Recognition of Human Motion and Form

BY

QINGQUAN WU
B.S., Tsinghua University, China, 2000
M.S., Tsinghua University, China, 2002

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2012

Chicago, Illinois

Defense Committee:

Jezekiel Ben-Arie, Chair and Advisor
Daniel Graupe
Michael Stroscio
Milos Zefran
Robert Sloan, Computer Science

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

FIGURE **PAGE**

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

HMM	Hidden Markov Model
HPCA	Hybrid Principal Component Analysis
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
MEMS	Microelectromechanical systems
RISq	Recognition by Indexing and Sequencing
SFS	Shape From Shading
ZVC	Zero Velocity Compensation
ZVLC	Zero Velocity Linear Compensation

SUMMARY

Recognition of human features is a very important topic in Computer Vision and Artificial Intelligence research. Researchers have examined many common features including human motion, human form (faces, hands...), human speech, etc. In our research, we focus on two types of features: human motion and form (especially faces).

In our work for human motion recognition, we employ a novel method - RISq (Recognition by Indexing and Sequencing) [1][2], which was developed by Ben-Arie for the recognition of general vector sequences. These kinds of sequences are widely found in various pattern recognition problems, e.g. Gesture Recognition [3], Speech Recognition [4], etc. Hidden Markov Model (HMM) [5] is commonly used to solve these problems. We compare RISq with HMM and demonstrate that RISq is better than HMM in many aspects. First of all, the training of RISq requires only one example from each model and is much simpler and faster than the training of HMM, which needs large training sets and a long complex training procedure. Secondly, unlike HMM, usually a few sparse samples from a test sequence are sufficient for RISq to achieve robust recognition while HMM needs the whole sequence. This makes RISq essentially much less sensitive to missing vectors in a test sequence. Lastly as our experiments demonstrate, RISq outperforms HMM when dealing with a large number of models and high vector dimensions. Also RISq exhibits better noise robustness, shorter computation time, higher selectivity ratio,

etc. For additional evaluation in real applications, we compare RISq and HMM in the task of human gesture recognition.

Gesture Recognition has recently received increasing interest from both the research community and the industry. Within the last few years, inertial sensors started to play a more and more important role in gesture recognition. Manufacturers are integrating inertial sensors into more and more products for various purposes. Among those products, two most famous ones are the Nitendo Wii and Apple iPhone. Rudimentary systems for gesture recognition have been developed recently based on those products or other lab prototypes. However, those systems usually rely only on accelerometers and the raw acceleration data used for recognition results in deteriorating performance when gestures are performed in different accelerations. In our research, we incorporate gyroscopes to overcome this drawback. We use an Inertial Measurement Unit (IMU) equipped with accelerometers and gyroscopes to sense the motion of the operator's hand. The IMU is calibrated with a Nonlinear Data-Fitting method. Gesture trajectories are reconstructed from inertial sensor measurements using the Inertial Navigation System (INS) theory. We also develop a novel method, named Zero Velocity Linear Compensation (ZVLC), to improve the trajectory reconstruction accuracy. Experimental results show that ZVLC provides more accurate reconstruction than the widely used Zero Velocity Compensation (ZVC) method. At the recognition stage, RISq is applied to recognize the trajectories

and achieves a recognition rate at 92%. HMM is also tested and only achieves 83% recognition rate.

In the third part of this research, we develop a novel method for 3D human head reconstruction and view-invariant recognition from single 2D images. We employ a deterministic Shape From Shading (SFS) method with initial conditions estimated by Hybrid Principal Component Analysis (HPCA). Our HPCA algorithm provides good initial estimates of 3D range mapping for the SFS optimization and yields much improved 3D head reconstruction. This part also describes a novel method in SFS handling of variable and unknown surface albedo. The problem of varying albedo received in the past only unsatisfactory solutions by prevalent SFS methods. In our experiments, we reconstruct 3D head range images from 2D images in different views. The 3D reconstructions are then used to recognize stored models of persons. Given a picture of a person from one direction, our SFS method enables one to identify the person by a picture taken from an entirely different direction. Empirical results show that our HPCA based SFS method provides 3D head reconstructions that notably improve the accuracy compared to other approaches. 3D reconstructions derived from images of 40 persons are tested against 80 3D head models and a recognition rate of over 90% is achieved. This part of the work already yielded two journal papers and two conference papers.

CHAPTER 1

INTRODUCTION

Recognition of human features is an important topic in Computer Vision and Artificial Intelligence research. Researchers have examined various human features including motion, form (faces, hands...), speech, etc. Our research focuses on two of them: human motion and form (especially faces). In this work, we present novel contributions in three aspects:

1. Comparison between RISq (Recognition by Indexing and Sequencing) [1][2] and HMM (Hidden Markov Model)
2. Human Gesture Recognition
3. 3D Face Reconstruction and Recognition

The gesture recognition we develop employs RISq (Recognition by Indexing and Sequencing) for recognition. RISq is a novel method invented by Ben-Arie for recognizing general vector sequences, which are commonly seen in various pattern recognition problems, e.g. recognition of gestures [3], speech [4], and human activities [6]. Researchers usually employ Hidden Markov Model (HMM) [5] to solve this kind of problems. Although HMM is quite successful in many areas including those mentioned above, it also has serious drawbacks. In this chapter, we compare RISq to HMM from both theoret-

ical and experimental perspectives. The experimental comparison includes tests of the following variables:

1. recognition rate vs. number of test samples
2. recognition rate vs. number of models
3. recognition rate vs. vector dimension
4. recognition rate vs. noise
5. missing vector test
6. computation time
7. selectivity ratio

The data used in the experiments is synthetic stochastic data. For further comparison, we will use real data derived from human gestures and compare the performance of both methods in recognizing those gestures.

Gesture Recognition is a topic that recently received increasing attention from both the research community and the industry with the introduction of inertial sensors into this area. Manufacturers are integrating inertial sensors into more and more products for various purposes, one of which is to implement gesture recognition for various functions. Among those products, two of the most famous ones are Nitendo Wii and Apple iPhone. Few gesture recognition systems have been developed with these products or with other lab prototypes. However, these applications usually rely only on the accelerometers and

the raw acceleration data for recognition, which results in deteriorating performance when gestures are performed in different acceleration. We incorporate another inertial sensor - gyroscopes - to overcome this drawback. We use an Inertial Measurement Unit (IMU) equipped with accelerometers and gyroscopes to sense the motion of the operator's hand. The IMU is calibrated with the help of Nonlinear Data-Fitting method. Gesture trajectories are reconstructed from inertial sensor measurements using the Inertial Navigation System (INS) theory. We also develop a novel method, Zero Velocity Linear Compensation (ZVLC), to improve the trajectory reconstruction accuracy. The reconstructed trajectories are recognized by the RISq method.

In the third part of this report, we propose a novel method for 3D head reconstruction and view-invariant recognition from single 2D images. We have published two journal papers [7][8] about this work and two other conference papers [9][10]. In this work, we employ a deterministic Shape From Shading (SFS) method with initial conditions estimated by Hybrid Principal Component Analysis (HPCA). Our HPCA algorithm provides good initial estimates of 3D range mapping for the SFS optimization and yields much improved 3D head reconstruction. This chapter also describes a novel method in SFS handling of variable and unknown surface albedo, a problem with unsatisfactory solutions by prevalent SFS methods. In the experiments, we reconstruct 3D head range images from 2D single images in different views. The 3D reconstructions are then used

to recognize stored models of persons. This enables one to recognize faces in wide range of views having only a single 2D picture of the person from another view.

CHAPTER 2

RECOGNITION BY INDEXING AND SEQUENCING AND THE COMPARISON TO HIDDEN MARKOV MODEL

2.1 Introduction

In this chapter, we compare the most popular sequence recognition method, HMM (Hidden Markov Model) [5], to a novel method called RISq (Recognition by Indexing and Sequencing) [1][2]. HMM is widely used in recognition of vector sequences. Such sequences have many applications in speech recognition [5] [11] [12] [13], computer vision [14] [15] [16] [17] [18], bioinfomatrics [19] [20] [21], Finance [22], etc. However, HMM has also quite a few disadvantages. Some of these disadvantages that hamper HMM are [5] [23]:

1. HMM usually needs a lot of data for training. Therefore, it is suitable only for recognition of sequences which have many training examples.
2. HMM is not suitable for tasks that require fast adaptation to different training data, e.g. adaptation to different speakers in speech recognition.
3. The training of HMM involves EM optimization and a lot of computation.

4. Every model sequence must be represented by a separate HMM. Thus, in the recognition phase, all the models need to be evaluated for each input sequence. This slows down the recognition process, especially if there are many models.

5. HMM requires the user to define the beginning and the end of each test sequence to be recognized. This is not suitable for many applications such as ours where the test sequence is part of a continuous stream of signals.

As a comparison, the RISq (Recognition by Indexing and Sequencing) method proposed by Ben-Arie in [1][2] is free of these unfavorable drawbacks. RISq is a general-purpose method for classification of vector sequences. The author applied RISq for Human Activity Recognition in [1]. In RISq, the model vector sequences are saved in a hash table. Each vector in a model is stored into a bin, of which multidimensional address is equal to the vector value. This bin stores the information on the model's ID and the frame timing of the vector in the model's sequence. At the recognition stage, vectors in the test sequence are used as indices to index into the hash table. Votes are calculated based upon multidimensional Mahalanobis distances between the test vector and the models' vectors. Each test vector yields a vote vector for every model in the database. The overall vote for each model is obtained by finding the optimal matching between the test sequence and each of the model sequences using dynamic programming with sequencing constraints. The model with the highest vote is recognized as the winning model.

RISq is substantially different from HMM and presents several advantages over HMM. First, A robust recognition can be achieved with RISq when training is performed with only one example sequence per class. On the contrary, HMM might need tens of sequences or even more. Second, the training process for RISq is as simple as converting vectors of the model sequences into entries in a hash table. The computation involved is much smaller than that in training a HMM model. Another advantage of RISq is the tremendous flexibility it provides in sampling sequences. There are no strict requirements either on the number of sampled vectors or on the vector intervals of the test sequence. Only a set of sparsely sampled representative observations are necessary for recognition. The organization of the model database also results in tremendous reduction of recognition time, since all the models are stored in the same hash table and examined at one voting action in parallel. Lastly, RISq also outperforms HMM in terms of recognition rate, computation time, robustness, etc as later shown in this chapter.

The rest of this chapter is arranged as follows: in Section 2.2, we describe some fundamentals about HMM; in Section 2.3, we describe the theory behind RISq; Section 2.4 presents the experimental comparison results between RISq and HMM.

2.2 Hidden Markov Model

This section briefly describes HMM. Figure 1 shows a HMM example [24]. Basically a HMM consists of states and observations. States are hidden but observations can be perceived and measured. A state either transits to another state or stays on the same

state. Meanwhile the observation might also change as a reflection of the state change. The most important concepts for HMM are defined as follows:

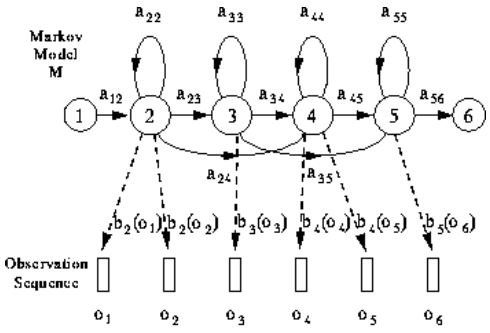

Figure 1. **A HMM example**

1. State q_t at time instant t. $q_t = S_i$ where $1 \leq i \leq N$ and N is the number of possible states of a HMM.
2. Observation O_t at time instant t. $O_t = v_j$ where $1 \leq j \leq M$ and M is the number of possible observation symbols of a HMM.
3. The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P(q_1 = S_i), 1 \leq i \leq N \tag{2.1}$$

π_i is the probability of the first state q_1 being S_i.

4. The state transition probability $A = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \tag{2.2}$$

a_{ij} is the probability of the state changing from S_i to S_j.

5. The observation symbol probability distribution at state S_i, $B = \{b_i(k)\}$ where

$$b_i(k) = P(O_t = v_k | q_t = S_i), 1 \leq i \leq N, 1 \leq k \leq M \tag{2.3}$$

i.e. $b_i(k)$ is the probability of observing v_k when the state is S_i.

A HMM λ can be represented by the parameters defined above, i.e.

$$\lambda = \{A, B, \pi\} \tag{2.4}$$

Given an observation sequence $O = O_1, O_2, ..., O_t, ...O_T$, three basic but complicated problems need to be solved:

1. How do we calculate $P(O|\lambda)$, the probability of observing the sequence O given the model λ? Once the probabilities are calculated for every model, the sequence O is then determined as belonging to the model with the highest probability.

2. How do we calculate a HMM $\lambda = \{A, B, \pi\}$ to maximize the probability $P(O|\lambda)$? Solving this problem is essentially to train a HMM model.

3. For an observation sequence O and a HMM λ, what state sequence $Q = q_1 q_2 ... q_T$ is optimal in a meaningful sense (i.e. best "explains" the observations)?

We will only describe the procedures to solve Problem 1 and 2, which are compared to RISq later. Problem 1 is often handled using the Forward-Backward Procedure described below. A new concept - forward variable $\alpha_t(i)$ - needs to be introduced for the convenience of explaining the Forward-Backward Procedure.

$$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = S_i|\lambda) \tag{2.5}$$

$a_t(i)$ is the probability of observing the partial sequence $O_1, O_2, ..., O_t$ and having state S_i at time instant t given the HMM model λ. The Forward-Backward Procedure now can be stated as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N. \tag{2.6}$$

2. Induction:

$$a_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N \tag{2.7}$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.8}$$

The steps above solve problem 1. As for problem 2, it is harder to resolve. As a matter of fact, it is the most difficult one among these 3 problems. Problem 2 is often handled by iteratively reestimate the model parameters $\{A, B, \lambda\}$. Before we go into the details of solving problem 2, three new variables need to be defined.

1. The first variable is called the backward variable $\beta_t(i)$

$$\beta_t(i) = P(O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda) \tag{2.9}$$

   $\beta_t(i)$ denotes the probability of the partial observation sequence from time $t+1$ to T being O_{t+1} to O_T, given state S_i at time t and the model λ.

2. The second variable is

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \tag{2.10}$$

   which is the probability of the state at time t being S_i, given the observation sequence O and the model λ.

3. The third variable is

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{2.11}$$

$\xi_t(i,j)$ is the probability of the states at time t and $t+1$ being S_i and S_j respectively, given the observation sequence O and the model λ.

$\gamma_t(i)$ and $\xi_t(i,j)$ can be further expressed as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{2.12}$$

$$\begin{aligned} \xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \end{aligned} \tag{2.13}$$

Now the procedure for the reestimation of the model parameters can be stated as follows:

$$\overline{\pi}_i = \gamma_1(i) \tag{2.14}$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.15}$$

$$\overline{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)|_{s.t.O_t=v_k}}{\sum_{t=1}^{T} \gamma_t(j)} \tag{2.16}$$

where $\overline{\pi}_i, \overline{a}_{ij}, \overline{b}_j(k)$ are the reestimated HMM parameters. The iterations are stopped when the stopping criteria is reached. For further details on HMM, readers could refer to the excellent description in [5].

As we can see from above, the training of HMM and the recognition using HMM are complicated and computationally demanding. As a comparison, the training of RISq is very easy as we will show next in this chapter. The recognition using RISq is also less complicated but yield better recognition results.

2.3 Recognition by Indexing and Sequencing

In this section, we describe the theoretical foundation of RISq (Recognition by Indexing and Sequencing).

RISq could be used for the recognition of general vector sequences. Suppose for a certain pattern recognition problem, the model sequences are $\{\mathbf{Y}_j = \mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_q}; j \in [1, J]\}$ where J is the number of models in the database. Also the test sequence is $\{\mathbf{Y}_t = \mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \cdots, \mathbf{x}_{t_q}\}$. The problem of recognition can be formulated as a Maximum Likelihood Sequence Estimation (MLSE) problem. In order to solve this problem, the following two assumptions are made:

1. The random differences between the subvectors $\mathbf{x}_t$ and $\mathbf{x}_j$ can be described as multivariate zero mean Gaussian distribution.
2. These variations are conditionally independent from sample to sample.

Thus the likelihood function for the sequence $P(\mathbf{Y}_t|\mathbf{Y}_j)$ can be written as

$$\begin{aligned} P(\mathbf{Y}_t|\mathbf{Y}_j) &= P(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \cdots, \mathbf{x}_{t_q}|\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_q}) \\ &= \prod_{i=1}^{q} \frac{e^{[\frac{-1}{2}(\mathbf{x}_{t_i}-\mathbf{x}_{j_i})^T C_x^{-1}(\mathbf{x}_{t_i}-\mathbf{x}_{j_i})]}}{(2\pi)^{\frac{N}{2}}|C_x|^{\frac{1}{2}}} \end{aligned} \tag{2.17}$$

where C_x is the covariance matrix of the distribution of the training set for $\mathbf{x}_j$, N is the dimension of the vectors $\mathbf{x}_j$ (also $\mathbf{x}_t$), and q is the number of vectors in the sequence. Using the log-likelihood function we get

$$\log P(\mathbf{Y}_t|\mathbf{Y}_j) = \sum_{i=1}^{q} [\frac{-1}{2}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})^T C_x^{-1}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})] - qG \tag{2.18}$$

where G is the logarithm of the denominator in Equation 2.17 given by

$$G = \log\left[(2\pi)^{\frac{N}{2}}|C_x|^{\frac{1}{2}}\right] \tag{2.19}$$

The most likely sequence Ω is found by the maximum likelihood approach,

$$\Omega = \arg\max_j\left(\sum_{i=1}^{q}\left[\frac{-1}{2}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})^T C_x^{-1}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})\right]\right) \tag{2.20}$$

2.3.1 Indexing, Voting and Sequencing

Finding the most likely model can now be solved by an indexing-based voting approach. In such voting, a model j will accumulate an incremental vote of

$$-\frac{1}{2}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})^T C_x^{-1}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i}) - G \tag{2.21}$$

for each test vector $\mathbf{x_{t_i}}$. This process is repeated by voting for all the vectors $\mathbf{x_{t_i}}$ in the test sequence. In our method, we even simplify this voting further by **v**oting only on a few representative test vectors which are sparsely sampled from the test sequence and robust recognition is still achieved. In the remainder of this section, we describe the three major components of RISq: indexing, voting and sequencing.

2.3.1.1 Multidimensional Indexing and Voting

In RISq, the models are saved in a hash table. The vectors in the models are quantized into multidimensional bins to form indices into the hash table. The model information

stored in the hash table contains a pair of values which denote the model number $\{j; j \in [1, J]\}$ and the time instant $\{t; t \in [0, T_j - 1]\}$ of the vector within the model sequence, where J is the number of models in the database and T_j represents the number of vectors in model j. This information is stored at the bin whose address corresponds to the vector that pertains to the model j at the particular instant t. The hash table is updated using the vectors from each model. In the hash table, every entry may include a set of different models which pertain to the same vector. This arrangement of the hash table is quite efficient for storage since it includes all the models in the same table and also enables robust recognition.

Our recognition scheme consists of two stages: The first stage involves voting for the vectors in the test sequence. The second stage calculates the final vote using dynamic programming, which is described in next section.

In the first stage, we index into the hash table. The voting scheme employs J 1D arrays $V_{jk}(t), j \in [1, J]$, where each array corresponds to a different model and to k that is the frame number of the test sequence. One may have several items in the same hash table bin that correspond to the same index, such items may correspond to different models and/or may pertain to different time instants.

In order to tolerate slight variations that may exist in sequences from the same model, it is necessary to consider also the neighboring bins of the indices derived from the vectors of test sequence. Suppose the vectors have 4 dimensions and $b_i^k = (q_1^k, q_2^k, q_3^k, q_4^k)$

denotes the quantized bin for the vector at time instant k, and let $b'_i = (q'_1, q'_2, q'_3, q'_4)$ denote a neighboring bin in the hash table. We define $f(b, c, d, e)$ as a mapping function from a bin's offset b, c, d, e to the f range $[0, -\infty)$. The mapping function is chosen to be a logarithm of a 4D Gaussian (assuming uncorrelated covariance matrix C_x) which conforms with our assumed model in Equation 2.17,

$$f(b, c, d, e) = log[e^{\frac{-1}{2}[(\frac{b-b_0}{\sigma_b})^2+(\frac{c-c_0}{\sigma_c})^2+(\frac{d-d_0}{\sigma_d})^2+(\frac{e-e_0}{\sigma_e})^2]}] \tag{2.22}$$

where $\sigma_b, \sigma_c, \sigma_d, \sigma_e$ denote the scale of the Gaussian along the respective axes, (b_0, c_0, d_0, e_0) represent the center of the function. $\sigma_b, \sigma_c, \sigma_d, \sigma_e$ are determined by experiments. In the voting process, a model j with time instant t receives a vote from the test vector b_i^k according to

$$V = (f(|q_1^k - q'_1|, |q_2^k - q'_2|, |q_3^k - q'_3|, |q_4^k - q'_4|)) \tag{2.23}$$

This voting mechanism is illustrated in Figure 2. As shown, there could be several items in the same hash table bin (same index). These items may pertain to different models and/or different time instants. Each vector in the test sequence yields a vote vector for each model in the database. This vote vector is a temporal depiction of the log-likelihood that the indexed vector belongs to the model. After the voting, the individual vote vectors of the test vectors have to be optimally combined with sequencing constraints.

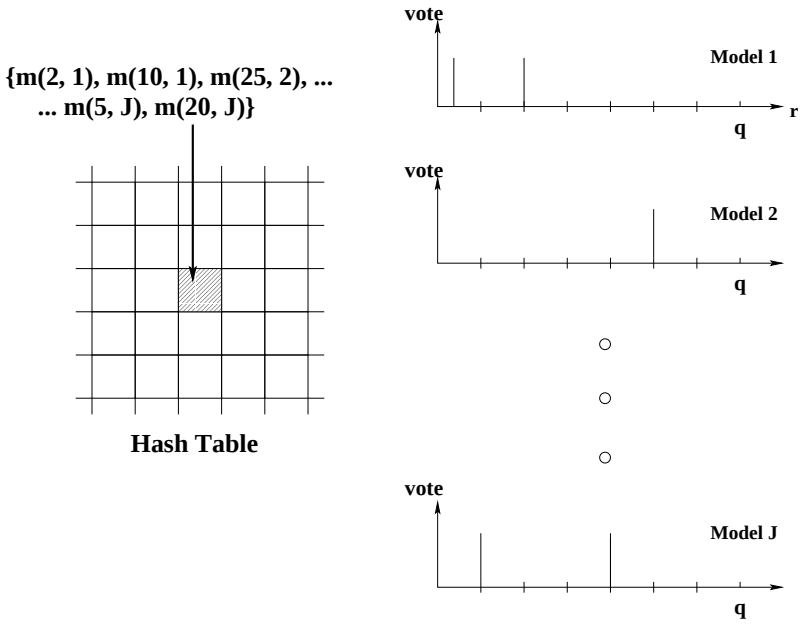

Figure 2. **A voting example. The current vector indexes to the center square, which has been indexed by certain models at certain specific time instants. The resulting votes are described in the right side of the figure.**

2.3.1.2 Optimal Sequencing Based on Dynamic Programming

The process of indexing the test vectors results in J (Total number of models) vote vectors for each of the p sample vectors in the input sequence $\mathbf{Y}_t$. These vote vectors have to be sequenced optimally to obtain the maximum vote of the input sequence for each model and the input sequence is recognized as the model receiving the highest vote. In this section, we describe the optimal technique based on dynamic programming to integrate the vote vectors of the input sequence, subject to the sequencing constraints.

For convenience, vectors in a test sequence $\mathbf{Y}_t$ is represented as

$$\mathbf{t}(n)\ ; n \in [1, p] \tag{2.24}$$

where p is the number of vectors in $\mathbf{Y}_t$. In order to find the vote of the input sequence for a particular model, we consider the vote vectors of all the test vectors corresponding to that model. Each entry in the vote vector is score of match between the test vector $\mathbf{t}(n)$ and $\mathbf{m}(r, j)\,\{r \in [1, q]\}$ referred to as vector pair matching score $\{\mathbf{t}(n), \mathbf{m}(r, j)\}$. Now, the problem of finding the vote for each model can be phrased as finding the sum of a sequence of vector pair matching scores $\ldots \{\mathbf{t}(n), \mathbf{m}(r, j)\} \ldots \{\mathbf{t}(a), \mathbf{m}(b, j)\} \ldots$ such that $(n \neq a)$ and if $(n < a)$ then $(r \leq b)$ or if $(n > a)$ then $(r \geq b)$

To find such a sequence of matching scores between vector pairs, we associate a cell comprising of four variables, as illustrated in Figure 3, to each matching vector pair of vote vectors in consideration. The description of the various variables are as follows,

$N = n$ stores the location of the test vector $\mathbf{t}(n)$ in the input sequence $\mathbf{Y}_t$, $R = r$ stores the location of the matching model vector $\mathbf{m}(r, j)$ in the model sequence Y_j, $W(nrj)$ stores the matching score of the test vector and model vector that are related to the cell, $O(nrj)$ is the sum of partial sequences of matching vector pair scores, indicating the highest sum of vector matching scores of a partial vector pair sequence that ends with the cell related matching vector pair.

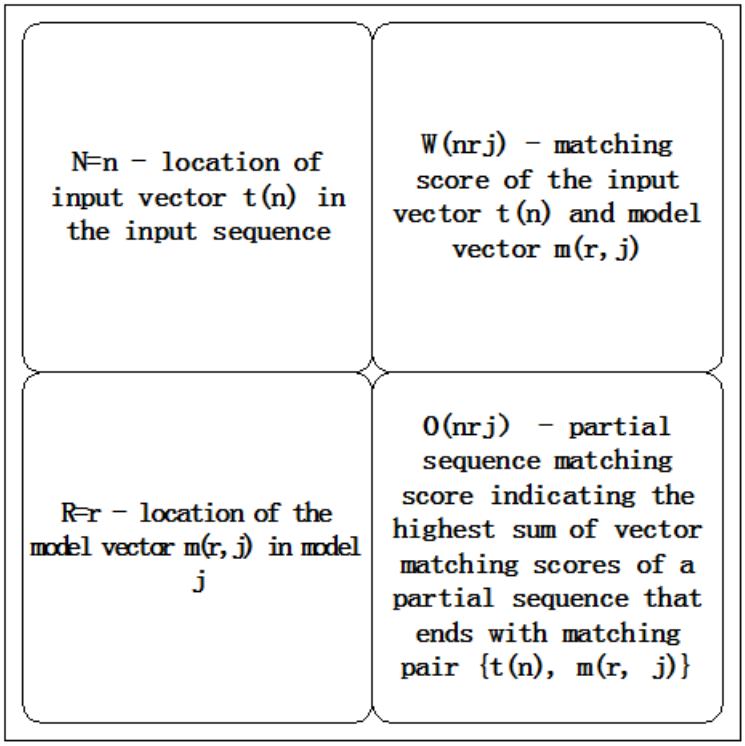

Figure 3. **Structure of cell corresponding to each matching vector pair**

Cells of matching vector pairs corresponding to a certain model are constructed into an array of cells $A\{j\}$. $A\{j\}$ holds all the required information for obtaining the vote of the input sequence for a particular model Y_j. This construction procedure has to be repeated for each model sequence Y_j $\{j; j \in [1, J]\}$.

As already mentioned, our sequencing technique is based on the principle of Dynamic Programming (DP), which states that any partial sequence of matching vector pairs with an optimal score includes only sub-partial sequences that are also optimal. It means that any pair that precedes the last pair of any optimal-partial sequence of vector pairs, is also an ending of an optimal-partial sequence of vector pairs. It implies that a DP-based algorithm has to gradually build partial sequences that each ends with a terminal vector pair by searching backwards and finding the preceding pair that contributes the most to the aggregate sum of matching scores that pertains to that terminal vector pair. Since the algorithm has to find for each pair only one preceding pair that maximizes the aggregate score, DP significantly reduces the computations required. The sequencing requirements further reduce the search space.

Figure 4 illustrates the various steps involved in this process. The basic idea in the approach is to process each cell of the array $A\{j\}$ and find the highest sum of pair-matching scores of a partial sequence that ends with matching pair $\{\mathbf{t}(n), \mathbf{m}(r, j)\}$ that pertains to that particular cell. A variable **p**ointer1 holds the information of the current cell that is being processed and is initialized to the address of the first cell of array

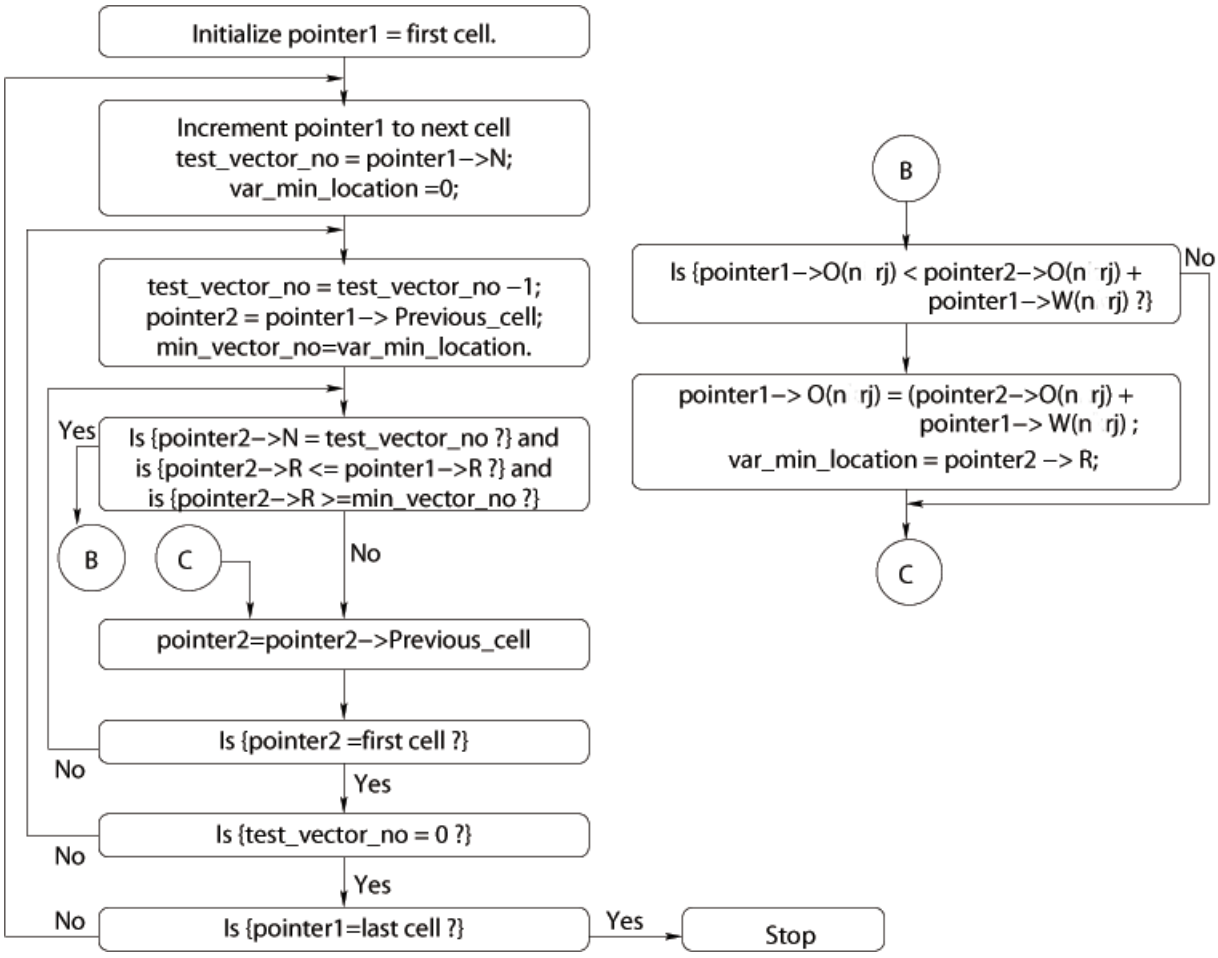

Figure 4. **Flow diagram depicting the dynamic programming based sequencing technique**

$A\{j\}$ and is subsequently incremented to process subsequent cells. Thus **p**ointer1 holds the current cell whose partial vector sequence is being optimized. The procedure is repeated until all the cells have been considered. In order to find the sum of matching scores of a partial sequence that ends with matching pair $\{\mathbf{t}(n), \mathbf{m}(r, j)\}$, cells with test vector locations less than n and model vector location less than or equal to r have to be considered for constituting the preceding matching vector pair of the optimal partial sequence. During the process of consideration, cells generated due to matching vector pairs of $\mathbf{t}(n-1)$ are considered before pairs of $\mathbf{t}(n-2)$ and so on until the null vector $\mathbf{t}(0)$ is reached. While considering vector pairs with the test vector $\mathbf{t}(n-1)$, only vector pairs that pertain to cells with $\mathbf{R}$ greater than 0 as lower bound and $\mathbf{R}$ less than r as the higher bound are considered. But for pairs with subsequent test vectors the range could be even narrower with the lower bound tending towards r. This saves considerable computations as the number of cells considered in finding highest sum of matching scores of partial sequence that ends with matching pair $\{\mathbf{t}(n), \mathbf{m}(r, j)\}$. Eligible vector pairs are checked for giving a higher sum of scores of partial sequence of matching vector pairs if they would be the preceding matching vector pair of the optimal sequence ending with matching pair $\{\mathbf{t}(n), \mathbf{m}(r, j)\}$. If they yield a higher score of partial sequence of matching vector pairs, then the resulting score is updated in $O(nrj)$.

The above process is repeated for all arrays $A\{j\}\,\{j; j \in [1, J]\}$ for the entire model set. The cell among all the cells of the entire set having the highest sum of matching score

$O(nrj)$ is selected as the highest sequence matching score and the input is recognized as most similar to the winning model Y_j.

2.4 Comparison between RISq and HMM

In this section, RISq and HMM are compared in various aspects using synthetic stochastic data. The reason we use synthetic stochastic data is that it allows us to create all kinds of data needed for various comparisons. We use the HMM code developed by Daniel DeMenthon and Marc Vuilleumier [25] in our experiment.

In these experiments, we use a random number generator to generate a set of vector sequences and each one of the sequences Y_j is used as a model for RISq. As for HMM, more than one training sequences are needed for each model. To obtain the additional training sequences, we add random Gaussian noise to the sequence Y_j to generate another 39 new sequences. The SNR (Signal to Noise Ratio) is controlled at 20dB. The original sequence Y_j and the 39 new sequences are then used to train and obtain a HMM model. Similarly, test sequences for both RISq and HMM are generated by adding Gaussian noise to the sequence Y_j. The SNRs of the test sequences varies between 10dB to 20dB in our experiments.

2.4.1 Recognition Rate vs. Number of Test Samples

The first part of the comparison demonstrates an important feature of RISq, that is RISq could achieve robust recognition with only a few sparse samples. The parameters for the experiments conducted in this section are shown in Table I and further explained

as follows. 3 models are used. Each model sequence has 50 vectors and every vector has 2 dimensions. 100 test sequences are generated from each model sequence by adding Gaussian noise to the model sequence. A total of 300 test sequences are generated and then tested.

Experiment results are shown in Table II and Figure 5. As we can see, the recognition rate of RISq increases as more samples are used for test. Noticeably, 100% recognition rate is achieved when 13 or more samples (out of the total 50 vectors) are used. On the other hand, HMM uses all 50 vectors for recognition and achieves a lower recognition rate at 95.6%. The difference here is not as obvious as it will be in other comparisons detailed later in this chapter. However, readers should keep in mind that RISq uses much less samples than what HMM uses for test but still achieves higher accuracy. As later can be seen in other experiments, RISq can reach 100% recognition rate with even less test samples when the vector dimensions are increased.

TABLE I

PARAMETERS FOR TESTS IN Table II

Number of models	3
Number of vectors in a sequence	50
Vector dimension	2
SNR (dB)	20

TABLE II

RECOGNITION RATES vs. NUMBER OF TEST SAMPLES FOR RISq

Number of samples used for test	2	3	4	5	6	7	8	9	10	13	14
RISq Recognition Rate (%)	56.3	60.3	77.3	87.7	87	92	97.7	97	96	100	100

2.4.2 Recognition Rate vs. Number of Models

The second experiment compares the recognition rates of RISq and HMM when the number of models are varied between 3 and 50. Similar parameters from Table I are used except the vector dimension is increased from 2 to 5. Detailed parameters are listed in Table III. The experiment results are shown in Table IV and Figure 6. As we can see, the performance of HMM deteriorates significantly when the number of models increases. On the other hand, the performance of RISq only deteriorates slightly, especially in the case when 10 test samples are used. In that case, RISq still achieves almost 100% recognition rate with 50 models while HMM only reaches 81.6%. As can be seen, a large number of models adversely affect both methods. However, this effect could be alleviated in RISq by simply increasing the number of test samples.

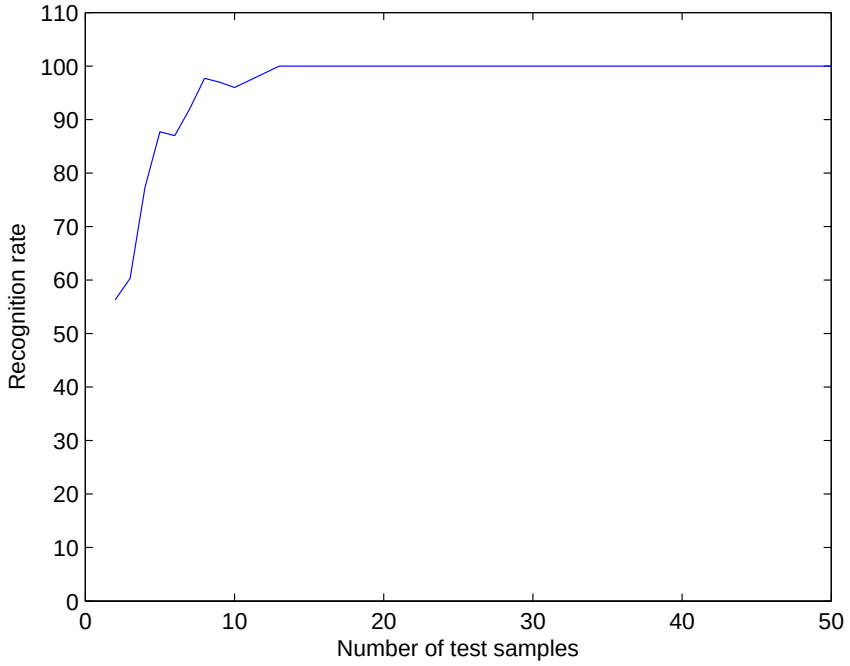

Figure 5. **RISq recognition rate vs. number of test samples**

TABLE III

PARAMETERS FOR TESTS IN Table IV, Table XI AND Table XII

Number of vectors in a sequence	50
Vector dimension	5
Number of training sequences (50 vectors each) for HMM	40
Number of test sequences (50 vectors each) for each model	100
SNR (dB)	20

TABLE IV

RECOGNITION RATES vs. NUMBER OF MODELS

Number of models	3	10	20	50
RISq (uses 6 test samples)	100	97.7	97.6	94.0
RISq (uses 10 test samples)	100	100	99.9	99.8
HMM (uses all 50 test samples)	97.7	97.4	91.9	81.6

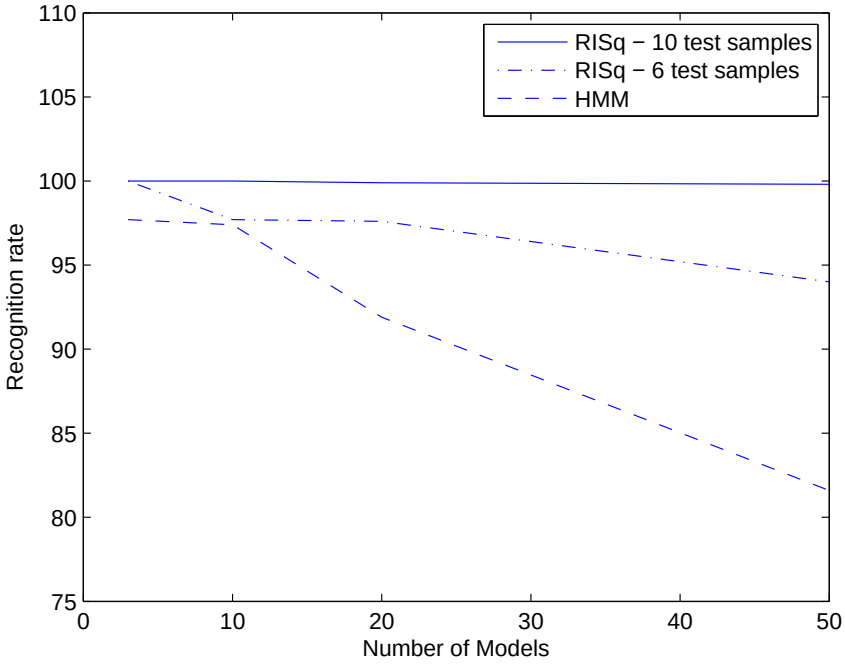

Figure 6. **RISq/HMM recognition rate vs. number of models**

2.4.3 Recognition Rate vs. Vector Dimension

In the experiments described in this section, the vector dimension is varied and the recognition rates of RISq and HMM are compared. Parameters for the experiments are shown in Table V. Results are shown in Table VI and Figure 7. As shown, RISq always achieves 100% recognition rate when 13 out of 50 vectors are used for test no matter the vector dimension is 2, 5 or 15. When 6 samples are used, the recognition rate of RISq also reaches 100% when the vectors have 5 or 15 dimensions. As for HMM, the recognition rate also improves as the vector dimension increases. However, HMM never achieves 100% recognition rate. RISq outperforms HMM in this comparison as well.

TABLE V

PARAMETERS FOR VECTOR DIMENSION TESTS IN Table VI

Number of models	3
Number of vectors in a sequence	50
Number of training sequences (50 vectors each) for HMM	40
Number of test sequences (50 vectors each) for each model	100
SNR (dB)	20

TABLE VI

RECOGNITION RATES vs. VECTOR DIMENSION

Vector Dimension	2	5	15
RISq (use 6 test samples)	87	100	100
RISq (use 13 test samples)	100	100	100
HMM (use all 50 test samples)	95.7	97.7	97.0

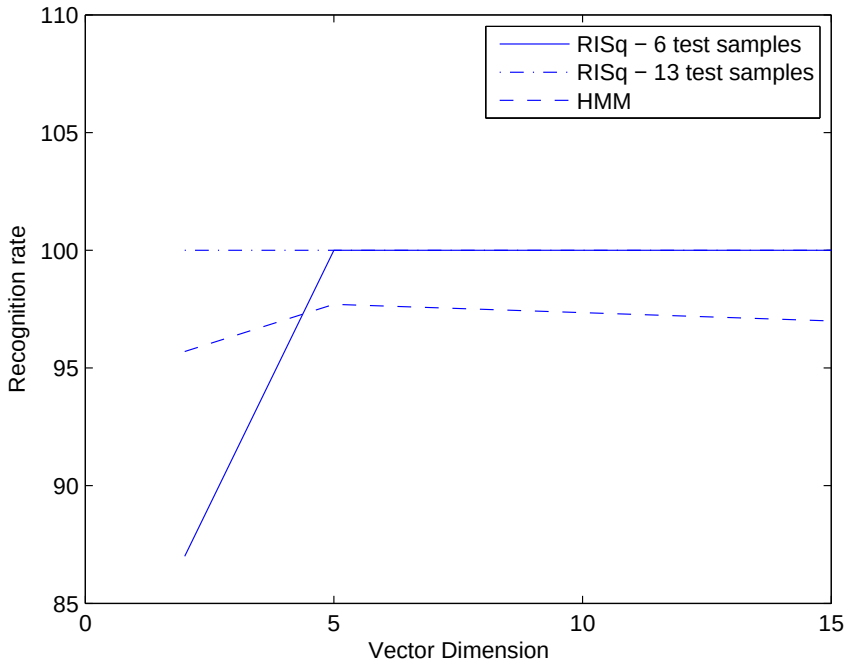

Figure 7. **RISq/HMM recognition rate vs. vector dimension**

2.4.4 Recognition Rate vs. Noise

In this section, the recognition rates of RISq and HMM are compared when different levels of noise are present. Experiment parameters are shown in Table VII. The results are shown in Table VIII and Figure 8. As can be observed, RISqs with different numbers of test samples all outperform HMMs. Even in a very noisy scenario when SNR=12dB, RISq still achieves a recognition rate of 88% when using all 50 available vectors. On the other hand, the recognition rate of HMM is only 32.3%. These results indicate that RISq is much more robust to the noise than HMM.

TABLE VII

PARAMETERS FOR TESTS IN Table VIII

Number of models	3
Vector dimension	5
Number of vectors in a sequence	50
Number of training sequences (50 vectors each) for HMM	40
Number of test sequences (50 vectors each) from each model	100

2.4.5 Missing Vector Test

In this section, we test the performance of RISq and HMM when certain vectors are missing. Experiment parameters are listed in Table IX. RISq either uses 6 samples for

TABLE VIII

RECOGNITION RATES vs. SNR

SNR (dB)	20	15	12	10
RISq (use 6 test samples)	100	64.7	48.3	36.3
RISq (use 13 test samples)	100	89.3	53.3	37.0
RISq (use 25 test samples)	100	98.67	66.7	40.7
RISq (use 50 test samples)	100	100	88.0	45.7
HMM (use all 50 test samples)	97.7	47	32.3	26.3

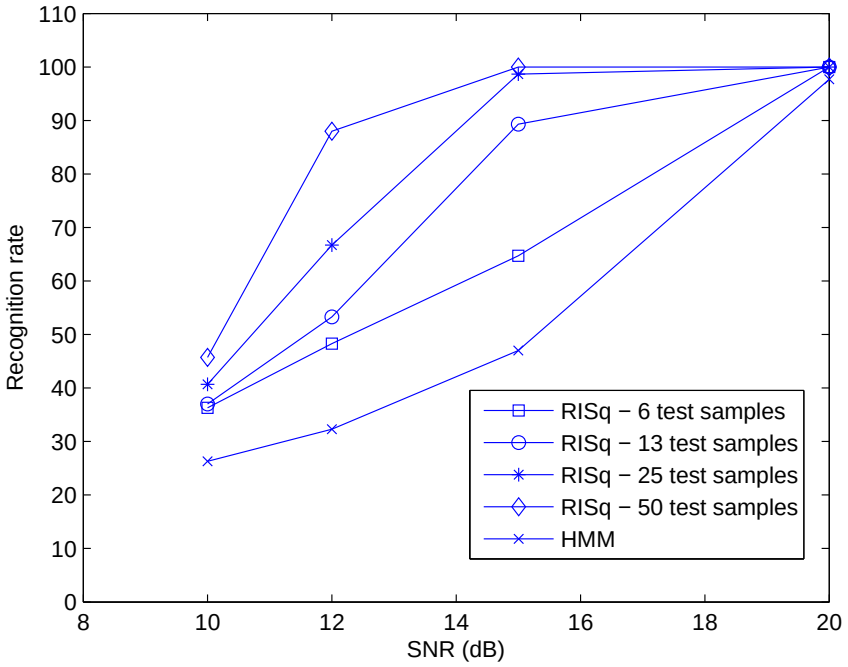

Figure 8. **RISq/HMM recognition rate vs. SNR**

test (when there're at least 6 remaining vectors) or all available remaining vectors (when there're less than 6 vectors remaining). The results are shown in Table X and Figure 9.

As we can observe, the recognition rate of RISq stays at 100% until 44 out of the 50 vectors are missing. When more vectors are missing, the performance of RISq starts to deteriorate slightly.

With regards to HMM, the recognition rate drops to unacceptable levels when more than 44 out of the 50 vectors are missing. RISq also surpasses HMM in this aspect.

TABLE IX

PARAMETERS FOR TESTS IN Table X

Number of models	3
Number of vectors in a sequence	50
Vector dimension	5
Number of training sequences (50 vectors each) for HMM	40
Number of test sequences (50 vectors each) from each model	100
Number of test samples for RISq	Minimum(6, number of remaining vectors)
SNR (dB)	20

TABLE X

RECOGNITION RATES vs. NUMBER OF MISSING VECTORS

Number of missing vector	0	5	16	25	44	45	46
RISq	100	100	100	100	100	99.3	98.9
HMM	97.7	95.7	90	88.7	74.7	54.4	72.7

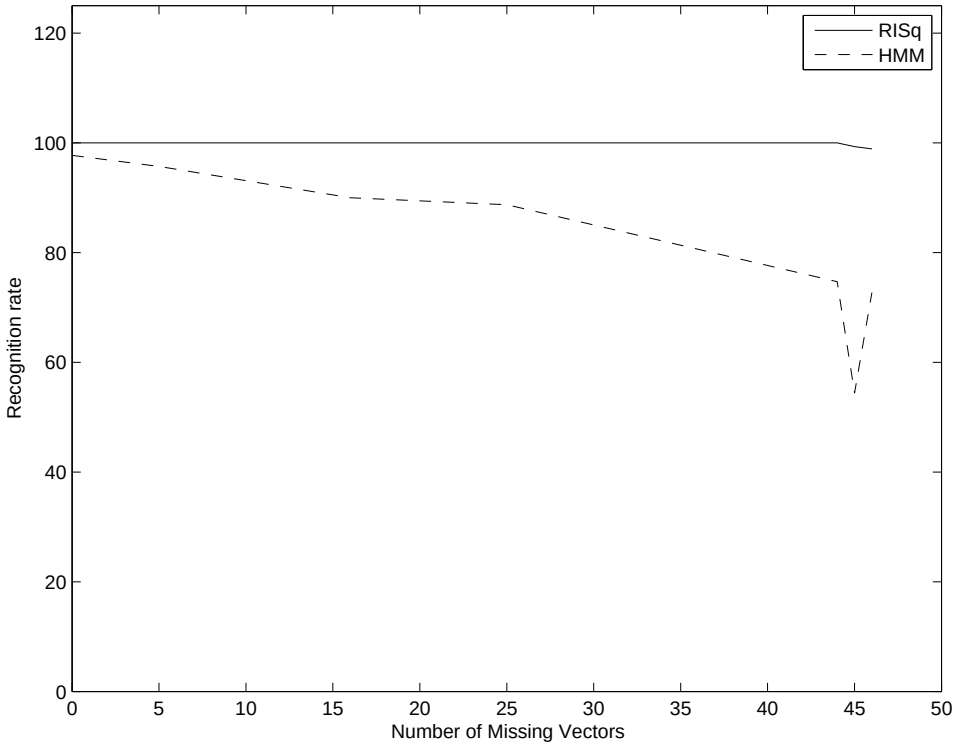

Figure 9. **RISq/HMM recognition rate vs. number of missing vectors**

2.4.6 Computation Time

We also compare the CPU time for recognition using RISq and HMM. The experiment parameters in Table III are used. The average CPU time for recognizing a sequence is shown in Table XI. From the results, we can see that RISq is also much more efficient and faster than HMM in terms of computation time.

TABLE XI

COMPUTATION TIME

Number of models	20	50
RISq (ms)	22.8	73.2
HMM (ms)	82.0	301.1

2.4.7 Selectivity Ratio

In the last comparison, we compare the Selectivity Ratios of both methods. The Selectivity Ratio of RISq is defined as the ratio between the highest matching score and the second highest matching score. For HMM, it's defined as the ratio between the highest probability and the second highest probability. In the experiments, 3 models are used and the other parameters are identical to those in Table III. The results are shown in Table XII. As can be seen from the table, the Selectivity Ratio of RISq are about

100 times higher than that of HMM when SNR is 20dB. When the SNR turns 15dB, the Selectivity Ratio of RISq drops to 4.69 but still remains higher than that of HMM. In a word, RISq is better than HMM in terms of Selectivity Ratio as well.

TABLE XII

SELECTIVITY RATIO TEST RESULTS

SNR (dB)	20	15
RISq	319.8	4.69
HMM	3.1	1.62

2.4.8 Comparison Summary

From the description and comparison above, we could see that RISq is superior to HMM in the following aspects:

1. It is much easier to train RISq models than to train HMM models. RISq requires only one example from each class for training and the training is as simple as indexing into a hash table and saving vector attributes into corresponding bins.
2. RISq only needs a few sparse samples to achieve robust recognition. That allows RISq to maintain high recognition rate even when most of the vectors are missing. HMM performs much worse under this kind of condition.
3. HMM's recognition rate degrades notably as the number of models increases. RISq is more immune to this change. When the number of models is high, RISq achieves much higher recognition rate than HMM.
4. When enough test samples are used, RISq always achieves higher recognition rate than HMM regardless of the vector dimension.
5. RISq is much more robust to noise than HMM.
6. RISq is more efficient and selective than HMM.

CHAPTER 3

GESTURE RECOGNITION

In the following few chapters, we present our work on Gesture Recognition. The goal of our work is to recognize hand gestures using an inertial system. We develop here a novel method, called Zero Velocity Linear Compensation (ZVLC), for improving gesture trajectory reconstruction accuracy. We also employ the novel RISq method described in last chapter to achieve robust recognition of the reconstructed gestures.

Gesture recognition has been an important topic in the Human Machine Interface (HMI) research since it has applications in many areas, such as communication with the deaf, communication with robot assistants [26][27], control of machines (home electronics, appliances [28], handheld devices [29], etc), human vehicle interaction [30], etc. Many solutions have been proposed for gesture recognition. The most popular ones are methods based on Computer Vision, or on Magnetic Sensing or on inertial sensors.

3.1 Computer Vision based Approaches

Computer Vision contributes the largest number of papers to gesture recognition. Among the methods used in those papers, HMM is the most popular one. Lee and Kim [31] proposed an HMM model to classify the gestures by a combination of all the states from all trained gesture models. However, this method doesn't work well with complex background. Yoon et al. [32] developed a HMM classifier to recognize gestures depict-

ing alphanumeric characters and graphic elements (circle, triangle, etc). The program localized and tracked the hand in the incoming video and used features such as hand location, angle and velocity. Bauer and Karl-Friedrich [33] split the signs in the American Sign Language into subunits and employed HMM to model the subunits other than the signs directly. Markers and colored gloves were used. The feature vector for HMM included information of hand locations, distance, marker distances, colored glove area size, etc. Ramamoorthy et al. integrated static shape recognition, Kalman-filter-based hand tracking, and HMM to recognize single-handed hand gestures in [34]. Kelly et al. [35] also employed HMM to recognize 8 gestures from American Sign Language (ASL). Features used included hand position relative to eyes, hand direction, hands distance, etc. Liu et al. [36] studied the effects of different HMM model structures and training algorithms on a gesture recognition system, which recognized the letters from A to Z. Elmezain et al. [37][38] developed a gesture recognition system for Arabic numbers (0-9) in stereo color image sequences using HMM. Orientations of gesture trajectories were quantized and used as features for HMM. The same authors employed a different method - Conditional Random Fields - for the recognition in [39]. Other works that employed HMM includes [40] and [41].

In addition to HMM, researchers also investigated other methods to recognize gestures from videos. Holte and Moeslund [42] used both a range camera and a regular camera to recognize gestures. Motion primitives were identified from the images and a

probabilistic Edit Distance classifier was employed to identify 4 gestures. Rajko et al. [43] applied semantic network model (SNM), a generalization of HMM to recognize 6 mouse/pen gestures. The unit vector in the direction of mouse/pen movement was used in the recognition. Wong and Cipolla [44] proposed a method to recognize gestures by first converting a portion of a video into a motion gradient orientation image and then classifying it into one of the 9 gestures by a sparse Bayesian classifier. Bhuyan et al. [45] proposed for gesture recognition a 10 dimensional feature vector, which consists of gesture trajectory length, hand orientations, maximum/minimum velocities, etc. Suk et al. [46] employed Dynamic Bayesian Network to recognize 10 gestures. Features used included hand motion direction code, hand-hand positional relation, hand-face positional relation, etc. There are some other methods that have been examined for gesture recognition. These methods include Finite-State Machine (FSM) [47][48][49], Condensation Algorithm [50], Hidden Conditional Random Fields [51], Latent-Dynamic Discriminative Models [52], Neural Network [53], SVM [53], Continuous Dynamic Programming [54], Volume Motion Template [55], etc.

3.2 Magnetic Sensing based Approaches

As mentioned earlier, magnetic sensing based methods are also commonly seen for gesture capture and recognition. Magnetic sensors has been used in motion capture for a long time [56][57][58]. Ma et al. developed a magnetic hand motion tracking system in [59]. Permanent magnets were fixed on nails and tracked by magnetic sensors installed

on a wristband. A hand geometric model was employed to compute the hand postures. Kevin et al. [60] developed a gesture recognition system using a pair of CyberGloves and three magnetic trackers. The CyberGloves measured 18 hand-joint angles and the magnetic trackers determined 3-D hand positions. The system utilized the condensation algorithm and used 48-D feature vectors to classify directional movement and static hand shape gestures. Corradini and Cohen [61] developed a gesture recognition system using a 6 DOF tracker device based on magnetic fields and a Recurrent artificial neural networks (RNNs) as the recognizer. Pirkl et al. [62] used a pair of sender and receiver based on magnetic resonant coupling to study Tai Chi moves. The magnetic receiver and the sender were placed on the chest and the arm of a tester respectively. The system calculated variables indicating sender to receiver distance, receiver rotation angles, etc and passed the variables to a recognition program based on Decision Tree. Bobick and Wilson described a few gesture recognition experiments in [63] and one of those used a magnetic spatial position and orientation sensor. Dynamic programming was employed for recognition.

3.3 Inertial Sensor based Approaches

Despite the popularity of the methods based on Computer Vision and Magnetic Sensing, these methods have few drawbacks. Visual approaches suffer from occlusion, varied illumination and cluttered background. The performance of this kind of methods largely depends on the performance of the preprocessing stage, which segments, localizes, and

tracks the hands and arms (sometimes even face and eyes are necessary). In some cases, the gesture performer has to wear colored gloves and short-sleeve shirt with simple texture. Furthermore, the gesture performer has to stay within the view of the camera and most of the systems are view dependent. As for magnetic sensing, any electric motors or just ferromagnetic and metallic objects close to the sensors will affect the electromagnetic fields and the measurements. This renders the magnetic tracking useless in cockpits, cars and industrial environments. Also the accuracy diminishes as the gesture performer moves away from the emitter.

As a comparison, the methods using inertial sensors are not affected by the surroundings, which are more suitable than the Computer Vision-based recognition in complex, cluttered environments. Moreover, inertial measurements are not affected by external signal jamming or magnetic fields. Inertial measurement units can be used as totally self-contained input devices [64]. These advantages have shifted more and more attention towards the inertial sensing approach. Inertial sensors are now being used in electronic products such as Nitendo Wii, Apple iPhone, BlackBerry phones, etc. The manufacturers rely on the inertial sensors and employ simple motion recognition techniques to achieve effective interaction between the products and the users.

Even before the introduction of integrated inertial sensors by the consumer electronics industry, researchers have been using these powerful devices. For example, Sakaguchi et al. used accelerometers and gyroscopes to calculate the angular velocity and the

angular displacement of a human arm model [65]. However, the authors did not examine their method on real human arms and non-circular motion. Heinz et al. discussed the possibility of using accelerometer and gyroscope data to automatically analyze Wing Tsun movements in [66]. Features calculated from sensor data and derivatives were studied and discussed. However, gesture trajectory and recognition were not investigated. Benbasat et al. [67] looked at the gesture recognition problem from a different angle and found a set of atomic gestures according to the acceleration curve characteristics. The gestures were recognized after their atomic gestures were identified. Sama et al. proposed to use a glove equipped with accelerometers, gyroscopes and bend sensors to control the movement of an animated glove [68]. However, the translation and rotation of the glove has not yet been implemented when the paper was published. Reifinger et al. developed a system to control the translation and rotation of objects in augmented reality with the help of a 2-axis accelerometer and a single axis gyroscope [69]. Pandit et al. also used a pair of gloves with built-in accelerometers to convert hand gestures into computer commands [70], e.g. copy, paste, scroll, etc. Bang et al. [71] developed a pen-shape input device for capturing characters written in the air by the operator. The device relied on both accelerometers and gyroscopes to reconstruct the pen trajectory. Gabayan et al. also developed a gesture recognition system to recognize arabic numbers drawn by an operator holding an Inertial Measurement Unit (IMU) [3] with a 3-axis accelerometer and a 2-axis gyroscope. Zhang et al. [72] developed a system to control the operation

of handheld devices by recognizing gestures like swaying and drawing Arabic numbers. Ouchi et al. [28] at Toshiba developed the MagicWand based on accelerometers to control a few home appliances by recognizing simple gestures like circle, arrow, and so on. In [73][74], authors use the acceleration and trajectory direction as features and employed HMM to recognize gestures corresponding to graphical elements (square, circle, etc) or directional movements. In [75], Dong et al. performed Discrete Cosine Transform (DCT) on the acceleration data to extract features for HMM. Gestures depicting number 0-4 were examined. In addition to the methods described above, other techniques including Dynimic Time Warping (DTW) [76] and Maximum Entropy Models [77] were investigated by researchers as well.

As mentioned earlier in this chapter, in last few years inertial sensors started to emerge in the consumer electronics products. As can be seen from Figure 10, the application of MEMS (Micro-electro-mechanical systems, including inertial sensors) sensors in consumer and mobile electronics has been growing steadily in last few years and expects faster growing in the coming years [78]. Among these applications, two of the most famous ones are the Nitendo Wii gaming system and the Apple iPhone, which employs an accelerometer to detect the device's orientation. Researchers used the inertial sensors on these products and developed several gesture recognition systems. Liu et al. developed such a system based on the Wii remote [79]. In their experiments, an operator held a Wii remote and performed different gestures. Acceleration data from the 3-axis accelerometer

on the Wii remote were transmitted to a PC via Bluetooth connection. Dynamic Time Warping (DTW) was employed to recognize the gestures. Other researchers also developed gesture recognition systems based on the Wii remote. The methods used include HMM [80][81][82][83][28], DTW [84], SVM [28] and K-Nearest Neighbors [85]. Besides the Wii remote, other handheld devices with built-in inertial sensors were also investigated. Jang and Park [29] developed on a PDA a gesture recognition program based on state machine. Different gestures were used to control the sound or scrolling, or to switch between application windows. Agrawal et al. [86] planned to develop the Nokia N95 cell phone into a gesture performing tool. The user held the cell phone and wrote in the air. The authors planned to reconstruct what the user writes using data from the built-in 3-axis accelerometer. The recognition was not completed when the paper was published. Kauppila et al. also proposed a gesture recognition system using a series 60 cell phone [87]. He et al. [88] processed the acceleration data from a cell phone with Wavelet packet decomposition and FFT. SVM was employed to recognize the cell phone user's gestures in order to control the cell phone operations. In [89], simple gestures like writing Arabic numbers were classified using data from accelerometers in a cell phone to operate the phone, e.g. speed dialing, navigation, delete message, etc. These gesture recognition systems based on the Wii remote and the cell phones use only accelerometers, which limits the performance in terms of speed-invariance, pose-invariance, etc. As well known, accelerometer reading varies under different acceleration/deceleration. Therefore,

the recognition deteriorates significantly when gestures are performed in different acceleration/deceleration. Moreover, accelerometer reading also varies from pose to pose due to the influence of the earth gravity. Depending on the accelerometer pose, the gravity has different components on the axes of a 3-axis accelerometer even if the accelerometer stays stationary. So the recognition rate of the systems above also depends greatly on the way the user is holding the device.

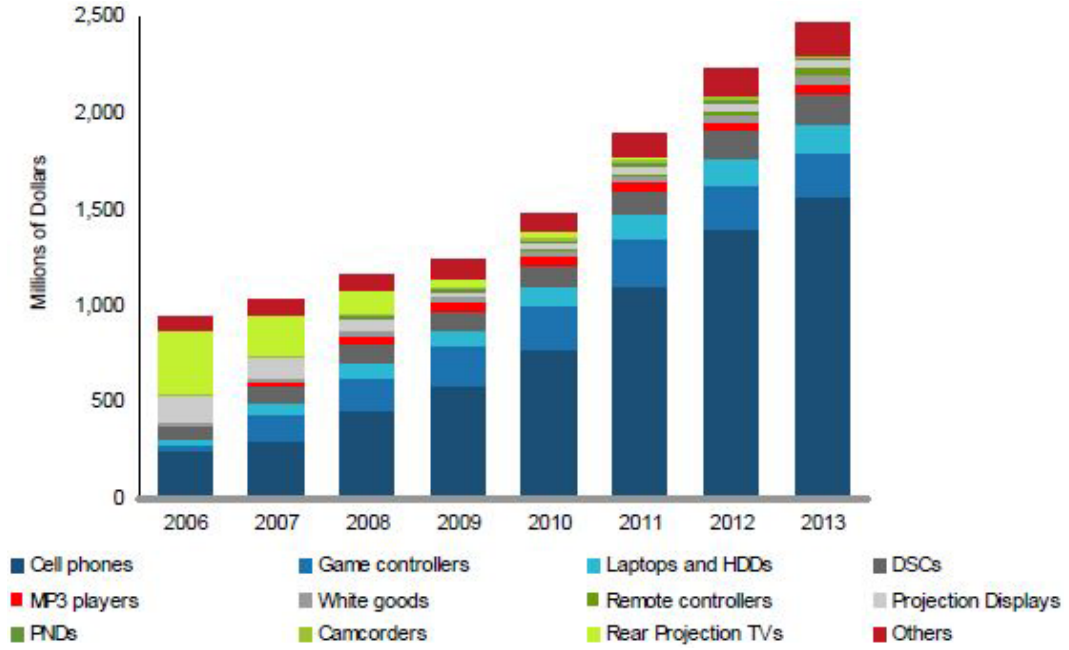

Figure 10. **Consumer and mobile MEMS market by application. (Source: iSuppli)**

3.4 Our Approach

In order to overcome these drawbacks, we propose to reconstruct the trajectory of the handheld device and recognize the gesture by recognizing its trajectory. We use both accelerometers and gyroscopes, which allows us to reconstruct the trajectory more accurately than using only accelerometers. We also propose a method we developed recently, called Zero Velocity Linear Compensation (ZVLC), for improving trajectory reconstruction accuracy. Another major difference of our work from other works is that we employ for the recognition the novel method that we described in previous chapter - RISq (Recognition by Indexing and Sequencing) [1].

In the following chapters, we present our work on Gesture Recognition using an Inertial Measurement Unit (IMU). In Chapter 4, we describe the Inertial Measurement Unit (IMU) and the calibration process. In Chapter 5, we present the methods we developed for trajectory reconstruction. In Chapter 6, we describe our RISq-based gesture recognition method and present the results.

CHAPTER 4

INERTIAL MEASUREMENT UNIT FOR GESTURE RECONSTRUCTION

In this chapter, we briefly describe the IMU we use in our work and the calibration process based on Nonlinear Data-Fitting. An inertial measurement unit, or IMU, is an electronic device that calculates a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes [90]. IMUs are typically used to maneuver aircraft, including UAVs, among many others, and spacecraft, including shuttles, satellites and landers.

4.1 CHR-6d Inertial Measurement Unit

The IMU that we use is the low-cost entry-level CHR-6d from CH Robotics [91]. The CHR-6d, shown in Figure 11, is a complete 6-axis Inertial Measurement Unit (IMU), with 3 axes of angular acceleration measurements, and 3 axes of acceleration measurements. The specifications of the onboard inertial sensors are listed in Table XIII.

The CHR-6d includes a ARM Cortex processor running at 64M Hz. All sensor channels are oversampled and decimated to provide 16-bit resolution. A Parks-McClellan window low-pass FIR filter is then applied to remove additional noise. The corner frequency of the filter can be adjusted in 10 Hz increments from 10 Hz to 140 Hz for each channel independently. The number of taps used in each filter can also be adjusted from

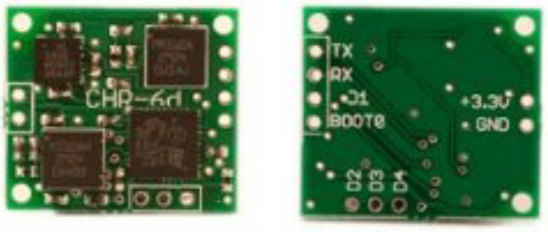

Figure 11. **CHR-6d Inertial Measurement Unit**

8, 13, 32, and 64 taps. The filter can also be disabled if preferred. Communication is performed over a simple TTL UART interface at 115200 Baud [92].

TABLE XIII

CHR-6d ONBOARD SENSOR SPECIFICATIONS

Sensor	Part Number	Measurement Range
Accelerometer	ADXL335	$\pm 3g$
Pitch/roll rate gyroscope	LPR510AL	$\pm 100 deg/s$
Yaw rate gyroscope	LY510ALH	$\pm 100 deg/s$

4.2 Nonlinear Data-Fitting based Calibration

The 16-bit measurements from the sensors are unsigned positive numbers. As shown in Table XIII, the accelerometer measures up to $\pm 3g$ and the gyroscopes measure up

to ± 100 deg/s. In order to convert the positive 16-bit numbers into positive/nagative acceleration or angular acceleration, biases have to be subtracted and scaling factors have to be multiplied. The biases and scaling factors recommended by the IMU manufacturer usually need to be adjusted for a particular IMU. Therefore, a calibration to find these parameters becomes necessary.

For gyroscopes, the calibration is straightforward. The bias of a gyroscope is the output when it is stationary. The scaling factors recommended by the IMU manufacturer are used for the gyroscopes.

For accelerometers, the calibration becomes non-trivial due to the gravity influence. A stationary 3-axis accelerometer yields readings corresponding to the gravity component on each axis. Suppose A_x, A_y, and A_z are the 16-bit accelerometer outputs for axis x, y and z respectively. k_x, k_y and k_z are the scaling factors. b_x, b_y, and b_z are the biases. When an 3-axis accelerometer sits still, we have

$$\sqrt{(k_x(A_x - b_x))^2 + (k_y(A_y - b_y))^2 + (k_z(A_z - b_z))^2} = g \tag{4.1}$$

where g is the gravity. So the accelerometer reading is a mixture of the bias and the gravity component. To obtain the biases and scaling factors, we turn to the nonlinear

data-fitting method. We formulate the problem into a Nonlinear Least-Squares Curve Fitting problems of the form

$$\min_x \|f(X)\|_2^2 = \min_x (f_1(X)^2 + f_2(X)^2 + \ldots + f_n(X)^2) \tag{4.2}$$

where

$$\begin{aligned} f_i(X) &= \sqrt{(k_x(A_{xi} - b_x))^2 + (k_y(A_{yi} - b_y))^2 + (k_z(A_{zi} - b_z))^2} - g; i \in [1, n] \\ X &= [k_x, b_x, k_y, b_y, k_z, b_z] \end{aligned}$$

We measure the accelerometer outputs at a few different poses when the IMU stays stationary. The number of poses should not be smaller than the number of biases and scaling factors, which is 6. The readings are substituted into Equation 4.2. Gauss-Newton algorithm is used to find the solution. The resulting biases and scaling factors are applied to the 16-bit measurements. Figure 12 shows an example of the result after applying the biases and scaling factors.

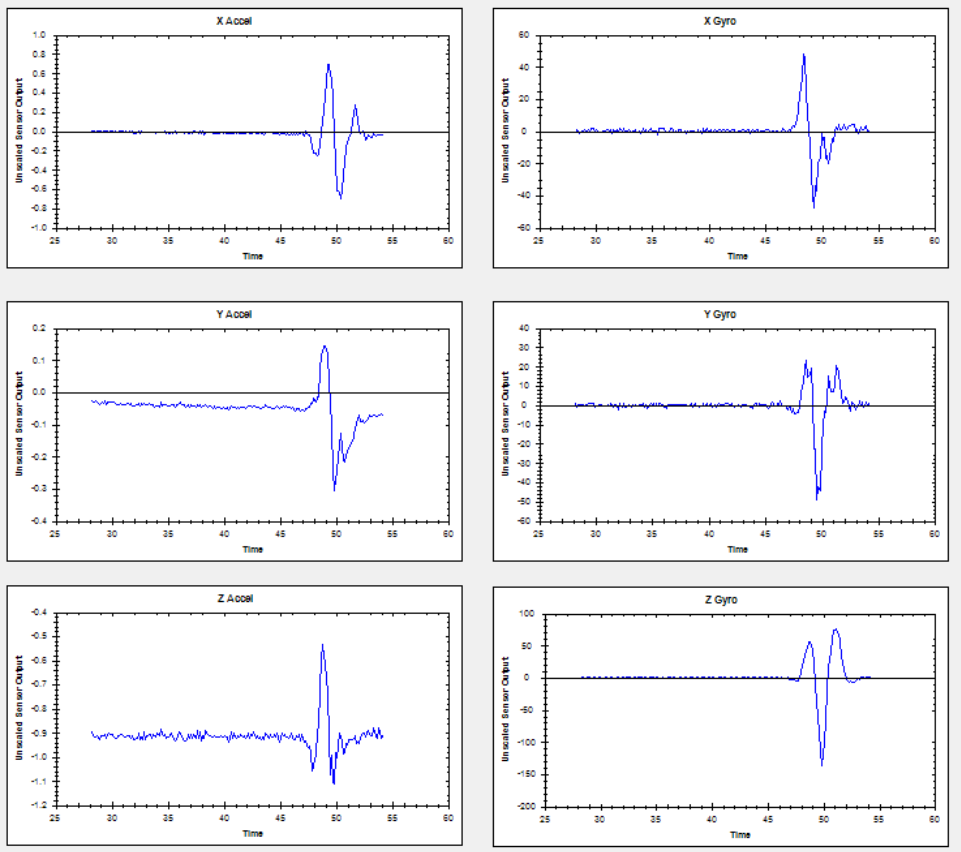

Figure 12. **Outputs of accelerometers and gyroscopes when writing digit 2 in the air. The 3 figures on the left hand side show the linear acceleration data for x, y, and z axis. The 3 figures on the right hand side show the angular acceleration data.**

CHAPTER 5

3D GESTURE TRAJECTORY RECONSTRUCTION

In this chapter, we describe the methods for reconstructing gesture trajectories, including a novel method we developed, Zero Velocity Linear Compensation (ZVLC), for improving trajectory reconstruction accuracy. As later shown in this chapter, ZVLC yields better reconstruction results than other methods.

After the sensor measurements are obtained, we can now calculate the trajectory of the IMU using the Inertial Navigation System (INS) theory [93][94]. An Inertial Navigation System (INS) is a navigation aid that uses a computer, motion sensors (accelerometers) and rotation sensors (gyroscopes) to continuously calculate the position, orientation, and velocity (direction and speed of movement) of a moving object without the need for external references [95]. The calculation involves two coordinate systems - the navigation coordinate system **n** and the body coordinate system **b**, which are shown in Figure 13. The origin of the navigation coordinate system **n** is the starting point of the trajectory. X_n, Y_n, and Z_n are the 3 axes. Z_n is parallel to the gravity. The body coordinate system **b** is a right-handed coordinate system on the moving object. If the moving object is an aircraft, **b** is usually defined as Figure 13 shows. X_b is along the longitudinal axis. Y_b is to the right along the lateral axis. Z_b is along the vertical axis. In our case, our moving object is the IMU. The body coordinate system **b** on the IMU is defined to be the same

as the one on an aircraft as demonstrated in Figure 13. All the measurements from the IMU sensors are based on **b**.

Next we will discuss how the trajectory is calculated. Table XIV shows the nomenclature used on aircrafts [93]. We use this nomenclature as well. In addition to the variables in Table XIV, the following is a list of other important variables that will be used in calculating the trajectory. $\mathbf{P_n} = [P_{nx}, P_{ny}, P_{nz}]^T$ - position of the origin of **b** in **n** (also the position of the IMU).

TABLE XIV

AIRCRAFT NOMENCLATURE

	X-axis Longitudinal axis Roll axis	Y-axis Lateral axis Pitch axis	Z-axis Vertical axis Yaw axis
Velocity components	u	v	w
Angular accelerations	Roll rate p	Pitch rate q	Yaw rate r
Euler angles	Roll angle ϕ	Pitch angle θ	Heading angle ψ
Accelerations	A_x	A_y	A_z

$\mathbf{V_n} = [u, v, w]^T$ - velocity of the origin of **b** in **n** (also the velocity of the IMU).

$\mathbf{A_n} = [A_{nx}, A_{ny}, A_{nz}]^T$ - acceleration of the origin of **b** in **n** (also the acceleration of the IMU).

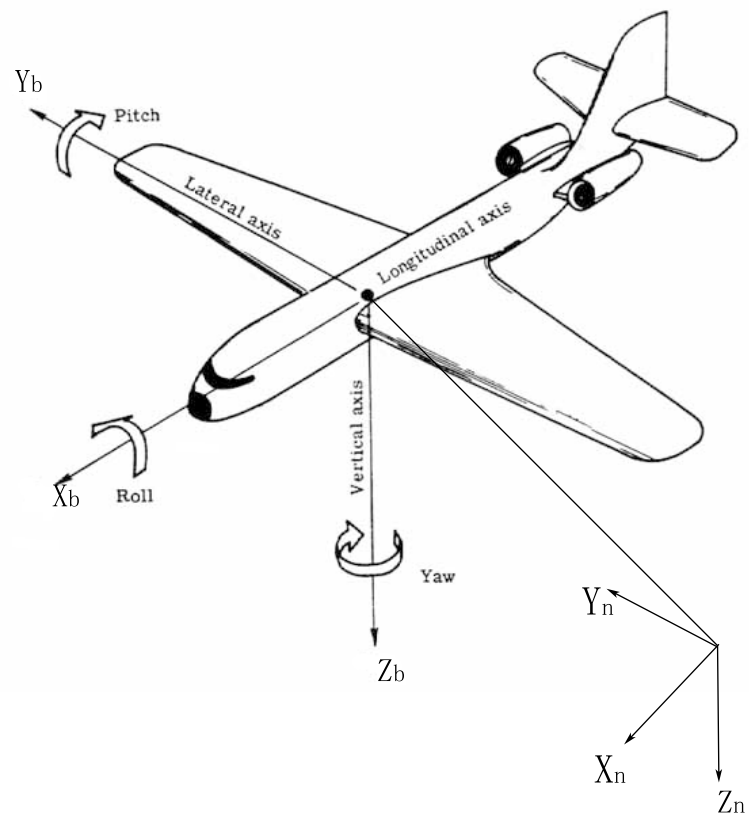

Figure 13. **Navigation Coordinate System n and Body Coordinate System b**

$\mathbf{A_b} = [A_{bx}, A_{by}, A_{bz}]^T$ - acceleration measurements from the IMU accelerometers (based on **b**).

$\omega_\mathbf{b} = [p, q, r]^T$ - angular acceleration measurements from the IMU gyroscopes (based on **b**).

$[\phi, \theta, \psi]$ - roll, pitch, and yaw angles of the IMU in **n**. The definitions of roll, pitch, and yaw are illustrated in Figure 14 [96]. Figure 13 further shows the positive directions of these three angles [97].

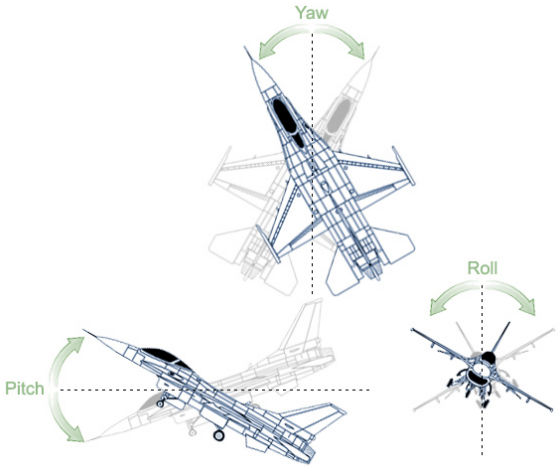

Figure 14. **Yaw, pitch and roll**

According to the INS theory, we have the following equations

$$\dot{\mathbf{P}}_{\mathbf{n}} = \mathbf{V}_{\mathbf{n}} \tag{5.1}$$

$$\dot{\mathbf{V}}_{\mathbf{n}} = \mathbf{A}_{\mathbf{n}} \tag{5.2}$$

$$\mathbf{A}_{\mathbf{n}} = \mathbf{C}_{\mathbf{b}}^{\mathbf{n}}\mathbf{A}_{\mathbf{b}} - \mathbf{G} \tag{5.3}$$

$$\dot{\psi} = \frac{q\sin\phi + r\cos\phi}{\cos\theta} \tag{5.4}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{5.5}$$

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta \tag{5.6}$$

where

$$\mathbf{C}_{\mathbf{b}}^{\mathbf{n}} = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

$$= \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi & \cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\psi & \sin\psi\sin\theta \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix}$$

$$= \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

$$\mathbf{G} = (0, 0, g)^T = (0, 0, 9.8)^T$$

To calculate the trajectory, i.e. position $\mathbf{P_n}$, we perform integrations on Equation 5.1 through Equation 5.6 as follows:

$$\mathbf{P_n}(i+1) = \mathbf{P_n}(i) + \frac{\mathbf{V_n}(i+1) + \mathbf{V_n}(i)}{2} \times t(i) \tag{5.7}$$

$$\mathbf{V_n}(i+1) = \mathbf{V_n}(i) + \frac{\mathbf{A_n}(i+1) + \mathbf{A_n}(i)}{2} \times t(i) \tag{5.8}$$

$$\psi(i+1) = \psi(i) + \frac{\dot{\psi}(i+1) + \dot{\psi}(i)}{2} \times t(i) \tag{5.9}$$

$$\theta(i+1) = \theta(i) + \frac{\dot{\theta}(i+1) + \dot{\theta}(i)}{2} \times t(i) \tag{5.10}$$

$$\phi(i+1) = \phi(i) + \frac{\dot{\phi}(i+1) + \dot{\phi}(i)}{2} \times t(i) \tag{5.11}$$

where $t(i)$ is the time elapsed between time instant i and $i+1$.

As seen from above, as many as three steps of integrations are involved in this process. A small error on $\dot{\psi}, \dot{\theta}$, and $\dot{\phi}$ will cause the final error on position $\mathbf{P_n}$ to grow exponentially after all these integrations. Fortunately, this problem could be overcome using our method, Zero Velocity Linear Compensation (ZVLC), as described later in this chapter.

5.1 Initial Pose Calculation

The iterative process in Equation 5.7 through Equation 5.11 needs initial values for the yaw, pitch and roll angles (ψ, θ, ϕ). This section will describe how we obtain the initial values for these 3 angles.

The initial yaw angle $\psi(0)$ can't be determined using only the accelerometers and gyroscopes. Therefore, we assume the initial yaw to be zero, i.e.

$$\psi(0) = 0$$

With regard to the initial pitch $\theta(0)$, it could be calculated as

$$\theta(0) = \arctan(\frac{A_{bx}(0)}{\sqrt{A_{by}^2(0) + A_{bz}^2(0)}}) \tag{5.12}$$

if the IMU stays stationary at the beginning of a gesture [98]. $A_{bx}(0)$, $A_{by}(0)$ and $A_{bz}(0)$ are the measured acceleration from x, y, z axis in $\mathbf{b}$ at time instance 0.

As for the initial roll $\phi(0)$, it could be derived from Equation 5.3. We expand Equation 5.3 here for convenience of discussion.

$$A_n = \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

$$* \begin{pmatrix} A_{bx} \\ A_{by} \\ A_{bz} \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

Since the initial yaw $\psi(0)$ is zero, we have

$$\sin\psi(0) = 0, \cos\psi(0) = 1$$

Also if the IMU is held still at the beginning of a gesture, the acceleration A_n in $\mathbf{n}$ is zero. So Equation 5.3 could be written as

$$0 = \begin{pmatrix} \cos\theta(0) & \sin\theta(0)\sin\phi(0) & \sin\theta(0)\cos\phi(0) \\ 0 & \cos\phi(0) & -\sin\phi(0) \\ -\sin\theta(0) & \cos\theta(0)\sin\phi(0) & \cos\theta(0)\cos\phi(0) \end{pmatrix} \begin{pmatrix} A_{bx}(0) \\ A_{by}(0) \\ A_{bz}(0) \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

$$\begin{pmatrix} \cos\theta(0) & \sin\theta(0)\sin\phi(0) & \sin\theta(0)\cos\phi(0) \\ 0 & \cos\phi(0) & -\sin\phi(0) \\ -\sin\theta(0) & \cos\theta(0)\sin\phi(0) & \cos\theta(0)\cos\phi(0) \end{pmatrix} \begin{pmatrix} A_{bx}(0) \\ A_{by}(0) \\ A_{bz}(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

which could be expressed as

$$A_{bx}(0)\cos\theta(0) + A_{by}(0)\sin\theta(0)\sin\phi(0) + A_{bz}(0)\sin\theta(0)\cos\phi(0) = 0 \quad (5.13)$$

$$A_{by}(0)\cos\phi(0) - A_{bz}(0)\sin\phi(0) = 0 \quad (5.14)$$

$$-A_{bx}(0)\sin\theta(0) + A_{by}(0)\cos\theta(0)\sin\phi(0) + A_{bz}(0)\cos\theta(0)\cos\phi(0) = g \quad (5.15)$$

We could pick either Equation 5.13 or Equation 5.14 to solve for the initial roll $\phi(0)$. For simplicity, we pick Equation 5.14, which could be written as

$$A_{by}(0)\cos\phi(0) = A_{bz}(0)\sin\phi(0) \quad (5.16)$$

$$\tan\phi(0) = \frac{A_{by}(0)}{A_{bz}(0)} \quad (5.17)$$

$$\phi(0) = \arctan(\frac{A_{by}(0)}{A_{bz}(0)}) \quad (5.18)$$

5.2 Zero Velocity Linear Compensation

In this section, we describe our novel method, Zero Velocity Linear Compensation (ZVLC), for reducing trajectory reconstruction error. As described earlier, the process to reconstruct the trajectory involves 3 integration steps, which causes the reconstruction error to grow out of control quickly given any error on the sensor measurements. The biases and scaling factors obtained in Section 4.2 can't be perfectly accurate due to temperature drift, sensor errors (Package Alignment Error, Inter-axis Alignment Error, Cross-Axis Sensitivity [99][100]), etc. Any deviation from the real biases and scaling fac-

tors will cause inaccurate measurements of the accelerations and angular accelerations, therefore inaccurate trajectories. Examples of this kind of trajectories are demonstrated in Figure 15 and Figure 16. Those two figures show reconstructed trajectories from our experiments. In these experiments, a subject draws digits 0 to 9 in the air. Trajectories for every digit are calculated using 3 different methods and shown in Figure 15 and Figure 16. The first trajectory is the reconstruction result using unprocessed acceleration and angular acceleration data. The other two trajectories are results of applying Zero Velocity Compensation (ZVC) and Zero Velocity Linear Compensation (ZVLC), which will be explained later in this section. As we can see, the reconstructions using unprocessed sensor data are not satisfactory, especially those for digits 1, 4, 5, 7, and 9. The reconstructed digit 4 is not even close to the correct trajectory.

5.2.1 Zero Velocity Compensation

In order to improve the accuracy of the reconstructed trajectories, researchers have developed many methods. One method to correct the errors is to add additional sensors [101], e.g. GPS or optical sensors. Another method is to rely on known information at certain time instances to correct the errors. Coordinates, orientations, and velocities are commonly used for this purpose [102]. For instance, Zero Velocity Updates (ZUPT) method [102] uses velocities. When the IMU stops at certain time instances, ZUPT will update the calculated velocities at those instances to zeros although the original calculated velocities might not be zeros. In [103], Frank proposes a more complicated

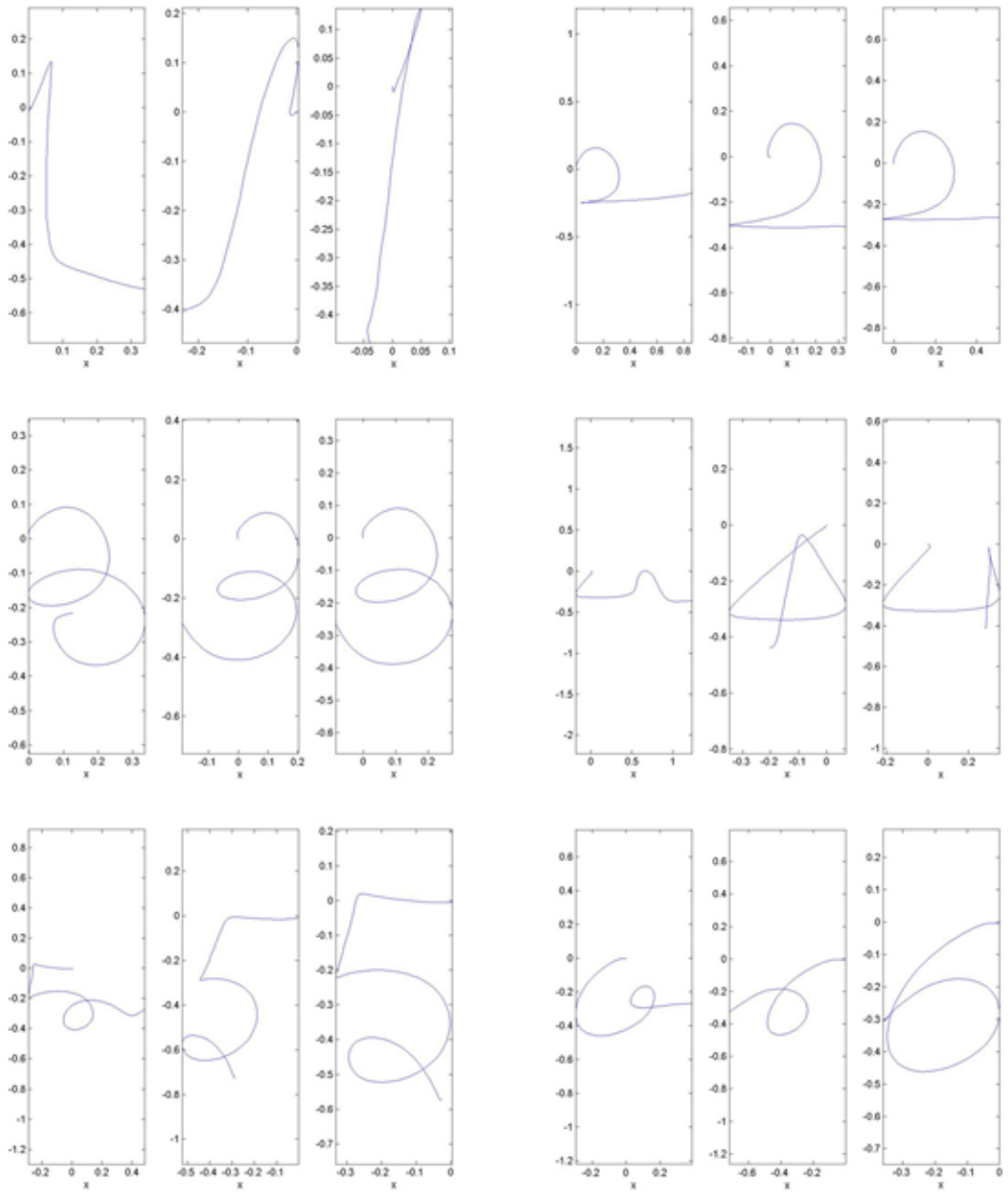

Figure 15. **Reconstructed trajectories for digit 1-6. 3 trajectories for every digit are reconstruction results using unprocessed sensor data, ZVC and ZVLC respectively.**

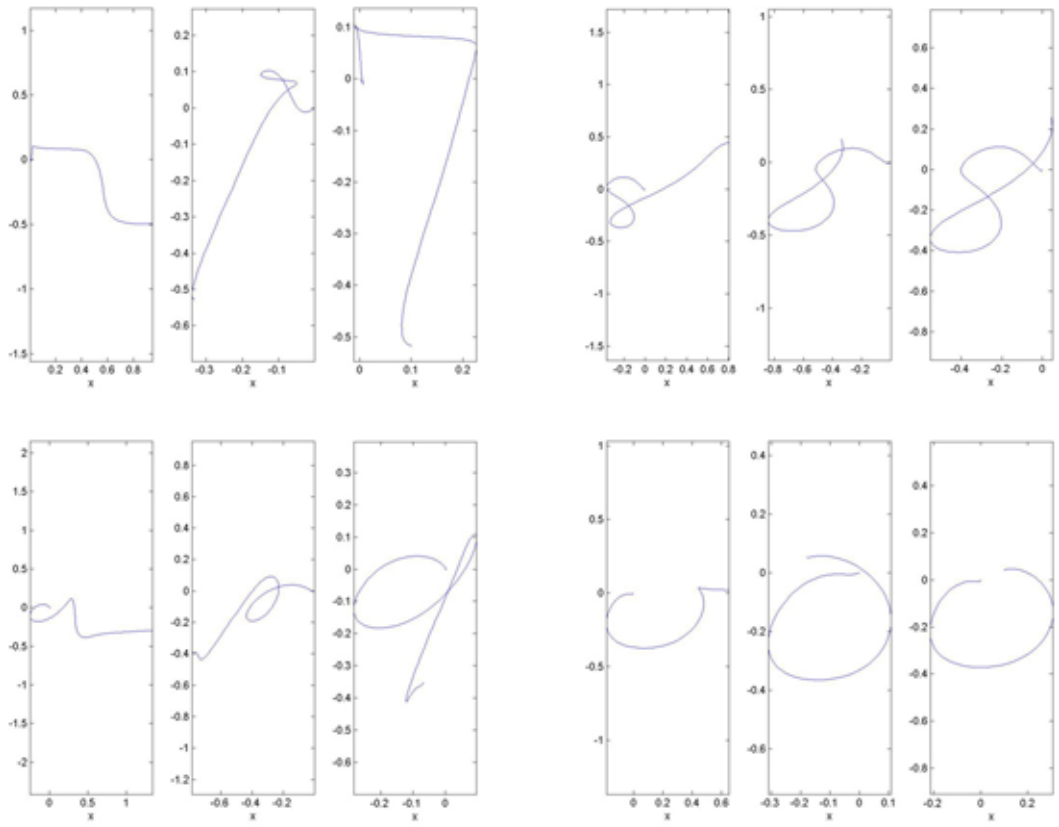

Figure 16. **Reconstructed trajectories for digit 7, 8, 9 and 0. 3 trajectories for every digit are reconstruction results using unprocessed sensor data, ZVC and ZVLC respectively.**

velocity updating scheme in which velocities are updated on every time instance on the trajectory. The method is named Zero Velocity Compensation (ZVC) in [71]. ZVC requires the beginning and ending velocities on a trajectory to be both zeros. The details of ZVC are illustrated in Figure 17 [71]. For the acceleration $A_n(k)$ for $k_1 \leq k \leq k_2$, the compensated acceleration is [71]

$$\tilde{A}_n(k) = A_n(k) - \frac{V_n(k_2) - V_n(k_1)}{(k_2 - k_1)T_s} \tag{5.19}$$

The second term on the right hand side of Equation 5.19 could be considered as a constant acceleration error in the navigation coordinate system $\mathbf{n}$.

We implement ZVC in our trajectory reconstruction experiment. The subject is require to stop the IMU at the beginning and the end of gestures to satisfy the requirements of ZVC. The reconstruction results are shown in Figure 15 and Figure 16 - the second figure for each digit is the result of applying ZVC. As can be seen, most ZVC results are better than those from using unprocessed acceleration data. This is especially obvious for digit 4 and 5. However, the reconstructions are still not accurate enough for some of the digits, e.g. digit 1, 4, 7 and 9. This indicates that it's not accurate enough to model the acceleration error as a constant term in this case.

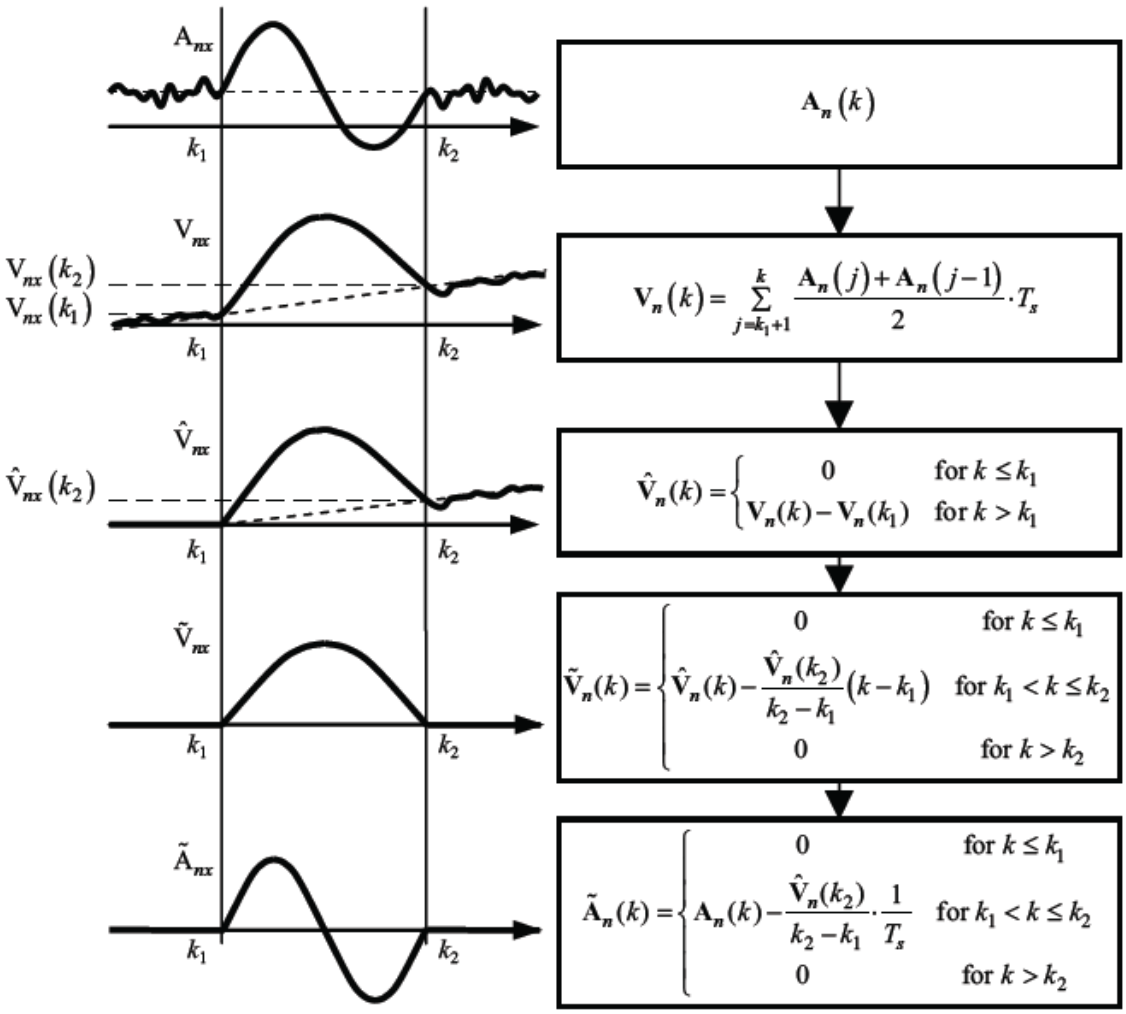

Figure 17. **Zero Velocity Compensation**

5.2.2 Zero Velocity Linear Compensation

In order to achieve better results, we propose a new method, which models the error as a linear term, i.e.

$$A_{err} = c \times t = c(k - k_1)T_s \tag{5.20}$$

where c is the coefficient and t is the time. We name our method Zero Velocity Linear Compensation (ZVLC). To determine the linear factor c in Equation 5.20, we first repeat one of the equations in Figure 17.

$$\hat{V}_n(k) = \begin{cases} 0 & \text{for } k \leq k_1 \\ V_n(k) - V_n(k_1) & \text{for } k > k_1 \end{cases}$$

$\hat{V}_n(k)$ is the velocity error caused by the acceleration error. At the end of the trajectory (time instance k_2), the velocity error is

$$V_{err}(k_2) = \hat{V}_n(k_2) = V_n(k_2) - V_n(k_1) \tag{5.21}$$

Also

$$\begin{aligned} \dot{V}_{err} &= A_{err} = c \times t \\ V_{err} &= 0.5c \times t^2 + d \end{aligned} \tag{5.22}$$

Since $V_{err}(0) = 0$,

$$0.5c \times 0^2 + d = 0 \rightarrow d = 0$$

So

$$V_{err} = 0.5c \times t^2$$

$$V_{err}(k_2) = 0.5c(k_2 - k_1)^2 T_s^2 \tag{5.23}$$

Considering both Equation 5.21 and Equation 5.23, we have

$$\begin{aligned} 0.5c(k_2 - k_1)^2 T_s^2 &= V_n(k_2) - V_n(k_1) \\ c &= \frac{2(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2 T_s^2} \\ A_{err} &= c \times t \\ &= \frac{2(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2 T_s^2}(k - k_1)T_s \\ &= \frac{2(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2 T_s}(k - k_1) \end{aligned} \tag{5.24}$$

Therefore, the compensated acceleration and velocity are

$$\begin{aligned}
\tilde{A_n}(k) &= A_n(k) - A_{err} \\
&= A_n(k) - \frac{2(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2 T_s}(k - k_1) \qquad (5.25) \\
\tilde{V_n}(k) &= V_n(k) - V_{err} \\
&= V_n(k) - \frac{1}{2} c \times t^2 \\
&= V_n(k) - \frac{1}{2}\frac{2(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2 T_s^2}(k - k_1)^2 T_s^2 \\
&= V_n(k) - \frac{(V_n(k_2) - V_n(k_1))}{(k_2 - k_1)^2}(k - k_1)^2 \qquad (5.26)
\end{aligned}$$

The results of applying ZVLC are shown in Figure 15 and Figure 16 - the third trajectory for each digit is the reconstruction result after applying ZVLC. As can be observed, ZVLC reconstructs digit 1, 7, and 9 accurately while the ZVC reconstructions are not accurate at all. The ZVLC reconstructions for digit 4 and 6 are also much more improved from the ZVC reconstructions. For the remaining digits, ZVLC reconstructions show improvements over ZVC reconstructions as well.

CHAPTER 6

GESTURE RECOGNITION BY INDEXING AND SEQUENCING

In this chapter, we describe a novel Gesture Recognition method based on RISq and present the recognition results.

After the trajectories are reconstructed by applying ZVLC, we apply RISq (Recognition by Indexing and Sequencing), as described in Chapter 2, to recognize the trajectories. In our experiment, the subject holds the IMU and draws digits 0 to 9 in the air. These gestures are used for training. Next, subjects are asked to repeat the gestures in the recognition phase. We use the projections of the 3D trajectories on a certain plane for recognition since humans tend to write on a virtual plane when writing in the air. In our experiments, we require the subject to try to write in the $x - z$ plane. The IMU is held in a certain way so that the subject could write in this plane. The direction α at each point along this 2D projection is used as the feature for RISq. We use the four-quadrant inverse tangent as shown in Equation 6.1 and Figure 18.

$$\alpha_i = \arctan 2(z_{i+1} - z_i, x_{i+1} - x_i) \tag{6.1}$$

where $\alpha_i \in (-\pi, \pi]$.

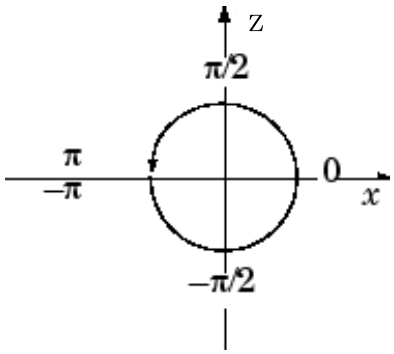

Figure 18. **Four-quadrant inverse tangent**

Next we describe how RISq is applied to recognize the 2D projections. Suppose we have a test sequence $\{\mathbf{Y}_t = \mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \cdots, \mathbf{x}_{t_q}\}$ and a set of model sequences $\{\mathbf{Y}_j = \mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \cdots, \mathbf{x}_{j_q}; j \in [1, J]\}$ where J is the number of models in the database. According to Equation 2.20, the vote for the test sequence $\mathbf{Y}_t$ is

$$\begin{aligned} \Omega &= \arg\max_j \Big(\sum_{i=1}^{q} [-\frac{1}{2}(\mathbf{x}_{t_i} - \mathbf{x}_{j_i})^T C_x^{-1} (\mathbf{x}_{t_i} - \mathbf{x}_{j_i})]\Big) \\ &= \arg\max_j \Big(\sum_{i=1}^{q} [-\frac{1}{2\delta}(\alpha_{t_i} - \alpha_{j_i})^2]\Big) \end{aligned} \tag{6.2}$$

where δ is the variance of the direction α. α falls into the range of $(-\pi, \pi]$ and is discontinuous at π (also $-\pi$). In order to overcome this discontinuity problem, we calculate another two votes by adding 2π to α_{t_i} or subtracting 2π from α_{t_i}, i.e.

$$\begin{aligned} v_{1i} &= -\frac{1}{2\delta}(\alpha_{t_i} - \alpha_{j_i})^2 \\ v_{2i} &= -\frac{1}{2\delta}((\alpha_{t_i} + 2\pi) - \alpha_{j_i})^2 \\ v_{3i} &= -\frac{1}{2\delta}((\alpha_{t_i} - 2\pi) - \alpha_{j_i})^2 \end{aligned}$$

We pick the maximum vote v_i

$$v_i = max(v_{1i}, v_{2i}, v_{3i})$$

So Equation 6.2 changes into

$$\Omega = \arg\max_j (\sum_{i=1}^{q} v_i) \tag{6.3}$$

The model with the highest vote Ω is classified as the winning model.

6.1 Experiments

We pick digits 0 to 9 as the gestures and the gesture shapes are designed for high recognition rates and convenience of writing. Figure 19 and Figure 20 show more recon-

structed gesture trajectories by applying ZVLC. As can be seen, the trajectories look smooth and reasonably accurate.

Figure 19. **Examples of reconstructed trajectories (by ZVLC)**

6.1.1 Recognition by Indexing and Sequencing

We apply RISq to recognize the trajectories. For every digit, we use one trajectory for training and another 19 trajectories for testing. RISq achieves 92% recognition rate.

6.1.2 Recognition by Hidden Markov Model

In order to compare RISq with HMM, we also apply HMM to recognize the same trajectories. For every model, 10 sequences are used to train a HMM model and the other 10 sequences are used for testing. The number of states are varied to find the highest recognition rate. The results are shown in Table XV. As can be seen from the

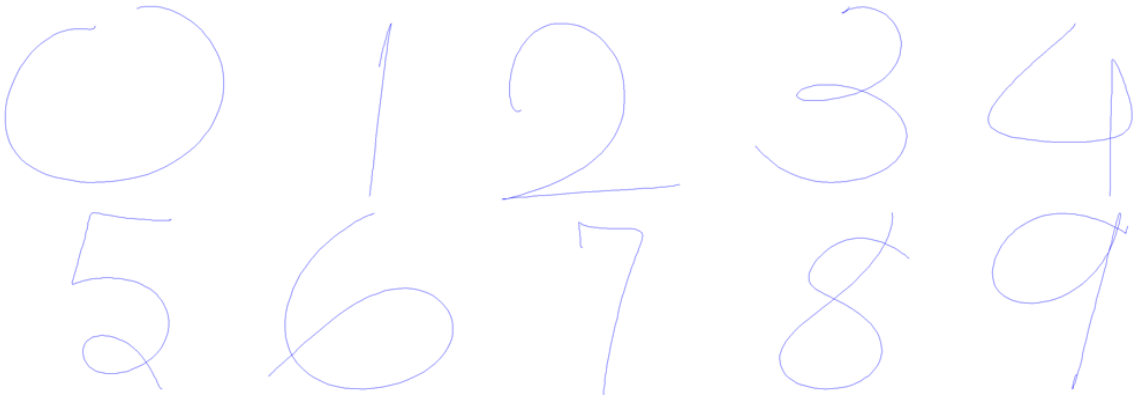

Figure 20. **More examples of reconstructed trajectories (by ZVLC)**

table, HMM reaches the highest recognition rate of 83% when 18 hidden states are used, which is much lower than the 92% recognition rate achieved by RISq in the previous section.

TABLE XV

HMM TEST RESULTS

Number of States	2	4	6	8	10	12	14	16	18	20	25	30
Recognition Rate	56	70	74	77	78	76	75	76	83	76	75	75

CHAPTER 7

VIEW INVARIANT HEAD RECOGNITION BY HYBRID PRINCIPAL COMPONENT ANALYSIS BASED RECONSTRUCTION

In this chapter, we propose a novel method for 3D head reconstruction and view-invariant recognition from single 2D images. The method described in this chapter was published in two journal papers [7][8] and two other conference papers [9][10]. In this work, We employ a deterministic Shape From Shading (SFS) method with initial conditions estimated by Hybrid Principal Component Analysis (HPCA). Our HPCA algorithm provides good initial estimates of 3D range mapping for the SFS optimization and yields much improved 3D head reconstruction. The chapter also describes a novel method in SFS handling of variable and unknown surface albedo, a problem with unsatisfactory solutions by prevalent SFS methods. In the experiments, we reconstruct 3D head range images from 2D single images in different views. The 3D reconstructions are then used to recognize stored model persons. This enables one to recognize faces in wide range of views. Empirical results show that our HPCA based SFS method provides 3D head reconstructions that notably improve the accuracy compared to other approaches.

7.1 Introduction

3D face reconstruction from one or multiple 2D face images is an interesting topic that receives a lot of attention. Blanz and Vetter proposed a morphable model for 3D

faces reconstruction using an analysis-by-synthesis approach in [104] and later developed a face recognition method in [105], which is based on matching eigenvector coefficients. This method is different from ours in this chapter, which matches 3D geometry for recognition. Jiang et al. [106] used detected face features to determine coefficients for synthesis from shape eigenvectors. Hu et al. [107] utilized a generic 3D face model and detected face features to reconstruct 3D faces with the help of a Shape From Shading (SFS) method and Radial Basis Functions (RBFs). The last two methods reconstruct faces only from frontal face images. In addition, surface albedo was assumed constant in [107], which led to inaccurate range on some feature points. Smith and Hancock [108] used an image normalization algorithm to decouple surface normal directions from variable surface albedo. Illumination cones and a Point Distribution Model were employed in a geometric SFS method to refine estimated normals. However, the overall performance of this method is determined by the accuracy of the normalization process. As we can see from above, pose and variable albedo are major concerns in 3D face reconstruction. We propose in this chapter a novel method for 3D head reconstruction by SFS which addresses these concerns. Our method reconstructs facial range images from 2D face images in any pose. Furthermore, our approach provides the capability to estimate variable surface albedo. Such a capability is absent in most of the prevalent SFS methods.

Research on SFS has been conducted for decades. Ikeuchi and Horn [109] proposed to recover shape information by minimizing a cost function. The stereographic plane was

employed in their method to express orientations of surface patches. In [110], Horn and Brooks applied the calculus of variations to solve SFS problems. Zheng and Chellappa [111] proposed to estimate illumination direction, albedo, and surface shape by minimizing a cost function with a new smoothness constraint, which was aimed at decreasing the gradient difference between the reconstructed intensity image and the input image. Kimmel and Bruckstein proposed the Level Sets method in [112]. A Lambertian smooth surface is recovered by numerically propagating an almost arbitrarily initialized surface and tracking the level sets. Worthington and Hancock [113] replaced estimated normals with the closest normalized vector on illumination cones to ensure accuracy of recovered surface normals. Samaras and Metaxas [114] incorporated illumination constraints with deformable models in resolving SFS problems. Crouzil et al. [115] developed a multiresolution SFS method, in which cost functions were minimized by fuzing deterministic and stochastic minimization approaches. During the examination of these SFS methods, we find that surface albedo was assumed either constant or given. Assuming constant albedo results in inaccurate reconstruction of surfaces with variable albedo as was demonstrated by the experiments in [116].

Existing SFS methods almost always yield unsatisfactory results when applied to realistic imagery when the initial estimation of the true surface is unavailable or inaccurate. In experiments described in this chapter, we demonstrate that providing an good initial estimation in SFS methods yields much better results. The Hybrid Principal Component

Analysis algorithm provides good head surface estimations. These estimations serve as initial conditions for our multiple-level optimization. The introduction of HPCA and the multiple-level optimization combined with albedo estimation, are the innovative parts of our approach.

The rest of the chapter is arranged as follows: The HPCA (Hybrid Principal Component Analysis) algorithm is described in section 7.2; In section 7.3, we present results from HPCA; Section 7.4 describes the SFS (Shape From Shading) method; 7.5 describes the face recognition module based on Iterative Closest Point (ICP); Section 7.6 presents SFS results and face recognition results; Section 7.7 concludes the chapter.

7.2 Hybrid Principle Component Analysis

In this section, we describe the HPCA algorithm. To perform the HPCA algorithm, we need a set of M training images. Each image is a hybrid composed of a 2D $[n \times m]$ gray scale image and a corresponding $[n \times m]$ range image. These training images are lexicographically reordered into M pairs of vectors, denoted by

$$\{\overrightarrow{f_i}, \overrightarrow{r_i}\} \qquad i = 1, 2, \cdots, M \tag{7.1}$$

where $\overrightarrow{f_i}$ is the vector that represents the i^{th} gray scale image and $\overrightarrow{r_i}$ is the vector that represents the corresponding range image. These two vectors are concatenated to generate a $2nm$ dimensional hybrid vector $\overrightarrow{h_i}$,

$$\overrightarrow{h_i} = \left(\begin{array}{c} \overrightarrow{f_i}^T, \overrightarrow{r_i}^T \end{array} \right)^T \tag{7.2}$$

The training set H for HPCA consists of all the M hybrid vectors $\overrightarrow{h_i}$. The mean vector $\overrightarrow{\mu_h}$ and covariance matrix C_h for H are calculated as follows

$$\overrightarrow{\mu_h} = \frac{1}{M} \sum_{i=1}^{M} \overrightarrow{h_i} \tag{7.3}$$

$$C_h = \frac{1}{M} \sum_{i=1}^{M} (\overrightarrow{h_i} - \overrightarrow{\mu_h})(\overrightarrow{h_i} - \overrightarrow{\mu_h})^T \tag{7.4}$$

Next, the eigenvectors $\{\overrightarrow{v_j}; j = 1, 2, \cdots, \omega\}$ for C_h are computed. $\omega \leq 2nm$ is the rank of C_h. The first P eigenvectors, which correspond to the P largest eigenvalues, are taken as the principal eigenvectors. Every eigenvector $\overrightarrow{v_j}$ is then split into two sub-vectors with nm dimensions each. We name the two sub-vectors as the top vector $\overrightarrow{t_j}$ and the bottom vector $\overrightarrow{b_j}$ respectively, i.e.

$$\overrightarrow{v_j} = \left(\begin{array}{c} \overrightarrow{t_j}^T, \overrightarrow{b_j}^T \end{array} \right)^T \qquad \overrightarrow{t_j} \in R^{nm}, \overrightarrow{b_j} \in R^{nm} \tag{7.5}$$

The vector set $\{\overrightarrow{t_j}\}$ corresponds to the gray scale images while the set $\{\overrightarrow{b_j}\}$ corresponds to the range images. Similarly, the mean vector $\overrightarrow{\mu_h}$ is split into two sub-vectors as well.

$$\overrightarrow{\mu_h} = \left(\overrightarrow{\mu_f}^T, \overrightarrow{\mu_r}^T \right)^T \tag{7.6}$$

We also perform PCA on the set of range images $\{\overrightarrow{r_i}; i = 1, 2, \cdots, M\}$ and obtain P principal eigenvectors $\{\overrightarrow{e_j}; j = 1, 2, \cdots, P\}$ for the range image space $S_r \in R^{nm}$. Obviously, The set $\{\overrightarrow{e_j}\}$ for S_r would be different from the set $\{\overrightarrow{b_j}\}$ for the hybrid space. However, we can approximate $\{\overrightarrow{e_j}\}$ with $\{\overrightarrow{b_j}\}$, i.e., we use $\{\overrightarrow{b_j}\}$ as an estimation of the principal eigenvectors for the range image space S_r. Similarly, we use $\{\overrightarrow{t_j}\}$ as an estimation of the principal eigenvectors of the gray scale image space $S_g \in R^{nm}$.

The underlying principle of HPCA (Hybrid Principal Component Analysis) is that a range image $\overrightarrow{r}$ can be approximated by a linear combination of the set $\{\overrightarrow{b_j}\}$ using the projection coefficients obtained by projecting the corresponding gray scale image $\overrightarrow{f}$ onto the set $\{\overrightarrow{t_j}\} \in S_g$, namely

$$\overrightarrow{f} = T\overrightarrow{d} + \overrightarrow{\mu_f} \tag{7.7}$$

$$\Longrightarrow \quad \overrightarrow{d} = (T^T T)^{-1} T^T (\overrightarrow{f} - \overrightarrow{\mu_f}) \tag{7.8}$$

$$\Longrightarrow \quad \overrightarrow{r} = B\overrightarrow{d} + \overrightarrow{\mu_r} \tag{7.9}$$

where $T = (\overrightarrow{t_1}\ \overrightarrow{t_2} \cdots \overrightarrow{t_\omega}), B = (\overrightarrow{b_1}\ \overrightarrow{b_2} \cdots \overrightarrow{b_\omega})$ and $\overrightarrow{d}$ is the coefficient vector.

7.3 Experimental Results for Hybrid Principal Component Analysis

To achieve reconstruction of heads in different poses, we need a set of gray scale images and corresponding range images taken in intervals of few degrees about the vertical axes of the heads. For this purpose, we synthesize the gray scale images from a 3D head model library provided by USF [117]. A few synthetic gray scale images are illustrated in 7.3. These synthetic images still look realistic since the variable albedo is also taken into account.

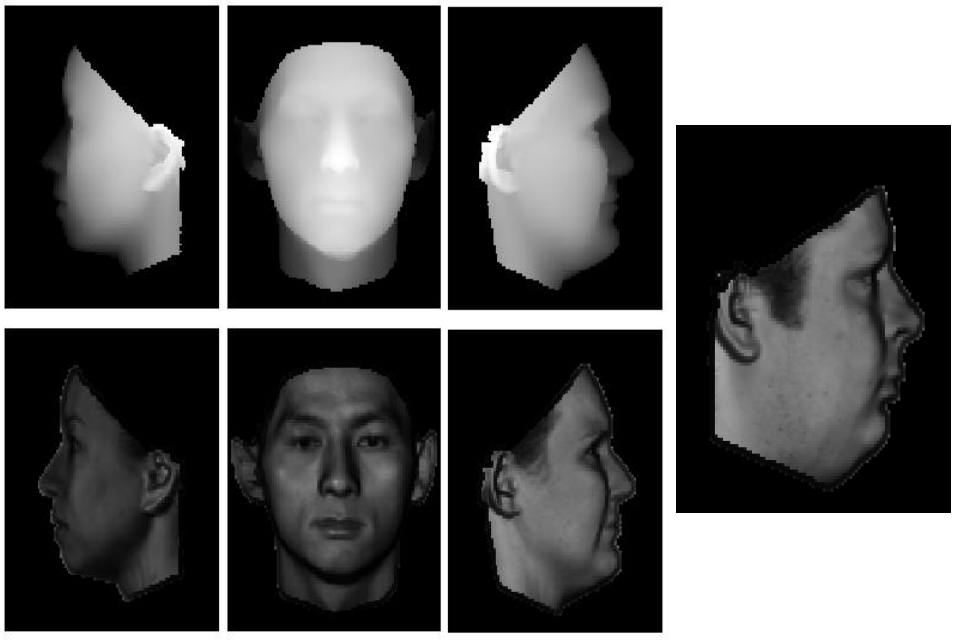

Figure 21. **(a) 3 pairs of training images (Top row - Range images; Bottom row - Corresponding gray scale images); (b) A test image**

The library from USF includes 100 3D head models and corresponding texture maps. We use 40 head models in the experiment. Every model is rotated about the vertical axis from -90 to +90 degrees in a step size of 5 degrees. A gray scale image and a range image are generated for every pose, which leads to 1480 hybrids for all 40 models. Few pairs of training images are illustrated in 7.3 and Figure 22.

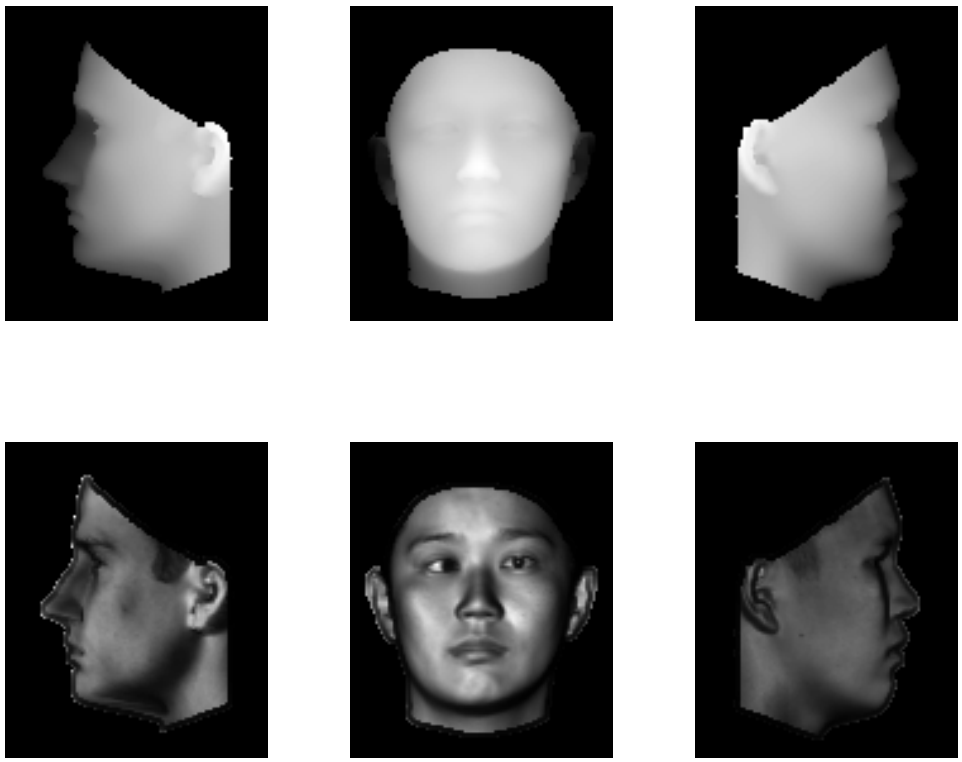

Figure 22. **More examples of the training images (Top row - Range images; Bottom row - Corresponding gray scale images)**

It is straightforward to obtain model range images while it is more difficult to obtain model gray scale images. Here we need to make two assumptions:

1. Head surfaces exhibit Lambertian reflectance.

2. The light is perpendicular to the image plane.

With these two assumptions, the gray scale value at a point A can be calculated as

$$R_A = \rho\alpha(\overrightarrow{l} \cdot \overrightarrow{n}) = u(\overrightarrow{l} \cdot \overrightarrow{n}) \tag{7.10}$$

where ρ is the illuminant strength, α is the surface albedo, u is the composite albedo [111] representing the product of ρ and α, $\overrightarrow{l} = (0, 0, 1)^T$ represents the illuminant direction, and $\overrightarrow{n}$ is the normalized surface normal at point A.

$$\overrightarrow{n} = \frac{1}{\sqrt{1 + p^2 + q^2}} \times (-p, -q, 1)^T \tag{7.11}$$

p and q are the surface derivatives along x and y axes respectively. Substitute Equation 7.11 into Equation 7.10, we get

$$R_A = \frac{\rho\alpha}{\sqrt{1 + p^2 + q^2}} = \frac{u}{\sqrt{1 + p^2 + q^2}} \tag{7.12}$$

illuminant strength $\rho = 1$ is used in our experiment. As for the surface albedo α, note that there is a texture map for every model in the library from USF. We map the texture

to the 3D model and take the normalized gray scale value at every point as its albedo. The gray scale images for testing are obtained in a similar manner from 3D head models which are not included in the training. A test image is shown in 7.3b. Two views of the corresponding original range image are shown in Figure 23a and Figure 23d. The range image reconstructed by HPCA is illustrated in Figure 23b and Figure 23e. As can be observed, the reconstruction is close to the original range image except that it misses fine details and adds spurious noise in the extremes. The noise results from the discontinuity between the background and the face surfaces in the range images for training. However, it's later shown that the noise can be eliminated almost completely by the SFS (Shape From Shading) method described in section 7.4.

7.4 Surface Reconstruction by Shape from Shading

7.4.1 Shape from Shading: Cost Function and the Minimization

Starting from the initial reconstruction provided by HPCA (Hybrid Principal Component Analysis), we further improve the reconstruction using a SFS method. SFS is usually modeled as an optimization problem [116], in which cost functions are minimized subject to various constraints. Existing methods either try to estimate the surface height directly [118], or to divide the problem into two subproblems[115] [113] . First, to compute the surface's gradient field and then to calculate surface height from the gradient field. Our method belongs to the second category.

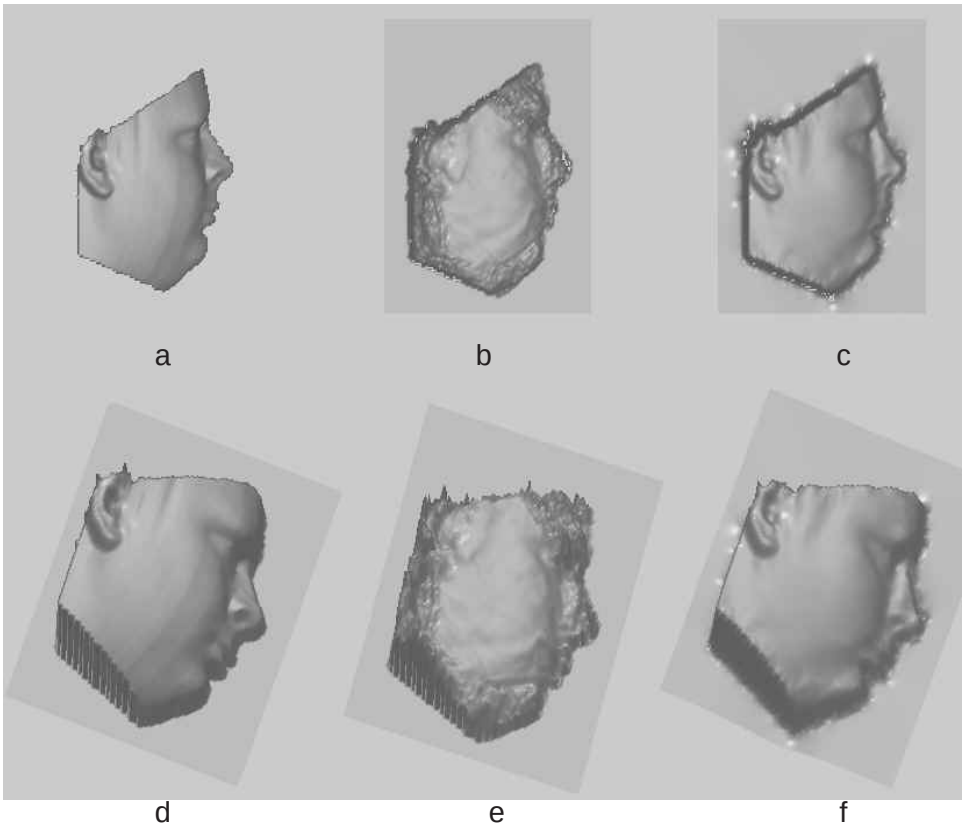

Figure 23. **(a)&(d) The original range image in the x-y and 3D view; (b)&(e) The reconstructed range image from HPCA in the x-y and 3D view. This reconstruction serves as the initial estimation for the optimization algorithm; (c)&(f) The x-y and 3D view of the final reconstruction after applying the SFS optimization**

Henceforth, we refer to the estimated 3D surface as $z(x, y)$, where z is the surface height and x, y are lateral coordinates. The two components of the gradient field are

$$p(x, y) = \partial z(x, y)/\partial x \qquad q(x, y) = \partial z(x, y)/\partial y \tag{7.13}$$

We present the fundamental equation for SFS, Image Irradiance Equation [110], as

$$I(x,y) = R(\overrightarrow{l}, p(x,y), q(x,y)) \tag{7.14}$$

where R and I are the reflectance map and the input gray scale image respectively, and $\overrightarrow{l}$ is the illuminant direction. The cost function shown below is minimized to find the gradient field in image domain Ω ($\Omega = \{(x,y)\}, 1 \leq x \leq m, 1 \leq y \leq n$) for a $n \times m$ image.

$$\begin{aligned} C_1(p,q) &= \iint_\Omega [R(\overrightarrow{l}, p(x,y), q(x,y)) - I(x,y)]^2 dxdy \\ &= \iint_\Omega [\frac{u(x,y)}{\sqrt{1+p^2(x,y)+q^2(x,y)}} - I(x,y)]^2 dxdy \end{aligned} \tag{7.15}$$

where $u(x,y)$ is the composite albedo. To get a well-posed solution, an integrability constraint and a smoothness constraint are usually added [110] [115] and the augmented cost function is given below.

$$\begin{aligned} C_2(p,q) &= C_1(p,q) + \lambda_i \iint_\Omega [p_y(x,y) - q_x(x,y)]^2 dxdy \\ &+ \lambda_{s1} \iint_\Omega [p_x^2(x,y) + p_y^2(x,y) + q_x^2(x,y) \\ &+ q_y^2(x,y)] dxdy \end{aligned} \tag{7.16}$$

where

$$p_x(x,y) = \partial p(x,y)/\partial x \tag{7.17}$$

$$p_y(x,y) = \partial p(x,y)/\partial y \tag{7.18}$$

$$q_x(x,y) = \partial q(x,y)/\partial x \tag{7.19}$$

$$q_y(x,y) = \partial q(x,y)/\partial y \tag{7.20}$$

λ_i is the integrability factor and λ_{s1} is the smoothing factor. Both factors are set to positive values. The second and the third terms on the right-hand side represent the integrability constraint and the smoothness constraint respectively.

To handle the variable albedo of faces, we add to the cost function $C_2(p,q)$ a smoothness constraint for the composite albedo u and estimate the gradient field and u simultaneously. The new constraint for the composite albedo u is inspired by the observation that abrupt changes of albedo usually only occur on boundaries between special face regions (e.g. lips, eyebrows, eyes or pigmentation). Other than that, the albedo usually varies smoothly, especially on cheeks and foreheads. As a matter of fact, even albedo inside some special regions, e.g. lips, doesn't vary abruptly. As for illuminant strength, in most cases it remains constant or varies smoothly on faces. Therefore, u should also

vary smoothly on most face regions. Hence, imposing a smoothness constraint on u is justified almost everywhere. The cost function with the new constraint is

$$C = C_2(p,q) + \lambda_{s2} \iint_{\Omega} [u_x^2(x,y) + u_y^2(x,y)] dxdy \tag{7.21}$$

where λ_{s2} is the smoothing factor for u.

Next we will convert the cost function into discrete form. Hereafter we will use subscripts to indicate the coordinates of pixels. We apply either forward or backward finite difference method in the conversion. Without loss of generality, we will discuss the case where forward finite difference method is applied. According to the definition of forward finite difference method, we have

$$p_x|_{(i,j)} = p(i,j+1) - p(i,j) = p_{i,j+1} - p_{i,j} \tag{7.22}$$

$$p_y|_{(i,j)} = p(i+1,j) - p(i,j) = p_{i+1,j} - p_{i,j} \tag{7.23}$$

Hence, Equation 7.21 changes into

$$C = \sum_{(i,j)\in\Phi} c_{i,j} \tag{7.24}$$

where Φ is the discrete image domain and $c_{i,j}$ is the cost component for the pixel at (i,j) given by:

$$\begin{aligned} c_{i,j} &= [\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]^2 \\ &+ \lambda_i[(p_{i+1,j}-p_{i,j})-(q_{i,j+1}-q_{i,j})]^2 \\ &+ \lambda_{s1}[(p_{i+1,j}-p_{i,j})^2+(p_{i,j+1}-p_{i,j})^2 \\ &+ (q_{i+1,j}-q_{i,j})^2+(q_{i,j+1}-q_{i,j})^2] \\ &+ \lambda_{s2}[(u_{i+1,j}-u_{i,j})^2+(u_{i,j+1}-u_{i,j})^2] \end{aligned} \tag{7.25}$$

To minimize the cost function C with respect to $(p_{i,j}, q_{i,j}, u_{i,j})$ at every pixel is very difficult if the image is large. Therefore, we choose to split a large image into small patches and run the optimization patch by patch. The process is illustrated in Figure 24. In the figure, the big squares in solid lines represent the image. We use 10×10 patches in our experiment. Starting from the patch (the small square in solid lines in Figure 24a) at the lower right corner of the image, a window is moved row-wise from right to left, from bottom to top. The window is moved by 5 pixels every time so that the patch in the window always has a half overlapping with any neighboring patch. As shown in Figure 24b, the second patch (the small square in solid lines) overlaps with the first patch (the small square in dash lines) at the lower right corner. When the left boundary of the

image is reached, the window is moved upward by 5 pixels and rightward to the right boundary of the image, as shown in Figure 24c. The patch (the small square in solid lines) in the current window also overlaps with the patch (the small square in dash lines) at the lower right corner. When the upper left corner is reached, the window is moved in the other direction from left to right (as shown in Figure 24d), from top to bottom. In this way, the window is moved back-and-forth between the two corners and the cost on every patch is minimized. The iteration is stopped when the norm of the changes in the gradient field between two iterations is smaller than a predefined threshold. In our experiments, the convergence is usually achieved within 6 iterations.

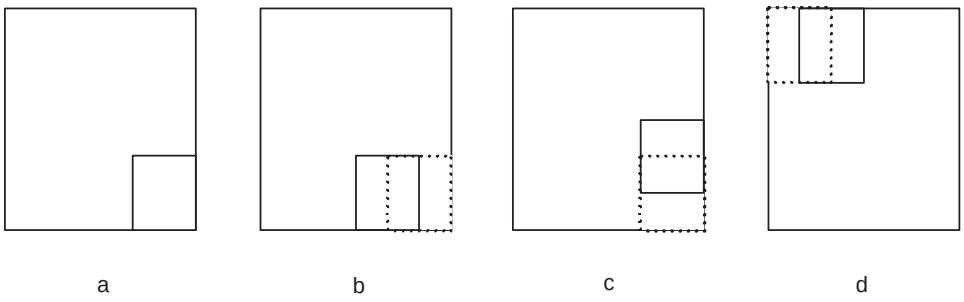

Figure 24. **Patch-by-patch minimization**

We assume that the gradients on the global boundary of an image are zero. When the window is moved from the lower right corner to the upper left corner, the right and

bottom boundary conditions of patches can be obtained from either the image boundary conditions or from results of previous patches. That allows us to impose these two boundary conditions during the optimization. As explained in Section 7.4.2, we apply forward finite difference method in this case to simplify the calculation of the gradient for the cost function. When the window is moved from the upper left corner to the lower right corner, the left and top boundary conditions of patches are available instead. In this case, these two boundary conditions are imposed and then backward finite difference method is applied.

Direct minimization of the cost function is performed on those patches using Nonlinear Polak-Ribière Conjugate Gradient method [119]. For every patch, we specify the initial vector set for the optimization as

$$\nu^{(0)} = \{\nu_{i,j}^{(0)}\} = \{(p_{i,j}^{(0)}, q_{i,j}^{(0)}, u_{i,j}^{(0)})_{(i,j)\in\Phi}\} \tag{7.26}$$

where the gradient field $(p_{i,j}^{(0)}, q_{i,j}^{(0)})_{(i,j)\in\Phi}$ is derived from the initial range image provided by HPCA (Hybrid Principal Component Analysis) and $u_{i,j}^{(0)}$ is set to zero, i.e. $(u_{i,j}^{(0)} = 0)_{(i,j)\in\Phi}$. Beginning with $\nu^{(0)}$ and an initial set of constraint factors $(\lambda_i, \lambda_{s1}, \lambda_{s2})$, the Nonlinear Polak-Ribière Conjugate Gradient method is carried out to find a minimum of the cost function and the corresponding vector set $\nu^{(1)}$. Next, the set of constraint factors are reduced by a factor of 2 and the optimization is repeated. After each iteration, the constraint factors are reduced by half. The process is stopped when the constraint factors

fall below predefined thresholds. After that, the factors are set to zero (no constraints are imposed) and a final iteration of minimization is carried out on each pixel individually. Removing constraints allows the gradient field and the composite albedo to vary more freely to account for abrupt changes on the face.

Evidently, patches on faces have different smoothness. Another advantage of our patch-by-patch method is that different initial constraint factors can be applied to handle different smoothness. The smoothness can be roughly estimated from the range ξ and the standard deviation σ of the gray scale values in a patch, which is taken from the normalized input image. The smaller ξ and σ, the smoother the patch is. Therefore, large factors are used if ξ and σ are small. On the other hand, smaller factors are used if ξ and σ are large.

The Nonlinear Polak-Ribière Conjugate Gradient method is outlined as follows [119]:

1. $g^{(0)} = s^{(0)} = -\nabla C(\nu^{(0)})$ where $\nabla C(\nu^{(0)})$ is the gradient of the cost function at the vector set $\nu^{(0)}$. Details on gradient calculation can be found in Section 7.4.2.
2. Perform line search to find a value $\gamma^{(m)}$ that minimizes $C(\nu^{(m)} + \gamma^{(m)} g^{(m)})$ using the Secant method. $\nu^{(m)}$ is the vector set at iteration $m (m = 0, 1, 2, \cdots)$
3. $\nu^{(m+1)} = \nu^{(m)} + \gamma^{(m)} g^{(m)}$
4. $s^{(m+1)} = -\nabla C(\nu^{(m+1)})$
5. $\delta^{(m+1)} = max\{\frac{(s^{(m+1)})^T (s^{(m+1)} - s^{(m)})}{(s^{(m)})^T s^{(m)}}, 0\}$
6. $g^{(m+1)} = s^{(m+1)} + \delta^{(m+1)} g^{(m)}$

The error minimization is stopped when $\|g^{(m+1)}\|$ falls below a predefined threshold.

After the gradient field is estimated, we calculate the final surface using the M-estimators algorithm [120]. The reconstructed range image is further fed into a face recognition program described in Section 7.5.

7.4.2 Cost Function Gradient Calculation

Here we demonstrate how to calculate the gradient of the cost function C in Equation 7.24 with respect to variables $p_{i,j}, q_{i,j}, u_{i,j}$ for $(i,j) \in \Phi$. We limit our discussion here to patches with the right and bottom boundary conditions imposed. Similar derivation can be developed for patches when top and left boundary conditions are imposed. Without loss of generality, we will demonstrate the calculation of the derivative with respect to $p_{i,j}$. Due to the application of forward finite difference method, there are 4 different cases for pixels in a $N \times N$ patch.

1. For pixels (i,j) where $i,j \neq 1, N$: from Equation 7.22 and Equation 7.23, we know that $p_{i,j}$ is involved in the derivative approximation at and only at these 3 locations: $(i,j), (i-1,j), (i,j-1)$. For the latter two locations, we have

$$p_y|_{(i-1,j)} = p_{i,j} - p_{i-1,j}$$

$$p_x|_{(i,j-1)} = p_{i,j} - p_{i,j-1}$$

Hence,

$$\frac{\partial C}{\partial p_{i,j}} = \frac{\partial c_{i,j}}{\partial p_{i,j}} + \frac{\partial c_{i-1,j}}{\partial p_{i,j}} + \frac{\partial c_{i,j-1}}{\partial p_{i,j}} \tag{7.27}$$

2. For the pixel $(1, 1)$:

$$\frac{\partial C}{\partial p_{1,1}} = \frac{\partial c_{1,1}}{\partial p_{1,1}} \tag{7.28}$$

3. For pixels (i, j) where $i = 1$ and $j \neq 1, N$:

$$\frac{\partial C}{\partial p_{i,j}} = \frac{\partial c_{i,j}}{\partial p_{i,j}} + \frac{\partial c_{i,j-1}}{\partial p_{i,j}} \tag{7.29}$$

4. For pixels (i, j) where $i \neq 1, N$ and $j = 1$:

$$\frac{\partial C}{\partial p_{i,j}} = \frac{\partial c_{i,j}}{\partial p_{i,j}} + \frac{\partial c_{i-1,j}}{\partial p_{i,j}} \tag{7.30}$$

Here we will explain why we use forward finite difference method when the right and bottom boundary conditions are imposed. Let us take a look at a pixel (i, j) on the left boundary of a patch. This pixel doesn't have a neighboring pixel $(i, j - 1)$ on the left. As a result, derivatives along the horizontal direction can't be calculated if backward finite difference method is applied. However, forward finite difference method doesn't pose such a problem in this case and therefore is employed.

The calculation of $\frac{\partial c_{i,j}}{\partial p_{i,j}}$, $\frac{\partial c_{i-1,j}}{\partial p_{i,j}}$, and $\frac{\partial c_{i,j-1}}{\partial p_{i,j}}$ is further demonstrated below.

1). For pixel (i,j), we calculate $\frac{\partial c_{i,j}}{\partial p_{i,j}}$ from Equation 7.25

$$\begin{aligned}
\frac{\partial c_{i,j}}{\partial p_{i,j}} &= -2[\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]\frac{u_{i,j}p_{i,j}}{\sqrt{(1+p_{i,j}^2+q_{i,j}^2)^3}} \\
&- 2\lambda_i[(p_{i+1,j}-p_{i,j})-(q_{i,j+1}-q_{i,j})] \\
&+ 2\lambda_{s1}[2p_{i,j}-p_{i,j+1}-p_{i+1,j}] \qquad (7.31) \\
\frac{\partial c_{i,j}}{\partial q_{i,j}} &= -2[\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]\frac{u_{i,j}q_{i,j}}{\sqrt{(1+p_{i,j}^2+q_{i,j}^2)^3}} \\
&+ 2\lambda_i[(p_{i+1,j}-p_{i,j})-(q_{i,j+1}-q_{i,j})] \\
&+ 2\lambda_{s1}[2q_{i,j}-q_{i,j+1}-q_{i+1,j}] \qquad (7.32) \\
\frac{\partial c_{i,j}}{\partial u_{i,j}} &= 2[\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]\frac{1}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} \\
&+ 2\lambda_{s2}[2u_{i,j}-u_{i,j+1}-u_{i+1,j}] \qquad (7.33)
\end{aligned}$$

2). For pixel $(i-1,j)$

$$\begin{aligned}
c_{i-1,j} &= [\frac{u_{i-1,j}}{\sqrt{1+p_{i-1,j}^2+q_{i-1,j}^2}} - I_{i-1,j}]^2 \\
&+ \lambda_i[(p_{i,j}-p_{i-1,j})-(q_{i-1,j+1}-q_{i-1,j})]^2 \\
&+ \lambda_{s1}[(p_{i-1,j+1}-p_{i-1,j})^2+(p_{i,j}-p_{i-1,j})^2 \\
&+ (q_{i-1,j+1}-q_{i-1,j})^2+(q_{i,j}-q_{i-1,j})^2] \\
&+ \lambda_{s2}[(u_{i-1,j+1}-u_{i-1,j})^2+(u_{i,j}-u_{i-1,j})^2] \qquad (7.34)
\end{aligned}$$

So

$$\begin{aligned}
\frac{\partial c_{i-1,j}}{\partial p_{i,j}} &= 2\lambda_i[(p_{i,j} - p_{i-1,j}) - (q_{i-1,j+1} - q_{i-1,j})] \\
&+ 2\lambda_{s1}[p_{i,j} - p_{i-1,j}] \qquad (7.35) \\
\frac{\partial c_{i-1,j}}{\partial q_{i,j}} &= 2\lambda_{s1}[q_{i,j} - q_{i-1,j}] \qquad (7.36) \\
\frac{\partial c_{i-1,j}}{\partial u_{i,j}} &= 2\lambda_{s2}[u_{i,j} - u_{i-1,j}] \qquad (7.37)
\end{aligned}$$

3). For pixel $(i, j-1)$

$$\begin{aligned}
c_{i,j-1} &= [\frac{u_{i,j-1}}{\sqrt{1 + p_{i,j-1}^2 + q_{i,j-1}^2}} - I_{i,j-1}]^2 \\
&+ \lambda_i[(p_{i+1,j-1} - p_{i,j-1}) - (q_{i,j} - q_{i,j-1})]^2 \\
&+ \lambda_{s1}[(p_{i,j} - p_{i,j-1})^2 + (p_{i+1,j-1} - p_{i,j-1})^2 \\
&+ (q_{i,j} - q_{i,j-1})^2 + (q_{i+1,j-1} - q_{i,j-1})^2] \\
&+ \lambda_{s2}[(u_{i,j} - u_{i,j-1})^2 + (u_{i+1,j-1} - u_{i,j-1})^2] \qquad (7.38)
\end{aligned}$$

So

$$\frac{\partial c_{i,j-1}}{\partial p_{i,j}} = 2\lambda_{s1}[p_{i,j} - p_{i,j-1}] \tag{7.39}$$

$$\begin{aligned}\frac{\partial c_{i,j-1}}{\partial q_{i,j}} &= -2\lambda_i[(p_{i+1,j-1} - p_{i,j-1}) - (q_{i,j} - q_{i,j-1})] \\ &+ 2\lambda_{s1}[q_{i,j} - q_{i,j-1}]\end{aligned} \tag{7.40}$$

$$\frac{\partial c_{i,j-1}}{\partial u_{i,j}} = 2\lambda_{s2}[u_{i,j} - u_{i,j-1}] \tag{7.41}$$

Therefore, by adding Equation 7.31, Equation 7.35 and Equation 7.39 together, we get

$$\begin{aligned}\frac{\partial C}{\partial p_{i,j}} &= \frac{\partial c_{i,j}}{\partial p_{i,j}} + \frac{\partial c_{i-1,j}}{\partial p_{i,j}} + \frac{\partial c_{i,j-1}}{\partial p_{i,j}} \\ &= -2[\frac{u_{i,j}}{\sqrt{1 + p_{i,j}^2 + q_{i,j}^2}} - I_{i,j}]\frac{u_{i,j}p_{i,j}}{\sqrt{(1 + p_{i,j}^2 + q_{i,j}^2)^3}} \\ &+ 2\lambda_i[-p_{i+1,j} + 2p_{i,j} + q_{i,j+1} - q_{i,j} - p_{i-1,j} \\ &- q_{i-1,j+1} + q_{i-1,j}] \\ &+ 2\lambda_{s1}[4p_{i,j} - p_{i,j+1} - p_{i+1,j} - p_{i-1,j} - p_{i,j-1}]\end{aligned} \tag{7.42}$$

Similarly, Equation 7.32 + Equation 7.36 + Equation 7.40 leads to

$$\begin{aligned}
\frac{\partial C}{\partial q_{i,j}} &= \frac{\partial c_{i,j}}{\partial q_{i,j}} + \frac{\partial c_{i-1,j}}{\partial q_{i,j}} + \frac{\partial c_{i,j-1}}{\partial q_{i,j}} \\
&= -2[\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]\frac{u_{i,j}q_{i,j}}{\sqrt{(1+p_{i,j}^2+q_{i,j}^2)^3}} \\
&+ 2\lambda_i[p_{i+1,j} - p_{i,j} - q_{i,j+1} + 2q_{i,j} - p_{i+1,j-1} \\
&+ p_{i,j-1} - q_{i,j-1}] \\
&+ 2\lambda_{s1}[4q_{i,j} - q_{i,j+1} - q_{i+1,j} - q_{i-1,j} - q_{i,j-1}] \qquad (7.43)
\end{aligned}$$

Equation 7.33 + Equation 7.37 + Equation 7.41 leads to

$$\begin{aligned}
\frac{\partial C}{\partial u_{i,j}} &= \frac{\partial c_{i,j}}{\partial u_{i,j}} + \frac{\partial c_{i-1,j}}{\partial u_{i,j}} + \frac{\partial c_{i,j-1}}{\partial u_{i,j}} \\
&= 2[\frac{u_{i,j}}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} - I_{i,j}]\frac{1}{\sqrt{1+p_{i,j}^2+q_{i,j}^2}} \\
&+ 2\lambda_{s2}[4u_{i,j} - u_{i,j+1} - u_{i+1,j} - u_{i-1,j} - u_{i,j-1}] \qquad (7.44)
\end{aligned}$$

The gradient of the cost function C now can be calculated as

$$\begin{aligned}
\nabla C(\nu) &= (\frac{\partial C}{\partial p_{1,1}}, \cdots, \frac{\partial C}{\partial p_{n,m}}, \frac{\partial C}{\partial q_{1,1}}, \cdots, \frac{\partial C}{\partial q_{n,m}}, \frac{\partial C}{\partial u_{1,1}}, \\
&\quad \cdots, \frac{\partial C}{\partial u_{n,m}})^T \qquad (7.45)
\end{aligned}$$

for a $n \times m$ image.

7.5 Recognition based on Iterative Closest Point Algorithm

The subject for the reconstructed range image is recognized by a recognition program based on the Iterative Closest Point (ICP) algorithm [121] [122]. In this algorithm, two 3D point sets are registered by iteratively transforming one set and finding the geometric transformation (translation and rotation) that minimizes the average minimal error. The average minimal error is defined here as the average of the minimal distances of all the control points in the first set with the second set. The steps in the ICP algorithm are detailed as follows:

1. Select control points from the first point set
2. To each control point in the first set, find the closest point in the second set
3. Calculate the transformation that minimizes the average distance between the two point sets
4. Transform the second point set
5. repeat step 1 to 4 until convergence is reached (i.e. the difference in the average distances of two consecutive iterations is below a predefined threshold)

To find the model that best fits the test set, we match the reconstructed test range image with every 3D model in a library. The average matching error is computed by summing for every 3D point in the test range image its minimal distance with the transformed model. The model with the lowest average matching error is considered as the matching model.

7.6 Experiments

For the test image in 7.3b, results after SFS (Shape From Shading) optimization are shown in Figure 23c and Figure 23f. Compared to the HPCA results, we can see that details on the face are improved and noise is reduced significantly. Two additional examples are illustrated in Figure 26 and Figure 28 respectively. These two examples are reconstructed from a frontal face image in Figure 25 and a profile image in Figure 27 respectively. We can observe from all these results that the reconstructions are very close to the original range images. To further measure the reconstruction errors, we calculate an error image r_e, which is the difference between the original range image r_o and the reconstructed range image r_r. The overall error is measured as follows

$$e = \frac{RMS(r_e)}{MAX(r_o)} \tag{7.46}$$

where $RMS(r_e)$ is the RMS (Root Mean Square) of the error image r_e and $MAX(r_o)$ is the maximum of the original range image r_o.

In our experiments, 40 profile images and 40 frontal face images are synthesized from 40 new 3D head models that are not included in the training. Reconstructions are performed and the errors are measured. We find out that the reconstruction errors are all

Figure 25. **A test image in the frontal view**

less than 7%, which implies that the reconstructed range images are close to the original images.

To further verify the reconstruction accuracy, we use the reconstructions in face recognition experiments, in which the reconstructed range images are tested against 3D head models, which include the 40 new models and those used in training. 73 out of 80 test images are recognized correctly. The recognition rate is 91%, which is satisfactory for such a hard task, in which 2D images are tested against 3D models. The good performance on recognition demonstrates the accuracy of our reconstruction method as well.

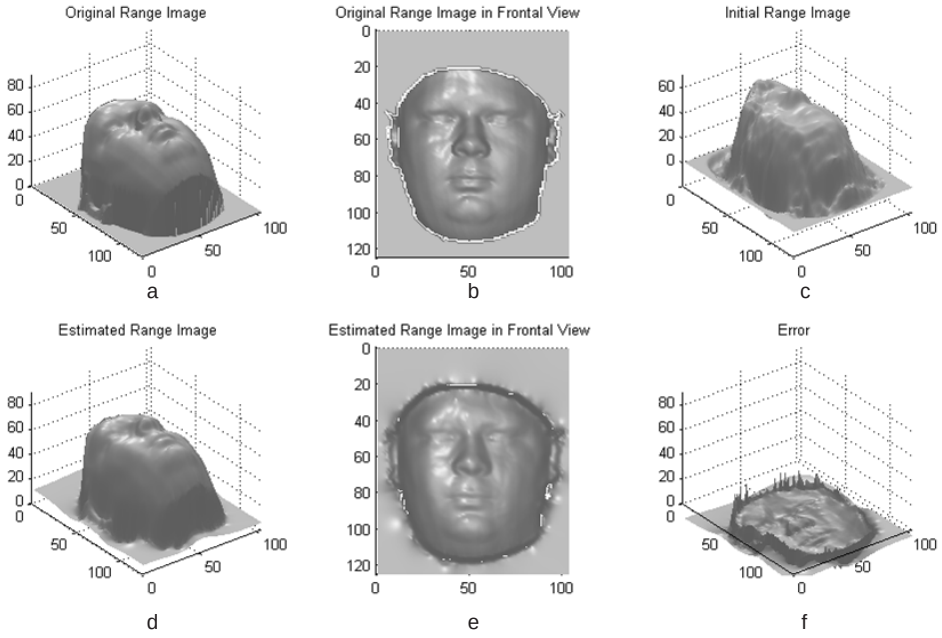

Figure 26. **(a) The original range image; (b)The original range image in the x-y view; (c) The reconstructed range image from HPCA. This reconstruction serves as the initial estimation for the optimization algorithm; (d) The SFS reconstruction; (e) The SFS reconstruction in the x-y view; (f) The reconstruction error**

Figure 27. **A test image in the side view**

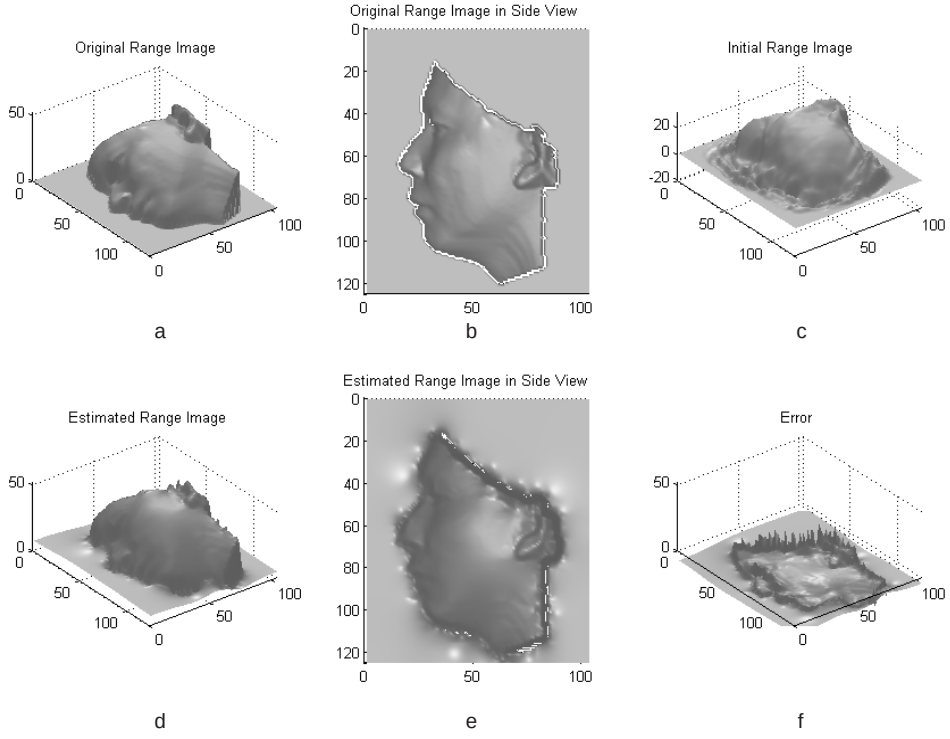

Figure 28. **(a) The original range image; (b)The original range image in the x-y view; (c) The reconstructed range image from HPCA. This reconstruction serves as the initial estimation for the optimization algorithm; (d) The SFS reconstruction; (e) The SFS reconstruction in the x-y view; (f) The reconstruction error**

CHAPTER 8

CONCLUSIONS

In this report, we first compare the novel RISq method to HMM. Both methods are used for the recognition of general vector sequences. Our comparison shows that RISq performs better than HMM in many aspects. The training of RISq requires only one example from each class and the training is much simpler than training HMM models. Also a few sparse samples from a test sequence are usually sufficient for RISq to achieve robust recognition while HMM needs the entire sequence. This makes RISq essentially much less sensitive to missing vectors. Lastly, our experiments demonstrate that RISq outperforms HMM in term of handling large sets of models or with many vectors dimensions. RISq is also better in noise robustness, computation time, selectivity ratio, etc.

For human gesture recognition, we develop a novel system to recognize different hand gestures. We use an Inertial Measurement Unit (IMU) equipped with accelerometers and gyroscopes to sense the motion of the operator's hand. The IMU is calibrated with the help of Nonlinear Data-Fitting method. Gesture trajectories are reconstructed from inertial sensor measurements using the Inertial Navigation System (INS) theory. We develop a novel method named Zero Velocity Linear Compensation (ZVLC) to improve trajectory reconstruction accuracy. Experimental results show that ZVLC provides more

accurate reconstruction than the widely used method of Zero Velocity Compensation (ZVC). At the recognition stage, the novel RISq method is employed to recognize the reconstructed gesture trajectories and achieves a recognition rate of 92%.

In the third part of this thesis, we describe a novel method for 3D head reconstruction and view-invariant recognition, which is based on Shape From Shading (SFS) combined with Hybrid Principal Component Analysis (HPCA). Our novel HPCA algorithm provides good initial estimates of 3D range mapping for the SFS optimization and yields much improved 3D head reconstruction. Additional contribution of our chapter is the successful handling of variable and unknown surface albedo in SFS. Experimental results show that our HPCA based SFS method provides accurate 3D head reconstructions and high recognition rates. Our work could have many practical applications such as person recognition from side views when only frontal views are available for modeling.

CITED LITERATURE

1. Ben-Arie, J., Wang, Z., Pandit, P., and Rajaram, S.: Human activity recognition using multidimensional indexing. IEEE Trans. Pattern Anal. Mach. Intell., 24(8):1091–1104, 2002.

2. Ben-Arie, J.: Method of recognition of human motion, vector sequences and speech. United States Patent 7366645, April 2008.

3. Gabayan, K. and Lansel, S.: Programming-by-example gesture recognition. Website, 2006. `http://www.stanford.edu/~slansel/projects/MachineLearningPaper.pdf`.

4. Franzini, S. and Ben-Arie, J.: Speech recognition by indexing and sequencing. In Soft Computing and Pattern Recognition, 2010. Proceedings. International Conference on, pages 93–98, Dec. 2010.

5. Rabiner, L. R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Readings in speech recognition, pages 267–296, 1990.

6. Ben-Arie, J., Wang, Z., Pandit, P., and Rajaram, S.: Human activity recognition using multidimensional indexing. IEEE Trans. Pattern Anal. Mach. Intell., 24(8):1091–1104, 2002.

7. Wu, Q. and Ben-Arie, J.: View invariant head recognition by Hybrid PCA based reconstruction. Journal of Integrated Computer-Aided Engineering, 15(2):97–108, 2008.

8. Wu, Q. and Ben-Arie, J.: View invariant head recognition by Hybrid PCA based reconstruction. In Analysis and Modeling of Faces and Gestures, volume 4778 of Lecture Notes in Computer Science, pages 46–57. Springer Berlin / Heidelberg, 2007.

9. Wu, Q. and Ben-Arie, J.: Hybrid PCA based shape from shading for 3D head reconstruction. In the proceedings of 2007 IEEE International Conference on Electro/Information Technology, pages 407 –411, May. 2007.

CITED LITERATURE (Continued)

10. Wu, Q. and Ben-Arie, J.: Reconstruction and recognition of 3D heads employing hybrid principal component analysis. In the Proceedings of 2006 IASTED Conference on Visualization, Imaging, and Image Processing, August 2006.

11. Benzeghiba, M., Mori, R. D., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., Fissore, L., Laface, P., Mertins, A., Ris, C., Rose, R., Tyagi, V., and Wellekens, C.: Automatic speech recognition and speech variability: A review. Speech Commun., 49(10-11):763–786, 2007.

12. Liu, J., Wang, Z., and Xiao, X.: A hybrid SVM/DDBHMM decision fusion modeling for robust continuous digital speech recognition. Pattern Recogn. Lett., 28(8):912–920, 2007.

13. Pisarn, C. and Theeramunkong, T.: An HMM-based method for Thai spelling speech recognition. Comput. Math. Appl., 54(1):76–95, 2007.

14. Yu, L. and Wu, L.: Comments on "A Separable Low Complexity 2D HMM with Application to Face Recognition". IEEE Trans. Pattern Anal. Mach. Intell., 29(2):368, 2007.

15. Bicego, M. and Murino, V.: Investigating Hidden Markov Models' capabilities in 2D shape classification. IEEE Trans. Pattern Anal. Mach. Intell., 26(2):281–286, 2004.

16. Ivanov, Y. A. and Bobick, A. F.: Recognition of visual activities and interactions by stochastic parsing. IEEE Trans. Pattern Anal. Mach. Intell., 22(8):852–872, 2000.

17. Horowitz, B. and Pentland, A.: Recovery of non-rigid motion and structure. IEEE Trans. Pattern Anal. Mach. Intell., 6(3):325–330, 1991.

18. Caelli, T. and McCane, B.: Component analysis of Hidden Markov Models in computer vision. In ICIAP '03: Proceedings of the 12th International Conference on Image Analysis and Processing, page 510, Washington, DC, USA, 2003. IEEE Computer Society.

19. Krogh, A., Larsson, B., von Heijne, G., and Sonnhammer, E.: Predicting transmembrane protein topology with a Hidden Markov Model: application to complete genomes, 2001.

CITED LITERATURE (Continued)

20. LaFramboise, T., Hayes, D., and Tengs, T.: Statistical analysis of genomic tag data. Statistical Applications in Genetics and Molecular Biology, 3(1):34, 2007. available at http://ideas.repec.org/a/bep/sagmbi/32007134.html.

21. Maddimsetty, R. P., Buhler, J., Chamberlain, R. D., Franklin, M. A., and Harris, B.: Accelerator design for protein sequence HMM search. In ICS '06: Proceedings of the 20th annual international conference on Supercomputing, pages 288–296, New York, NY, USA, 2006. ACM.

22. Hassan, M. R., Nath, B., and Kirley, M.: A fusion model of HMM, ANN and GA for stock market forecasting. Expert Syst. Appl., 33(1):171–180, 2007.

23. Kadous, M. W.: Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. Doctoral dissertation, University of New South Wales, 2002.

24. Young, S., Odell, J., Ollason, D., Valtchev, V., and Woodland, P.: HTK Book. Website, 1997. `http://labrosa.ee.columbia.edu/doc/HTKBook21/HTKBook.html`.

25. DeMenthon, D. and Vuilleumier, M.: Lamp_hmm v. 0.9 (c++). Website, 2003. `http://www.cfar.umd.edu/~daniel/Site_2/Code.html`.

26. Zefran, M., Ben-Arie, J., Eugenio, B. D., and Foreman, M. D.: Effective communication with robotic assistants for the elderly: Integrating speech, vision and haptics. NSF Grant #0905593, 2009.

27. Fong, T., Nourbakhsh, I., and Dautenhahn, K.: A survey of socially interactive robots. Robotics and Autonomous Systems, 42(3-4):143 – 166, 2003.

28. Liang, X., Zhang, S., Zhang, X., and Geng, W.: Motion-based perceptual user interface. In Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on, volume 1, pages 247 –251, 2009.

29. Jang, I. and Park, W.: Signal processing of the accelerometer for gesture awareness on handheld devices. In Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on, 2003.

CITED LITERATURE (Continued)

30. Pickering, C. A., Burnham, K. J., and Richardson, M. J.: A research study of hand gesture recognition technologies and applications for human vehicle interaction. In Automotive Electronics, 2007 3rd Institution of Engineering and Technology Conference on, pages 1 –15, 2007.

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

32. Yoon, H.-S., Soh, J., Bae, Y. J., and Yang, H. S.: Hand gesture recognition using combined features of location, angle and velocity. Pattern Recognition, 34(7):1491 – 1501, 2001.

33. Bauer, B. and Karl-Friedrich, K.: Towards an automatic sign language recognition system using subunits. In Gesture and Sign Language in Human-Computer Interaction, eds. I. Wachsmuth and T. Sowa, volume 2298 of Lecture Notes in Computer Science, pages 123–173. Springer Berlin / Heidelberg, 2002.

34. Ramamoorthy, A., Vaswani, N., Chaudhury, S., and Banerjee, S.: Recognition of dynamic hand gestures. Pattern Recognition, 36(09):2069–2081, 2003.

35. Kelly, D., McDonald, J., and Markham, C.: Continuous recognition of motion based gestures in sign language. In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pages 1073 – 1080, sep. 2009.

36. Liu, N., Lovell, B., Kootsookos, P., and Davis, R.: Model structure selection training algorithms for an HMM gesture recognition system. In Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on, pages 100 – 105, oct. 2004.

37. Elmezain, M., Al-Hamadi, A., and Michaelis, B.: Hand trajectory-based gesture spotting and recognition using HMM. In Image Processing (ICIP), 2009 16th IEEE International Conference on, pages 3577 –3580, nov. 2009.

38. Elmezain, M., Al-Hamadi, A., Appenrodt, J., and Michaelis, B.: A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, pages 1 –4, dec. 2008.

CITED LITERATURE (Continued)

39. Elmezain, M., Al-Hamadi, A., and Michaelis, B.: A robust method for hand gesture segmentation and recognition using forward spotting scheme in conditional random fields. In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 3850 –3853, aug. 2010.

40. Yamato, J., Ohya, J., and Ishii, K.: Recognizing human action in time-sequential images using Hidden Markov Model. In Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on, pages 379 –385, jun. 1992.

41. Rajko, S. and Qian, G.: HMM parameter reduction for practical gesture recognition. In Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on, pages 1 –6, sep. 2008.

42. Holte, M. and Moeslund, T.: View invariant gesture recognition using 3D motion primitives. In Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on, 31 2008.

43. Rajko, S., Qian, G., Ingalls, T., and James, J.: Real-time gesture recognition with minimal training requirements and on-line learning. In Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, pages 1 – 8, jun. 2007.

44. Wong, S.-F. and Cipolla, R.: Continuous gesture recognition using a sparse bayesian classifier. In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 0 2006.

45. Bhuyan, M., Ghosh, D., and Bora, P.: Feature extraction from 2D gesture trajectory in dynamic hand gesture recognition. In Cybernetics and Intelligent Systems, 2006 IEEE Conference on, pages 1 –6, 2006.

46. Suk, H.-I., Sin, B.-K., and Lee, S.-W.: Robust modeling and recognition of hand gestures with dynamic Bayesian network. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, pages 1 –4, dec. 2008.

47. Davis, J. and Shah, M.: Visual gesture recognition. In Vision, Image and Signal Processing, IEE Proceedings -, volume 141, pages 101 –106, apr. 1994.

CITED LITERATURE (Continued)

48. Yeasin, M. and Chaudhuri, S.: Visual understanding of dynamic hand gestures. Pattern Recognition, 33(11):1805 – 1817, 2000.

49. Hong, P., Turk, M., and Huang, T.: Gesture modeling and recognition using finite state machines. In Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on, pages 410 – 415, 2000.

50. Black, M. and Jepson, A.: Recognizing temporal trajectories using the condensation algorithm. In Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, pages 16 –21, apr. 1998.

51. Wang, S. B., Quattoni, A., Morency, L.-P., Demirdjian, D., and Darrell, T.: Hidden conditional random fields for gesture recognition. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, 2006.

52. Morency, L.-P., Quattoni, A., and Darrell, T.: Latent-dynamic discriminative models for continuous gesture recognition. In Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, pages 1 –8, 2007.

53. Dadgostar, F., Sarrafzadeh, A., Fan, C., Silva, L. D., and Messom, C.: Modeling and recognition of gesture signals in 2D space: A comparison of NN and SVM approaches. In Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on, pages 701 –704, 2006.

54. Li, H. and Greenspan, M.: Segmentation and recognition of continuous gestures. In Image Processing, 2007. ICIP 2007. IEEE International Conference on, 16 2007.

55. Roh, M.-C., Shin, H.-K., Lee, S.-W., and Lee, S.-W.: Volume motion template for view-invariant gesture recognition. In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, 0 2006.

56. Yabukami, S., Kikuchi, H., Yamaguchi, M., Arai, K., Takahashi, K., Itagaki, A., and Wako, N.: Motion capture system of magnetic markers using three-axial magnetic field sensor. Magnetics, IEEE Transactions on, 36(5):3646 –3648, sep. 2000.

CITED LITERATURE (Continued)

57. Mitobe, K., Kaiga, T., Yukawa, T., Miura, T., Tamamoto, H., Rodgers, A., and Yoshimura, N.: Development of a motion capture system for a hand using a magnetic three dimensional position sensor. In SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters, page 102, New York, NY, USA, 2006. ACM.

58. Hashi, S., Toyoda, M., Yabukami, S., Ishiyama, K., Okazaki, Y., and Arai, K.: Wireless magnetic motion capture system for multi-marker detection. Magnetics, IEEE Transactions on, 42(10):3279 –3281, oct. 2006.

59. Ma, Y., Jia, W., Li, C., Yang, J., Mao, Z.-H., and Sun, M.: Magnetic hand motion tracking system for human-machine interaction. Electronics Letters, 46(9):621 –623, apr. 2010.

60. Kevin, N., Ranganath, S., and Ghosh, D.: Trajectory modeling in gesture recognition using CyberGloves and magnetic trackers. In TENCON 2004. 2004 IEEE Region 10 Conference, volume A, pages 571 – 574 Vol. 1, nov. 2004.

61. Corradini, A. and Cohen, P.: Multimodal speech-gesture interface for handfree painting on a virtual paper using partial recurrent neural networks as gesture recognizer. In Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on, volume 3, pages 2293 –2298, 2002.

62. Pirkl, G., Stockinger, K., Kunze, K., and Lukowicz, P.: Adapting magnetic resonant coupling based relative positioning technology for wearable activitiy recogniton. In Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on, pages 47 –54, sep. 2008.

63. Bobick, A. and Wilson, A.: A state-based approach to the representation and recognition of gesture. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(12):1325 –1337, dec. 1997.

64. Kim, J.-H., Thang, N. D., and Kim, T.-S.: 3-D hand motion tracking and gesture recognition using a data glove. In Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on, pages 1013 –1018, 2009.

65. Sakaguchi, T.; Kanamori, T. K. H. S. K. I. S.: Human motion capture by integrating gyroscopes and accelerometers. In Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on, pages 470–475, 1996.

CITED LITERATURE (Continued)

66. Heinz, E. A., Kunze, K., Gruber, M., Bannach, D., and Lukowicz, P.: Using wearable sensors for real-time recognition tasks in games of martial arts - an initial experiment. In in Proceedings of the 2nd IEEE Symposium on Computational Intelligence and Games (CIG), pages 98–102. IEEE Press.

67. Benbasat, A. Y. and Paradiso, J. A.: An inertial measurement framework for gesture recognition and applications. In GW '01: Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction, pages 9–20, London, UK, 2002. Springer-Verlag.

68. Sama, M., Pacella, V., Farella, E., Benini, L., and Riccó, B.: 3dID: a low-power, low-cost hand motion capture device. In DATE '06: Proceedings of the conference on Design, automation and test in Europe, pages 136–141, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

69. Reifinger, S. Laquai, F. R. G.: Translation and rotation of virtual objects in augmented reality: A comparison of interaction devices. In Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on, pages 2448 – 2453, 2008.

70. Pandit, A., Dand, D., Mehta, S., Sabesan, S., and Daftery, A.: A simple wearable hand gesture recognition device using iMEMS. Soft Computing and Pattern Recognition, International Conference of, 0:592–597, 2009.

71. Bang, W.-C., Chang, W., Kang, K.-H., Choi, E.-S., Potanin, A., and Kim, D.-Y.: Self-contained spatial input device for wearable computers. In ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers, page 26, Washington, DC, USA, 2003. IEEE Computer Society.

72. Zhang, S., Yuan, C., and Zhang, Y.: Self-defined gesture recognition on keyless handheld devices using MEMS 3D accelerometer. In Natural Computation, 2008. ICNC '08. Fourth International Conference on, volume 4, pages 237 –241, 2008.

73. Kallio, S., Kela, J., and Mantyjarvi, J.: Online gesture recognition system for mobile interaction. In Systems, Man and Cybernetics, 2003. IEEE International Conference on, volume 3, pages 2070 – 2076 vol.3, 2003.

CITED LITERATURE (Continued)

74. Mirabella, O., Brischetto, M., and Mastroeni, G.: MEMS based gesture recognition. In Human System Interactions (HSI), 2010 3rd Conference on, pages 599 – 604, May 2010.

75. Dong, Z., Wejinya, U., Zhou, S., Shan, Q., and Li, W.: Real-time written-character recognition using MEMS motion sensors: Calibration and experimental results. In Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on, pages 687 –691, 2009.

76. Akl, A. and Valaee, S.: Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, and compressive sensing. In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, pages 2270 –2273, 2010.

77. Liu, A., Yang, J., and Boda, P.: Poster abstract: Gesture recognition via continuous maximum entropy training on accelerometer data. In Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on, pages 371 –372, 2009.

78. Analyst: What downturn for consumer, wireless MEMS? Website, 2009. http://www.electroiq.com/index/display/nanotech-article-display/366410/articles/small-times/electronics/main-stories/2009/07/analyst-what-downturn-for-consumer-wireless-mems.html.

79. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., and Vasudevan, V.: uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive Computing and Communications, IEEE International Conference on, 0:1–9, 2009.

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

81. Mlch, J.: Wiimote gesture recognition. In Proceedings of the 15th Conference and Competition STUDENT EEICT 2009 Volume 4, pages 344–349. Faculty of Electrical Engineering and Communication BUT, 2009.

82. Schreiber, M., Wilamowitz-Moellendorff, M., and Bruder, R.: New interaction concepts by using the Wii remote. In Proceedings of the 13th

CITED LITERATURE (Continued)

International Conference on Human-Computer Interaction. Part II, pages 261–270, Berlin, Heidelberg, 2009. Springer-Verlag.

83. VARCHOLIK, P., M. J.: Gestural communication with accelerometer-based input devices and tactile displays. In Proceedings of the 26th Army Science Conference, 2008.

84. Lorente-Leal, A., Ferna andndez Rodrigues, J., and Montero, J.: Development of a Wiimote-based gesture recognizer in a microprocessor laboratory course. In Education Engineering (EDUCON), 2010 IEEE, pages 451 –455, 2010.

85. Bascetincelik, A.: WiiRobot: Controlling robots with Wii gestures. Brown University, Masters Thesis, 2009.

86. Agrawal, S., Constandache, I., Gaonkar, S., and Choudhury, R. R.: PhonePoint pen: using mobile phones to write in air. In MobiHeld '09: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds, pages 1–6, New York, NY, USA, 2009. ACM.

87. Kauppila M, Inkeroinen T, P. S. J.: Mobile phone controller based on accelerative gesturing. In Adjunct Proceedings Pervasive, pages 130–133, 2008.

88. He, Z., Jin, L., Zhen, L., and Huang, J.: Gesture recognition based on 3D accelerometer for cell phones interaction. In Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on, 30 2008.

89. Choi, E.-S., Bang, W.-C., Cho, S.-J., Yang, J., Kim, D.-Y., and Kim, S.-R.: Beatbox music phone: gesture-based interactive mobile phone using a tri-axis accelerometer. In Industrial Technology, 2005. ICIT 2005. IEEE International Conference on, pages 97 –102, 2005.

90. Inertial measurement unit. Website, 2010. http://en.wikipedia.org/wiki/Inertial_Measurement_Unit.

91. CHRobotics: CHR-6d inertial measurement unit. Website, 2010. http://www.chrobotics.com/CHR6d.php.

92. CHRobotics: CHR-6d datasheet. Website, 2010. http://www.chrobotics.com/docs/chr6d_datasheet.pdf.

CITED LITERATURE (Continued)

93. Jitendra R. Raol, Gopalrathnam Girija, J. S. J. S.: Modelling and parameter estimation of dynamic systems. Institution of Engineering and Technology, 2004.

94. Dissanayake, G., Sukkarieh, S., Nebot, E., and Durrant-Whyte, H.: The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. Robotics and Automation, IEEE Transactions on, 17(5):731–747, 2001.

95. Inertial navigation system. Website, 2010. http://en.wikipedia.org/wiki/Inertial_navigation_system.

96. ACME Worldwide Enterprises, I.: Fixed wing true Q dynamic motion seats for simulators. Website, 2010. http://www.acme-worldwide.com/dynamic_motion_seat_Fixed.htm.

97. Mahoney, M.: Pointing an instrument on an airborne platform. Website, 2010. http://mtp.jpl.nasa.gov/notes/pointing/pointing.html.

98. Tuck, K.: Tilt sensing using linear accelerometers. In Freescale Semiconductors Inc. application note AN3461, 2007.

99. AnologDevices: ADXL335 accelerometer datasheet. Website, 2010. http://www.analog.com/static/imported-files/data_sheets/ADXL335.pdf.

100. STMicroelectronics: LPR510AL/LY510ALH gyroscope datasheet. Website, 2009. http://www.st.com/stonline/products/literature/ds/15811/lpr510al.pdf.

101. Skaloud, J. and Schwarz, K.-P.: Application of Inertial Technology to Underground Positioning: The Soudan Mine Shaft Survey. Zeitschrift fr Vermessungswesen, 8:292–299, 2000.

102. Huddle, J. R.: Trends in inertial systems technology for high accuracy AUV navigation. In Workshop Autonomous Underwater Vehicle, 1998.

103. Frank, M.: Position refinement algorithm. U.S. Patent 6292 751, 2001.

CITED LITERATURE (Continued)

104. Blanz, V. and Vetter, T.: A morphable model for the synthesis of 3D faces. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 187–194, 1999.

105. Blanz, V. and Vetter, T.: Face recognition based on fitting a 3D Morphable Model. IEEE Trans. Pattern Anal. Mach. Intell., 25(9):1063–1074, 2003.

106. Jiang, D., Hu, Y., Yan, S., Zhang, L., Zhang, H., and Gao, W.: Efficient 3D reconstruction for face recognition. Pattern Recognition, 38(6):787–798, 2005.

107. Hu, Y., Zheng, Y., and Wang, Z.: Reconstruction of 3D face from a single 2D image for face recognition. In Proceedings 2nd Joint IEEE International Workshop on VS-PETS, pages 217–222, 2005.

108. Smith, W. A. P. and Hancock, E. R.: Recovering facial shape using a statistical model of surface normal direction. IEEE Trans. Pattern Anal. Mach. Intell., 28(12):1914–1930, 2006.

109. Ikeuchi, K. and Horn, B. K. P.: Numerical shape from shading and occluding boundaries. Shape from shading, pages 245–299, 1989.

110. Horn, B. K. P. and Brooks, M. J.: The variational approach to shape from shading. Comput. Vision Graph. Image Process., 33(2):174–208, 1986.

111. Zheng, Q. and Chellappa, R.: Estimation of illuminant direction, albedo, and shape from shading. IEEE Trans. Pattern Anal. Mach. Intell., 13(7):680–702, 1991.

112. Kimmel, R. and Bruckstein, A. M.: Tracking level sets by level sets: a method for solving the shape from shading problem. CVGIP: Computer Vision and Image Understanding, 62(1):47–58, 1995.

113. Worthington, P. L. and Hancock, E. R.: New constraints on data-closeness and needle map consistency for shape-from-shading. IEEE Trans. Pattern Anal. Mach. Intell., 21(12):1250–1267, 1999.

114. Samaras, D. and Metaxas, D.: Incorporating illumination constraints in deformable models for shape from shading and light direction estimation. IEEE Trans. Pattern Anal. Mach. Intell., 25(2):247–264, 2003.

CITED LITERATURE (Continued)

115. Crouzil, A., Descombes, X., and Durou, J.-D.: A multiresolution approach for shape from shading coupling deterministic and stochastic optimization. IEEE Trans. Pattern Anal. Mach. Intell., 25(11):1416–1421, 2003.

116. Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M.: Shape from shading: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(8):690–706, 1999.

117. USF HumanID Face Database. University of South Florida, Tampa, FL, USA.

118. Szeliski, R.: Fast shape from shading. CVGIP: Image Underst., 53(2):129–153, 1991.

119. Shewchuk, J. R.: An introduction to the conjugate gradient method without the agonizing pain. Carnegie Mellon University, Pittsburgh, PA, 1994.

120. Agrawal, A., Raskar, R., and Chellappa, R.: What is the range of surface reconstructions from a gradient field. In Proceedings of the 9th European Conference on Computer Vision, pages 578–591, Graz, Austria, 2006.

121. Besl, P. and McKay, N.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell., 14(2):239–256, February 1992.

122. Chen, Y. and Medioni, G.: Object modeling by registration of multiple range images. In CRA91, pages 2724–2729, 1991.

VITA

Name:	Qingquan Wu
Education:	B.S., Electrical Engineering, Tsinghua University, Beijing, China, May 2000
	M.S., Electrical Engineering, Tsinghua University, Beijing, China, May 2002
	PhD, Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL, Dec 2012
Experience:	University of Illinois at Chicago, Teaching Assistant & Research Assistant, 2002 to 2007
Publications:	Wu, Q. and Ben-Arie, J.: Reconstruction and recognition of 3D heads employing hybrid principal component analysis. In the Proceedings of 2006 IASTED Conference on Visualization, Imaging, and Image Processing, August 2006.
	Wu, Q. and Ben-Arie, J.: Hybrid PCA based shape from shading for 3D head reconstruction. In the proceedings of 2007 IEEE International Conference on Electro/Information Technology, pages 407 –411, May. 2007.
	Wu, Q. and Ben-Arie, J.: View invariant head recognition by Hybrid PCA based reconstruction. In Analysis and Modeling of Faces and Gestures, volume 4778 of Lecture Notes in Computer Science, pages 46–57. Springer Berlin / Heidelberg, 2007.
	Wu, Q. and Ben-Arie, J.: View invariant head recognition by Hybrid PCA based reconstruction. Journal of Integrated Computer-Aided Engineering, 15(2):97–108, 2008.