ISOMAP TRACKING WITH PARTICLE FILTER

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Nikhil Rane
May 2007

Accepted by:
Dr. Stanley Birchfield, Committee Chair
Dr. Adam Hoover
Dr. Ian Walker

ABSTRACT

The problem of tracking an object in an image sequence involves challenges like translation, in-plane and out-of-plane rotations, scaling, variations in ambient light and occlusions. A model of an object to be tracked is built off-line by making a training set with images of the object with different poses. A dimensionality reduction technique is used to capture the variations in the training images of the object. This gives a low-dimensional representation of high-dimensional data. Isometric feature mapping, also known as Isomap, is the unsupervised nonlinear dimensionality reduction technique used to capture the true degrees of freedom in high-dimensional data. Once the training data is reduced to low-dimensions it forms a part of the state-vector of the object to be tracked. Tracking is done in a stochastic recursive Bayesian framework. Particle filters, which are based on the recursive Bayesian framework, track the state of the object in presence of nonlinearity and non-Gaussianity. The focus of this thesis is the problem of tracking a person's head and also estimating its pose in each frame using Isomap for dimensionality reduction and particle filter for tracking. 'Isomap tracking with particle filter' algorithm is capable of handling rapid translation and out-of-plane rotation of a person's head with a relatively small amount of training data. The performance of the tracker is demonstrated on an image sequence with a person's head undergoing translation and out-of-plane rotation.

DEDICATION

I dedicate this work to my family who have always provided great support to me.

Acknowledgments

I am indebted to my advisor, Dr. Stanley Birchfield, for his never-ending support and guidance throughout my Masters. I also thank Dr. Ian Walker and Dr. Adam Hoover for being a part of my thesis committee and for their helpful suggestions. I would also like to thank my Graduate Coordinator, Ms. Bradley, for all her help.

TABLE OF CONTENTS

Page

TITLE PAGE i

ABSTRACT iii

DEDICATION v

ACKNOWLEDGMENTS vii

LIST OF FIGURES xi

CHAPTER

1 INTRODUCTION 1

2 RELATED WORK 5

3 DIMENSIONALITY REDUCTION 7

3.1 Principal Components Analysis (PCA) 8

3.1.1 PCA algorithm 8

3.1.2 Performance of PCA on face-data 10

3.2 Multi-dimensional Scaling (MDS) 12

3.2.1 MDS algorithm 12

3.2.2 Finding dimensionality recovered by MDS 13

3.2.3 Performance of MDS on face-data 13

3.3 Isometric Feature Mapping (Isomap) 14

3.3.1 Isomap algorithm 14

3.3.2 Finding Isomap dimensionality 17

3.3.3 Performance of Isomap on face-data 17

3.4 Locally Linear Embedding (LLE) 17

3.4.1 LLE algorithm 18

3.4.2 Finding LLE dimensionality 21

3.4.3 Performance of LLE on face-data 21

Table of Contents (Continued)

Page

3.5 Comparing PCA, MDS, Isomap and LLE 23

4 TRACKING IN A RECURSIVE BAYESIAN FRAMEWORK . . . 27

4.1 Bayesian Framework 28
4.2 Kalman filter 29
4.2.1 Discrete Kalman filter algorithm 29
4.3 Extended Kalman filter 30
4.3.1 Extended Kalman filter algorithm 31
4.4 Markov Chain Monte Carlo simulations 32
4.4.1 Markov Chain 32
4.4.2 Monte Carlo simulation 33
4.4.3 Importance Sampling for Monte Carlo simulations 33
4.5 Particle filtering 34
4.5.1 Sequential Importance Sampling (SIS) 34
4.5.2 Degeneracy problem of SIS algorithm 36

5 ISOMAP TRACKING WITH PARTICLE FILTER 39

5.1 Training data 40
5.2 Isomap of the training data 41
5.3 Particle filter algorithm 42
5.3.1 Condensation algorithm 42
5.3.2 Design of the particle filter 46

6 EXPERIMENTAL RESULTS 51

6.1 Effects of resampling 51
6.2 Tracking results 54

7 CONCLUSION 57

BIBLIOGRAPHY 59

LIST OF FIGURES

Figure Page

3.1 Sample images from face-data used for comparing the performance of PCA, MDS, LLE, Isomap. 8

3.2 The four largest principal components recovered by PCA. 10

3.3 Plot of top six eigenvalues of the covariance matrix vs. dimensionality recovered by PCA. 11

3.4 (a) Original image. (b) Image recovered by PCA using the equation $x_D = Ex_K$ with 5 eigenvectors. 11

3.5 Plot of residual variance vs. dimensionality recovered by MDS . . . 14

3.6 2D plots of low-dimensional points given by MDS with the corresponding high-dimensional images superimposed. 15

3.7 Plot of residual variance vs. dimensionality recovered by Isomap. . 18

3.8 2D plots of low-dimensional points given by Isomap with the corresponding high-dimensional images superimposed. 19

3.9 Plots of e^d/K where d is the dimensionality and K is the number of neighbors for four random points. It can be seen that K does not scale with ϵ^d for any of the points and therefore it is difficult to determine the dimensionality using the property $K_\epsilon \propto \epsilon^d$. 22

3.10 Plot of eigenvalues of the cost matrix M vs. dimensionality. The plot does not show a sharp enough elbow to determine the dimensionality of data using the eigenvalue method. 23

3.11 2D plots of low-dimensional points given by LLE with the corresponding high-dimensional images superimposed. 24

4.1 Performance of a Kalman tracker tracking an object moving in a trajectory $x = 3t$ with random accelerations of standard deviation = 0.5, where x is the position and t is the time. The measurement noise had a standard deviation of 10. 31

5.1 Sample images from the training set of a person's head used to train the tracker. 41

List of Figures (Continued)

Figure Page

5.2 (a) Plot of the residual variance vs. dimensionality for the training data. It can be seen that the curve elbows at dimensionality $d = 3$. (b) The low-dimensional structure recovered by Isomap from the high-dimensional data. It resembles a tetrahedron. 43

5.3 (a) and (b) show two different two-dimensional views of the low-dimensional structure with training templates, corresponding to some points, superimposed to help visualize how the high-dimensional 151×151 points are embedded in 3D. 44

5.4 Plot of the residual variance vs. dimensionality for the training data with head moving from left to right and down. The images with head moving up were removed from the training data. 45

5.5 (a) Image from the test sequence. (b) The image patch associated with a particle. (c) The template closest to the particle. (d) Image patch with the mask of the training template (background pixels = 0) applied to it. 49

5.6 (a) Training template after Canny edge detection. (b) Test image after Canny edge detection. (c) The chamfer transform of (b). (d) Image patch from (c) corresponding to a particle. The chamfer weight is computed by finding the distance between (a) and (d). 49

5.7 Block diagram of 'Isomap tracking with particle filter' algorithm for a single particle. 50

6.1 In the images, the blue dots show the particles. These images show the behavior of the particles in absence of resampling. 52

6.2 In the images, the blue dots show the particles. These images show the behavior of the particles when resampling is used. 53

6.3 The blue square is the tracker. The vector inside the circle at the top-left corner gives the 2D out-of-plane rotation. In the circle, the vertical axis represents the up-down pose and horizontal axis represents right-left pose. The length of the vector is proportional to the amount of rotation and the angle gives the pose. The top-right corner of the images show the closest training template. The tracking results are continued in the next Figure. 55

6.4 Tracking results continued from the previous Figure. 56

Chapter 1

INTRODUCTION

In computer vision, locating an object in every frame of an image sequence is a challenge. Two techniques exist to locate an object in every image frame: detecting the object afresh in every frame without using past information, or detecting the object in the first frame and then tracking it in subsequent frames. Tracking uses the information obtained about an object in previous frames to predict the state of the object in future frames whereas detection just locates the object in every frame independent of the information about the object obtained in previous frames. Under the assumption that the state of an object does not change drastically from one frame to the other tracking is more efficient and less prone to distraction.

Tracking and estimating a person's head pose in a 2D image sequence involves tracking an object with seven degrees of freedom (three rotational degrees of freedom, two translational degrees of freedom in the image, scaling and variation in the lighting conditions). The problem becomes harder when occlusion is involved. Tracking can be done in a deterministic or a stochastic framework. In a deterministic framework the tracking equations do not contain any random element. Tracking in a stochastic framework is necessary because of noise due to unpredictable movement.

Detecting and tracking a person's head can be done by taking cues from the shape of the head, color of the skin, texture of the face, features (e.g., eyes, nose, ears, mouth) or by simply matching face templates. In this work the face is detected and tracked using a combination of template matching and edge information of a person's head in a stochastic framework. A training set of images of a person's head to be tracked is built off-line. The training templates are high-dimensional images. In order to use these templates in a stochastic framework the dimensionality of these training templates needs to be reduced. Isometric feature mapping or Isomap is the technique used to discover the variation or the true degrees of freedom in the training set. Isomap computes the geodesic distances between high-dimensional points and gives a low-dimensional structure such that the geodesic distances between the high-dimensional points are preserved in the low-dimensional structure. Isomap of the training data is a low-dimensional representation of the images. Once an Isomap is built from the training data a state-vector x_t can be defined for the object (head) using these low-dimensions as its parameters. Tracking the head in an image sequence is then done with a particle filter. A particle filter is based on the recursive Bayesian framework. It is basically a cloud of n particles x_t^n with weights w_t^n associated with them. These weighted particles provide an estimate of the probability distribution of the state of the object (head) at each frame. The generic particle filter consists of two stages: 1) time-update and 2) measurement-update. The time-update stage predicts the state of the system for time t using the probability density function of the state obtained at $t-1$ and a system model which gives the evolution of state of the particles in time. The measurement-update stage corrects the prediction of the state obtained from time-update stage based on the measurement received at time t using a measurement model which relates the measurement to the state of the object. The system model has a deterministic component (which predicts the state transition for a particle) and a stochastic component (which accounts

for the process noise). The measurement model has a stochastic component which accounts for the noisy measurements. The weights for all particles are updated after every measurement at time t and these weights then give the probability density function for the state of the system at time t.

The rest of the text is organized as follows. Chapter 2 describes related work in tracking and dimensionality reduction. In Chapter 3, PCA, MDS, ISOMAP and LLE algorithms for dimensionality reduction are discussed in detail and their performance is compared on a data set of training images. In Chapter 4 Kalman filter, extended Kalman filter and particle filter (SIS algorithm) are explained. Chapter 5 describes the 'Isomap Tracking with Particle filter' algorithm. Experimental results are demonstrated in Chapter 6, and conclusions are presented in Chapter 7.

Chapter 2

RELATED WORK

As described in the previous section, tracking can be done deterministically or stochastically. For head-tracking, cues can be taken from shape, color, texture and features (like eyes, nose, mouth). Birchfield [4] uses magnitude gradients and color histograms to track a person's head using an elliptical head-tracker. Cascia et al. [16] use a texture mapped surface model to approximate the 3D pose of the head. They formulate tracking as an image registration problem in the model's texture map. Pardas et al. [19] propose a technique to track a person's face using active contours or snakes. All the techniques described above are deterministic approaches. The stochastic approach to tracking is based on Bayesian recursive estimation. In 1960, R.E. Kalman introduced Kalman filtering in his classic paper [14] which has since been a subject of extensive research. The Kalman filtering technique is a Bayesian technique which assumes the posterior density function to be Gaussian and the process and measurement models to be linear. Kiruluta et al. [15] investigate the feasibility of using a Kalman filtering model to predict motion of the head (both abrupt and smooth). The linearity problem of Kalman filter is solved in the extended Kalman filter and the unscented Kalman filter [27], both of which allow nonlinear dynamics. Particle filtering techniques which are also based

on Bayesian estimation do not assume linearity or Gaussianity in a process. Arulampalam et al. [1] give an overview of the basic particle filtering technique and its variations. Isard and Blake use the Condensation algorithm [10] to track curves in a cluttered environment using the principle of resampling. They also came up with the ICondensation algorithm [11] in which they augment Condensation to incorporate the principle of importance sampling, in which an importance function is used to sample from the posterior density function of the state. Nummiaro et al. [18] demonstrate the use of Condensation algorithm in face tracking based on color.

Dimensionality reduction techniques can be classified as linear and nonlinear. Jollife [13] describes the use of Principal Components Analysis, which is a linear dimensionality technique, for dimensionality reduction. PCA tries to capture maximum variance in data along the principal components which are the eigenvectors of the covariance matrix computed from the data. Torgerson came up with Classical Multidimensional Scaling (MDS) [26] which is another linear technique for dimensionality reduction. In this technique the inter-point Euclidean distances between all the high-dimensional points are preserved in the low-dimensional structure. The Isomap algorithm given by Tenenbaum et al. [25] is an improvement over Classical MDS. Instead of using Euclidean distances it uses geodesic distances for better performance on nonlinear structures. Roweis et al. [21] propose nonlinear dimensionality reduction using locally linear embedding. LLE computes low-dimensional neighborhood preserving embeddings of high-dimensional points without any knowledge of the global structure. Souvenir et al. [22] propose image distance functions for improving the results of these dimensionality reduction techniques.

Chapter 3

DIMENSIONALITY REDUCTION

The purpose of dimensionality reduction is to express high-dimensional data (such as images, audio, etc.) in lower dimensions in such a way as to highlight the similarities and dissimilarities in the data. Raw high-dimensional observations may actually have fewer degrees of freedom than their dimensionality suggests. Consider images of a person's head undergoing 3D rotation. Although the dimensionality of these images is very high, all of images lie in a 3D manifold. Each high-dimensional image can therefore be represented by its 3D pose with just three parameters $\{\theta_0, \theta_1, \theta_2\}$ which makes the data very convenient to work with. Thus, it is possible to represent a set of images in significantly fewer dimensions if the variation in the data is correctly captured. The dimensionality of the data can be reduced using techniques like PCA [13], MDS [26], LLE [20] and Isomap [25]. This chapter describes each of these techniques. The performance of these techniques is compared using a data set of synthetic images of a persons's head undergoing out-of-plane rotation and lighting changes. This data set is available at [12]. Let us call this data set as face-data. Every image in face-data is 64×64 pixels thus having a dimensionality of 4096. Figure 3.1 shows 12 images from face-data.

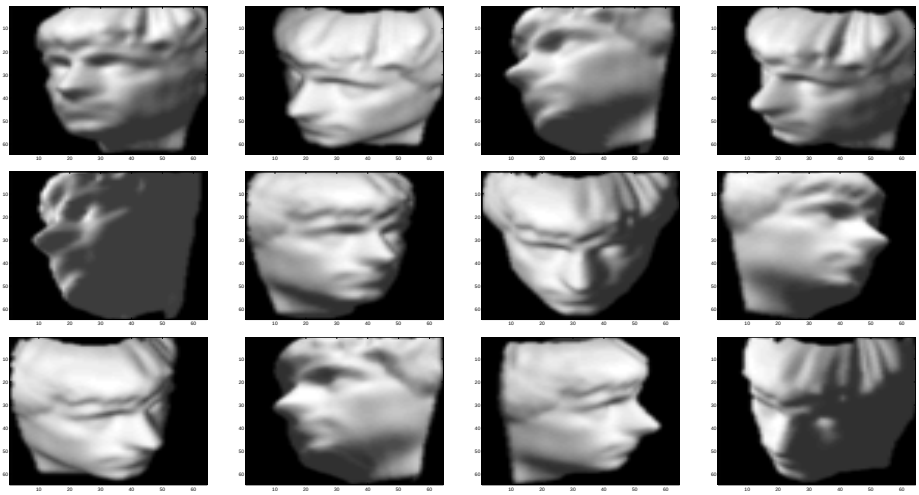


Figure 3.1: Sample images from face-data used for comparing the performance of PCA, MDS, LLE, Isomap.

3.1 Principal Components Analysis (PCA)

Principal Components Analysis [13] is a linear dimensionality reduction technique. PCA transforms the data into a new coordinate system in which the largest variance in the data lies on the first coordinate, second largest on the second coordinate and so on. The PCA algorithm is given below.

3.1.1 PCA algorithm

- *Step 1: Compute the mean value for all dimensions of the data set*

  $\overline{x^i} = \sum_{k=1}^{N} x_k^i$ where x_k^i is the k^{th} point of the i^{th} dimension and N is the total number points of the i^{th} dimension.

- *Step 2: Subtract the mean value of the data from every data-point*

  $x_j^{i*} = x_j^i - \overline{x^i}$. This is done to avoid non-uniqueness due to location.

- *Step 3: Compute the covariance matrix C*

  Covariance between any two variables x, y is given by

  $$C(x,y) = \sum_i (x_i - \overline{x})(y_i - \overline{y}) / (N-1)$$

  where i goes from 1 to N (total number of points) and $\overline{x}$ and $\overline{y}$ are the means. Note that the divisor is $N-1$ instead of N because the data set has finite number of samples. Every entry $C_{i,j}$ of the covariance matrix C is given by $C_{i,j} = Cov(x_i, x_j)$ where i, j are the i^{th} and j^{th} dimensions respectively. Note $\overline{x}$ and $\overline{y}$ are zero because the mean was already subtracted in step 2.

- *Step 4: Find the eigenvectors and eigenvalues of the covariance matrix*

  Next step is to compute the eigenvectors and eigenvalues of the covariance matrix C. Eigenvector of a square matrix is defined as the vector x which satisfies the equation $Cx = \lambda x$ and the constant λ is the corresponding eigenvalue.

- *Step 5: Choose the principal components and form the feature vector*

  Arrange the eigenvalues in descending order. Choose the first K significant eigenvalues where K depends on how much detail is needed in the low-dimensional data. Construct a matrix E with the eigenvectors corresponding to these first K significant eigenvalues.

  $$E = [eigenvector_1, eigenvector_2, \ldots eigenvector_K]$$

  Note that the eigenvectors are orthogonal to each other.

- *Step 6: Generate a low-dimensional point from a high-dimensional point*

  The final step is to compute a low-dimensional point x_K from a high dimensional point x_D where K is the new dimensionality of the data and D

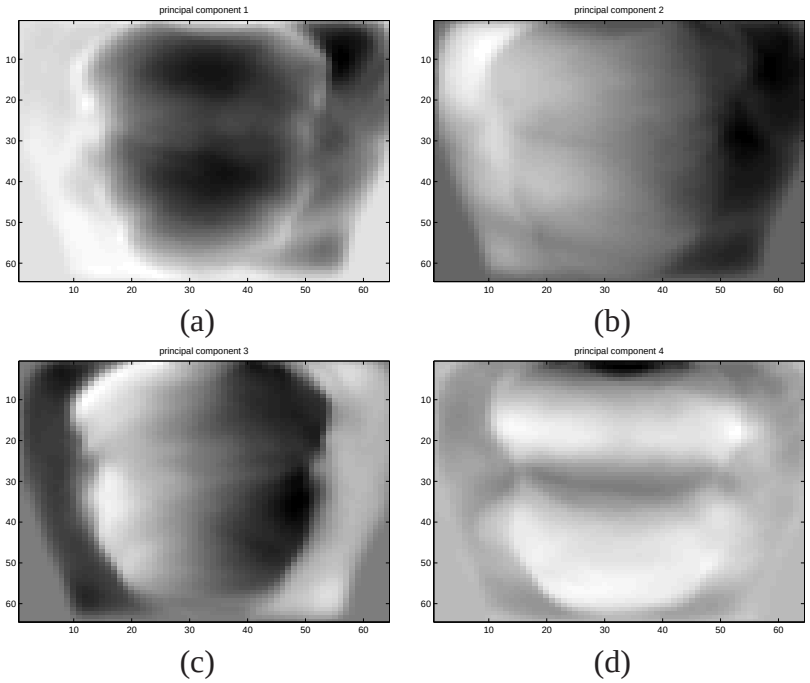


Figure 3.2: The four largest principal components recovered by PCA.

is the old dimensionality of the data. The low-dimensional vector is given by $x_K = E^T x_D$ where E^T is the transpose of matrix E computed in step 5 of the algorithm.

3.1.2 Performance of PCA on face-data

Performance of PCA was evaluated on face-data. Figure 3.2 shows first 4 principal components recovered by PCA. Figure 3.3 shows the first 6 eigenvalues given by step 4 of the algorithm. It can be seen that the curve elbows at the dimensionality of 5 which suggests that the 4096-dimensional face-data can be expressed in 5 dimensions. It is a significant reduction in the dimensionality but the true degrees of freedom is 3. Figure 3.4 shows the original image fed to PCA and the image recovered by PCA from the reduced dimensions. One can go back from lower dimensions to higher dimensions using the equation $x_D = Ex_K$ where E, K, D are defined in step 5 and 6 of the PCA algorithm.

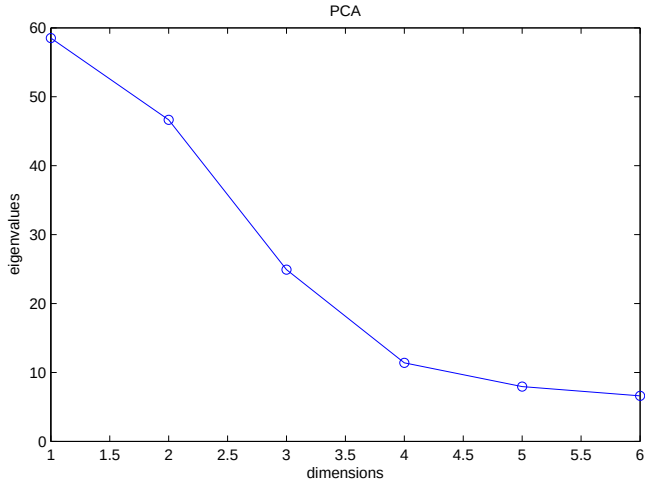
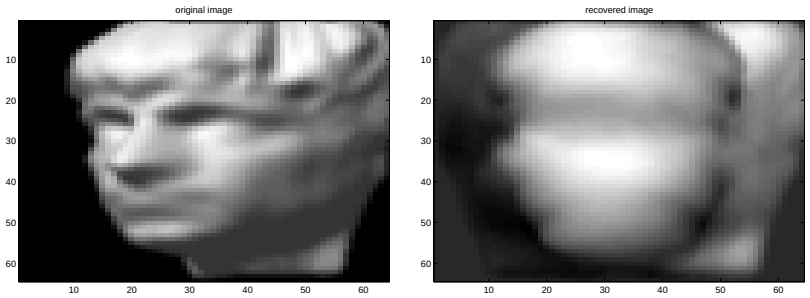


Figure 3.3: Plot of top six eigenvalues of the covariance matrix vs. dimensionality recovered by PCA.



(a) (b)

Figure 3.4: (a) Original image. (b) Image recovered by PCA using the equation $x_D = Ex_K$ with 5 eigenvectors.

3.2 Multi-dimensional Scaling (MDS)

Multi-dimensional scaling is a linear dimensionality reduction technique. The basic idea of MDS is to preserve the inter-point distances between the high-dimensional points in the low-dimensional structure. There are many variations of MDS. In this thesis only Classical MDS [26] is studied.

3.2.1 MDS algorithm

The Classical MDS algorithm [24] is given below. Let the original dimensionality of the data be D, the reduced dimensionality be K and total number of high-dimensional points be N.

- *Step 1: Compute the Similarity Matrix S*

  The similarity matrix S can be computed by taking the Euclidean distance between high-dimensional points. Each entry $\delta_{i,j}$ of matrix S is given by $\sqrt{\sum_l (x_{i,l} - x_{j,l})^2}$ where x_i and x_j are points in high dimensions and l goes from 1 to D.

- *Step 2: Compute the inner product matrix B*

  Compute the inner product matrix $B = -\frac{1}{2}JSJ$ where $J = I_N - \frac{1}{N}11^T$. J is called the centering matrix. I_N is the identity matrix of size N and 1 is a vector of ones of length N.

- *Step 3: Decompose B to obtain its eigenvalues and eigenvectors*

  Decompose B as $B = V\Lambda V$ where $\Lambda = diag(\lambda_1, \lambda_2, \ldots \lambda_N)$ is the diagonal matrix of eigenvalues of B. $V = [v_1, v_2, \ldots v_N]$ is a matrix of corresponding unit eigenvectors.

- *Step 4: Extract the first K eigenvalues and eigenvectors*

  Extract the first K eigenvalues $\Lambda_K = diag\,(\lambda_1, \lambda_2, \ldots \lambda_K)$ and corresponding eigenvectors $V_K = [v_1, v_2 \ldots v_K]$.

- *Step 5: Form the new low-dimensional coordinates of the points*

  The new coordinates of the points are in the matrix $X_{N\times K} = [x_1, x_2 \ldots x_N]^T = V_K \Lambda_K^{1/2}$. Using the top d-eigenvectors as the low-dimensional coordinates globally minimizes the difference in the distances between high-dimensional and low-dimensional points.

3.2.2 Finding dimensionality recovered by MDS

Hidden lower dimensionality in the data can be estimated using a quantity known as residual variance R. Residual variance basically means variance in the data due to unknown reasons. R is calculated at various low dimensions to find the dimensionality at which it stops decreasing significantly. The residual variance is calculated using the correlation coefficient c_i between the vectorized distance matrix Y for dimensionality i and the vectorized distance matrix S, defined in step 1 of the algorithm. Residual variance is then given by $R_i = 1 - c_i$.

3.2.3 Performance of MDS on face-data

Figure 3.5 shows that the residual variance stopped decreasing significantly only at dimensionality = 4. So the lower-dimensionality recovered by MDS for the 4096-dimensional face-data, although better than PCA, still does not reflect the true degrees of freedom in face-data. Figure 3.6 shows the plots of the low-dimensional points taking two significant dimensions at a time and keeping the third dimension constant, with the corresponding images from the face-data superimposed. In Figure 3.6 it can be seen that although MDS does a decent job at grouping similar head

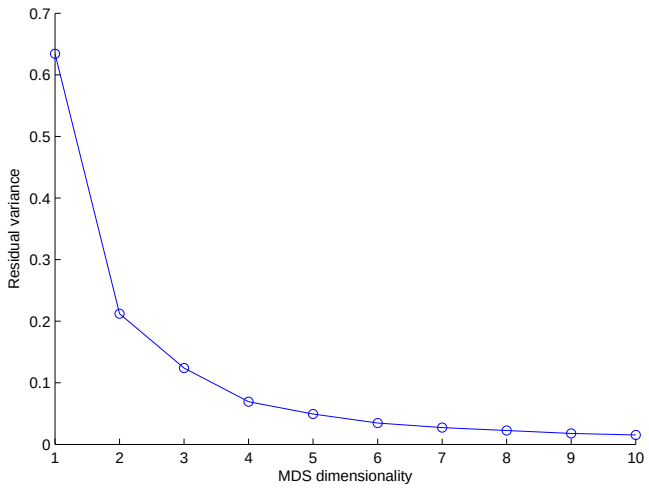


Figure 3.5: Plot of residual variance vs. dimensionality recovered by MDS

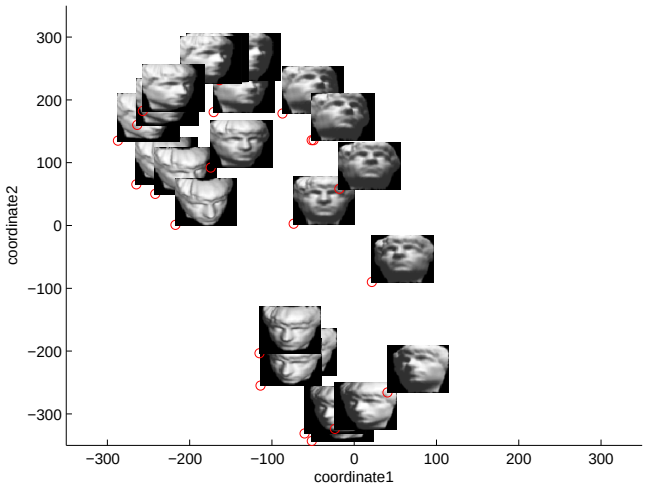
poses together but none of the low-dimensions have any meaning. The images superimposed on their respective 2D points show that no particular head-pose remains constant when a dimension for the points is kept constant. It can be concluded that MDS did a poor job at reducing the dimensionality of face-data.

3.3 Isometric Feature Mapping (Isomap)

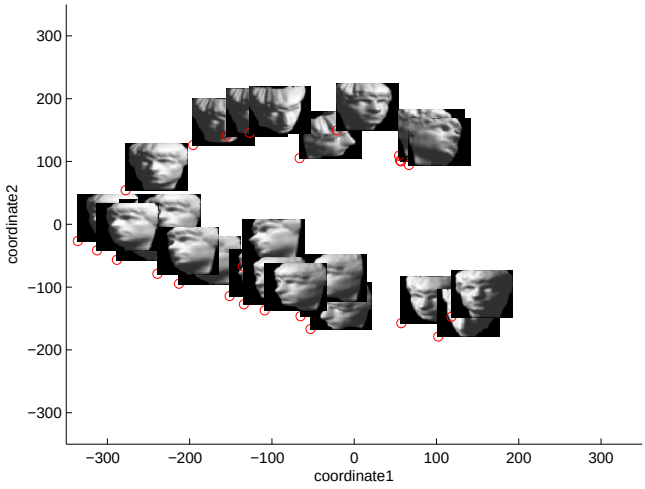
The Isomap technique, described in [25], is an improvement of the Classical MDS algorithm. The idea is to use Euclidean pairwise distances between high-dimensional points to compute the geodesic distances between those points. These are then fed to the Classical MDS algorithm, described in section 3.2.1, to find the true low-dimensionality of the data.

3.3.1 Isomap algorithm

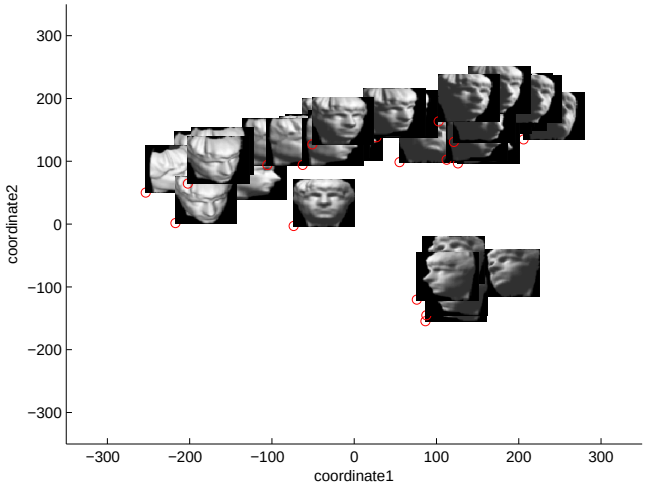
The Isomap algorithm takes as input the distances $d_x(i,j)$ between all pairs i,j from N data points in the high-dimensional input space X, measured in some domain-specific metric e.g. Euclidean distance. The algorithm outputs coordinates y_i in a



(a) Points with $z - coordinate \approx 0$.



(b) Points with $x - coordinate \approx 0$.



(c) Points with $y - coordinate \approx 0$.

Figure 3.6: 2D plots of low-dimensional points given by MDS with the corresponding high-dimensional images superimposed.

d-dimensional Euclidean space Y which best represents the intrinsic geometry of the data. The 3 steps of the algorithm are as follows:

- *Step 1: Construct neighborhood graph*

  This first step determines which points are neighbors on manifold M, based on the distances $d_x(i,j)$ in the input space. Two ways to do that are to connect each point to all points within a distance ϵ ($\epsilon - Isomap$) of that point, or to all of its K nearest neighbors ($\kappa - Isomap$). ϵ *or* κ is the only free parameter in the algorithm. The neighborhood relations are represented as a weighted graph G over the data points, with edges of weight $d_x(i,j)$ between neighboring points.

- *Step 2: Compute shortest paths*

  Compute the geodesic distances $d_M(i,j)$ between all pairs of points on the manifold M by computing their shortest path distances given by $d_G(i,j)$ in graph G. Computation: Initialize $d_G(i,j) = d_x(i,j)$ if i,j are linked by an edge; $d_G(i,j) = \infty$ otherwise. For each value of $k = 1, 2, \ldots, N$ in turn replace all entries $d_G(i,j)$ by

  $$min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$$

  The final matrix $D_G = d_G(i,j)$ will contain the shortest path distances between all pairs of points in G.

- *Step 3: Construct d-dimensional embedding*

  The last step applies classical MDS, described in section 3.2.1, to the matrix of graph distances $D_G = \{d_G(i,j)\}$, constructing an embedding of the data in a d-dimensional Euclidean space that best preserves the manifold's intrinsic geometry.

3.3.2 Finding Isomap dimensionality

The idea of using residual variance to find the true low-dimensionality of the data was described in section 3.2.2. Since Isomap is an extension of MDS, the same procedure can be used for Isomap.

3.3.3 Performance of Isomap on face-data

Figure 3.7 shows that the residual variance curve for face-data elbows at dimensionality = 3. So Isomap reflects the true dimensionality of the data, i.e., it finds 3 degrees of freedom in face-data. Figure 3.8 shows the plots of the low-dimensional points taking 2 significant dimensions at a time and keeping the third dimension constant, with the corresponding images from the face-data superimposed. In Figure 3.8 it can be seen that Isomap does a good job at grouping similar poses together and the low-dimensions correspond to the true degrees of freedom. When the first (x) coordinate is kept constant there is no variation in the left-right pose. When the second (y) coordinate is kept constant there is no variation in the left-right pose. When the third (z) coordinate is kept constant there is no variation in the lighting conditions. It can be concluded that Isomap correctly recovered the dimensionality of face-data.

3.4 Locally Linear Embedding (LLE)

Locally Linear Embedding [21] is a nonlinear dimensionality reduction technique which computes low-dimensional embedding with the property that nearby points in high-dimensional space are similarly co-located with respect to one another in low-dimensional space. This technique optimizes the embedding to preserve local configurations in high-dimensional data. LLE recovers the global nonlinear structure of the data from locally linear fits.

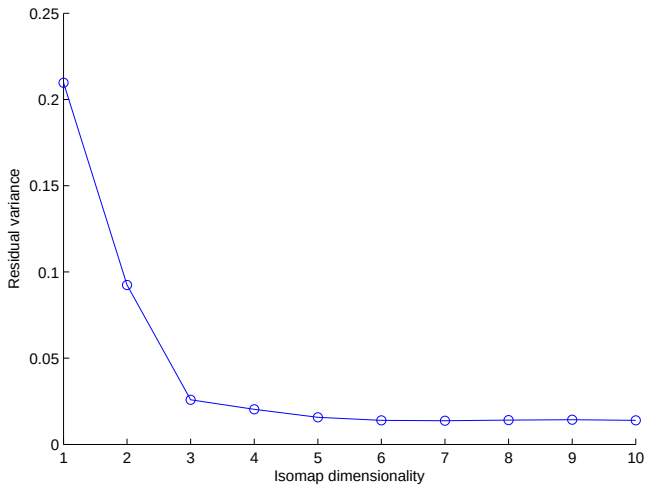


Figure 3.7: Plot of residual variance vs. dimensionality recovered by Isomap.

3.4.1 LLE algorithm

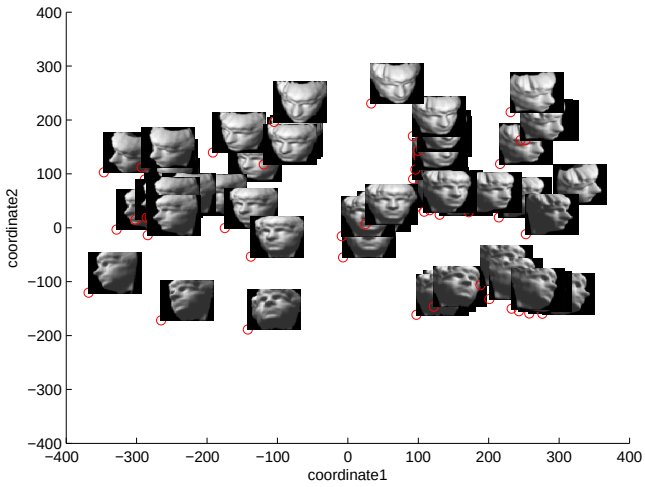
Suppose the data consists of N real-valued high-dimensional vectors $\vec{X}_i$ (inputs) each of dimensionality D. This algorithm maps those inputs to N low-dimensional vectors $\vec{Y}_i$ with dimensionality d such that the local geometry is preserved everywhere. The LLE algorithm is as follows:

- *Step 1: Compute the neighbors of each data point* $\vec{X}_i$

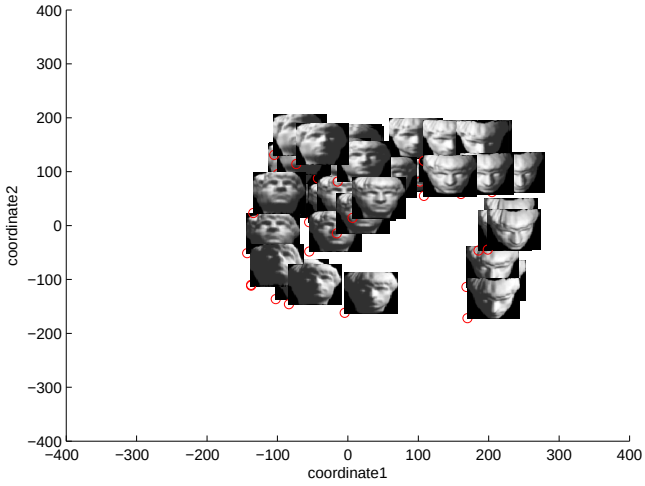
  For every point $\vec{X}_i$ identify K nearest neighbors based on the Euclidean distance between the points. K is chosen such that the data is well sampled. The rough estimate of K is $K \geq 2d$ where d is defined above.

- *Step 2: Compute the weights* W_{ij} *that best reconstruct each data point from its neighbors*

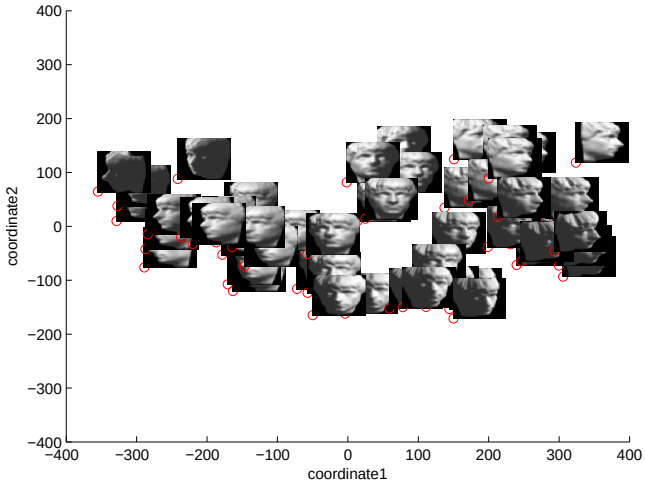
  The weights W_{ij} are chosen to minimize the cost function E given by the equation $E(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$. The weight W_{ij} summarizes the contribution of j^{th} data point to i^{th} reconstruction. The cost function is minimized subject to two constraints: a sparseness constraint and an invariance constraint. The sparseness constraint is that each data point $\vec{X}_i$ is reconstructed only from its neighbors, i.e., $W_{ij} = 0$ if the points are not neighbors. The invariance



(a) Points with $z - coordinate \approx 0$.



(b) Points with $x - coordinate \approx 0$.



(c) Points with $y - coordinate \approx 0$.

Figure 3.8: 2D plots of low-dimensional points given by Isomap with the corresponding high-dimensional images superimposed.

constraint is that the rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$. The constrained weights are invariant to rotations, rescalings and translations (by invariance constraint) of that data point and its neighbors. For a particular data point $\vec{x}$ with K nearest neighbors $\vec{\eta}_j$ and reconstruction weights w_j that sum to one, the reconstruction error is $\epsilon = \sum_{ij} w_j w_k G_{jk}$ where G is the local Gram matrix given by $G_{jk} = \left(\vec{x} - \vec{\eta}_j\right)\left(\vec{x} - \vec{\eta}_k\right)$. The optimal reconstruction weights can be computed as $w_j = \frac{\sum_k G^{-1}_{jk}}{\sum_{lm} G^{-1}_{lm}}$.

- *Step 3: Compute the vectors $\vec{Y}_i$ best reconstructed by the weights W_{ij}*

  Since the whole idea behind the algorithm is to preserve local geometry of the high-dimensional data, the same weights W_{ij} that reconstruct the input $\vec{X}_i$ in D dimensions should also reconstruct its embedded manifold coordinates in d dimensions. This is done by choosing the d-dimensional coordinates of each output $\vec{Y}_i$ to minimize the embedding cost function $\Phi(Y) = \sum_i |\vec{Y}_i - \sum_j W_{ij}\ \vec{Y}_j|^2$ where W_{ij}, computed in step 2, are fixed. To optimize the cost function Φ it can be re-written in the quadratic form $\Phi(Y) = \sum_{ij} M_{ij}\left(\vec{Y}_i \cdot \vec{Y}_j\right)$ where M is NxN square matrix given by: $M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}$. $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The output $\vec{Y}_i$ is subjected to 2 constraints: 1) $\sum_i\ \vec{Y}_i = \vec{0}$ removes the translational degree of freedom by requiring the outputs to be centered at the origin and 2) $\frac{1}{N}\sum_i\ \vec{Y}_i \vec{Y}_i^T = I$, where I is a $d \times d$ identity matrix, removes the rotational and scaling degree of freedom of the output. The cost function Φ can be minimized by finding the bottom $d + 1$ (corresponding to smallest eigenvalues) eigenvectors of cost matrix M. The bottom eigenvector is discarded as it represents the free translation mode of eigenvalue zero.

3.4.2 Finding LLE dimensionality

Saul and Roweis [21] give techniques to estimate the dimensionality d of the data. Let us consider two points lying within the distance ϵ to be neighbors. Then, for small value of ϵ, K should scale according to equation $K_\epsilon \propto \epsilon^d$. K is the number of neighbors within distance ϵ and d is the intrinsic dimensionality. Another way is to use the technique similar to the one used in PCA, described in 3.1.2, i.e., to estimate d by the number of eigenvalues comparable in magnitude to the smallest non-zero eigenvalue of the cost matrix M. The dimensionality, if previously known, can also be enforced to bias the embedding. This can be done by modifying the second step of the LLE algorithm. The idea is to project its neighbors into a d-dimensional subspace of maximal variance before performing the least squares reconstruction. The subspace is computed from d-dominant eigenvectors of Gram matrix G, defined in step 2, effectively limiting the rank of G before solving for reconstruction weights.

3.4.3 Performance of LLE on face-data

Section 3.4.2 describes the techniques given in [21] by Saul and Roweis to either find or force the dimensionality on the manifold. Figure 3.9 shows the plots of ϵ^d/K for four random high-dimensional points of face-data. It can be seen that ϵ^d/K is not a constant for dimensionality one to six. It can be concluded that LLE could not reduce the dimensionality to less than or equal to six dimensions and therefore could not discover the true degrees of freedom in face-data. Figure 3.10 shows the plot of eigenvalues of the cost matrix M vs. the dimensionality of face-data. Again, this technique, as described in section 3.4.2, was unable to find the true dimensionality of face-data. This may be due to the nonlinear nature of the manifold. Saul and Roweis [21] mention that this technique works well only for linear manifolds. Figure 3.11 shows the plots of the low-dimensional points recovered by LLE taking 2 significant dimensions at a time and keeping the third dimension constant,

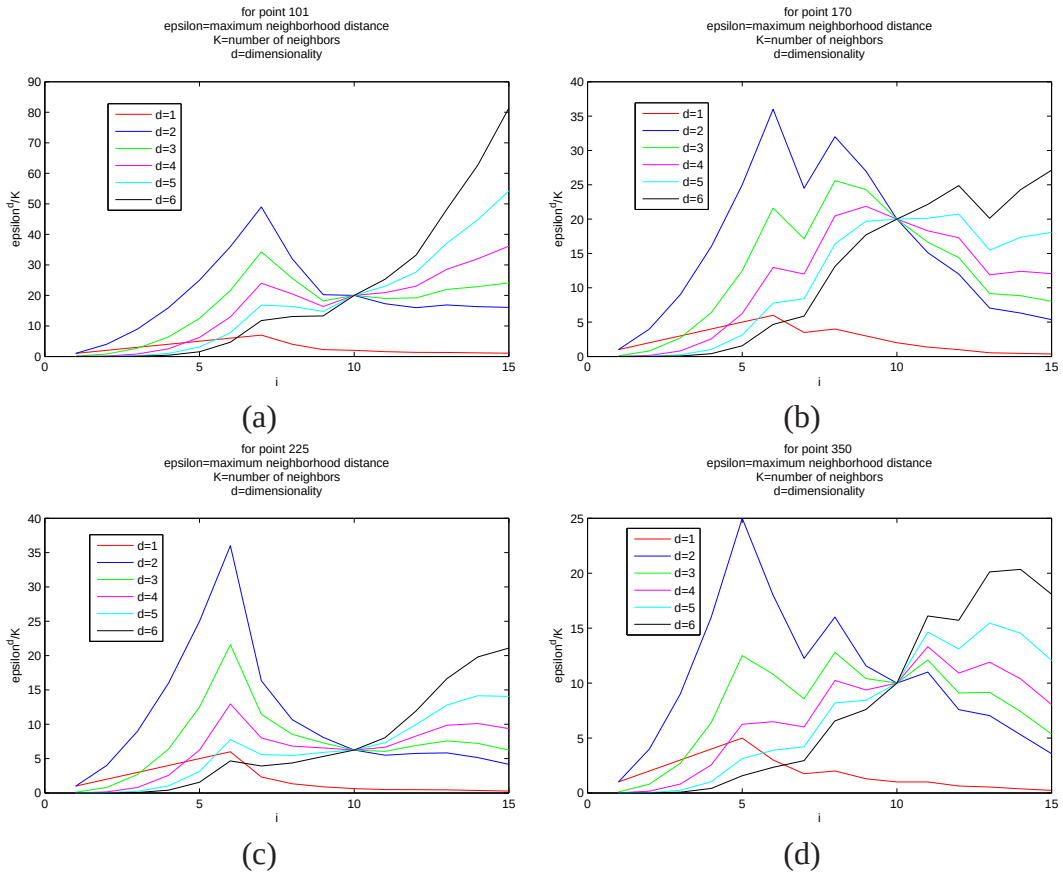


Figure 3.9: Plots of e^d/K where d is the dimensionality and K is the number of neighbors for four random points. It can be seen that K does not scale with ϵ^d for any of the points and therefore it is difficult to determine the dimensionality using the property $K_\epsilon \propto \epsilon^d$.

with the corresponding images from the face-data superimposed. In Figure 3.11 it can be seen that LLE shows some grouping of similar poses but none of the low-dimensions have any meaning. The images superimposed on their respective 2D points show that no particular head-pose remains constant when a dimension for the points is kept constant. It can be concluded that LLE could not find the true degrees of freedom in face-data.

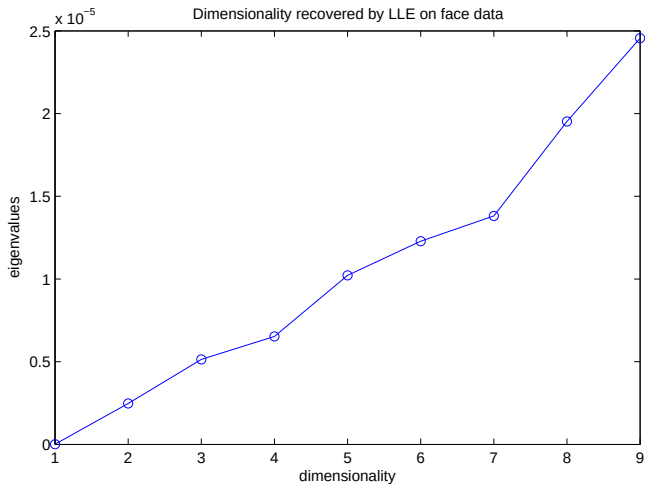
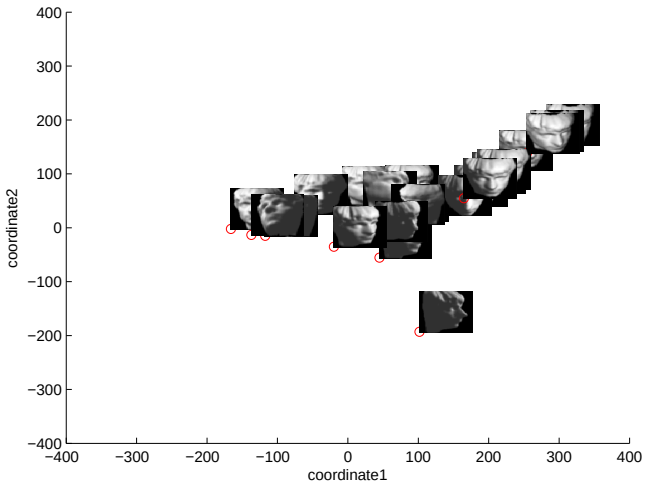


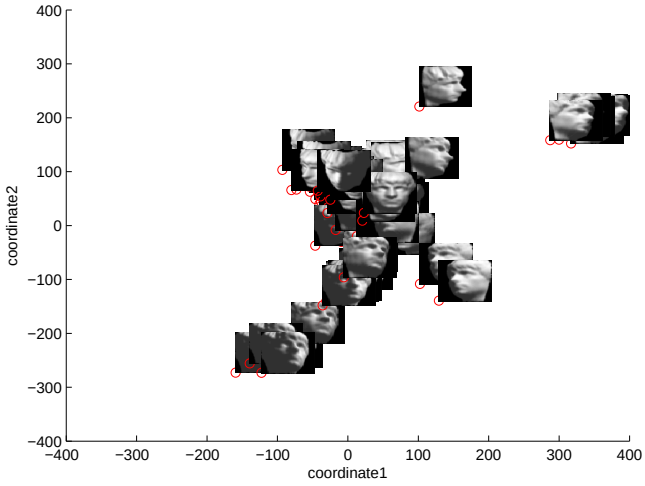
Figure 3.10: Plot of eigenvalues of the cost matrix M vs. dimensionality. The plot does not show a sharp enough elbow to determine the dimensionality of data using the eigenvalue method.

3.5 Comparing PCA, MDS, Isomap and LLE

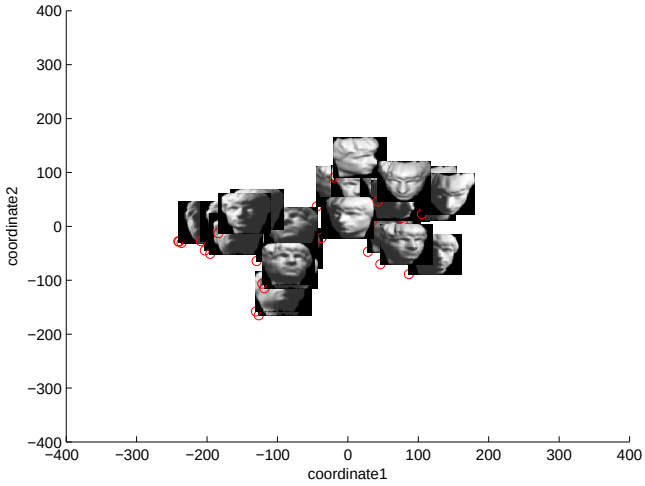
In the above sections the performance of PCA, MDS, Isomap and LLE was demonstrated on face-data shown in Figure 3.1. PCA and MDS performed poorly in capturing the true dimensionality of face-data. This result is not unexpected since PCA and MDS are essentially linear dimensionality reduction techniques and could not recover the nonlinear structure of face-data. Between Isomap and LLE, which are nonlinear dimensionality reduction techniques, Isomap gave better results on face-data. The reason for the poor performance of LLE could be poor sampling due to the nature of the training data as LLE assumes that each data-point and its neighbors lie on locally linear patches. Isomap on the other hand has no such assumption. The only approximation in Isomap algorithm is the estimation of the geodesic distance between the data points using pairwise Euclidean distances, which is still good if the data is dense enough. Going from low-dimensions to high-dimensions in PCA for any point (including points not present in the training set), was discussed in section 3.1.2. For MDS, Isomap and LLE, a high-dimensional point corresponding to a low-dimensional point, not present in the training set, can be constructed by



(a) Points with $z - coordinate \approx 0$.



(b) Points with $x - coordinate \approx 0$.



(c) Points with $y - coordinate \approx 0$.

Figure 3.11: 2D plots of low-dimensional points given by LLE with the corresponding high-dimensional images superimposed.

weighted combination of its nearest neighbors (in low dimensions). Out of the 4 techniques studied for dimensionality reduction, it can be concluded that Isomap gave the best results.

Chapter 4

TRACKING IN A RECURSIVE BAYESIAN FRAMEWORK

Approaches to tracking the state of a system can be broadly classified as deterministic and stochastic. The basic difference between the two classes is the use of random variables. Deterministic tracking does not use any random variable and the system is described by fixed values while stochastic tracking makes use of random variables and the system is described by a probability distribution. In this thesis the focus is on stochastic tracking in a Bayesian framework.

In the Bayesian framework, a two stage approach is used to track the state of the system in the presence of noisy measurements. A posterior probability density function (pdf) which gives an estimate of the state of a system, given the noisy measurements, is constructed at every step in time. The first stage is prediction, which creates a pdf of the state of the system at time t based on the system model and the posterior pdf from time $t-1$. The second stage is update which updates the pdf created in the prediction stage using the newly obtained measurement and the measurement model.

4.1 Bayesian Framework

A tracking problem can be defined as the evolution of the state of the system $\{x_k, k \in N\}$ where x_k is the state vector containing all the information about the system at time k. Equation 4.1 is the system model f_k for the evolution of state of the system where $f_k : R^{n_x} \times R^{n_v} \rightarrow R^{n_x}$ can be a linear or a nonlinear function and v_k is the process noise sequence. Equation 4.2 is the measurement model h_k which relates the state of the system to the measurements z_k obtained at time k. In equation 4.2, $h_k : R^{n_x} \times R^{n_n} \rightarrow R^{n_z}$ can be linear or a nonlinear function and n_k is the measurement noise sequence. For tracking any system f_k, h_k, v_k and n_k should be known. The process noise v_k and the measurement noise n_k can be constant or variable over time. Equation 4.3 is the Chapman-Kolmogorov equation which describes the prediction stage. The quantity $p(x_k \mid z_{1:k-1})$ is the pdf of the state given the system model $p(x_k \mid x_{k-1})$ and the posterior $p(x_{k-1} \mid z_{1:k-1})$ from time $k-1$. When the measurement z_k becomes available at time k the pdf $p(x_k \mid z_{1:k-1})$ is updated using the Bayes' rule given by equation 4.4. In equation 4.4, the posterior $p(x_k \mid z_{1:k})$ is constructed by updating the pdf $p(x_k \mid z_{1:k-1})$ (also called the prior in Bayes' equation) using the likelihood function $p(z_k \mid x_k)$. The quantity $p(z_k \mid z_{1:k-1})$ is the normalizing constant. The next sections of this chapter give an overview of Bayesian algorithms: Kalman filtering, extended Kalman filtering and particle filtering. Since the focus of this thesis is particle filtering, it is explained in detail.

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{4.1}$$

$$z_k = h_k(x_k, n_k) \tag{4.2}$$

$$p(x_k \mid x_{k-1}, z_{1:k-1}) = \int p(x_k \mid x_{k-1})\, p(x_{k-1} \mid z_{1:k-1})\, dx_{k-1} \tag{4.3}$$

$$p(x_k \mid z_{1:k}) = \frac{p(z_k \mid x_k)\, p(x_k \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})} \tag{4.4}$$

4.2 Kalman filter

The Kalman filtering algorithm [14] is based on recursive Bayesian framework explained in section 4.1. Let us examine the quantities in equations 4.1, 4.2 for a Kalman filter.

- system model f_k and measurement model h_k are linear
- process noise v_k and measurement noise n_k are assumed to be additive Gaussian noises
- posterior density $p(x_k \mid z_k)$ is estimated to be Gaussian

4.2.1 Discrete Kalman filter algorithm

The Kalman filter is a two stage predictor-corrector algorithm. The first stage is the 'Time update' (Predict) and second stage is the 'Measurement update' (Correct). P. Maybeck [17] explains the probabilistic origins of the algorithm. Let the error between the estimated state $\widehat{x}_k$ and the true state x_k be e_k at time k. The algorithm tries to minimize the error covariance $P_k = E\left[e_k e_k^T\right]$ at every time-step. Let Q be the process noise covariance, R be the measurement noise covariance, A be the state transition matrix, H be the matrix which relates the measurements to the state vector, and K be the Kalman gain. The super-script minus (x_k^-) indicates the a priori state estimate at step k, given the knowledge of the system dynamics f. The algorithm is given below. Figure 4.1 shows the performance of a Kalman tracker tracking an object moving in a trajectory $x = 3t$ with random accelerations of standard deviation = 0.5, where x is the position and t is the time. The measurement noise had a standard deviation of 10.

- Time update

- *Step 1: Project the state ahead*

  $\widehat{x_k}^- = A\widehat{x_{k-1}}$

- *Step 2: Project the error covariance ahead*

  $P_k^- = AP_{k-1}A^T + Q$, with P_0 set to Q.

- Measurement update

  - *Step 1: Compute Kalman gain*

    $K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1}$

  - *Step 2: Update estimate with measurement*

    $\widehat{x_k} = \widehat{x_k}^{-1} + K_k \left(z_k - H\widehat{x_k}^{-1}\right)$

  - *Step 3: Update the error covariance* $P_k = (I - K_k H)\, P_k^-$

A Kalman filter is restricted in its use because it requires the system and the measurement model to be linear and it also estimates the posterior density of the state to be Gaussian at every time-step. The first problem is taken care of in the extended Kalman filter which can handle nonlinear system dynamics. For a process, whose posterior density of the system-state is non-Gaussian, the Gaussian estimates of Kalman filtering give large tracking errors.

4.3 Extended Kalman filter

As the name suggests the extended Kalman filter is just an extension of the Kalman filter to handle nonlinear processes. The nonlinearity can be linearized for every current estimate using the first term of the Taylor expansion of nonlinear function by taking the partial derivatives of the system and measurement models. This is what an extended Kalman filter does. It linearizes a nonlinear stochastic difference equation at every time step. Let us examine the quantities in equations 4.1, 4.2 for the extended Kalman filter.

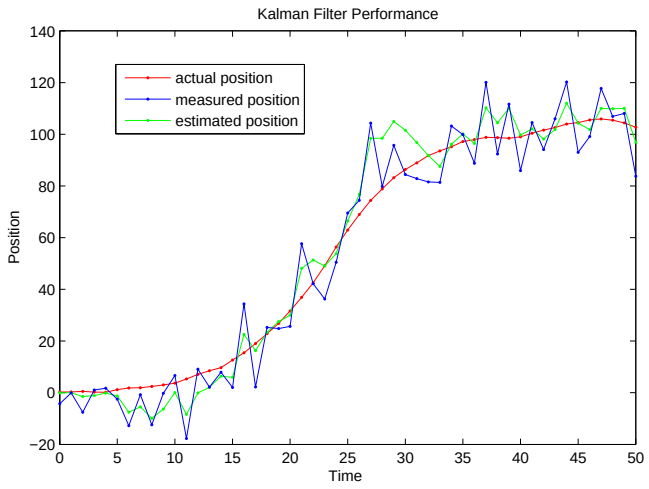


Figure 4.1: Performance of a Kalman tracker tracking an object moving in a trajectory $x = 3t$ with random accelerations of standard deviation = 0.5, where x is the position and t is the time. The measurement noise had a standard deviation of 10.

- system model f_k and measurement model h_k can be nonlinear
- process noise v_k and measurement noise n_k no longer remain Gaussian due to nonlinearity
- posterior density $p(x_k \mid z_k)$ is estimated to be Gaussian at all times

4.3.1 Extended Kalman filter algorithm

For an extended Kalman filter, let A be the Jacobian matrix of partial derivatives of process model f with respect to state x, V be the Jacobian matrix of partial derivatives of f with respect to process noise v, H be the Jacobian matrix of partial derivatives of measurement model h with respect to x, N be the Jacobian matrix of partial derivatives of h with respect to measurement noise n. All the other notations remain the same as described in section 4.2.1. The algorithm is given below.

- Time update
  - *Step 1: Project the state ahead*

    $\widehat{x_k}^- = f(\widehat{x_{k-1}}, 0)$

- *Step 2: Project the error covariance ahead*

  $P_k^- = AP_{k-1}A^T + W_kQW_k^T$

- Measurement update

  - *Step 1: Compute Kalman gain*

    $K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R V_k^T\right)^{-1}$

  - *Step 2: Update estimate with measurement*

    $\widehat{x_k} = \widehat{x_k}^- + K_k \left(z_k - h(\widehat{x_k}^-, 0)\right)$

  - *Step 3: Update the error covariance*

    $P_k = (I - K_k H_k) P_k^-$

Although the extended Kalman filter handles nonlinearity by linear approximations, the tracking errors increase as the nonlinearity increases. Like the Kalman filter, it still estimates the posterior density of the system-state to be Gaussian and therefore does not track a system with non-Gaussian posterior well. This problem is solved by particle filtering techniques, described in section 4.5.

4.4 Markov Chain Monte Carlo simulations

4.4.1 Markov Chain

Markov property of a process is the conditional independence of the future states of the process on the past states. According to the property, the future state of the system can be estimated given the current state and the current measurement. Mathematically, a stochastic process X_n is a Markov chain in discrete time if for every $n \in N$ the probability of going from state i to state j is given by the equation $P[X_{n+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n] = P[X_{n+1} = j \mid X_n = i_n]$. For a Markov chain it is assumed that the initial distribution X_0 is known. Assume a finite set

$S = \{1, \ldots, m\}$ to be the values that a system can take and let $t_{i,j}$ be the probability of going from state i to j. A transition matrix T can be defined such that $t_{i,j} > 0 \forall (i,j) \in S^2$ and $\sum_{j \in S} p_{ij} = 1 \forall i \in S$. This transition matrix gives the system model for a Markov process.

4.4.2 Monte Carlo simulation

Monte Carlo simulation is a technique to approximate an expected value of a function using simulated random variables. Let X be a random variable on a set of values Y with a pdf $F(x)$ and let $g(x)$ be a function of X. The expected value $E(g(X)) = \sum_{x \in Y} g(x)F(x)$ can be approximated as a MC simulation $\widetilde{g_n(x)} = \frac{1}{N} \sum_{i=1}^{N} g(x_i)$ where x_i is chosen randomly using any distribution $G(x)$. By the strong law of large numbers as N approaches ∞, the difference between the expected value and the value approximated by Monte Carlo simulation approaches zero.

4.4.3 Importance Sampling for Monte Carlo simulations

The Monte Carlo estimator $\widetilde{g_n(x)}$ of $E(g(x))$ can be thought of as a random variable. The variance of the Monte Carlo estimator needs to be minimized for decreasing the estimation error. This can be done by choosing a suitable distribution to sample the random variable X (defined in section 4.4.2) from. The idea behind importance sampling [23] is that certain values have bigger impact on Monte Carlo estimate and by choosing these values frequently, variance of the estimate can be reduced. This results in a biased simulation but results are weighted to correct for the biased distribution. The weight W_i is the ratio of $g(X_i)$ (true distribution) and $g^{'}(X_i)$ (importance distribution) at X_i. One good choice for reducing variance on the estimate is choosing $g^{'}(x)$ such that $g^{'}(x) > 0$ and $g^{'}(x) \propto \mid g(x) \mid$.

4.5 Particle filtering

The Kalman filtering algorithms estimate the posterior density of the state to be Gaussian. A Kalman filter also assumes the measurement and the process noise to be Gaussian. It is clear that these are highly restrictive conditions for a process. For processes with non-Gaussian pdfs the Gaussian estimates may prove unfit for tracking. In such scenarios tracking can be done with particle filtering methods [1] which do not assume any of the above conditions. There are many variations of the particle filtering algorithms. This section describes the sequential importance sampling (SIS) algorithm based on Monte Carlo simulations (explained in section 4.4.2) with some of the problems associated with it and ways to solve those problems.

4.5.1 Sequential Importance Sampling (SIS)

The SIS algorithm [1] is a Monte Carlo Method (explained in section 4.4.2) based on Bayesian estimation. The idea here is to represent a posterior density function $p(x_{0:k} \mid z_{1:k})$ with a set of random samples (particles) with associated weights $\{x^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ where $\{x^i_{0:k}, i = 0, \ldots N_s\}$ is the set of N_s random samples with weights $\{w^i_k, i = 0, \ldots N_s\}$. The pdf $p(x_{0:k} \mid z_{1:k})$ is approximated as $p(x_{0:k} \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w^i_k \delta(x_{0:k} - x^i_{0:k})$. For many processes $p(x_n \mid y_{1:n})$ cannot be expressed in an analytical form. For such cases it is difficult to draw samples from $p(x_n \mid y_{1:n})$. This problem can be solved with the principle of importance sampling [7] (explained in section 4.4.3). Suppose $q(x_n \mid y_{1:n})$ is a pdf from which samples can be easily drawn and $q(x_n \mid y_{1:n}) \propto p(x_n \mid y_{1:n})$. The weights are then defined to be

$$w^i_k \propto \frac{p(x_n \mid y_{1:n})}{q(x_n \mid y_{1:n})} \tag{4.5}$$

For a sequential case, let the importance density be chosen to factorize as

$$q(x_{0:k} \mid z_{1:k}) = q(x_{0:k} \mid x_{0:k-1}, z_{1:k}) q(x_{0:k-1} \mid z_{1:k-1}) \tag{4.6}$$

Using Bayes' rule given in equation 4.4, for a sequential case, we have

$$p(x_{0:k} \mid z_{1:k}) \propto p(z_k \mid x_k) p(x_k \mid x_{k-1}) p(x_{0:k-1} \mid z_{1:k-1}) \tag{4.7}$$

Using equations 4.5, 4.7, 4.7 we get

$$w_i^k \propto \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i) p(x_{0:k-1}^i \mid z_{1:k-1})}{q(x_{0:k}^i \mid x_{0:k-1}^i, z_{1:k}) q(x_{0:k-1}^i \mid z_{1:k-1})}$$

This can be written as a recursive weight update equation for every sample x_k^i

$$w_k^i = w_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_{0:k}^i \mid x_{0:k-1}^i, z_{1:k})}$$

Assuming the process to be a first order Markov process (explained in section 4.4.1) the equation simplifies to equation 4.8 where w_k^i is the normalized weight of each sample.

$$w_k^i = w_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)} \tag{4.8}$$

For a process with Markov property, equation 4.9 gives an approximation for the posterior $p(x_k \mid z_{1:k})$. For a Monte Carlo simulation as $N_s \to \infty$ the approximation approaches the true posterior.

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{4.9}$$

The SIS algorithm is given below

- For $k = 1 : t$

– For $i = 1 : Ns$

* Draw sample x_k^i from the importance density $q(x_k^i \mid x_{k-1}^i, z_k)$
* Calculate weight w_k^i for every sample using equation 4.8

– End For

- Estimate the posterior density $p(x_k \mid z_{1:k})$ using equation 4.9
- Estimate the moment of the tracked position at every time-step k
- End For

4.5.2 Degeneracy problem of SIS algorithm

A. Doucet [6] shows that with limited number of particles the variance of the importance weights can only increase over time. After a few iterations over time of the SIS algorithm many particles (or samples) drop to negligible weight. This is known as the degeneracy phenomenon. Significant amount of computation is devoted to sampling and updating these particles whose contribution to the posterior $p(x_k \mid z_{1:k})$ is negligible. An approximate measure of degeneracy is $N_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2}$ where w_k^i is the normalized weight calculated in equation 4.8. M. Arulampalam [1] describes two techniques to circumvent the degeneracy problem: 1) choosing a good importance density $q(.)$ and 2) resampling.

1. *Choosing a good importance density*

   The optimal importance density which reduces the variance on the weights is $q(x_k \mid x_{k-1}^i, z_k) = p(x_k \mid x_{k-1}^i, z_k)$. But this solution works only in cases where it is possible to sample from $p(x_k \mid x_{k-1}^i, z_k)$. It is possible however to have sub-optimal approximation to the optimal importance density. One such approximation is $q(x_k \mid x_{k-1}^i, z_k) = p(x_k \mid x_{k-1})$. Substituting this equation

in equation 4.8 we get $w_k^i \propto w_{k-1}^i p(z_k \mid x_k^i)$. The use of $p(x_k \mid x_{k-1})$ as the importance density function is convenient because it simplifies the weight update equation. This approach is used in bootstrap filtering [8].

2. *Resampling*

   Another approach to avoid the degeneracy problem is to resample the particles when the degeneracy approximator N_{eff} drops below a certain threshold N_T. The idea is to propagate particles with high weights to the next iteration in time by eliminating the particles with low weights. A new set x_k^{i*} is created from the original set x_k by resampling (with replacement) N_s times. This is done by constructing a cumulative distribution function (cdf) of the weights of particles. A particle is then selected using uniformly distributed random number u drawn from $U(0,1)$. With this procedure, particles with high weights get selected with higher probability than particles with low weights. Since u is uniformly distributed the new particles are weighted as $\frac{1}{N_s}$. The details of the algorithm are given below. Although resampling helps correct the degeneracy problem, it introduces another problem which is sample impoverishment. It duplicates particles with high weights but this process reduces the diversity of the sample-set. In fact, when the process noise is small the effects of resampling are severe as the particles tend to collapse to a single point within a few iterations. Since the diversity of paths of particles is reduced it is also difficult to get any smooth estimates of the paths over time. Resampling is used in Condensation algorithm [10].

   Resampling algorithm: $\left[\left\{x_k^{j*}, w_k^j\right\}_{j=1}^{N_s}\right] = RESAMPLE(\left[\{x_k^i, w_k^i\}_{i=1}^{N_s}\right])$

   - Initialize the CDF: $c_1 = 0$
   - For $i = 2 : N_s$

  - Construct CDF: $c_i = c_{i-1} + w_k^i$

- End For

- For $j = 1 : N_s$

  - Generate a random number u_j from $U(0, 1)$
  - Find smallest l for which $c_l \geq u_j$
  - Set $x_k^{j*} = x_k^l$
  - Set $w_k^{j*} = 1/N_s$

- End For

Chapter 5

ISOMAP TRACKING WITH PARTICLE FILTER

Tracking an object has several challenges. Some of these are translation of the object, changing view of the object at each frame, scaling, occlusions, background clutter and changing ambient light. The focus of this thesis is tracking a person in an image sequence. A good tracker should be able to handle the translational motion, in-plane and out-of-plane rotations of a person's face and the other challenges mentioned above. In this thesis, an algorithm is presented which can track a person's face in the presence of some of these challenges and also estimate the head-pose at each frame of the image sequence. The algorithm requires a training set of a person's face at various poses. Once the training-set is acquired the dimensionality of the training-set is reduced using a dimensionality reduction technique. The reduced dimensionality allows a more compact and meaningful representation of training data. This low-dimensional information can then be used by a stochastic tracking algorithm to track a person in an image sequence.

Isomap algorithm is selected over PCA, MDS and LLE to reduce the dimensionality of the training data because of its superior performance (described in section

3.5) in reducing the dimensionality of face-data. In chapter 4, Gaussian (Kalman filtering) and non-Gaussian (particle filtering) Bayesian recursive techniques were discussed. In an image sequence, the motion of a person's head and its pose can be very unpredictable. With background clutter, occlusions and lighting variations it is a kind of process which is hard to model. Under such conditions, assuming the posterior probability density of the state (face-pose) to be Gaussian could very well lead to large tracking errors. Particle filtering is chosen as the tracking algorithm because it does not assume the posterior density distribution of the state to be Gaussian. In fact, it gives an approximation to the true posterior density of the state. The tracker is based on the Condensation algorithm [10] which uses resampling to avoid the degeneracy problem explained in section 4.5.2. It also uses the prior as the importance density which makes the weight update equation much simpler. M. Isard and A. Blake [10] demonstrate the ability of the Condensation algorithm to track curves in dense visual clutter. 'Isomap tracking with particle filter' algorithm can be outlined in three steps:

1. Create a noise-free training set of the person's face (done off-line)

2. Use Isomap to reduce the dimensionality of the training set (done off-line)

3. Run the particle filter to track the person in a test image sequence using the training data and the low-dimensional structure (coordinates) recovered by Isomap (done on-line)

5.1 Training data

The training data should include the images of the person's head at various poses. Data should be dense enough to have a smooth variation in pose for best results by Isomap. There is no reason to include training images with in-plane rotation of the

Figure 5.1: Sample images from the training set of a person's head used to train the tracker.

person's face or varying ambient light. In-plane rotation can be approximated by rotating the face-templates. The variation in ambient light can also be handled by the tracker by choosing a good weighting function. This helps in reducing the size and complexity of the training set. When building a training set (since this is done off-line), images can be made noise-free, i.e., all the non-face pixels in the image can be made zero. Obviously, Isomap gives better results when the noise is reduced. The training data used for testing the algorithm has 145 grayscale images of size 151×151 pixels. Few samples are shown in Figure 5.1. These images include a person's head varying in up-down pose and left-right pose (out-of-plane rotations).

5.2 Isomap of the training data

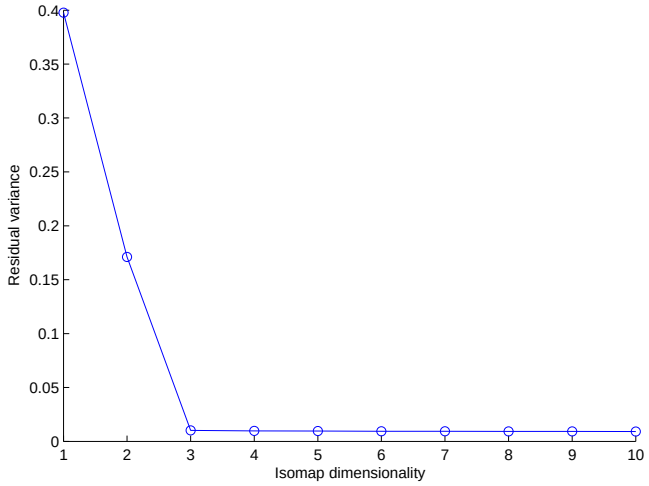
Isomap [25] of the training data is computed off-line as described in section 3.3.1. In this experiment, Euclidean distance (SSD) is used as the distance metric to com-

pute distance between any two pair of images. It was seen that neighborhood value $k \approx 2d$ (where d is the observed low-dimensionality of the training data) gives good results for Isomap. Sum of squared distances (SSD) is used to compute the distance between the training images. Figure 5.2.a shows that the residual variance vs. dimensionality plot for the training data elbows at dimensionality of 3 which means that the high-dimensional face-data can be well represented in 3 dimensions. Figure 5.2.b shows the arrangement of the points in 3 dimensions given by Isomap. With the training data varying in up-down pose and the left-right pose, one would expect Isomap to recover only 2 degrees of freedom but it recovers a dimensionality of 3. The reason for this might be the symmetry of a face. A face has a vertical axis of symmetry but no horizontal axis of symmetry and with many of the intermediate face images missing and the data being sparse, it recovers a dimensionality of three. When the images with head moving up were removed from the training data Isomap correctly recovered the dimensionality of two. This is illustrated in Figure 5.4. In Figure 5.2.b it can be seen that the 3D structure given by Isomap resembles a tetrahedron. Isomap placed faces moving in each direction on separate arms with the face looking straight ahead placed at the center. What is also interesting to note in Figure 5.2.b and Figure 5.3 is that the arms of the structure are almost linear. Since all human faces have similar structure it can be assumed that Isomap would give a similar result for similar training data of a different person.

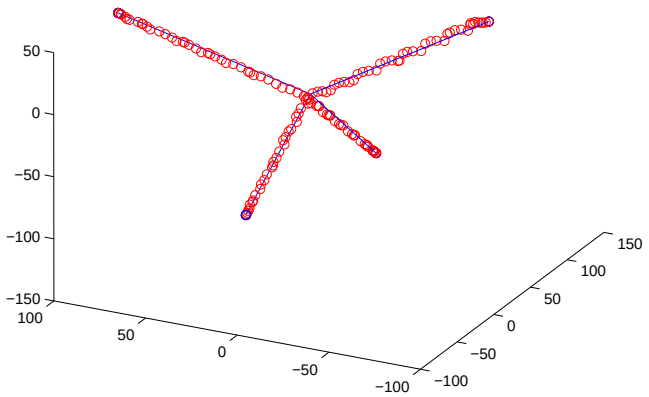
5.3 Particle filter algorithm

5.3.1 Condensation algorithm

The particle filtering algorithm used for this experiment is based on the Condensation algorithm [10]. In the Condensation algorithm, the output of every iteration (with measurement taken at time t) is a weighted set of particles denoted as

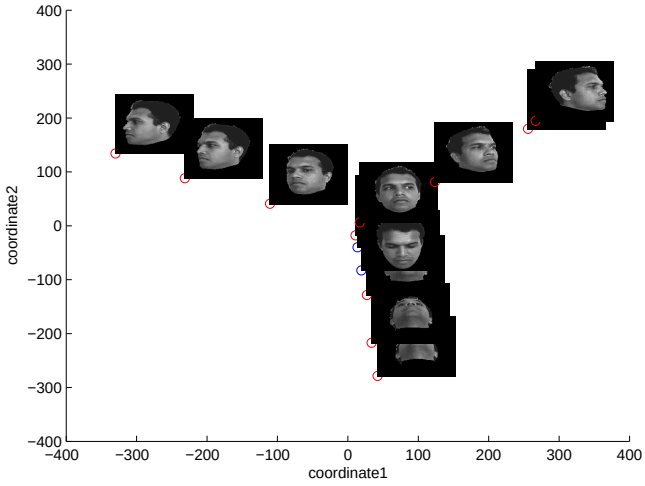


(a)

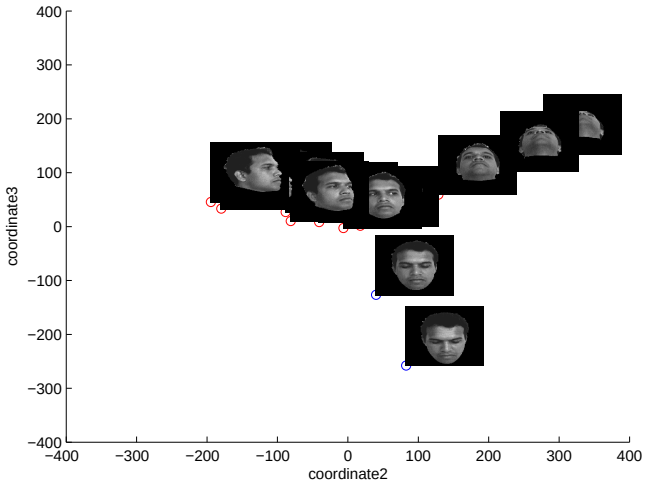


(b)

Figure 5.2: (a) Plot of the residual variance vs. dimensionality for the training data. It can be seen that the curve elbows at dimensionality $d = 3$. (b) The low-dimensional structure recovered by Isomap from the high-dimensional data. It resembles a tetrahedron.



(a)



(b)

Figure 5.3: (a) and (b) show two different two-dimensional views of the low-dimensional structure with training templates, corresponding to some points, superimposed to help visualize how the high-dimensional 151×151 points are embedded in 3D.

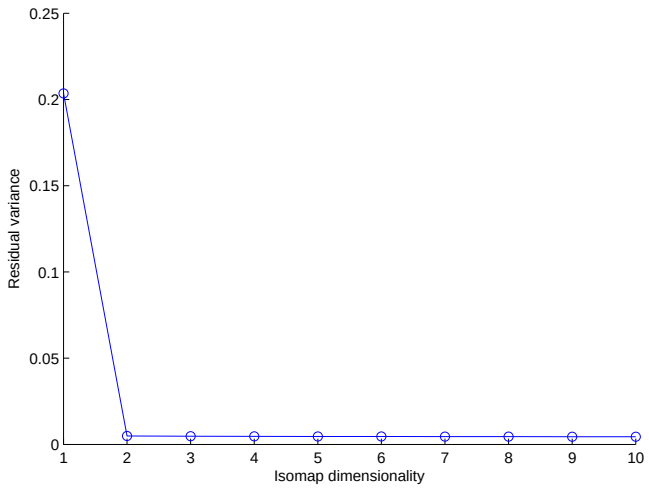


Figure 5.4: Plot of the residual variance vs. dimensionality for the training data with head moving from left to right and down. The images with head moving up were removed from the training data.

$\left\{s_t^{(n)}, n = 1, ..., N\right\}$ with weights $\pi_t^{(n)}$, representing the probability density $p\left(x_t \mid Z_t\right)$ where x_t is the state of the system, z_t is the measurement at time t and Z_t is $z_{1:t}$. An iterative process explained below is used to estimate the state of the system at each time step.

Iterate:

From old sample set $\left\{s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, .., N\right\}$ at time step $t-1$, construct a new sample-set $\left\{s_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}, n = 1, .., N\right\}$ for time t. Construct the n^{th} of the N new samples as follows:

1. **Resample**: Select a sample $s_t^{'(n)}$ as follows:

   (a) generate a random number $r \in [0, 1]$ uniformly distributed

   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^j \geq r$

   (c) set $s_t^{'(n)} = s_{t-1}^j$

2. **Predict** by sampling from $p(x_t \mid s_{t-1} = s_t^{'(n)})$. This is governed by the system model which includes the deterministic drift and the stochastic diffuse of the sample $s_t^{(n)}$.

3. **Measure** and weight the new position in terms of the measured features z_t as $\pi_t^{(n)} = p\left(z_t \mid x_t = s_t^{(n)}\right)$. Then normalize so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $\left\{s_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\right\}$ where $c_t^0 = 0\,;\;\; c_t^n = c_t^{n-1} + \pi_t^n;\;\; (n = 1, ..., N)$. Once the N samples have been constructed: **estimate** moments of the tracked position at time-step t as

$$E\left[f\left(x_t\right)\right] = \sum_{n=1}^{N} \pi_t^n f\left(s_t^n\right)$$

5.3.2 Design of the particle filter

1. *State vector* x_t

   The state vector for this experiment is seven dimensional. Two dimensions of x-y translation in the image, three dimensions given by the Isomap for out-of-plane rotation, and one dimension each for in-plane rotation and scaling of the tracker, make up the state vector.

2. *Initialization prior* $p(X_0)$

   An initialization prior should be known for the Condensation algorithm. There are various was to do it. A tracker can be manually initialized in the first frame with particles, with equal weights, randomly distributed near the person face. Or a background subtraction method can initialize the x-y position of the tracker in the first frame. Initialization of rotation parameters becomes easier under the assumption that the person is facing the camera in the first frame. The particles can be initialized randomly to have a range of scale values. In this experiment, manual initialization is used for the first frame.

3. *System model* $p(x_t \mid x_{t-1})$

   The system model has a deterministic component and a stochastic component. The transition matrix gives the deterministic component while the pro-

cess noise vector gives the stochastic component. For this experiment the system model is a linear stochastic difference equation based on a constant velocity assumption similar to the one used by Birchfield [3]. The process noise is modeled as a Gaussian with constant mean and variance throughout the sequence. Now the interesting question is how to handle any sort of head-pose which is absent from the training data, in a test sequence? The answer is, any such pose can be roughly approximated as an in-plane rotation of a training template. This makes up for the missing poses in the training data.

4. *Likelihood $p(z_n \mid x_n)$ measure*

   Every particle has a patch in the test image and a training template associated with it. The image patch is extracted, based on the translation, in-plane rotation and scaling parameter values of the particle. The training template is assigned based on the Isomap (out-of-plane rotation) coordinates of the particle. Training template is taken to be the template closest to the particle in the 3D Isomap coordinate system. Figure 5.5 gives an illustration of the patch and the template associated with a particle. The likelihood function (weighting function) has 2 components : a) SSD and b) chamfer distance.

   (a) SSD:

   The mask (background pixels = 0) of the training template is applied to the image patch for computing the SSD between the two. This is done to minimize the measurement noise of the background pixels. The SSD weight of a particle is calculated as the negative exponential of the SSD between the training template and the image patch, corresponding to the particle. This is illustrated in Figure 5.5.d.

   (b) chamfer distance:

   If only SSD is used to compute the distance between the training tem-

plate and the image patch the weights become very sensitive to noise (due to background pixels in the patch which could not be masked by the training template). Also, the SSD distance is unsuitable to handle any kind of ambient light variations. Changing lighting conditions can be handled using a distance measure which would just measure the distance between the binary thresholded edges of the training template and the image patch. Two possible candidates to this distance measure are Hausdorff distance [9] and chamfer distance [2]. For this experiment, chamfer matching is used. Edge detection in the training template and the training patch is done using Canny edge detector [5]. Figure 5.6 shows the results of Canny edge detection and chamfer transform on a training template and a test image. The chamfer-weight of a particle is calculated as the negative exponential of the sum of the product of the edge-detected training template and the chamfer transform of the image patch corresponding to a particle.

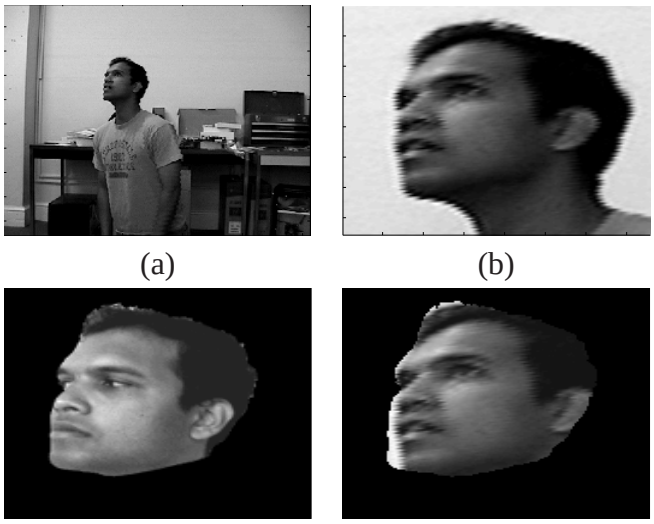
Weight of a particle is calculated as

$$W = (\textit{normalized SSD weight} + \textit{normalized chamfer weight})/2$$

5. *State estimation* $E\{g(x_t)\}$

   The true state x_t of the person's face can be estimated as the weighted mean of all the particles. The weighted mean of all the parameters gives position, pose and scale of the person's face.

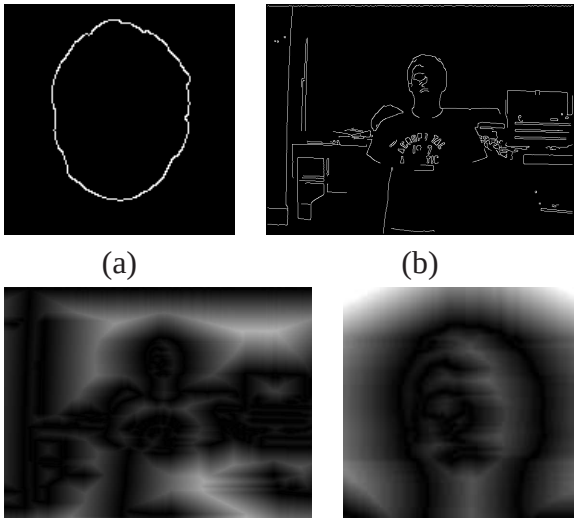
Figure 5.7 shows the block diagram of 'Isomap tracking with particle filter' algorithm for a single particle. It gives an overview of the steps in the algorithm. The details of the processes within the blocks are explained in section 5.3.2. The results of this algorithm on an image sequence are demonstrated in chapter 6.



(a) (b)

(c) (d)

Figure 5.5: (a) Image from the test sequence. (b) The image patch associated with a particle. (c) The template closest to the particle. (d) Image patch with the mask of the training template (background pixels = 0) applied to it.



(a) (b)

(c) (d)

Figure 5.6: (a) Training template after Canny edge detection. (b) Test image after Canny edge detection. (c) The chamfer transform of (b). (d) Image patch from (c) corresponding to a particle. The chamfer weight is computed by finding the distance between (a) and (d).

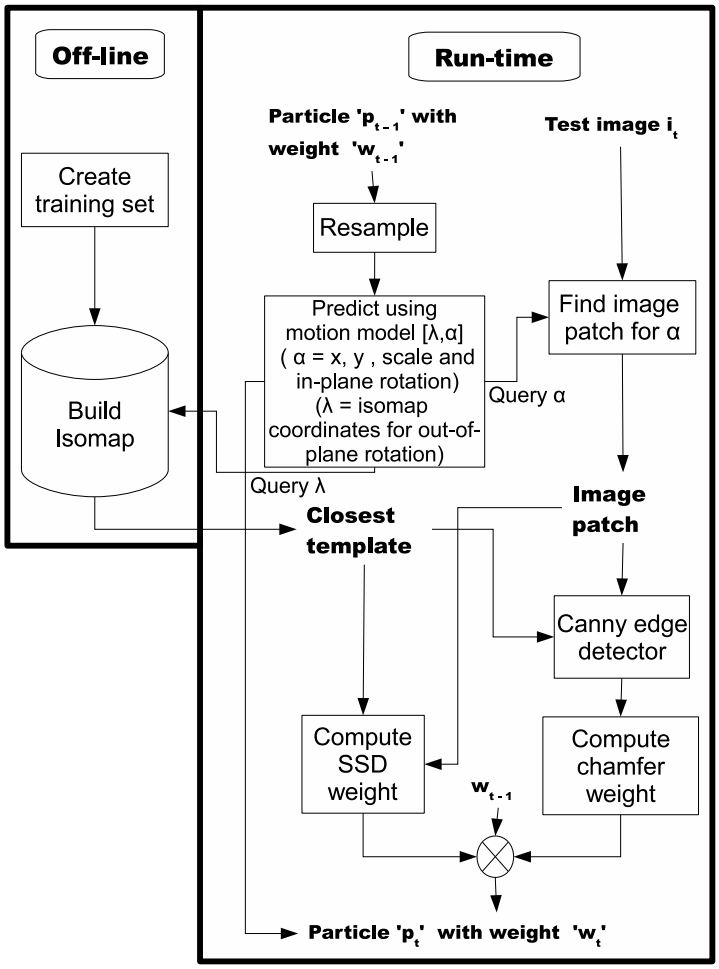


Figure 5.7: Block diagram of ‘Isomap tracking with particle filter’ algorithm for a single particle.

Chapter 6

EXPERIMENTAL RESULTS

6.1 Effects of resampling

The resampling technique, discussed in section 4.5.2, is used to reduce the variance on weights and to eliminate the weights that carry little information about the true state of the object being tracked. Figures 6.1 and 6.2 demonstrate the effectiveness of resampling. When resampling is not used, majority of the particles drift to regions which carry very little or no information about the person's head, within a few iterations. These particles have low weights and contribute very little in estimating the true posterior probability density of the person's face-pose, giving poor tracking performance. In Figure 6.1 it can be seen that many low-weight particles drift away from the face gradually. When resampling is used, most of the low-weighted particles get eliminated at every iteration. Thus, most of the remaining particles provide significant contribution to estimating the posterior density of the system-state. In Figure 6.2 it can be seen that resampling retains particles with high weights and thus helps in reducing the variance of weights. It can be concluded that when a good importance density cannot be computed for a process, resampling is a good technique to reduce the effects of degeneracy of particles.

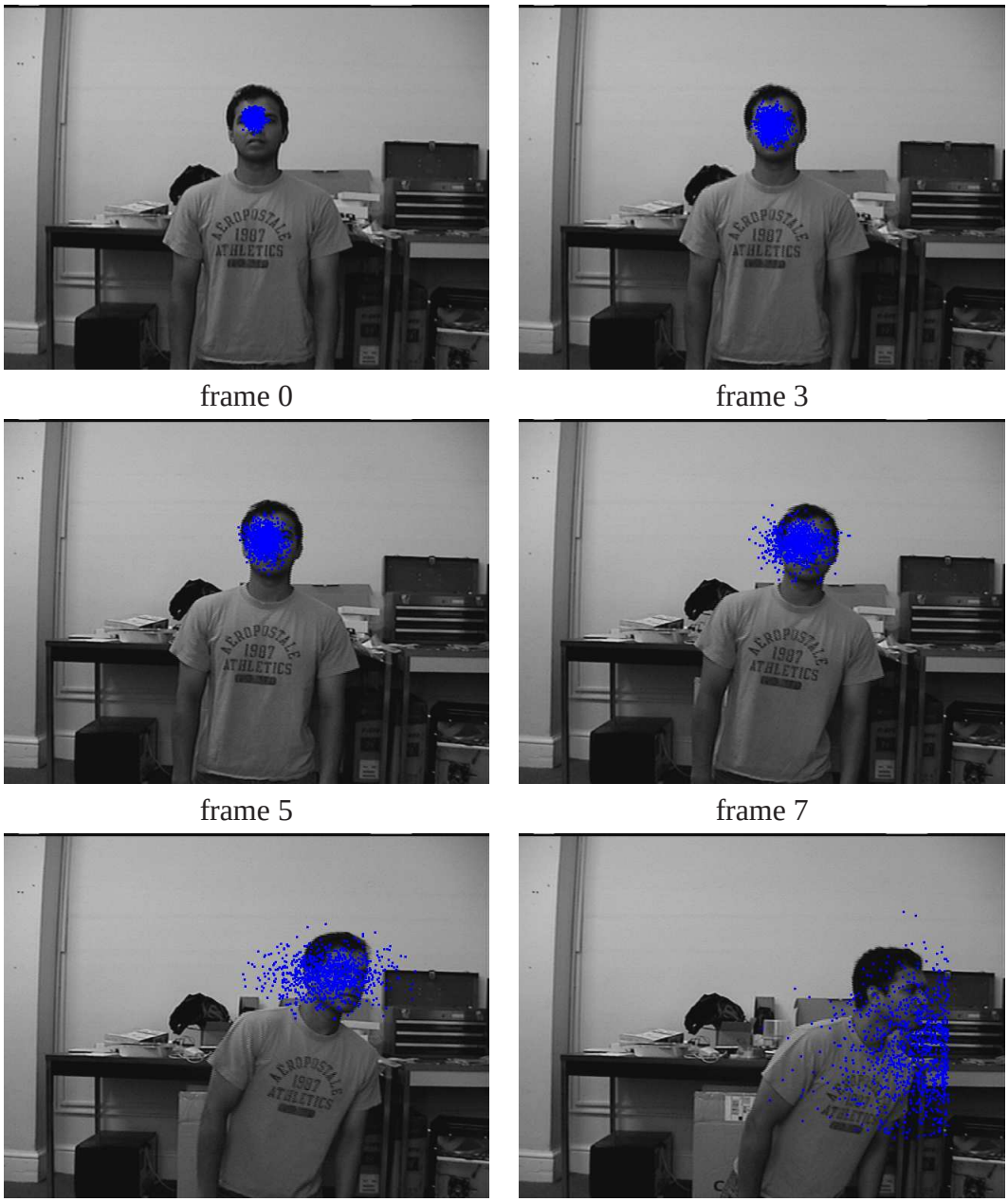


Figure 6.1: In the images, the blue dots show the particles. These images show the behavior of the particles in absence of resampling.

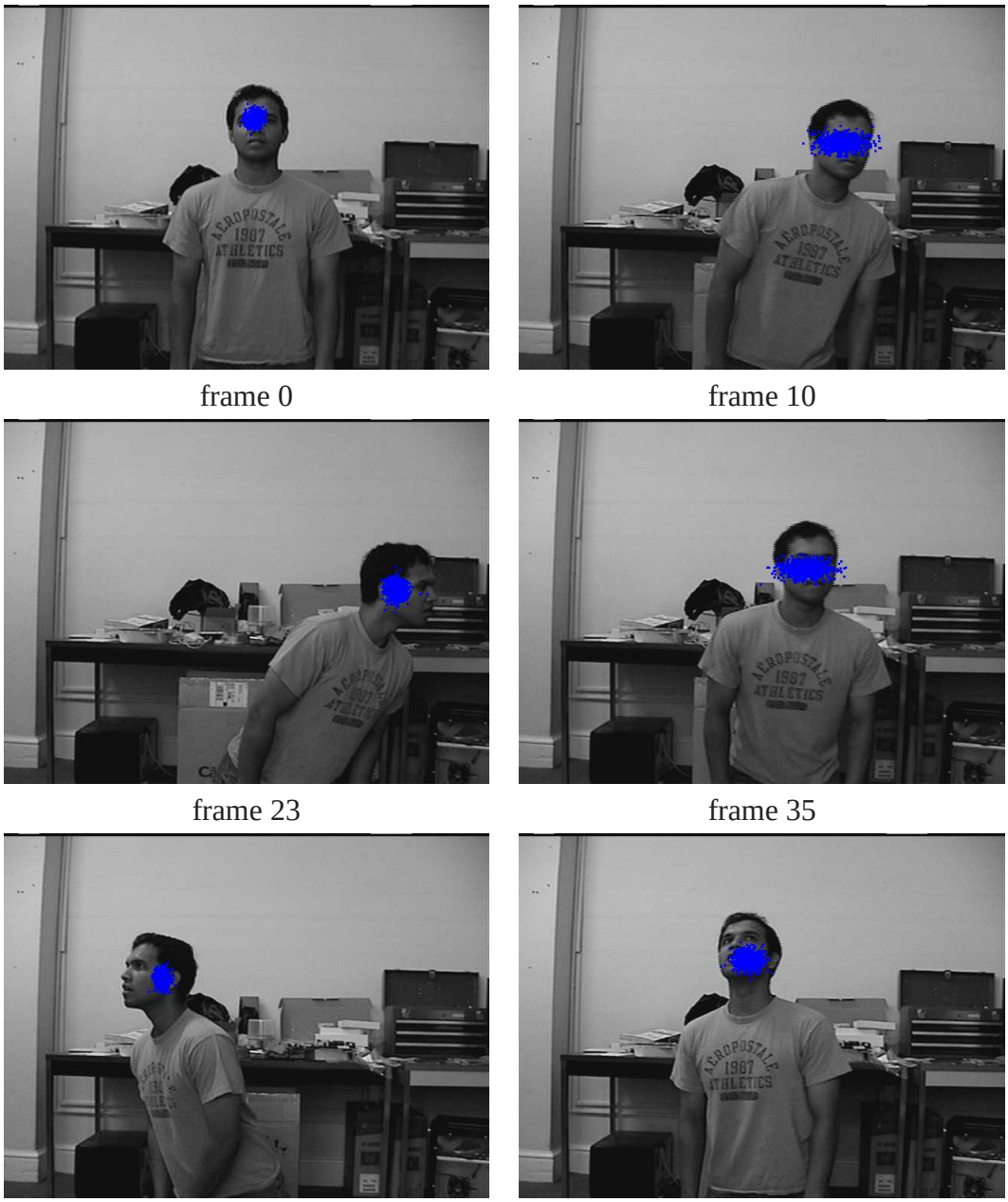
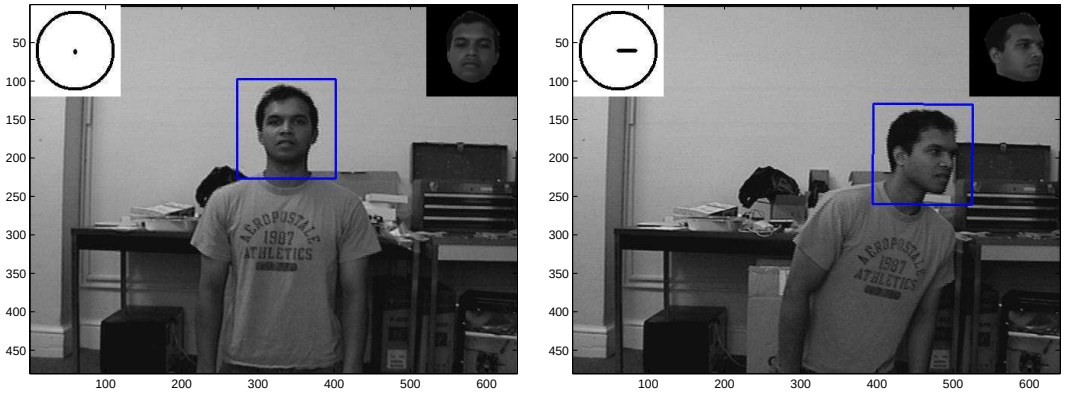
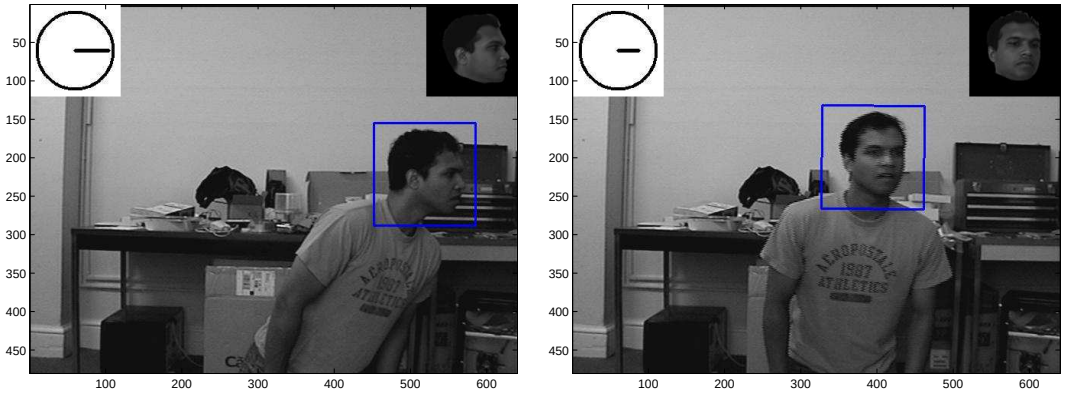


Figure 6.2: In the images, the blue dots show the particles. These images show the behavior of the particles when resampling is used.

6.2 Tracking results

A particle filter with 1000 particles was used for tracking. The tracker was tested on an image sequence with the person's head undergoing translation and out-of-plane rotation. Tracking results are shown in Figure 6.3 and 6.4. The vector inside the circle, in the left-hand corner of the test images, indicates the head-pose. The length of the vector is proportional to the amount of rotation of the head and the angle indicates the pose. The tracker successfully tracked the head of the person with a reasonably accurate estimation of the 3D pose. The tracker handled the out of plane rotation and the rapid translation of the person's head.





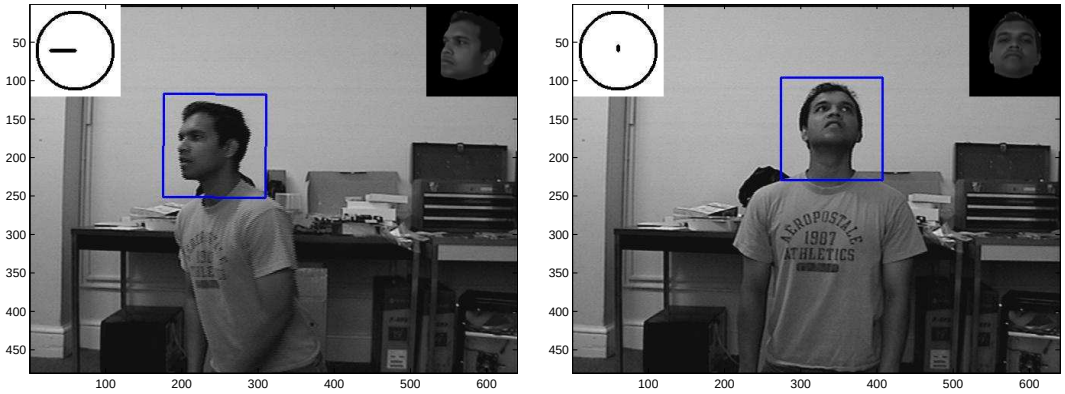


Figure 6.3: The blue square is the tracker. The vector inside the circle at the top-left corner gives the 2D out-of-plane rotation. In the circle, the vertical axis represents the up-down pose and horizontal axis represents right-left pose. The length of the vector is proportional to the amount of rotation and the angle gives the pose. The top-right corner of the images show the closest training template. The tracking results are continued in the next Figure.

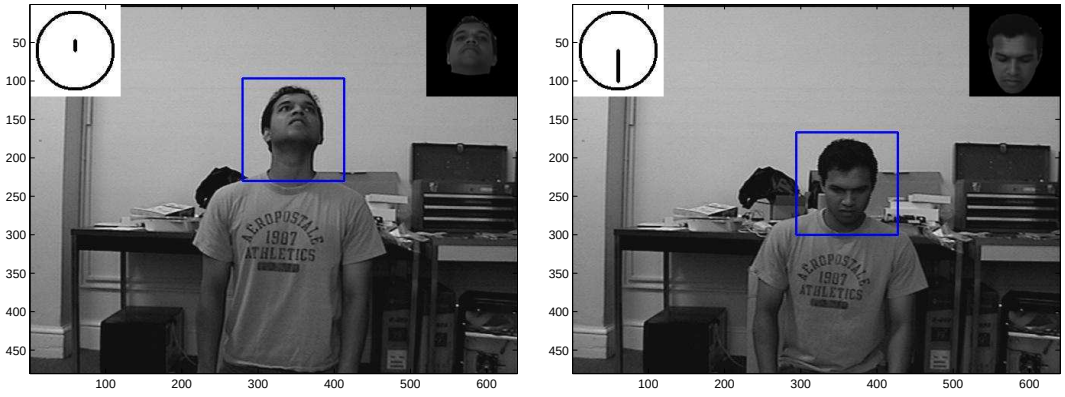


Figure 6.4: Tracking results continued from the previous Figure.

Chapter 7

CONCLUSION

This study of dimensionality reduction techniques showed that Isomap gives very good results for nonlinear dimensionality reduction. Isomap is an unsupervised learning technique with the only input to the algorithm being the neighborhood k. One drawback for Isomap is if new training data is to be added to the original training data the whole of Isomap has to be computed again. If this problem is solved, with the technique being unsupervised, training data can be build on-line with minimal starting knowledge of the data. This would be extremely useful because with a few images a tracker can start tracking and building a new training-set at the same time with no user intervention. The study of particle filtering showed that it is expected to perform better than Kalman filtering when the posterior density of the state cannot be estimated by a Gaussian. Another advantage of using a particle filter is that if the particles run independent of each other the system can be parallelized, thus offering high computational speeds. The 'Isomap tracking with particle filter' algorithm combines these two techniques and thus gives a robust tracking framework. The algorithm not only tracks a person's face in an image sequence but also gives a good estimate of the pose of the person at each frame. This technique can be extended to track any kind of a rigid object if its training data is available.

Bibliography

[1] M. Arulampalam, S. Maskell, N. Gordan, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian filtering. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2000.

[2] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. *Proc. 5th Int. Joint Conf. Artificial Intelligence*, 2:659–663, 1977.

[3] Stan Birchfield. An elliptical head tracker. In *Proceedings of the Thirty-first Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1710–1714, November 1997.

[4] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 232–237, June 1998.

[5] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[6] A. Doucet. On sequential Monte Carlo methods for Bayesian filtering. Technical Report Tech. Rep., Dept. Eng.,Univ. Cambridge, UK, 1998.

[7] A. Doucet, J.F.G. de Freitas, and N.J. Gordan. *Sequential Monte Carlo Methods in Practice*. New York : Springer-Verlag, 2001.

[8] H.C. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, April 1993.

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

[10] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, February 1998.

[11] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 893–908, 1998.

[12] V. de Silva J. B. Tenenbaum and J. C. Langford. Data sets for nonlinear dimensionality reduction. http://isomap.stanford.edu/datasets.html.

[13] Ian Jolliffe. *Principal component analysis*. New York : Springer-Verlag, 1986.

[14] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, 82:35–45, March 1960.

[15] Andrew Kiruluta, Moshe Eizenman, and Subbarayan Pasupathy. Predictive head movement tracking using a Kalman filter. *IEEE Transactions on Systems, Man, and CyberneticsPart B: Cybernetics*, 27(2):326–331, April 1997.

[16] M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3d models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):322–336, April 2000.

[17] P. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, Inc, 1979.

[18] K. Nummiaro, E. Koller-Meier, and L.J. Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, January 2003.

[19] M. Pardas and E. Sayrol. A new approach to tracking with active contours. In *International Conference on Image Processing*, volume 2, pages 259–262, 2000.

[20] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[21] Lawrence K. Saul and Sam T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[22] R. Souvenir and R. Pless. Image distance functions for manifold learning. *Image Vision Comput.*, 25(3):365–373, 2007.

[23] Rajan Srinivasan. *Importance sampling - Applications in communications and detection*. Springer-Verlag, 2002.

[24] Amarnag Subramanya and Stanley T. Birchfield. Extension and evaluation of mds for geometric microphone array calibration. In *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)*, September 2004.

[25] J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for non-linear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[26] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, December 1952.

[27] E. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. *Proc. of IEEE Symposium 2000 (AS-SPCC), Lake Louise, Alberta,Canada Oct. 2000*, pages 153–158, October 2000.