mathbf{R} = \{r_1, r_2, \ldots, r_N\}, N \in \mathbb{N}$ and
$\mathbf{T} = \{t_1, t_2, \ldots, t_M\}, M \in \mathbb{N}$ be reference and test sequences (sequence of set of joint positions in our case), respectively. The objective is to align the two sequences in time via a nonlinear mapping (*i.e.*, warping or alignment). Such a warping path can be illustrated as an ordered set of points as given below

$$p = (p_1, p_2, \ldots, p_L), \; p_l = (n_l, m_l),$$

where $p_l = (n_l, m_l)$, denotes mapping of r_{n_l} to t_{m_l}. $p_l \in [1:N] \times [1:M]$ for $l \in [1:L]$, where L is the number of mappings. The total cost D of a warping path p between **R** and **T** with respect to a *distance* function $d(r_i, t_j)$, $i \in [1:N]$ and $j \in [1:M]$, is defined as the sum of all distances between the mapped sequence elements

$$\mathrm{D_p} = \sum_{l=1}^{L} d(r_{n_l}, t_{m_l}), \tag{1.2}$$

where D_p is the total cost of the path p and $d(r_i, t_j)$ measures the distance between elements r_i and t_j. For gesture recognition, distance can be chosen as the distance between the corresponding joint positions (3D points) of the reference gesture, **R**, and the test gesture **T**.

A mapping can also be viewed as a path on a two-dimensional (2D) grid, also known as the cost matrix, which is of size $N \times M$ (see Figure 1.5), where grid node (r_i, t_j) denotes the distance between r_i and t_j. The node (r_1, t_1) which starts the alignment by matching the first sequence elements is conventionally placed on the left-bottom corner of the grid. Each path p on the 2D grid (*i.e.*, the cost matrix) is associated with a total cost D given in Eq. (1.2). Note that among all possible paths, we are mostly interested in the path which makes the total accumulated cost minimum while satisfying the desired constraints. Hence, optimal path denoted by p^* is the path with the minimum total cost. The DTW distance between two sequences is defined by the distance associated with a total cost D given in Eq. (1.2) using the optimal path, *i.e.*:

$$\mathrm{DTW}(\mathbf{R}, \mathbf{T}) = \mathrm{D_{p^*}}(\mathbf{R}, \mathbf{T}). \tag{1.3}$$

The optimal path specifies the optimal alignment between two sequences and is computed by finding the path that minimizes the total cost. One way to find the minimum cost path is to test every possible path on the 2D grid from the left-bottom corner to the right-top corner. However, this has exponential complexity. Dynamic programming reduces the complexity by taking advantage of Bellman's principle [9]. Bellman's optimality principle states that the optimal path from the starting grid node (r_1, t_1) to the ending node (r_N, t_M) through an intermediate point (r_n, t_m) can be expressed as the concatenation of the optimal path from (r_1, t_1) to (r_n, t_m), and the optimal path from (r_n, t_m) to (r_N, j_M). This implies that if we are given the optimal path from (r_1, t_1) to (r_n, t_m), we only need to search for the optimal path from (r_n, t_m) to (r_N, t_M) rather than searching for paths from (r_1, t_1) to (r_N, t_M). We will use Bellman's principle in the total cost computation with dynamic programming.

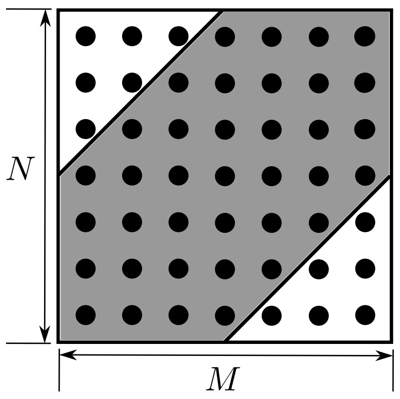

FIGURE 1.5: Accumulated cost matrix of two sequences **R** and **T** with sizes N and M, respectively. Global constraint region, R, *Sakoe-Chiba band* [49], is shown with gray color.

Some well-known restrictions on the warping path have been proposed to eliminate unrealistic correspondences between the sequences [40, 49]. The most fundamental constraints which are applied in various topics as well as gesture recognition, are the following:

(i) Boundary conditions: $p_1 = (1,1), p_L = (N,M)$.
(ii) Step size condition: $p_{l+1} - p_l \in \{(0,1),(1,0),(1,1)\}$ for $l \in [1 : L-1]$.

The boundary conditions require the whole reference sequence to be mapped to the whole test sequence, and can be modified if this is not strictly desired. The step size condition requires that only one element of both sequences can be skipped at each cost computation step of Bellman's principle. Hence, optimal path can progress from a restricted set of predecessor nodes as shown in Figure 1.6. Since all the elements are ordered in time, the set of predecessor nodes are to the left and bottom of a current node.

First, let's define $C(n_l, m_l)$ as below

$$\mathrm{C(n_l, m_l)} = \mathrm{DTW}(\mathbf{R}(1:n_l), \mathbf{T}(1:m_l)). \tag{1.4}$$

Note that $\mathrm{C(N,M)}$ is equal to $\mathrm{DTW}(\mathbf{R},\mathbf{T})$. Let's further assume that the total costs of the optimal paths to three predecessor nodes denoted by $(n_l - 1, m_l)$, $(n_l, m_l - 1)$, and $(n_l - 1, m_l - 1)$ have been computed. Since the $(l-1)$th position of the path (*i.e.*, (n_{l-1}, m_{l-1})) is restricted to be one of these three nodes on the 2D grid, Bellman's principle leads to

$$\begin{aligned}\mathrm{C(n_l, m_l)} = \min\{&\mathrm{C(n_l, m_l - 1)},\\ &\mathrm{C(n_l - 1, m_l)},\\ &\mathrm{C(n_l - 1, m_l - 1)}\} + d(r_{n_l}, t_{m_l}).\end{aligned} \tag{1.5}$$

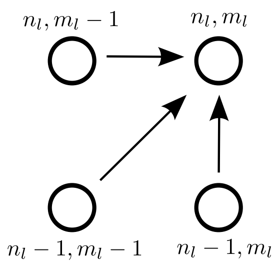

FIGURE 1.6: Predecessor nodes used in Bellman's principle where $n_l \in [1 : N]$, $m_l \in [1 : M]$ and $l \in [2 : L]$. Note that $(n_{l-1}, m_{l-1}) \in \{(n_l - 1, m_l), (n_l, m_l - 1), (n_l - 1, m_l - 1)\}$.

Finally, the minimum cost path aligning two sequences has cost $\text{DTW}(\mathbf{R}, \mathbf{T})$, and the test sequence is matched to the reference sequence that has the minimum cost among all reference sequences.

Although Eq. (1.5) outputs the minimum cost between two sequences, it does not output the optimal path. To find the optimal path, which can be used to map test sequence elements to reference sequence elements, one needs to backtrack the optimal path starting with the final node. Note that if the boundary condition is satisfied, *i.e.*, the whole test sequence is mapped to the whole reference sequence, than $(n_L, m_L) = (N, M)$ and $(n_1, m_1) = (1, 1)$.

1.4.1 Boosting The Reliability of DTW

Global constraints define a set of nodes on the 2D grid to be searched for finding the optimal path. Imposing global constraints not only reduces the DTW computational complexity, but also increases the reliability of DTW's dissimilarity measure by omitting unrealistic paths. We used a well-known global constraint region, *Sakoe-Chiba band* [49] given in Figure 1.5. The Sakoe-Chiba band effectively limits the warping amount, *i.e.*, slowing down or speeding up of a sequence in time. For example a gesture can be performed with different speeds in time depending on the performer but it is logical to expect that there is a limit to how slow or how fast a gesture is performed.

Another problem that degrades DTW's reliability in gesture recognition is due to unknown beginning and ending times of gesture samples. A gesture in a test sequence can often begin later or end sooner than the gesture in the reference sequence stored for that gesture class. Boundary conditions assume that all gestures start at the beginning of the sequence and finish at the ending of the sequence. Hence, imposing boundary conditions in such cases decreases the reliability of DTW costs. To boost the reliability, we relaxed the boundary conditions by changing the total cost given in Eq. (1.2) as below

$$\mathrm{D_p} = \sum_{l=1}^{L} \alpha_l d(r_{n_l}, t_{m_l}), \tag{1.6}$$

where α_l is a weight that is equal to 1 everywhere except the regions close to the starting node (*i.e.*, left-bottom node denoted by (r_1, t_1)) and the ending node (*i.e.*, right-top node denoted by (r_N, t_M)). To infer the proximity of the current node to starting and ending nodes the length of the path, $||p_l|| = \sqrt{n_l^2 + m_l^2}$, is utilized. The distance terms coming from the beginning and ending of the sequence is weighted down by computing α_l from the below formula

$$\alpha_l = \begin{cases} \frac{||p_l||}{\tau} & \text{if } ||p_l|| \quad < \tau \\ \frac{L-||p_l||}{\tau} & \text{if } L - ||p_l|| \quad < \tau \\ 1 & \text{otherwise,} \end{cases} \tag{1.7}$$

where L is the length of the longest path and τ is a threshold value.

1.4.2 Weighted DTW

The conventional DTW computes the dissimilarity between two time sequences by aligning the two sequences based on a sample based distance as in Eq. (1.5). If the sequence samples are multi-dimensional (18 dimensional for the gesture recognition problem), using an Euclidean distance gives equal importance to all dimensions. We propose to use a weighted distance in the cost computation based on how relevant a body joint is to a specific gesture class. The relevancy is defined as the contribution of a joint to the motion pattern of that gesture class. To infer a joint's contribution to a gesture class we compute its total displacement (*i.e.*, contribution) during the performance of that gesture by a trained user:

$$C_j^g = \sum_{n=2}^{N} Dist^j(\mathbf{f}_{n-1}^g, \mathbf{f}_n^g), \tag{1.8}$$

by where g is the gesture index, j is the joint index and n is the skeleton frame number. $Dist^j()$ computes the displacement of jth joint's two consecutive coordinates in feature vectors $\mathbf{f}_{n-1}^g$, and $\mathbf{f}_n^g$. By summing up these consecutive displacements one can find the total displacement of a joint in a selected reference gesture.

After the total displacements are calculated, we filter out the noise (e.g, shaking, trembling) and threshold them from the bottom and the top. This prevents our parametric weight model to output too high or low weights as given below

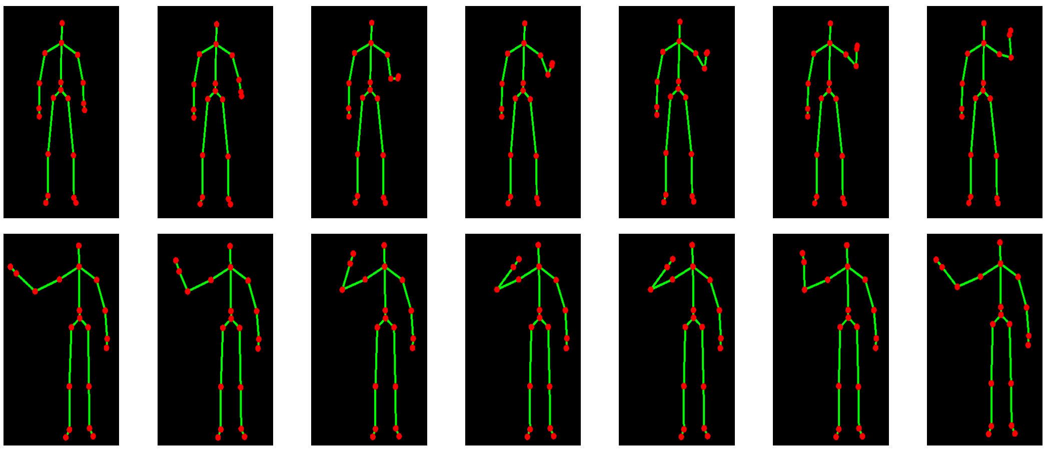

FIGURE 1.7: Two sample reference gestures in the gesture database: Right Hand Push Up and Left Hand Wave.

$$C_j^g = \begin{cases} C_a & \text{if } 0 \leq C_j^g < T_1 \\ \frac{C_j^g - T_1}{T_2 - T_1}(C_b - C_a) + C_a & \text{if } T_1 \leq C_j^g < T_2 \\ C_b & \text{otherwise,} \end{cases} \tag{1.9}$$

where C_a and C_b are threshold values.

Using the total displacement (*i.e.*, contribution) values of joints, the weights of class g are calculated via

$$w_j^g = \frac{1 - e^{-\beta C_j^g}}{\sum_k \left(1 - e^{-\beta C_k^g}\right)}, \tag{1.10}$$

where w_j^g is joint j's weight value for gesture class g. Note that in this formulation a joint's weight value can change depending on the gesture class. For example, for the right-hand-push-up gesture, one would expect the right hand, right elbow and right wrist joints to have large weights, but to have smaller weights for the left-hand-push-up gesture.

To incorporate these weights into the cost, the distance function $d(r_n, t_m)$ becomes a weighted average of joints distances between two consecutive frames and is defined to be

$$d(r_n, t_m) = \sum_j Dist^j(r_n, t_m) w_j^g, \tag{1.11}$$

which gives the distance between nth skeleton frame of reference gesture $\mathbf{R}$ and mth skeleton frame of test gesture $\mathbf{T}$, where $\mathbf{R}$ is a sequence known to be in gesture class g and $\mathbf{T}$ is an unknown test sequence.

The weights are obtained from the model given in Eq. (1.10), which has a single parameter β. Our objective is to choose a β value that minimizes the within-class variation while between-class variation is maximized. Between-class variation maximization and within-class variation minimization can be achieved by making irrelevant joints contribute less to the cost (*e.g.*, reducing the weights of right hand in left-hand-push-up gesture) and not reducing (or possibly increasing) the weights of joints that can help to discriminate different gestures. We try to achieve this goal by maximizing a discriminant ratio similar to Fisher's Discriminant Ratio [27]. To this end, we define $D_{g,h}(\beta)$, as the average weighted DTW cost between all samples of gesture class g and gesture class h using weights calculated with given β. Then between-class dissimilarity is the average of all $D_{g,h}(\beta)$'s $(h \neq g)$ as the following:

$$D_B(\beta) = \sum_g \sum_{\substack{h \\ h \neq g}} D_{g,h}(\beta). \tag{1.12}$$

Within-class dissimilarity is the sum of within-class variances $D_{g,g}(\beta)$ for all g,

$$D_W(\beta) = \sum_g D_{g,g}(\beta). \tag{1.13}$$

The discriminant ratio of a given β, $R(\beta)$, is then obtained by

$$R(\beta) = \frac{D_B(\beta)}{D_W(\beta)}. \tag{1.14}$$

The optimum β, β^*, is chosen as the one that maximizes R:

$$\beta^* = \arg\max_{\beta} R(\beta). \tag{1.15}$$

1.5 Results

We tested the performance of our feature pre-processing and proposed weight distribution method on our three discrete gesture databases to show the improvements separately: (i) Rotationally distorted gesture database: In this database we recorded a set of noisy gestures in terms of the rotational orientation of the body with respect to the Kinect sensor in X,Y and Z axes (See Figure 1.2). The gestures are performed by trained users. This database is designed in order to see the effect of pre-processing on the recognition performance. It has 12 different gesture classes and 21 gesture samples per gesture class. (ii) Relaxed gesture database: In this database there is no intentionally generated rotational distortion, instead, these gesture samples are performed more relaxed in terms of the movement of other body parts out of the active joints. For example in one sample of this database, performer scratches his head with his left hand while he performs the right-hand-push-up gesture. This database has 8 gesture classes and *1116* gesture samples in total. (iii) Rotationally distorted and relaxed gesture database: In this database performers recorded gestures *relaxed* in terms of both rotation and body movement. This database has 12 gesture classes and *198* gesture samples in total. We use this database to show the overall performance of the system. All the three databases are created using Microsoft Kinect Sensor. The databases are available online at `http://mll.sehir.edu.tr/mvaa2013`.

In addition to these databases, there is a set of reference samples per gesture class, performed properly by trained users without any rotational distortion and without any undesired movements. These reference samples are used in learning the total distance measures of each joint in each class, which is required by our weight model in Eq. (1.10). Two sample reference gestures are shown in Figure 1.7.

38 participants joined the gesture recording event. It took approximately one week to finish all the recordings. All participants performed 12 different gesture classes 6 per sample. Bad records, approximately 30% percentage of all recorded gestures, due to a bad gesture performance or Kinect's human-pose recognition failure, were manually deleted by using an OpenGL based gesture visualizer. The physical factors (*e.g.*, distance from the Kinect sensor to the user, illumination in the room) are kept constant during the recording for all records. Each gesture sample includes 20 joint positions per frame, and although we did not use in this work, the time difference between two consecutive frames. The gesture databases used in the experiments, source code for visualization of gestures, source code used to produce the results in this paper and more results are publicly available[1]. We are hoping that the databases can be used in testing other gesture recognition algorithms as well.

In the first experiment, we test our pre-processing method using the rotationally distorted gesture database. We first calculated the discriminant ratios (See Eq. 1.14) of 21 samples for each 12 gesture class without using any of the pre-processing methods. Then, we used the same gesture samples to calculate the discrimant ratios again, but this time using our proposed pre-processing methods. Note that uniform weights were used in order to see the performance of the pre-processing method alone. The effect of pre-processing on the discriminant ratio can be seen in Figure 1.8.

In the second experiment we compared our weighted DTW algorithm against the conventional DTW method and a weighted DTW method proposed by [47] using the relaxed gesture database. The confusion matrices for the three algorithms for six chosen gesture

[1]`http://mll.sehir.edu.tr/mvaa2013`

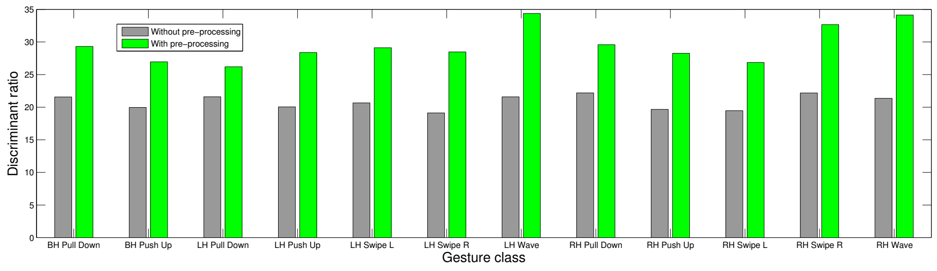

FIGURE 1.8: Discriminant ratios for with and without pre-processed gesture samples using the rotationally distorted gesture database. Note that the discrimant ratios are increased, on average, 42% with the proposed pre-processing method. There are 21 gesture samples in each gesture class. The gesture classes are, namely, Both Hands Pull Down, Both Hands Push Up, Left Hand Pull Down, Left Hand Push Up, Left Hand Swipe Left, Left Hand Swipe Right, Left Hand Wave, Right Hand Pull Down, Right Hand Push Up, Right Hand Swipe Left, Right Hand Swipe Right, Right Hand Wave, respectively.

TABLE 1.1: Confusion matrix for the conventional DTW.

	RH push up	LH push up	RH pull down	LH pull down	RH swipe L	LH swipe R
RH push up	93.9	0	0	2.3	3.8	0
LH push up	2.4	94.6	0.6	0	2.4	0
RH pull down	0	0	98.6	1.4	0	0
LH pull down	2	0	0.7	97.3	0	0
RH swipe L	0	0.8	0	4	95.2	0
LH swipe R	5.6	0	2.1	22.6	0.7	69

classes are given in Table 1.1, 1.2, and 1.3. After creating the confusion matrices, we computed the overall recognition accuracies according to the following formula:

$$A = 100 \cdot \frac{\text{Trace}(C)}{\sum_{i=1}^{m}\sum_{j=1}^{n} C(i,j)}, \tag{1.16}$$

where A denotes the accuracy, and C denotes the confusion matrix.

Our proposed method outperforms the weighted DTW method in [47] by a large margin as given in Table 1.4. The reason is that their weights are global weights, *i.e.*, a joint's weight is independent of the gesture class. However, in our proposed method a joint can have a different weight depending on the gesture class we are trying to align with. This degree of freedom in computing the associated DTW cost increases the reliability of DTW cost significantly.

In the third and the last stage, we tested the overall performance of our system using the rotationally distorted and relaxed gesture database. The purpose of this operation

Table 1.2: Confusion matrix for the weighted DTW in [47].

	RH push up	LH push up	RH pull down	LH pull down	RH swipe L	LH swipe R
RH push up	96.2	1.5	0	0.8	1.5	0
LH push up	3	97	0	0	0	0
RH pull down	0	1.4	98.6	0	0	0
LH pull down	2	0	0	98	0	0
RH swipe L	0	2.4	0	2.4	95.2	0
LH swipe R	7.8	0	0	25.3	0.7	66.2

Table 1.3: Confusion matrix for our proposed weighted DTW.

	RH push up	LH push up	RH pull down	LH pull down	RH swipe L	LH swipe R
RH push up	100	0	0	0	0	0
LH push up	0	100	0	0	0	0
RH pull down	0	0	100	0	0	0
LH pull down	0	0	0	100	0	0
RH swipe L	0.8	0	0	0	99.2	0
LH swipe R	0	0	0	0	2.8	97.2

Table 1.4: Accuracies of the three methods. Note that not only six gesture classes given in Table 1.1, 1.2, and 1.3 are used, but all eight gesture classes are taken into consideration.

Method	Accuracy
Classical DTW	84.41 %
State-of-the art	86.56 %
Proposed method	97.13 %

is to determine the overall improvement of the pre-processing and the weighting on the recognition performance using a larger database. These experiments clearly demonstrate the performance boost provided by our proposed techniques. The results are given in Table 1.5.

1.6 Conclusion

We have developed a weighted DTW method to boost the discrimination capability of DTW's cost, and shown that the performance increases significantly. The weights are based on a parametric model that depends on the level of a joint's contribution to a

Table 1.5: Overall performance comparison using the rotationally distorted and relaxed gesture database.

Method	Accuracy
Traditional DTW	62.41 %
Pre-processing + Traditional DTW	76.26 %
Weighted DTW	84.13 %
Pre-processing + Weighted DTW	96.64%

gesture class. The model parameter is optimized by maximizing a discriminant ratio, which helps to minimize within-class variation and maximize between-class variation. We have also developed a pre-processing method to cope with real life situations, where different body shapes and user orientations with respect to the depth sensor may occur.

Chapter 2

Low-Complexity Shape Recognition Using Random Forest Classifiers with Random Rectangle Features

2.1 Introduction

Shape recognition is an important problem encountered in various applications such as optical character recognition, gesture recognition, medical analysis, and drawing applications [3, 38, 42]. Large number of shape classes, high level of within-class variation, and low level of between-class variation exacerbates the problem. Large number of shape classes necessitates a larger set of training samples to learn which variation in the training dataset contributes to between-class variation. Hence, a good recognition algorithm learns the effect of (combination of) attributes on between-class variation and values attributes accordingly in performing the classification task. High level of within-class variance requires the features to be invariant to some extent to certain transformations and deformations. This requires the features used to be at least partially invariant to these variations. A low level of between-class variation requires the use of features with strong discrimination power or a cascaded application of relatively weaker features (*e.g.*, boosting with Adaboosting) and a larger database to learn such discriminative features. Hence, shape recognition requires a good learning algorithm using either strong features or many weak features. Strong features such as gradient-based statistical features or structural features can be used, but they often have high computational complexity and computing some of these feature can be as difficult as the original classification problem. To achieve fast execution times for real-time applications or low-cost implementations, weak features are utilized. To increase the accuracy of the classifier, cascaded computation of weak features is required. A good example for cascaded learning is the Viola-Jones framework for object detection, which uses a degenerate decision tree [60]. Another example is bagging of decision trees to increase the accuracy and reduce the variance [55]. In this paper, we use random forest classifiers with random rectangle features. Although rectangular features are used in [60] for detection, and random forest classifiers are used in many recognition applications with more complex features [12, 17], we propose to use random forest classifiers in combination with random rectangle features consisting of a single rectangle. We discuss how partial invariance, and stability is achieved with our random rectangle features as compared to other types of features, and how these two properties are related with cascaded learning characteristics of decision trees in Sections 2 and 3. We further discuss how the parameters of a decision tree and a random forest classifier can be optimized to achieve high accuracy, fast execution, and low computational complexity in Sections 4 and 5 by evaluating our proposed method on gesture recognition and optical character recognition. Shape recognition applications in consumer products require more efficient use of computation and memory resources without sacrificing on the quality. Our proposed method can enable applications in various fields requiring shape recognition with low memory and time budgets, due to its high accuracy, low complexity, and scalability.

2.2 Shape Features for Recognition

Features used in shape recognition can be loosely divided into statistical and structural shape descriptors. Statistical features are direction features utilized within a statistical framework. For example, the histogram of gradients in a locality is used as a descriptor for the orientation. These local gradient histograms can be aggregated via clustering to create global histograms so that not only local but global descriptors are also used as features [33, 35, 37]. Statistical features are invariant to within-class variations such as scaling, rotation, and illumination change [33, 36]. Directions are usually computed after low-pass filtering (*e.g.*, Gaussian filtering), which is performed to remove random variation and improve accuracy. A commonly used type of structural feature is (silhouette) contour descriptors which measure curvature, concavity, convexity, shape-part structure [21]. However, contour features are sensitive to nonlinear variations, structural changes, and articulation [5]. Contour features have lower dimensionality compared to the statistical features, and structural variations that degrade the feature quality can have a significant overall impact. Skeleton features are another form of structural features which extract the skeleton of the shape. Skeleton features perform better than contour features under structural variations but skeleton stability is often a problem and matching of skeleton graphs is still an open research area [6, 11, 24, 51]. Since statistical and structural features are fairly independent descriptors, using statistical and structural features in combination improves the recognition performance [34, 35, 56]. Although statistical and structural features are invariant up to an affinity, they are sensitive to image degradation. Moreover, statistical and structural features may not always be stable since both statistical and structural features are complex features and require an algorithm working on intensity image data. Oftentimes, a preprocessing (normalization) stage is needed to correct for translation, slant, and rotation. These steps may reduce the robustness due to noise, blur, and illumination changes, etc. On the other hand, random rectangle features are more stable and primitive as compared to statistical and structural features at the expense of being partially invariant.

Important attributes of tree-based rectangle features are (i) *cascaded learning*, corresponding to increasing structure and complexity (ii) *partial-invariance*, most samples of a given class will more likely give similar responses to similar features as they move down the tree (iii) *stability* to noise and other randomness since rectangle features do not depend on orientation or other intensity-gradient based features.

Statistical features such as Scale Invariant Feature Transform (SIFT) descriptors are invariant to image translation, scaling, rotation, and partially invariant to local affine distortion and illumination changes [37]. SIFT features uses Difference of Gaussians (DoG) function applied in scale space to find key points as feature candidates, which are reduced in the later processing stages. Gradient based descriptors compute quantized gradient histograms on Gaussian smoothed images. Although statistical features are invariant to certain variations, they might not be stable under nonlinear deformations and also require preprocessing stages such as normalization. Hence, statistical features are sensitive to degradation in intensity data such as blur, noise, compression artifacts or distortions in shape due to articulation or human errors, etc. Structural features need contour or skeleton extraction, and detection of structural parts. This task might as well be as complicated as detecting the whole structure, *i.e.*, the shape. For example, recognizing a hole or a line independently may be more difficult than recognizing them jointly as in character "d".

Rectangle features on the other hand are not invariant to translation, scaling, rotation, and projective perturbations in general. However, they are partially invariant and the degree of partial invariance increases with the area of rectangle. If we consider rotation as an example, the area of the shape that resides in a rectangle will vary as the shape is rotated but will vary to a lesser extent if the rectangle is enlarged. At the early stages of a decision tree, larger rectangle features are selected as splitting features, which improves the invariance. This is expected since at lower tree levels class label entropy will

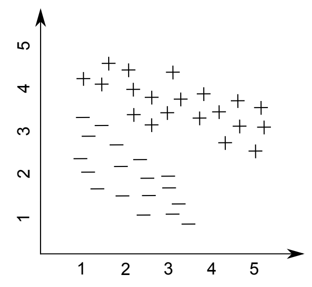

FIGURE 2.1: An example data instance

be larger, therefore same class samples will have high variance compared to same class samples that reside on tree nodes at higher tree levels. Cascaded learning in the form of a tree, will tend to favor larger rectangle features in lower levels. However at higher levels, due to the learning process, entropy of the class label c will be smaller and learning will become a more difficult problem, necessitating more "complex" rectangle features that are smaller or more oriented, *i.e.*, thinner in the horizontal or vertical direction.

2.3 Decision Tree Based Classification

2.3.1 Decision Trees and Random Forests

Decision trees are nonlinear classifiers and therefore aim at learning complex boundaries in the feature space using a training data set. The partitions formed by these boundaries are desired to be *pure* in the sense that each partition contains same class members. Classification of a future sample reduces to finding out which partition the sample lies in, and predicting the class label using the training data in that partition. If the feature space is under-partitioned, the partitions may not be pure enough to accurately predict the class label. On the other hand if the feature space is over-partitioned, partition boundaries might not reflect the true boundaries imposed by the data-generation process, and be affected by noise and other random variations in data. Under-partitioning, and over-partitioning corresponds to under-fitting and over-fitting respectively, which are terms used in classification literature. The under-partitioning leads to a high bias error with low variance in class prediction, and the over-partitioning case leads to a poor generalization performance.

Decision trees ask discriminative questions successively to infer the class of a data sample, and these questions are structured as a tree. Each tree node asks a question about

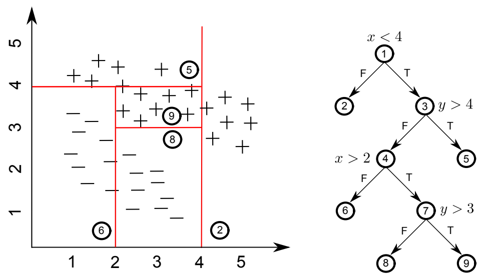

FIGURE 2.2: Univariate decision tree

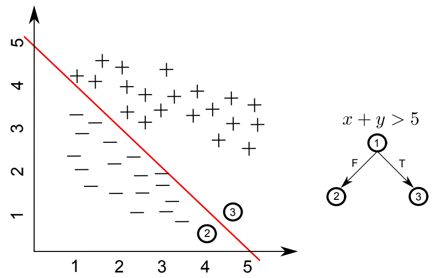

FIGURE 2.3: Multivariate decision tree

the features of a data sample[1]. The goal in the training phase is to choose the most discriminative question to ask at each node. The training data set associated with each tree node is split according to each data sample's answer. A binary decision tree asks YES/NO questions and therefore the training set associated with each node is split into two. A question can be about a single attribute or a combination of attributes of a data sample. The two question types lead to univariate or multivariate decision trees, respectively. The most discriminative question out of a specified set of questions is found as the maximizer of some *purity* measure such as the negative total entropy in the post-split data sets. A new data sample is classified by sending it down the tree by routing it according to its answers to the questions along its path from the root node to the leaf node. The leaf node predicts the class label using its training samples. Consider the training data samples shown in Figure 2.1. Let $\mathcal{I}$ denote a space of 2D points $\mathbf{I} = (x_1, x_2)$. Each $\mathbf{I} \in \mathcal{I}$ has a class label $c(\mathbf{I}) \in \mathcal{C} = \{+, -\}$. If single feature questions are learnt from the training data set, a univariate decision tree of depth four such as the one given in Figure 2.2 can be obtained. However, if questions involve linear combinations of attributes, a multivariate tree that consists of a single node as given in Figure 2.3 can be learnt, and the partitions formed by the decision tree is also shown. As can be seen from the figures, a univariate tree can only split the feature space at a node with a boundary that is orthogonal to the feature axes, resulting in space partitions that are hyperrectangles with sides parallel to the axes. However,

[1]A data sample can be a multi-dimensional vector rather than a scalar value, and each component of a data vector is called an attribute of that data sample. A question can involve a single attribute, or a combination of attributes. The first type of question is called a univariate feature, and the second type of question is called a multivariate feature. In this paper, we use *feature* and *question* interchangeably.

more discriminative questions can be asked using a linear combination of attributes, which splits the data space using hyperplanes, resulting in complex polyhedral space partitions. Since univariate questions are more restricted and therefore generally less discriminative than multivariate questions, univariate trees tend to be larger, *i.e.*, more univariate questions are needed to be asked to learn a discrimination boundary between samples of different classes.

Although multivariate trees can lead to a better partitioning of the data space, their training is more involved. Selecting attributes to be used in a linear combination for constructing a multivariate question is a difficult problem since the number of linear combinations grow exponentially with the attribute size. Moreover, choosing the combination weights is also a problem [14]. For example to classify an image sample, all combinations of attributes (*i.e.*, pixels) grows exponentially with the number of pixels in the image.

Training decision trees by maximizing a purity function at each node is a greedy heuristic which causes sensitivity: a small change in the training data set may result in a very different decision tree and data space partitioning. This means the classification performance depends on the particular instance of training data set leading to poor generalization performance. To reduce variance, bagging is used to train more than one decision trees on variants of the training set. Predictions of trees can be aggregated by letting each tree vote for a class and making the final decision in favor of the majority class. Another typical aggregation technique is to create histograms for each leaf node reach in tree t to approximate the probability distribution over the class labels $P_t(c|I)$, and compute the average histogram of all trees.

To improve the bagging performance, random forests reduce correlations between trees by randomization in their training. Randomization is achieved by randomly selecting a set of feature candidates for split decisions in addition to bootstrap techniques to create variants of the training set for each tree. Hence, randomization enforces each tree classifier ask different questions about the shape which improves the learning-from-data process. Below is a binary decision tree learning algorithm for a random forest

1. Randomly propose K splitting questions $\mathcal{Q} = \{Q_i\}$ if size of data set $\mathcal{I}$ is large enough

2. Split the set of examples $\mathcal{I}$ into left and right subsets according to their answer to each question

$$\begin{aligned} \mathcal{I}_l(Q_i) &= \{I|Q_i(I) = \text{YES}\} && (2.1) \\ \mathcal{I}_r(Q_i) &= \mathcal{I} \backslash \mathcal{I}_l(Q_i) && (2.2) \end{aligned}$$

3. Choose the question Q_i that maximizes a purity measure P

$$Q^* = \arg\max_{Q_i} P(Q_i) \tag{2.3}$$

$$P(Q_i) = -\sum_{s\in\{l,r\}} \frac{|\mathcal{I}_s(Q_i)|}{|\mathcal{I}|} H(\mathcal{I}_s(Q_i)), \tag{2.4}$$

   where negative entropy is used as the purity measure on the class label histograms derived from two split example sets, which are weighted with the cardinality of the two sets.

4. Recurse for left and right example subsets $\mathcal{I}_l(Q^*)$ and $\mathcal{I}_r(Q^*)$ if the depth in the tree has not exceeded a pre-defined maximum value.

The above algorithm randomly proposes K questions as candidates for splitting the examples, exits if a maximum number of depth in the tree is achieved or the pre-split example set does not have enough elements.

2.3.2 Random Forest Classifiers For Recognition

Randomized decision trees and forests have been used in multi-class classification problems due to their low complexity and high accuracy[31, 39, 54]. Using random forest classifiers, real-time performance can be achieved in difficult computer vision and machine learning problems such as human-pose recognition or gesture recognition [26, 55]. In [55], random forests are used to classify each depth pixel into intermediate body parts that are spatially localized near skeletal joints of interest. Pixel classification into 31 intermediate body parts transforms the human-pose recognition problem into a multi-class problem that can be efficiently solved using random forests in real-time. The features used in split decisions are depth differences of two pixels in the locality of the current pixel to be classified. The two pixels used in this bivariate feature are obtained by offsetting the current pixel $\mathbf{x}$, and the offset values are normalized using the current pixel's depth resulting in depth invariant features as given below.

$$f_\theta(I,\mathbf{x}) = d_I(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}) - d_I(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}), \tag{2.5}$$

where $d_I(\mathbf{x})$ is the depth at pixel $\mathbf{x}$ in image I. $\mathbf{u}$ and $\mathbf{v}$ represent the two-dimensional offset vectors, which are depth-normalized by $\frac{1}{d_I(\mathbf{x})}$. Bivariate features have weak discriminatory power (*e.g.*, if the above pixel is checked and found out to be in the background, the pixel can belong to the head or the two shoulders). Although these features have weak discriminatory power, cascaded use of these features as in the form of a decision tree reduces the bias of the classifier, *i.e.*, a decision tree classifier accurately disambiguates the body parts. Moreover, using an ensemble of randomized decision trees (*i.e.*, random forest), the variance of the classifier is reduced and the accuracy increases. This system runs at 200 frames per second on consumer hardware thanks to low-complexity depth features and random forest classifiers which enable parallel implementation.

Per-pixel random forest classifiers are recently used in hand shape recognition on depth image data and tested on American Sign Language (ASL) and hand gesture datasets [26]. High accuracy rates are reported without resorting to the use of color images. Similar to [55] as discussed above, random forest classifiers are used on depth images, and the same bivariate depth feature in (2.5) is used for making discriminative split decisions at split nodes.

Recognizing the shape (*i.e.*, human skeleton) in parts (*i.e.*, body joints/parts) necessitates per-pixel classification because more than one classes (*i.e.*, body parts) will exist in the same image. Hence, there are segments in the image with different class labels. To recognize each segment using more than one pixel, one needs to know where the segments are located, and their boundaries, etc. Therefore, a per-pixel based classification significantly simplifies the algorithm. However, the number of classification problems to be solved increases by the number of pixels. When there is one shape in the image or detection has already been priorly performed to find the region of interest, shape classification can be performed on the whole image without requiring per-pixel classification. Moreover, a bivariate feature as in (2.5) might not be reliable when the shape structure involves thin structural details oriented in varying directions, which makes the bivariate feature less invariant to structural and "pose" changes, or image degradations. We tailored the technique in [55] for Optical Character Recognition (OCR) by using per-pixel random forest classification together with bivariate features given in (2.5). The error rate was unacceptably high around 40% on the MNIST digit database. However, the same technique applied to Gesture Recognition (GR) on American Sign Langugage (ASL) dataset using only static ASL letters achieved a recognition rate of 85% using only depth data, which is significantly higher than the state-of-the-art recognition rates (*e.g.* 75% achieved in [43], see Section 2.5.2 for details). The better performance of per-pixel classification with bivariate features on GR is due to a lesser degree of fine details and structural variations in the hand gestures compared to hand-written digits which can have high degree of structural variation and fine details.

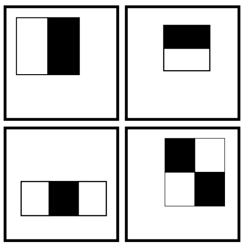

FIGURE 2.4: Example Viola-Jones features relative to the detection window. The sum of the pixel intensities in the grey rectangular regions are subtracted from the white rectangular regions.

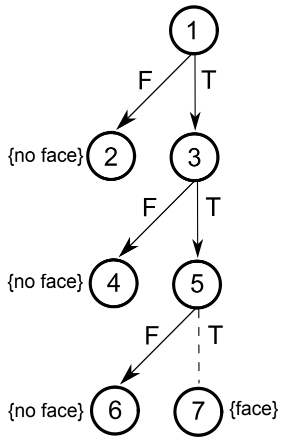

FIGURE 2.5: Cascaded application of classifiers enable focusing on object-like regions.

2.3.3 Viola-Jones Object Detection Framework

To improve the robustness, one can utilize features in the form of a filter that aggregates local information in its support. For example statistical features exploit pyramid filtering schemes by using steerable filters. Steerable filters can be oriented according to the structural details and can extract more useful information about orientation and shape in a locality [22]. Haar wavelets are examples of such filters. Haar wavelets are used in many applications for recognizing and detecting shapes in the image [4, 63, 67]. First used in face detection, and then used in object detection in general, Viola-Jones features are Haar-like features and compute differences of intensities in rectangular regions (see Figure 2.4 for some examples features) [60]. With the use of an image representation called the *integral image*, Viola-Jones features can be computed in constant time independent of feature scale (*i.e.*, rectangle size). Viola-Jones features consist of adjacent rectangular regions, and the total intensities inside adjacent rectangular regions are either summed or subtracted to compute the feature value. Adaboosting is used to train a weak classifier at each boosting stage. A weak classifier is constrained to use a single feature. As a result, classifier selection at each boosting stage reduces to a feature selection process. Each classifier is trained to have a low false negative rate of approximately 0% and a false positive rate of 40%. These classifiers are applied in cascade. Sub-windows rejected by a classifier at any stage of the cascade is not processed further thanks to the very low false negative rate. This method successively discards non-object regions and spends more processing time on regions that resemble the object of interest (see Figure 2.5), thereby achieving real-time execution. The classification structure of the Viola-Jones method is essentially a degenerate decision tree, in which the left nodes are always leaf nodes and are labeled NO. There exists a single leaf node labeled YES and it requires more computation to reach, compared to any other node in the tree.

In object detection the goal is to detect the object inside the image, and the object can be present at any scale and at any spatial location. Hence, the cascaded weak classifiers has to process various sub-windows in the image using classifiers of varying scale. This means Viola-Jones features of different sizes need to be evaluated at different spatial locations in the image. On the other hand object recognition assumes object detection has been performed before. Hence, the scale and the location is approximately known. In this sense recognition is a simpler problem compared to detection. But object recognition in general is a multiclass classification problem while object detection is a binary classification problem. Object detection algorithms such as Viola-Jones object detection framework take advantage of this by utilizing (weak) single-feature classifiers with low false negatives. The degenerate structure of the decision tree enables early termination for a NO label, which means that regions that are not object-like are reliably labeled early in the process. This is possible because in detection there are two classes and each classifier in the cascade is trained for achieving a low false negative rather than both low false negative and low false positive. However, in recognition there are more than two classes and it is difficult to find a (weak) single-feature classifier to recognize a class and create a leaf node for early termination. Hence, a degenerate decision tree would not be a good fit for an object recognition task. Cascaded application of more than one weak classifiers will be needed to recognize an object leading to more balanced decision trees.

2.4 Random Forest Classifiers with Random Rectangle Features

We use random forest classifiers with random rectangle features for shape recognition. Random forest classifiers have lower computational complexity compared to other classifiers such as support vector machines (SVMs), neural networks, or nearest-neighbor type classifiers [55]. Random forest classifiers with bivariate features fail to learn the shape structure when there are thin details in the structure because the bivariate (two

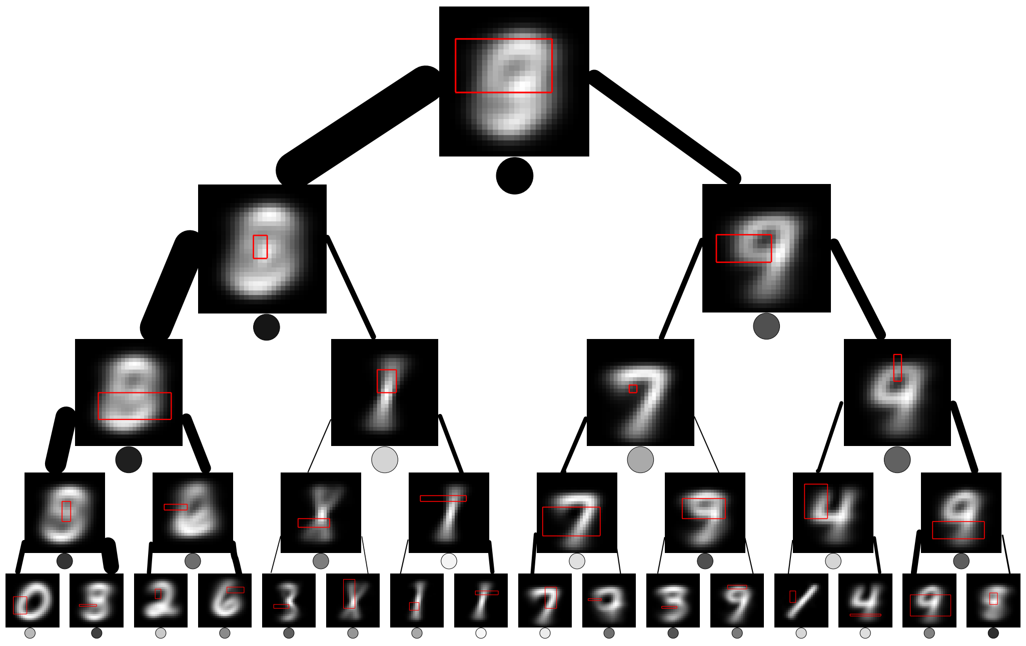

FIGURE 2.6: Visualization of a decision tree up to the fourth depth level trained on OCR data. The thickness of the edges connecting nodes is proportional to the number of images associated with the receiving node. The average of all training images at a node is displayed. The brightness of the circle below the average image is inversely proportional to the entropy at that node: the higher the entropy the darker the color of the circle.

pixel) feature may not be stable due to variations in the thin structural parts. For example using bivariate features for hand-written digit recognition performs poorly with an accuracy rate of 60%. However, rectangle features are more insensitive to in-class variations due to the aggregation of pixels inside a feature's rectangular region. We use a simple feature that consists of a single rectangle, which is even more primitive than the Viola-Jones features. Viola-Jones filters employ simple to complex single-feature classifiers starting from two-rectangle features to more complex features involving more rectangles and their various additive or subtractive combinations. This is required because Viola-Jones detection framework uses a degenerate decision tree whose NO-branch is always a leaf node detecting the non-existence of the searched shape. Hence, the false negative rate has to be kept extremely low, which requires asking more and more difficult questions (*i.e.*, more complex features as combinations of rectangles). However, a balanced decision tree will continue asking questions both on the YES and the NO branch. Therefore the questions do not need to become difficult: cascaded application of more primitive features will be able to perform successive splits and purify the label distribution.

A rectangle feature is a multivariate feature that uses a combination (summation) of all pixel intensities in a rectangle given by

$$f_{\mathbf{r}}(\mathbf{I}) = \mathbf{r}^T\mathbf{I}, \tag{2.6}$$

where **I** is a vector of image pixels, and **r** is a vector of zeros except nonzero elements of value one corresponding to the pixels inside the rectangle. A rectangle feature can be computed in constant time independent of its size using the integral image [60]. At each split node a set of random features $\{f_{\mathbf{r}}(\mathbf{I})\}$ and threshold candidates $\{T\}$ are created and out of the corresponding binary questions

$$\{Q = (f_{\mathbf{r}}(\mathbf{I}) < T) \text{ ? YES : NO}\}, \tag{2.7}$$

the best question is chosen. An example decision tree up to depth level four trained on OCR data set is shown in Figure 2.6. The red rectangle shows the best split rectangle depicted on the average image, which is the average of all images at that node. The data sets are purified at the higher depth levels and the average images start to resemble one of the ten digits.

2.4.1 Rectangle Feature As A Linear Combination

The *best* split rectangle feature is a multivariate feature which is a linear combination of attributes (*e.g.*, pixels) with all coefficients equal to one. There are two important problems in finding a good linear combination of attributes. The first is selecting the attributes to be included in the linear combination, and the second is learning their coefficients. To select the attributes there are two basic approaches: Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS). SFS is a bottom up search method that starts with zero attributes and adds the attribute that causes the biggest increase in the purity measure until a stopping criteria is met. On the other hand, SBS is a top down search method for attribute selection. To learn the coefficients of the linear combination, techniques such as Recursive Least Squares (RLS) which minimizes mean-squared error over the training data or CART which explicitly searches for a set of coefficients that maximize a purity measure is utilized. Both attribute selection and coefficient learning applied without any restriction will lead to general multivariate features that are not the sum of pixels in a rectangular area.

To be able to use rectangle features in combination with the integral image, we fix all coefficients to be one and randomly select attributes by creating a set of random rectangle features rather than a top down or a bottom up search technique. This randomization also improves the accuracy of the random forest classifier by reducing the correlation between trees. The best rectangle is chosen as the purity maximizer and then refined by perturbing its top-left and bottom-right corners. A typical corner perturbation can be a one pixel horizontal/vertical shift in the two-dimensional space, which results in a set of 25 perturbed rectangle candidates. The split rectangle feature is determined by refinement iterations using the corner perturbation technique. Rectangle refinement can produce discriminative rectangles, *e.g.*, the split rectangle of the rightmost node at depth level 2 as given in Figure 2.6, which most likely separates 4's and 9's in its data set.

Below is the algorithm for training a single tree using random rectangle features.

1. Randomly propose a set of rectangle features $\{\mathbf{r}\}$ and a set of candidate threshold $\{T\}$ for each $\mathbf{r}$ in $\{\mathbf{r}\}$
2. Create a question Q_i for each $\mathbf{r}$ and T
3. Split the set of examples $\mathcal{I}$ into left and right subsets according to their answer to each question

$$\begin{aligned} \mathcal{I}_l(Q_i) &= \{I \mid Q_i(I) = \text{YES}\} && (2.8) \\ \mathcal{I}_r(Q_i) &= \mathcal{I} \backslash \mathcal{I}_l(Q_i) && (2.9) \end{aligned}$$

4. Choose the question Q_i that maximizes a purity measure P

$$\begin{aligned} Q^* &= \arg\max_{Q_i} P(Q_i) && (2.10) \\ P(Q_i) &= -\sum_{s \in \{l,r\}} \frac{|\mathcal{I}_s(Q_i)|}{|\mathcal{I}|} H(\mathcal{I}_s(Q_i)), && (2.11) \end{aligned}$$

5. Create a new set of questions $\mathcal{Q}$ by perturbing left-top and right-bottom corners of $\mathbf{r}^*$ of Q^*
6. Find Q^{**} in $\mathcal{Q}$ by maximizing the purity measure
7. Go to step 5 until an exit criteria holds
8. Recurse for left and right example subsets $\mathcal{I}_l(Q^{**})$ and $\mathcal{I}_r(Q^{**})$ if the depth in the tree has not exceeded a pre-defined maximum value.

2.5 Experiments

In this section we describe the experiments performed to evaluate our Random Forest Classifier with Random Rectangle Features (RFCwRRF) method. We apply our method to Optical Character Recognition (OCR) and Gesture Recognition (GR), and evaluate the accuracy rate with respect to parameters such as number of trees, maximum depth, data set size, number of random rectangles used as split candidates at each node, etc. Unless otherwise stated, in all experiments the parameters of our RFCwRRF method are set as 20 trees of maximum depth 14, 100 candidate rectangles with 20 candidate thresholds per rectangle at each split node.

2.5.1 Optical Character Recognition Results

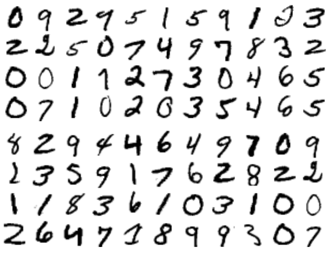

FIGURE 2.7: Sample digits from the MNIST database

We use the MNIST database for OCR, which is a digit database that consist of 60,000 training samples and 10,000 test samples [29]. Sample images from the database is given in Figure 2.7. Each sample is a 28×28 gray-scale image. We scaled the training database up by 4x by applying moderate rotational and affine transformations randomly. Increasing the database size helps with the overfitting problem, which is exacerbated by having deep trees (of maximum depth 14). Also, randomization more effectively reduces the correlation between the trees and therefore the accuracy of the forest classifier increases. Visualization of a trained tree is given in Figure 2.6.

Computation speeds and required memory space is important in analyzing the performance of a classifier, especially if the goal is to design a cost-effective real-time classifier. We compared our classification times against the classification times of the state-of-the-art given in [35]. We implemented the k-nearest-neighbor (k-NN) classifier on our system and used k-NN's classification time as a reference point in the comparison: by using the ratio between our k-NN classification time and the time given in [35], we updated other classification times in [35] accordingly.

The classification times are given in Table 2.1. [35] reports classification times using four different features. A principal component analysis (PCA) based feature, a 4-orientation gradient feature, an 8-orientation gradient feature, and pixel intensity. Our proposed RFCwRRF method implementation uses the image pixel intensities only. Rectangle features may as well operate on multi-dimensional features derived from the image data. And it is possible to extend the integral image technique to multi-dimensional data (*e.g.* gradient) to improve the accuracy of the classifier. However, our goal was to achieve a fast classifier with accuracy rates similar to the state-of-the-art. Hence, RFCwRRF results are given for intensity features (raw pixel data) only, while the other classifier results are given for all the four features. k-NN is the most expensive in terms of computation and memory resources since it has to compute the distance between a new image and all the (60K) training images in the database. Our result is about 40 times faster than the MLP (multilayer perceptron) classifier, which is the fastest among the classifiers in the table. The SVC (support vector classifier) is considerably slower than the other classifiers. This is because the number of support vectors is very large slowing SVC down to the level of k-NN classifier.

TABLE 2.1: Classification times (milliseconds) on an example image of size 28×28.

Classifier	Time
RFCwRRF	**0.010**
k-NN	49.66
Per-pixel RFC	7.832
MLP	0.402
RBF	0.518
PC	0.437
LVQ	0.719
LQDF	0.463
SVC-poly	8.098
SVC-rbf	31.589

Table 2.2 reports the accuracy of each classifier on the MNIST test set. RFCwRRF has an accuracy rate of 98.01%, which is slightly less than the highest accuracy classifier. The best performing classifier (SVC-rbf) which is a support vector classifier, has a 98.59% accuracy, while requires 3158.9 times more computation (see Table 2.1 and 2.2). Per-pixel random forest classifier (RFC) performs significantly lower than the rest. This is due to the use of bivariate features which have a significantly lower discrimination power on the thin details of digits and structural variation in hand-writings as discussed in Section 2.3.2. Moreover, per-pixel RFC requires high computational resources comparable to support vector classifiers since pixels rather than the whole image need to be classified.

Figure 2.8 shows accuracy as the number of trees is increased. The improvement slows down around 7 trees but accuracy increases approximately monotonically up to 30 trees. The maximum accuracy (98.01%) is reached with the maximum number of trees.

Figure 2.9 shows the effect of the number of candidate rectangles used during training on the test classification accuracy. The accuracy of a single tree trained with 300 candidates has 1.2% higher average accuracy compared to a tree trained with 50 candidates. The accuracy difference between the two configurations does not change significantly with increasing number of trees. One might expect that using 300 candidate rectangles at each

Table 2.2: Classification accuracies on the MNIST test set.

Classifier	Accuracy
k-NN	96.34
RFCwRRF	**98.01**
Per-pixel RFC	60.74
MLP	98.09
RBF	97.47
PC	98.36
LVQ	97.21
LQDF	98.03
SVC-poly	98.31
SVC-rbf	98.59

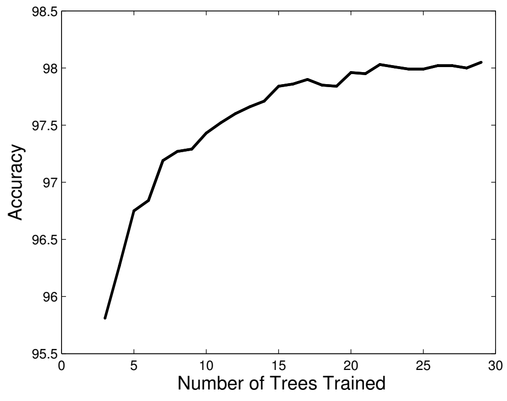

Figure 2.8: Accuracy as the number of trees is increased

split node will increase the correlation between the trees, hence the accuracy will improve to a lesser extent with increasing number of trees in the forest. However, Figure 2.9 shows that the accuracy difference only slightly decreases. Since the number of possible rectangles in a 28×28 image is 142884, increasing number of candidate rectangles from 50 to 300 does not have a significant impact on the correlation between the trees. Hence, the accuracy increases with the number of trees similarly in both training configurations.

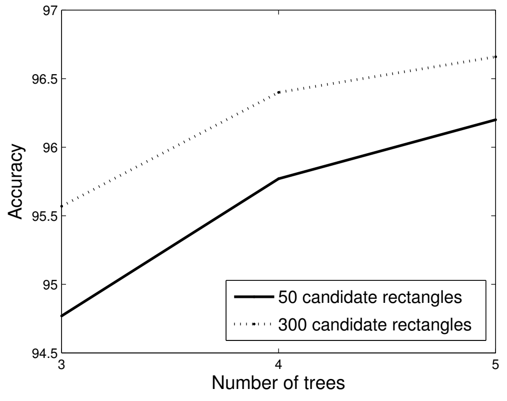

Figure 2.9: The effect of the number of candidate rectangles per each split node

Increasing the maximum depth increases the number of parameters to be learnt (*i.e.*, more questions to be asked), which gives more degrees of freedom to partition the feature space. However if the training set is not large enough, some split decisions may not be reliable and partitions may have too few training samples. Figure 2.10 shows accuracy as the maximum tree depth is increased using a random forest of five decision trees on two different training sets: 60k and 180k images. On the 60k training set, the accuracy goes down after maximum depth 12. However, on the 180k training set, accuracy of the forest keeps increasing despite a slowing down in the increase. This shows that increasing the maximum depth causes overfitting to training data rather than learning generalizable relations about the data distribution when 60k images are used. To avoid a decrease in the forest accuracy, more number of trees can be used as compared to five trees used to obtain the plot. Figure 2.11, gives the average per-tree accuracy on the training and testing sets for both 60k and 180k training sets. For both training sets when maximum depth is set to eight, both training and testing set accuracies are close but low, indicating that there is bias in $\hat{P}_t(c|I)$. However, as the maximum depth is increased training accuracy increases while testing accuracy converges, which shows that additional tree parameters (questions) are used to overfit the classifier to training data. The gap between testing and training accuracies is less using 180k training images, which shows the effectiveness of our artificially warped images in learning. There may be potential to increase the training set size even further by using a richer set of warping types.

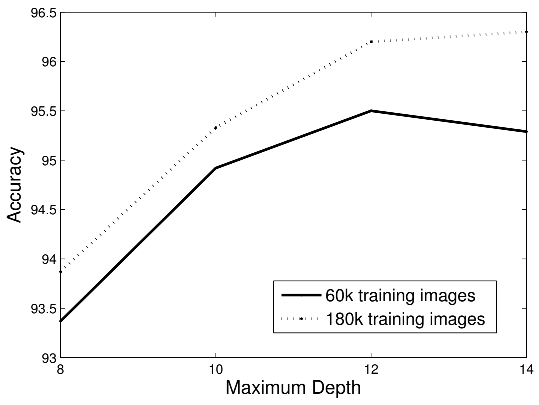

FIGURE 2.10: Accuracy versus maximum tree depth using a random forest of five randomized trees.

In Figure 2.12, a mixed set of features including features other than our single random rectangle feature is compared. Below we list the types of features used.

- Single rectangle features: Randomly created single rectangle features.

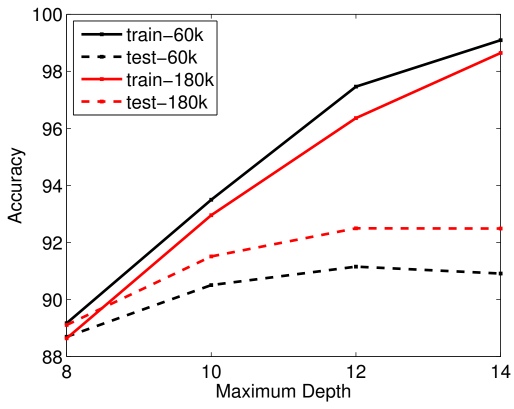

FIGURE 2.11: Average per-tree accuracy versus maximum tree depth

- Double rectangle features: Two random rectangles and random combination of the two rectangles by adding or subtracting the summed intensities of the rectangles. This is a generalization of two-rectangle Viola-Jones features by removing the adjacency requirement.
- Single rectangle features mixed with Viola-Jones type two-rectangle features: Two horizontally or vertically oriented, adjacent rectangles as Viola-Jones type features in addition to single random rectangle features.

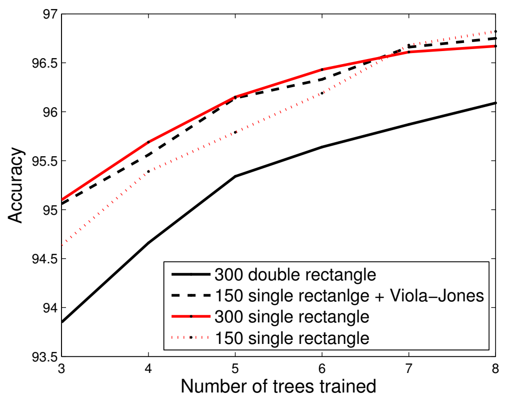

FIGURE 2.12: Accuracy versus number of trees using different features

In Figure 2.12, accuracy of the forest classifier as the number of trees changes, is given for single rectangle features using 300 and 150 candidates, double rectangle features using 300 candidates, and single rectangle features mixed with Viola-Jones features (150 candidates for each). We can see that the accuracy is lower when double rectangle features are used as opposed to single rectangle features (see the accuracy plots for 300 double and single rectangle features). This might seem counter intuitive in the sense that double rectangle features are more complex features as compared to single rectangle features, and hence can be expected to discriminate the classes better. The reason is that the number of possible double rectangle features N_d is much larger than the number of possible single rectangle features N_s, *i.e.*, $N_d = N_s^2$. To find *good* double rectangle features, more candidates needs to be tried. Hence, discriminative power of complex features goes down if the number of candidates is not increased accordingly[2]. By the same reason, the accuracy with 150 single rectangle candidates is lower than 300 single rectangle candidates using a small number of trees. However, as the number of trees in the forest grows, 150 single rectangle candidates outperforms 300 single rectangle candidates in terms of accuracy. Using 150 feature candidates instead of 300 feature candidates, the correlation between the decision trees goes down (*i.e.*, tree nodes are more likely to ask different questions), hence the forest aggregation improves the accuracy faster as the number of trees is increased. This tradeoff can be used to achieve a specified accuracy depending on the execution time requirements of the system, which determines the number of trees used. It is also important to note that using two-rectangle Viola-Jones features does not improve the performance. This is due to the fact that decision trees perform cascaded learning, and single rectangle features can achieve the same learning power as more complex features, but probably with more number of learning stages (*i.e.*, tree nodes). We also experimented with using 45 degrees rotated rectangles, which also did not improve the accuracy. More complex features such as Viola-Jones features, or rotated rectangles may help with a detection algorithm, which requires a significantly low false positive to cut down the computational complexity, but for a recognition problem they do not perform better than a simple feature such as a single rectangle.

2.5.2 Gesture Recognition

For testing our RFCwRRF method on gesture recognition, we use the publicly available ASL dataset. We only use the 24 static letters out of the 26 letters, since our focus is shape recognition, not motion patterns in an image sequence. We only use the depth images as input to our classifier but color images can as well be used with our method, as discussed above in using rectangle features on multi-dimensional gradient histogram data.

We compare our method with per-pixel random forest classifier as before, and also with Pugeault[43], which uses a random forest classifier on statistical features obtained by Gabor filters at four scales and four orientations. The accuracy results of the three methods are given in Table 2.3. Using RFC with statistical features performs the worst

[2]Breiman proposes a logarithmic relation between the number of candidate features and the number of all possible features [13]. Given that $N_d = N_s^2$, the number of candidate double rectangle features at each split node has to be twice as large as compared to the single rectangle features. We did not run experiments for 600 double rectangle candidates per node, as this slowed down the training phase tremendously.

although it uses the color images in addition to the depth images, and the other two methods operate on depth images only. Per-pixel RFC achieves 85% accuracy which improves Pugeault's results by about 10%. This is in contrast to the OCR results in which per-pixel RFC performed considerably worse. This is because the gesture images do not contain fine structures and thin details as compared to digit images, although there is a significant structural variation in the gesture samples depending on the person. RFCwRRF performs the best among the three methods, achieving an accuracy score of 91.3%.

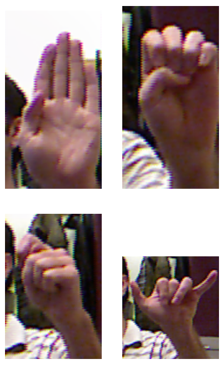

Figure 2.13: The characters in order b,e,t,y

Table 2.3: Gesture recognition accuracies

	Accuracy
Pugeault (color + depth)	75
Per-pixel RFC	85.04
RFCwRRF	91.33

2.6 Conclusion

We proposed using random rectangles as features for training random forest classifiers. Our rectangle features consist of a single rectangle which have a low computational complexity and can be computed in constant time. We compared our proposed method with other techniques that use rectangle based features and random forest classifiers in classification problems involving object/shape detection and recognition. We further studied the effect of classifier and feature parameters as well as the training dataset size on the performance.

Bibliography

[1] N. H. Adams, M. A. Bartsch, J. Shifrin, and G. H. Wakefield. Time series alignment for music information retrieval. In *ISMIR*, 2004.

[2] T. B. Amin and I. Mahmood. Speech Recognition using Dynamic Time Warping. In *International Conference on Advances in Space Technologies*, 2008.

[3] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[4] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 255–260. IEEE, 2005.

[5] X. Bai, X. Yang, D. Yu, and L. J. Latecki. Skeleton-based shape classification using path similarity. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(04):733–746, 2008.

[6] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38(15):2365–2385, 1998.

[7] L. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[8] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970.

[9] R. Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc*, 60(6):503–515, 1954.

[10] R. Bellman and R. Kalaba. On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):1–9, 1959.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[12] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[13] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[14] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine learning*, 19(1):45–77, 1995.

[15] Y.-J. Chang, S.-F. Chen, and J.-D. Huang. A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, 32(6):2566 – 2570, 2011.

[16] A. Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on*, pages 82–89. IEEE, 2001.

[17] R. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[18] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, 2007.

[19] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, pages 296–301, 1994.

[20] D. Gehrig, H. Kuehne, A. Woerner, and T. Schultz. Hmm-based human motion recognition with optical flow data. In *IEEE International Conference on Humanoid Robots (Humanoids 2009)*, Paris, France, 2009.

[21] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):1991–2005, 2006.

[22] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 222–228. IEEE, 1994.

[23] P. Hong, T. S. Huang, and M. Turk. Gesture modeling and recognition using finite state machines. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, FG '00, pages 410–, Washington, DC, USA, 2000. IEEE Computer Society.

[24] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, 1993.

[25] H. P. Jain, A. Subramanian, S. Das, and A. Mittal. Real-time upper-body human pose estimation using a depth camera. In *Proceedings of the 5th international conference on Computer vision/computer graphics collaboration techniques*, MIRAGE'11, pages 227–238, Berlin, Heidelberg, 2011. Springer-Verlag.

[26] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Randomized decision forests for static and dynamic hand shape classification. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 31–36. IEEE, 2012.

[27] S.-J. Kim, A. Magnani, and S. P. Boyd. Robust Fisher Discriminant Analysis. In *Neural Information Processing Systems*, 2005.

[28] A. Kuzmanic and V. Zanchi. Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system. In *EUROCON, 2007. The International Conference on" Computer as a Tool"*, pages 264–269. IEEE, 2007.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] H.-K. Lee and J. Kim. An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961 –973, oct 1999.

[31] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 775–781. IEEE, 2005.

[32] R. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 558–567. IEEE, 1998.

[33] K.-L. Lim and H. Galoogahi. Shape classification using local and global features. In *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*, pages 115–120, 2010.

[34] C. Liu, Y. Liu, and R. Dai. Preprocessing and statistical/structural feature extraction for handwritten numeral recognition. *Progress of Handwriting Recognition, World Scientific, Singapore*, pages 161–168, 1997.

[35] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[36] D. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.

[37] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[38] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, 2007.

[39] F. Moosmann, B. Triggs, F. Jurie, et al. Fast discriminative visual codebooks using randomized clustering forests. *Advances in Neural Information Processing Systems 19*, pages 985–992, 2007.

[40] M. Müller. *Information retrieval for music and motion*, volume 6. Springer Berlin, 2007.

[41] C. Myers and L. Habiner. A comparative study of several dynamic time-warping algorithms for connected-word. *Bell System Technical Journal*, 1981.

[42] M. Naf, O. Kubler, R. Kikinis, M. Shenton, and G. Székely. Characterization and recognition of 3d organ shape in medical image analysis using skeletonization. In *Mathematical Methods in Biomedical Image Analysis, 1996., Proceedings of the Workshop on*, pages 139–150. IEEE, 1996.

[43] N. Pugeault and R. Bowden. Spelling it out: Real-time asl fingerspelling recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1114–1119. IEEE, 2011.

[44] D. Quam. Gesture recognition with a dataglove. In *Aerospace and Electronics Conference, 1990. NAECON 1990., Proceedings of the IEEE 1990 National*, pages 755 –760 vol.2, may 1990.

[45] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–521 – II–527 vol.2, june 2003.

[46] J. Rekha, J. Bhattacharya, and S. Majumder. Shape, texture and local movement hand gesture features for indian sign language recognition. In *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pages 30 –35, dec. 2011.

[47] M. Reyes, G. Dominguez, and S. Escalera. Feature weighting in dynamic time warping for gesture recognition in depth data. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1182 –1188, nov. 2011.

[48] F. Ryden, H. J. Chizeck, S. N. Kosari, H. King, and B. Hannaford. Using kinect and a haptic interface for implementation of real-time virtual fixtures. In *Robotics Sciences and Systems, Workshop on RGB-D: Advanced Reasoning with Depth Cameras,*, Los Angeles,, June 2011.

[49] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43 – 49, feb 1978.

[50] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, pages 11–14, New York, NY, USA, 2008. ACM.

[51] T. B. Sebastian and B. B. Kimia. Curves vs. skeletons in object recognition. *Signal Processing*, 85(2):247–263, 2005.

[52] P. Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 2008.

[53] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 2, page 7, 2011.

[54] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[55] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[56] J. F. G. Srikantan and S. Srihari. Handprinted character/digit recognition using a multiple feature/resolution philosophy. In *Proc. Fourth Int'l Workshop Frontiers in Handwriting Recognition*, pages 57–66, 1994.

[57] T. Starner and A. Pentland. Real-Time American Sign Language Recognition from Video Using Hidden Markov Models. In *International Symposium on Computer Vision*, 1996.

[58] J. Stowers, M. Hayes, and A. Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 358 –362, april 2011.

[59] C. Tappert, C. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(8):787–808, 1990.

[60] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[61] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1521 – 1527, 2006.

[62] T. Wenjun, W. Chengdong, Z. Shuying, and J. Li. Dynamic hand gesture recognition using motion trajectories and key frames. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 3, pages 163 –167, march 2010.

[63] J. Whitehill and C. W. Omlin. Haar features for facs au recognition. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 5–pp. IEEE, 2006.

[64] Wikipedia. Dynamic Time Warping. `http://en.wikipedia.org/wiki/Dynamic_time_warping`, 2012. [Online;accessed 01-August-2008].

[65] A. D. Wilson. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA, 2010. ACM.

[66] A. D. Wilson and A. F. Bobick. Parametric Hidden Markov Models for Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:884–900, 1999.

[67] P. I. Wilson and J. Fernandez. Facial feature detection using haar classifiers. *J. Comput. Sci. Coll.*, 21(4):127–133, Apr. 2006.