Applications of Diffusion Wavelets

Sravan Kumar Naidu Gudivada

Submitted for the degree of Master of Science (M.Sc.)

Department of Computer Science

THE UNIVERSITY OF YORK

June 2011

Abstract

Diffusion wavelets have been constructed on graphs in order to allow an efficient multiscale representation. This MSc thesis outlines how the diffusion wavelet framework can be applied to dense and sparse optical flow estimation as well as to the eigendiffusion faces for face recognition and fingerprint authentication. Diffusion wavelets are used for multiscale dimensionality reduction at different scales for feature representation.

Local image features are recorded by the extended bases scale functions at different scales calculated from the graph Laplacian. These features are then used in a dense as well as in a sparse optical flow estimation algorithm. We also used the same multiscale extended bases method for getting the orthonormalized projections of the covariance matrix of the training set of faces or fingerprints and we called those projections as eigendiffusion faces. By using eigendiffusion faces we calculated the low dimensional space weight components which are used to recognise faces or fingerprints using the minimum Euclidean distance of the weight vectors.

The proposed methodology was applied on different image sequences such as: Middlebury database, Hamburg taxi sequence, Andrea Hurricane image sequence, Infra-red meteosat image sequence, for image registration in a set of medical images of eye's cornea as well as for the ORL face databases, Yale face database, fingerprint verification competition dataset (FVC2000).

Contents

1 Introduction **1**

2 Literature Review **3**

2.1 Wavelets on Graphs 3

2.2 Optical Flow Estimation 10

3 Diffusion Wavelets **13**

3.1 Manifold Learning 13

3.1.1 Principal Component Analysis 13

3.1.2 Graph Based Algorithms 14

3.1.3 Locally Linear Embedding 14

3.1.4 Isomap 15

3.1.5 Laplacian Eigenmaps 15

3.1.6 Diffusion Maps 16

3.2 Diffusion Wavelets 18

3.2.1 The construction of Diffusion Wavelets 19

3.3 Multiscale dimensionality reduction using diffusion wavelets 24

3.3.1 The main algorithm 24

3.4 Summary 27

4 Face Recognition using Eigendiffusion Faces **29**

4.1 Face Recognition using Correlation 29

4.2 Face Recognition using Eigenfaces 29

4.2.1 Calculating Eigenfaces 30

4.3 Face Recognition using Fisherfaces 31

4.4 Face Recognition using Eigendiffusion faces 33

4.5 Summary 38

5 Optical Flow Estimation using Diffusion Wavelets **39**

5.1 The Diffusion Kernel 39

5.2 Markov Process and Diffusion Extended Bases 40

5.3 Estimating Dense Optical flow 42

5.4 Estimating sparse optical flow from the Euclidean distances 43

5.4.1 Scale Invariant Feature Transform 45
5.4.2 Calculating Diffusion Wavelet Extended Bases 48
5.4.3 Optical flow using Euclidean distance in the diffusion wavelet space 48
5.5 Summary 49

6 Experimental Results 50
6.1 Diffusion extended bases functions as feature descriptor in images 51
6.1.1 Multiscale feature representation of animal on tree branch image . . 53
6.1.2 Multiscale feature representation of skiing person image 54
6.1.3 Multiscale feature representation of hand image 54
6.1.4 Multiscale feature representation of ballerina image 55
6.1.5 Multiscale feature representation of face image 56
6.2 Optical flow and Image Registration by using Extended diffusion bases functions 57
6.2.1 Dense Optical flow Estimation 58
6.2.2 Sparse Optical flow Estimation for Hamburg Taxi Image Sequence . 74
6.2.3 Image Registration 75
6.3 Human face recognition and fingerprint authentication using eigendiffusion faces 76
6.3.1 Face Recognition 78
6.3.2 Fingerprint Authentication 87

7 Conclusion and Future Scope 92
7.1 Summary 92
7.2 Future Scope 93

List of Figures

3.1 Spectral energy powers of **T** and their corresponding multiscale eigen-space decomposition(Fig 1 from [4]) 19
3.2 Diagram for downsampling, orthogonalization and operator compression (triangles are commutative by construction) (Fig 6 from [10]) 20
3.3 Multi scale dimensionality reduction flowchart 25

4.1 Face recognition using the diffusion wavelets flowchart 35

5.1 Gaussian functions of the normalized differences in pixel intensity for two scale values that were used in our experiments,(a)$\sigma = 0.003$ and (b)$\sigma = 0.0003$. 40
5.2 Dense optical flow estimation using Diffusion Wavelets flowchart 45
5.3 Sparse optical flow using Diffusion Wavelets flow chart 46

6.1 Football players image multiple scale feature representation 51
6.2 Deer image multiple scale feature representation 53
6.3 Multiple scale feature representation of animal on the tree branch 54
6.4 Skiing person feature multi scale reduction 55
6.5 Multi scale feature representation of hand 56
6.6 Multiple scale feature representation of a ballerina. 56
6.7 Multiple scale feature representation of a face. 57
6.8 Feature representation of 20 x 20 block parts in Hamburg taxi image 59
6.9 Estimated optical flow in Hamburg taxi sequence 60
6.10 Feature representation of 20 x 20 block parts in Andrea hurricane image . . 62
6.11 Estimated optical flow in Andrea Hurricane image sequence 63
6.12 Feature representation of 20 x 20 block parts in Meteosat image 64
6.13 Estimated fluid optical flow in Meteosat image sequence 65
6.14 Feature representation of 20 x 20 block parts in Dimetrodon image 68
6.15 Dimetrodon dense optical flow 69
6.16 colormap 70
6.17 Dimetrodon estimated and ground truth flow 71
6.18 Feature representation of 20 x 20 block parts in Venus image 71
6.19 Venus Optical flow 72
6.20 Venus estimated and ground truth flow 73
6.21 Sparse optical flow estimation in Hamburg taxi sequence 75

6.22 Dense Optical flow estimation in Cornea layers image sequence 77
6.23 Sparse Optical flow estimation in Cornea layers image sequence 78
6.24 ORL database 79
6.25 ORL Mean face 80
6.26 140 out of a total 198 of eigendiffusion faces of ORL training faces 80
6.27 Face recognition rate with respect to taken eigendiffusion faces from 198 in ORL 82
6.28 Reconstructed ORL database 83
6.29 Face recognition rate with respect to different ORL train set and test set as 100 % database i.e 400 faces 84
6.30 Face recognition rate with respect to different ORL train and test set cases 85
6.31 Yale Face Database B 86
6.32 Yale Mean face 87
6.33 66 out of a total 74 of eigendiffusion faces of Yale training faces 87
6.34 Reconstructed Yale database 88
6.35 sub sampled set B of database DB1 from the FVC2000 90
6.36 Mean fingerprint 90
6.37 16 out of a total 39 of eigendiffusion faces of fingerprints 91

List of Tables

6.1 Football players image multi scale reduction details 52
6.2 Deer image multi scale reduction details 53
6.3 Animal on branch of tree feature multi scale reduction 54
6.4 Skiing person feature multi scale reduction 55
6.5 Hand feature multi scale reduction 55
6.6 Multiple scale dimensionality reduction of a ballerina 57
6.7 Multiple scale dimensionality reduction of a face 58
6.8 Angular and flow error from the Dimetrodon and Venus sequences for the method proposed in this thesis, szymon paper [40] and method from Baker et.al [33] 74
6.9 Faces which are not correctly recognised when the eigendiffusion faces are 198,train set as 50 % of ORL database and test set as 100 % ORL database 81
6.10 Face recognition rate with respect to taken eigendiffusion faces from 198 where train set as 50 % of ORL database and test set as 100 % ORL database 82
6.11 Face recognition rate with respect to different ORL train set and test set as 100 % database i.e 400 faces 84
6.12 Face recognition rate with respect to different ORL train and test set cases 85
6.13 Yale face recognition rate with respect to different train set cases and and test set as 100 % database i.e 165 faces 89
6.14 Yale face recognition rate with respect to different train set and test set cases 89
6.15 Fingerprint Authentication rate with respect to different train set, where test set as 100 % database i.e 80 fingerprint images 91
6.16 Fingerprint Authentication rate with respect to different train and test set where test set = 80 - train set 91

List of Algorithms

3.1 Multiscale representation at different scales using diffusion wavelets construction 22
3.2 Modified Gram-Schmidt with pivoting columns 23
3.3 Multi scale dimensionality reduction by using Diffusion Wavelets 26
4.1 Face recognition using the diffusion wavelets 36
5.1 Dense optical flow using Diffusion Wavelets 44
5.2 sparse optical flow using Diffusion Wavelets 47

Acknowledgements

First of all I would like to thank my supervisor, Dr. Adrian G. Bors for his kind guidance throughout the course of my MSc, helping me understand research material and proof reading my documents and thesis. I am grateful to Dr. Richard Wilson for his suggestions during the literature presentation. I am very thankful to EURECA[1] scholarship under the Erasmus Mundus scheme for fully funding my masters degree here at University of York. Especially, I would like to thank Dr. Suresh Manandhar for his support as co-ordinator of EURECA. I acknowledge the wonderful multicultural and multilingual environment created by my fellow research students in Computer Science Department and CVPR group; especially Touqeer Ahmad, Abhishek Dutta, Dr. Ankur Patel, Eliza Xu, Lichi Zhang, Zhihong Zhang, and all others.

I have furthermore to thank people from Indian Institute of Information Technology and Management - Kerala, India. Especially, Dr. Elizabeth Sherly (Director (In Charge) and Professor), Dr. Venkatesh Choppella (was Associated Professor), Dr. T. K. Manoj Kumar (Associate Professor), and Mr. T. Radhakrishnan (Chief Technology Officer) for their help and suggestions to choose research at University of York under EURECA programme.

I am thankful for all the love, prayers and best wishes from people back at home and my friends.

[1]http://www.mrtc.mdh.se/eureca/

Declaration

I declare that all the work in this thesis is solely my own, except where attributed and cited to another author.

Chapter 1

Introduction

Modelling features as well as dimensionality reduction for data representation is very important in fields such as information theory and machine learning, image processing, computer vision, etc.

Diffusion Wavelets have been developed for representing data using multiscale extended bases with the aim to define their relational structure at each level of the multiscale representation [4]. The adjacency matrix representing the graph Markov transition matrix is constructed with the entries defined as the probability of transition between any pair of points from the given data set. Diffusion wavelets lead to a multiscale dimensionality reduction, which has the benefit of being able to handle non-symmetric Markov transition matrices. This approach is similar to Laplacian eigenmaps [24] and Locality Preserving Projections (LPP) [25]. Here, at each level we achieve a reduction in the dimensionality of the orthonormalized projections called extended bases of Markov transition matrix and these extended bases model image features. Diffusion wavelets are applied to represent local image features with the aim of the estimating optical flow from image sequence. Meanwhile we also use these for calculating eigendiffusion faces for face recognition and for authenticating fingerprints.

Estimating accurate optical flow is a very challenging task when the scenes present significant noise and illumination variation. Diffusion wavelets can be constructed on manifolds, graphs and allow an efficient multiscale representation by applying large powers of an operator on the Markov transition matrix. We describe in this report how can the extended bases functions at each level represents image features. The extended bases functions at last level preserve the better feature information in images than those at the previous levels, because the diffusion wavelets algorithm removes noise at each level.

For dense optical flow estimation, initially we segmented both frames $\mathbf{I}_1$ and $\mathbf{I}_2$ into blocks of pixels and calculate for each block a feature representation by using the diffusion wavelet algorithm, i.e we calculate the extended bases functions at the respective last level of orthonormalization for each block of pixels. Now, we considered each block in frame $\mathbf{I}_2$, and we consider a search area around the block from $\mathbf{I}_1$. We find the correspondence by considering the locations of two pixel blocks each from a different frame, which have the minimum Euclidean distance between their diffusion bases vectors. This is described in Section 5.3 .

For sparse optical flow estimation, initially we applied scale invariant feature transform (SIFT) [32] on frame $\mathbf{I}_2$ to find the key points in the frame. Then we define blocks of pixels by taking each key locations as the center of the block from $\mathbf{I}_2$ and consider its corresponding search window in frame $\mathbf{I}_1$. Now, we calculate the extended bases functions of each block. We find the block in the search window, which has the closest diffusion bases functions to that of the reference block in frame $\mathbf{I}_2$. We repeat the same step for all SIFT selected blocks in $\mathbf{I}_2$. This is described in Section 5.4. We use the same optical flow concept for image registration.

We applied the diffusion wavelets algorithm for calculating eigendiffusion faces for face recognition. We find the covariance matrix of the training set of faces (input data labelled with known classes) and then apply the diffusion wavelets algorithm on the covariance matrix. Then we get the extended bases functions for the last level, these functions are called as eigendiffusion faces. These eigendiffusion faces are used for calculating weights in the high dimensional space of each face in the training set in order to decrease the dimensional representation of faces. Now we have weights for each face in the training set. We process a face by our algorithm and estimate the weight of that face by using the eigendiffusion wavelets and then calculate the Euclidean distance of this weight to all the weights from the training set. Now we find which face class in the training set has the minimum Euclidean distance in the diffusion bases space and that face class is assigned to the respective face in the training set. This algorithm is described in Section 4.4 . We used the same method for fingerprint authentication.

In Chapter 2, we review various methods of wavelets on graph, spectral methods for dimensionality reduction, optical flow estimation method as well as face recognition methods. In Chapter 3, we describe the diffusion maps algorithm, construction of diffusion wavelets and multiscale dimensionality reduction using the diffusion wavelets. In Chapter 4, we describe the eigenfaces, fisherfaces and our proposed eigendiffusion faces, then we explain our proposed method for face recognition and fingerprint authentication using the eigendiffusion faces. In Chapter 5, we describe our proposed methods for dense and sparse motion estimation and in Chapter 6, we describe the entire methodology. In the experimental results chapter, we enclosed the results of feature representation by using multiscale dimensionality reduction, sparse and dense optical flow estimation for various image sequences and image registration results on images of the cornea layer, then we provide face recognition and fingerprint results by using the eigendiffusion faces. In the final chapter 7, we have the conclusion and discuss future research work.

Chapter 2

Literature Review

The applications of this work is to estimate dense and sparse optical flow, face recognition, and also for fingerprint authentication by using the diffusion wavelets. In this chapter we will give a brief review of spectral graph methods for dimensionality reduction, diffusion wavelets applications, dense and sparse optical flow earlier methods, face recognition using the various approaches.

2.1 Wavelets on Graphs

Wavelets [26] are a class of a functions used to localize a given function in both space and scaling. Wavelets can be constructed from a function, sometimes known as a mother wavelet, which is confined in a finite interval. The mother wavelet is used to generate a set of functions through the operation of scaling and dilation applied to the mother wavelet. This set forms an orthogonal or biorthogonal bases (in fact they can form frames as well), that allows using inner products to decompose any given signal like in Fourier analysis. Wavelets are better for modelling than Fourier analysis because wavelets are not loosing the space information when moving to the frequency domain. Classical wavelets are constructed by dilating and scaling a single mother wavelet. The transform coefficients are then given by the inner product of the input function with the dilated and scaled waveforms. Directly extending this construction to a arbitrary weighted graphs is problematic, as it is unclear how to define scaling and dilation on an irregular graph. To overcome this problem, we use spectral graph domain [22], by using the bases consisting of the eigenfunction of the graph Laplacian.

Laplacian Pyramids [27], describe a technique for image encoding by the local operators of several scales. Here, the code elements are localized in the spatial frequency as well as in space. Pixel to pixel correlations are first removed by subtracting a low pass filtered copy of the image from the image itself. The result is a net data compression since the difference, image has low variance and entropy. Further data compression is achieved by quantizing the difference image. These steps are then repeated to compress the low pass image. The encoding process is equivalent to sampling the image with Laplacian operators of many scales. Thus, the algorithm tends to enhance salient image features. It is well suited for many image analysis tasks as well as for image compression.

Maggioni and Coifman [4], proposed diffusion wavelets methodology and the general theory for diffusion wavelets decomposition based on compressed representation of dyadic powers of a diffusion operator. The diffusion wavelets were described with in a framework that can be applied on smooth manifolds as well as on graphs. Their construction interacts with the underlying graph or manifold space through repeated applications of a diffusion operator $\mathbf{T}$. In this report by using this algorithm we propose various applications such as dense and sparse optical flow estimation, face recognition, fingerprint authentication and image registration.

Maggioni and Mhaskar [41] have developed a theory of diffusion polynomial. They constructed a multiscale matrix based on orthonormal bases for the $\mathbf{L}_2$ space of a finite measure space. The approximation properties of the resulting multiscales are studied in the context of Besov approximation spaces, which were characterized both in terms of suitable K-functional and the frame transforms. The major condition required was the uniform boundedness of a summabilility operator. The authors provide sufficient conditions for this to hold in the context of a very general class of metric measure spaces.

Geller and Mayeli [42] studied a construction for wavelets on compact differentiable manifolds. They define scaling using the pseudo differential operator tLe^{-tL}, where t is a scale parameter and $\mathbf{L}$ is the manifold Laplace-Beltrami operator, and in order to obtain the wavelets they applied a pseudo differential operator to a delta impulse. Authors also studied the localization of the resulting wavelets.

Hammond, Vandergheynst, and Gribonval [43] , proposed a novel method for constructing wavelet transforms of functions defined on the vertices of an arbitrary finite weighted graph. This method was based on defining the scaling using the graph Laplacian $\mathbf{L}$. Given a wavelet generating kernel g and a scale parameter t, they define the scaled wavelet operator as $\mathbf{T}_g^t = g(tL)$. The spectral graph wavelets were then formed by localization in a small scale limit. Subject to an admissibility condition on g, this procedure defines an invertible transform. Authors explored the localization properties of the wavelets in the limit of fine scales and they also presented a fast Chebyshev polynomial approximation algorithm for computing the transform that avoids the need for diagonalizing $\mathbf{L}$. This method is closely related to the method [41] , in a more general quasi metric measure space setting.

Bremer et all [5] proposed the construction of diffusion wavelet packets, which generalize the classical wavelet packets, and enrich the diffusion scaling functions and wavelet bases of [4]. Authors explained construction of diffusion wavelet packets with two examples, first example was anisotropic diffusion on a circle which illustrates how the anisotropy can affect the structure of the wavelet packets and the associated time-frequency analysis, in particular the representation and compression of the functions. Secondly, they applied Laplace-Beltrami diffusion on a sphere and the operator **T** is obtained through the Laplacian-Beltrami normalization and they calculate diffusion wavelets and wavelet packets for the operator **T**. They explained about the best bases algorithm as it applies to diffusion wavelet packets and discussed its applications like data compression and de noising. Diffusion wavelet packets allow for flexible multiscale space-frequency analysis for the functions on the manifolds and graphs.

Mahadevan and Maggioni [6] proposed the problem of automatically constructing of an efficient representations of bases functions for approximating value functions based on analysing the structure and topology of the state space. Two approaches for approximating function, one is by using the eigenfunctions of the Laplacian, in effect performing a global Fourier analysis on the graph and the second approach is based on diffusion wavelets, which generalize classical wavelets to graphs using multiscale dilations induced by powers of a diffusion operator or random walk on the graph. These two approaches together form the new generation of methods for solving large Markov decision process, in which we can learn the underlying representation.

Szlam et all [7] proposed a top-down frame work for multiscale analysis on manifolds and graphs. The framework for building natural multiresolution structures on manifolds and graphs was introduced, that generalizes the construction of wavelets [4] and wavelet packets [5] in Euclidean spaces. This allows the study of the manifold and of the functions on it at various scales, which are induced naturally by the geometry of the manifold. This construction proceeds, bottom-up, from the finest scale to the coarser scale using the powers of the diffusion operator as dilation and the rank constraint to sample the multiresolution subspace. The top-bottom construction yields well-localized bases of smoothed Haar wavelets and other local cosines functions. The second eigenfunction of a diffusion on the manifold or graph is used to split the spaces into two parts. Then each part is recursively subdivided further, by using the second eigenfunction where the restriction of a diffusion operator to functions is essentially supported on each part. This yields a dyadic decomposition of the space i.e the dyadic decomposition of Euclidean spaces. This method yields associated local cosine packets on manifolds, generalizing local cosines in Euclidean spaces. These constructions have direct applications to the approximation, de noising, compression, and learning of functions on a manifold and this approach is promising for manifold approximation and dimensionality reduction.

Maggioni et all [8] proposed a biorthogonal diffusion wavelet for multiscale representation on manifolds and graphs. Initially they discussed about the diffusion-driven multiscale analysis on Manifolds and Graphs [7] and it has two types of approaches top-down construction and bottom-up construction. Bottom-up construction generalize orthogonal diffusion wavelets which is described in [4]. The construction of orthogonal diffusion wavelets builds smooth, local orthonormal bases for the scaling and wavelet spaces. It generalizes the construction to allow for biorthogonal bases as in the classical setting. It introduces an extra degree of flexibility. The particular interest, is the possibility of constructing sparser bases i.e bases whose elements have smaller support. One of the primary motivation for diffusion wavelets is the desire to build bases well adapted to the spectrum of a diffusion operator, but more compactly supported than bases consisting of eigenvectors. The diffusion bases space V_j spans approximately same space as the eigenvectors $\{e_l|\lambda_l^{2^j} \geq \epsilon\}$, but the diffusion wavelets are concentrated on a small set with exponential decay whereas the eigenvector e_l are supported on the whole graph. The orthonormal bases calculated in diffusion wavelets from sums of selected columns of the input matrices $\mathbf{T}_j$, but they are less compactly supported. This suggests to choose biorthogonal bases for the scaling space by choosing simply a set of columns of the input matrix $\mathbf{T}_j$. In the case of Markov chains it is convenient to represent states at a certain time scale in terms of probability distribution at the same scale and the columns of corresponding power of the Markov matrices are natural.

Maggioni and Mahadevan [9] proposed fast direct policy evaluation using multiscale analysis of Markov diffusion processes. Policy evaluation is a critical step to approximate solution of large Markov decision process, we require $O(|N|^3)$ to directly solve the Bellman systems of $|N|$ linear equations where, $|N|$ is the state space size in the discrete case and the sample size in the continuous case. In this paper they applied multiscale representation of diffusion wavelets for analysis on graphs to design the faster algorithm for policy evaluation.

Maggioni and Coifman [11] proposed multiscale analysis of data sets with Diffusion Wavelets. They explained multiscale analysis of document corpora, considered cloud of digital data. They given 1047 articles from Science News, from which they collected 2036 words chosen as being relevant for this body of document. A document-word matrix whose entry (i, j) is the frequency of the j^{th} word in the dictionary in the i^{th} document was constructed. Each document is categorized as belonging to one of the following fields: Anthropology, Astronomy, Social Sciences, Earth Sciences, Biology, Mathematics, Medicines or Physics. Initially they applied diffusion maps methodology [1] on this dataset to embed high-dimensional graph into Euclidean space. This embedding to be meaningful when the different categories appear well-separated. A simple K-means or hierarchical clustering algorithm ran on the Euclidean space vectors, yields clusters which match quite closely the given labels. This would correspond to a particular choice of kernel K-means or hierarchical clustering motivated by diffusion distance. We do not have space here and if we apply on multiscale construction on data then we get much more information to analyse high dimensional data and here the kernel is iterated over the set, induces a natural multiscale structure, that gets the data organized coherently, in space and scale. Here, author applied diffusion wavelets [4] on the data and described scaling functions at various scales represented on the set embedded in R^3.

Mahadevan [12] proposed Adaptive mesh compression in 3D computer graphics using multiscale manifold learning. They investigated compression of 3D objects in computer graphics using manifold learning. Spectral compression uses the eigenvectors of the graph

Laplacian of an object's topology to compress 3D objects. Object models can have $> 10^5$ vertices and computationally it is challenging for 3D compression. They explained spectral mesh compression by using Fourier analysis and it is compared with the spectral mess compression with wavelet bases. Fourier bases vectors are global, they do not provide multiscale analysis and poorly capture edges and local discontinuities. These limitations provide tangible consequences, it is hard to approximate piecewise smooth mesh geometries. To deal with the challenges of large graph, they used divide and conquer approaches for decomposing large graphs into a set of sub graphs and compute local bases function i.e calculating bases by applying diffusion wavelet methodology on each sub graph and combine all local bases functions.

Wild [13] proposed a multiscale, graph-based approach to 3D image analysis using diffusion wavelet bases. They described structure preserving compression of image sequences, regarded as a 3D, or more precisely a 2D+ time data set. They modelled the whole image sequence as a weighted graph, where the edge weights describes the local similarity between certain data points, which present the nodes of the graph. The vertices can be chosen as the whole set of pixels, or due to complexity considerations as a subset, e.g. using a downsampled version of the sequences by filtering or by feature point selection procedure. In order to define the edges of graph and their corresponding weights, they encoded the local relation between vertices from which algorithm will learn the global and multiscale structure. They used $w(u, v) = exp(-\rho(u, v)^2)$, where $\rho(u, v)$ may be the difference of the intensities in u,v. For complexity reason, it is reasonable to set $\rho(u, v) = \infty$, if v is not a neighbourhood of u. On the graph, they built diffusion wavelet bases. Instead of using the geometry of the data set R^3 (pixel distances on a regular grid), they used the structure of the data given by the connectivity of the graph nodes during a diffusion process. This diffusion process can be as learning the global structure of the data using local relationship. The diffusion operator is formed from the graph representing the input data, the wavelet bases depends on the data and the whole process is data-adaptive and non-linear. We can analyse the functions on the graph by computing coefficients from the constructed diffusion wavelet bases. In this way all the information of function is maintained in the sequence of coefficients. The salient information is reflected in the largest coefficients just like for the usual wavelet transform.

Coates, Pointurier and Rabbat [14] proposed a procedure for estimating a full set of network path metrics, such as loss or delay, from a limited number of measurements. This method achieves the strong spatial and temporal correlation observed in path level metric data, which arises due to shared links and stationary components of the observed phenomena. They applied diffusion wavelet on routing matrix to generate bases in which the functions are compressible. This allows to achieve powerful non-linear estimation algorithm that support for sparse solutions. They applied this approach on specific example, which is end to end delay estimation in a network whose topology is known. From the results, we can recover network mean end to end delay with 95% of accuracy while monitoring the 4% of the routes. They explained the three key points to the network monitoring framework, such as compressing transformation, which is achieves by using the diffusion wavelets, a nonlinear estimation scheme which favours to sparse solution, and a path selection algorithm for determining the optimal monitoring strategy.

Zhu et all [15] proposed a 3D shape retrieval approach based on diffusion wavelets which generalize wavelet analysis and associated signal processing techniques to functions on manifolds and graphs and it is a multiscale diffusion wavelet approach for 3D shape representation and matching. Previous 3D matching methods are based on either on the topological information of the models or on their scatter point distribution information. In this method they use both topological and point distribution information for more effective matching. They calculate multiscale feature representation vectors at each level of the training set and then calculated feature vector of test 3D object for matching. Practically, we calculate distance metric between test shape feature vector and each shape feature vector from the train data set. Then authors identify the class of test 3D object by setting the threshold i.e distance metric should be lesser than of the threshold. In this case we have multiscale feature vectors and diffusion wavelets gives feature representation from finer to coarser by increasing the level. In order to invariant and robust matching they considered radial bases function Gaussian kernel which obtain the rotation and shift invariance. On the other hand scale variance is allowed as the models are matched across different scales, although scale invariance can also be attained if we modify the the covariance matrix with a tangential covariance in the Fisher discriminant analysis ratio computing. The covariance matrix is calculated by the summation of covariance of train data and covariance test data, which is useful to calculate Fisher discriminant ratio at each scale. They calculate Fisher discriminant ratio is by ratio of square of mean difference between train and test object to the covariance coefficient from the above calculated covariance matrix. As the Fisher discriminant ratio values are low and very close to zero then models are well matched but it gives rise to the problem of overflow in the computing and ranking of values. So, they considered the inverse of above Fisher discriminant ratio and choose the high ratio of training set of 3D objects matched with those for the test 3D objects.

Wang and Mahadevan [16] proposed a multiscale dimensionality reduction based on diffusion wavelets. They called this method as diffusion projections (DP), which is automatically reveals the geometric structure of the data at different scales and provides multiscale embedded representation for both symmetric and non-symmetric relationship matrix. For the symmetric case this approach can automatically identify the most appropriate dimensions for embedding low dimensional representation. For the non-symmetric, we don't need to symmetrize and repeat the same step for embedding low dimensional representation of the high dimensional input data. This algorithm mainly comprises three steps such are, constructing relationship matrix which is nothing but Markov transition matrix, then apply multiscale diffusion wavelet algorithm on relationship matrix and finally, choose the best low dimensional representation which we get it generally at last levels. They applied this approach on toy example: faces and it is almost similar to eigenfaces. In eigenfaces method, we used linear dimensionality reduction method PCA on covariance matrix to find the eigenfaces, but in this approach they used multiscale diffusion wavelets approach in place of PCA.

Wang, Mahadevan [17] proposed an approach to multiscale manifold alignment. In this approach, a hierarchical alignment that preserves the local geometry of each manifold and matches the corresponding instances across manifolds at different temporal and spatial scales. This approach is non-parametric, data-driven and automatically generates multiscale alignment by analysing the intrinsic hierarchical shared structure of the given input data set. For example, we consider two data sets, $x_i \in R^p$; $X = \{x_1, x_2, ..., x_m\}$ is a $p \times m$ matrix and $y_i \in R^q$; $Y = \{y_1, y_2, ..., y_n\}$ is a $q \times n$ matrix. X_l and Y_l are in correspondence: $x_i \in X_l \longleftrightarrow y_i \in Y_l$, where $X_l = \{x_1, x_2, ..., x_l\}$ is a $p \times l$ matrix and $Y = \{y_1, y_2, ..., y_l\}$ is a $q \times l$ matrix. Now, calculate similarity, diagonal and Laplacian of both data sets and represented as W_x, D_x, L_x for data set X and W_y, D_y, L_y for data set Y. They defined diagonal matrix Ω having μ on the top l elements of the diagonal; Ω_1 is an $m \times m$ matrix; Ω_2 and Ω_3^T is an $m \times n$ matrix; Ω_4 is an $n \times n$ matrix. Now combined data set representation in the matrix $Z = \begin{pmatrix} X & 0 \\ 0 & Y \end{pmatrix}$ is a $(p+q) \times (m+n)$ matrix. Combined diagonal matrix defined $D = \begin{pmatrix} D_x & 0 \\ 0 & D_y \end{pmatrix}$ is a and combined Laplacian matrix $\mathbf{L} = \begin{pmatrix} L_x + \Omega_1 & -\Omega_2 \\ -\Omega_3 & L_y + \Omega_4 \end{pmatrix}$ are both $(m+n) \times (m+n)$ matrices. F can be constructed by SVD and it is a $(p \times q) \times r$ matrix, where r is the rank of ZDZ^T and $FF^T = ZDZ^T$. In order to construct a matrix representing the joint manifold: $T = F^+ZLZ^T(F^T)^+$, where $+$ represents the Moore-Penrose pseudoinverse. Then apply diffusion wavelet on matrix T to explore the intrinsic structure of the joint manifold. After this compute mapping functions by multiplying the inverse transpose of matrix T with extended bases at level k and it is a size of $(p \times q) \times p_k$. Finally, they applied mapping functions to find correspondences between X and Y.

Essafi, Langs and Paragious [18] proposed a novel approach for the representation of prior knowledge for image segmentation, using diffusion wavelets that can reflect arbitrary continuous interdependencies in shape data. To the shape variation observed in the training data by means of diffusion wavelet. They used wavelets to represent the variation of shapes and they learn topology of the wavelet domain from the training data instead of relying on a predefined manifold, and this wavelet representation of topology is encoded in a diffusion kernel. This defined kernel allows to learn and define arbitrary wavelet hierarchies, and thus to make optimal use of the training data. The diffusion operator **T** on the set of embedded in a metric space by using either of their mutual distance in the mean shape or their joint modelling behaviour. Now applied multiscale diffusion wavelet algorithm on diffusion operator **T** to calculate bases of scaling functions and we use this scaling functions to calculate diffusion wavelet coefficient on the deviation from the mean of aligned shapes. Once they have all training diffusion wavelet coefficients then build a a model of the variation by means of the orthomax criterion, which allows to obtain a simple and compact hierarchical representation through a rotation of the model parameter system. In the lowest level the coefficients provide information for a coarse approximation and the localized variations are captured by the high level coefficients in the hierarchy. In order to reduce the dimension of coefficients representation for all coefficients scales they used PCA. This results gives the eigenvector and the corresponding eigenvalue of the covariance matrix of the diffusion wavelets coefficients at each level and their coefficients represent each training shape in this coordinate system. We can reconstruct a shape based on the model parameters i.e eigenvector and eigenvalues.

Suen, Lau, Yue [19] proposed a system for which leverage this technique to differentiate web access requests generated by Denial of Service (DoS) attacks from legitimate ones. This algorithm comprises mainly as two major parts such as reference profile construction and real-time anomaly detection and response. Reference profile construction is a supervised learning based detection system where a set of purely normal reference data, presented in the reference profile, is used to compare with new data. The reference data is cleaned in the data preparation phase and it comprises 3 typical steps for preparing web access logs are first performed namely, Data Cleaning, User Session Recognition, and Path Completion. The Feature Extraction and Embedding (FEE) then converts the cleansed user access sequence into feature vectors, and then projects them onto a reduced data-space via diffusion wavelets. In order to compare new incoming user session to the reference ones, they all have to be projected on to the same feature space. Therefore, the same FEE engine is used for processing new incoming user access session in anomaly detection and response. Real-time anomaly detection and response governs the the daily operation of our system by evaluating and processing all incoming requests. The user session of an incoming request is first identified and then Session Filter tries to matches the user session to a set of previously detected abnormal sessions and if it is found to be abnormal then drops that session. If the request is not abnormal then it will be combined with the access history of same session, and go through the Data Preparation which is used in earlier part. After that by using the diffusion wavelet we project the feature space for this session i.e FEE engine used for this. Distance based outlier scores are then computed for the user session by comparing them to the legitimate sessions in the reference profile. Finally, the session is passed for threshold based anomaly detection, where abnormal ones are added to the blocked list.

2.2 Optical Flow Estimation

Optical flow is the distribution of an apparent velocity of movement of the brightness patterns in sequence of image.

Horn and Schunck [48] proposed a method for determining optical flow. It is a method for finding the optical flow pattern is presented which assumes that the velocities of the brightness pattern varies smoothly almost every part in the image. It is based on the observation that the flow velocity has two components and that the basic equation for the rate of change of image brightness only provides one constraint. Then the smoothness of the flow was introduced as a constraint. An iterative method is to solve the resulting equation was then developed. This optical flow method is somewhat inaccurate since it is based on noisy, quantized measurements.

Corpetti, Memin, and Perez [49] proposed a new method for estimating fluid flows from image sequences. This method is the extension of the standard minimization-based approaches, where a two-fold robust objective function is minimized. The two parts, data term and the regularizer of the novel cost function specifically designed to suit image sequences of fluid flows. The data term is based on the continuity equation and it is alternative to the brightness constancy assumption. Concerning the regularization, they argued that only a second order regularizer is able to preserve completely the vorticity and divergence of the unknown flow. In this paper, they demonstrated merit of two ingredients on both synthetic and real satellite images.

Lucas and Kanade [50] proposed an image registration algorithm. It is the second seminal algorithm to estimate optical flow. Here, they take feature based approach as their algorithm matches the local windows in a sequence of image. Spatial intensity gradient information is used to direct the search for the position that yields best match using a type of Newton-Raphson iteration.

Besnerais and Champagnat [51] proposed a method for dense optical flow estimation by using the iterative local window registration. In this paper, they showed the usual Iterative-Warping Scheme encounters the divergence problems and proposed a modified scheme with better behaviour. It yields good results with a lower computational cost than the dense Lucas-Kanade algorithm.

Zitnick, Jojic, and Kang [52] proposed a method for jointly computing optical flow and segmentating video while accounting for a mixed pixels. It is a stochastic approach to address the above issue. Here, they used the a generational model with spatio and temporal constraints to produce the consistent segmentation from this, they estimated optical flow. This technique generally applicable to video since it uses only colour consistency and similarity in extracting flow. This technique fails in the presence of occlusion and if the colours or intensities change dramatically. In order to overcome intensity changes problem , need to preprocessed frames to match their histogram.

Ren [53] proposed a local grouping for optical flow. They applied a local boundary operator and an asymmetric intervening contour scheme to compute the affinity between points. Pairwise affinity is defined a local spatial and scale-adaptive support for motion integration, and allow accurate recovery of flow near motion boundaries and in weak contrast regions.

Vidal, Tron, and Hartley [54] proposed a geometric algorithm for 3D motion segmentation from multiple affine views, which deals with the complete and incomplete data, and independent, partially dependent, full and degenerate motions. Initially, find the five dimensional subspace of high dimensional feature points of frames by using SVD (complete data) or PowerFactorization (incomplete data) or RANSAC (data with outliers). After that, they calculated multi body motion estimation via polynomial fitting and then cluster the feature points by applying the spectral clustering to the similarity matrix. Then applied standard factorization approach for affine cameras to each one of the several group of features to obtain motion and structure parameters.

Black and Anandan [39] proposed an approach for robust estimation of multiple motions. In particular, area based regression technique can be made robust to multiple motions resulting from occlusion, transparency, and specular reflections and the piecewise-smooth flow fields can be recovered by using a robust gradient-based algorithm. It allows to treat the effects of multiple motions on the data conservation and spatial coherence constraints in a uniform manner. It also provides an alternative interpretation of line processes and weak continuity constraints while generalizing their application to cope with non spatial outliers. This approach also allow us to detect the model violations and hence to recover motion boundaries.

We have explained some of the dimensionality reduction methods in Chapter 3 and also explained various face recognition methods in Chapter 4, which are almost similar to our eigendiffusion faces method.

Chapter 3

Diffusion Wavelets

In this chapter, we begin by reviewing manifold learning when employing various methods and the theoretical frame work of Diffusion Wavelets by Coifman and Maggioni [4] which provide the algorithm used for the data decomposition into: "scaling" functions span the column space of the input matrix at a given level and "wavelet" functions which span the orthogonal complement of the matrix column space. After description of diffusion wavelets, this chapter continues with a section about the multiscale dimensionality reduction using diffusion wavelets [17, 16].

3.1 Manifold Learning

High dimensional complex data is hard to model. For example, a set of face images might be governed by many parameters including lighting variation, affine geometric transformations, person mood and display of affections, etc. However, such variations may be recovered by non-linear dimensionality reduction techniques. In contrast to linear methods such as principal component analysis (PCA) or linear discriminant analysis (LDA) [44], non-linear methods do not ignore convexity or concavity of the data and because of this reason, non-linear methods are able to handle a broader range of data sets. Linear methods assume that the data lie on a low dimensional manifold representing a topological space that is locally Euclidean. Dimensionality reduction methods consist of finding a mapping from the original M-dimensional data $\mathbf{X}$ to a smaller dimensional space $\mathbf{Y}$ in which local distances are preserved as much as possible.

3.1.1 Principal Component Analysis

PCA is a method to reduce the data space to an orthogonal subspace which preserves the variance of the initial data set. For linearly dependent data no information is lost through this transformation. But, if we apply PCA on non-linear data set then data is lost through the implied projection. Another problem of PCA is that it tries to preserve large distances between data points. However in most cases, distances are only meaningful in the local neighbourhood. The following Section 3.1.2 presents graph based for data reduction.

3.1.2 Graph Based Algorithms

Graph based algorithms consists in general of three steps:

1. Undirected similarity graph is calculated from the high dimensional complex data, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.

2. Define the weight matrix $\mathbf{W}$ in order to represent the weighted similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$, where w_{ij} represents the weight of the edge between vertices i and j. Weights are calculated such that have the following properties:

   - symmetry: $k(x, y) = k(y, x)$.
   - positivity preserving: $k(x, y) > 0$.
   - represents the similarity between the points in the data set.

   In the weighted matrix, the weight of zero means that the two vertices are not connected.

3. Calculating a global embedding which preserves local properties.

There are three techniques for building the weight matrix. Firstly, there is the ϵ - neighbourhood graph, let us consider x_i and x_j are vertices in graph . In order to calculate the weight between these two vertices we use threshold ϵ . If the distance between two vertices satisfies the condition, $||x_i - x_j||^2 < \epsilon$ then we represent that distance as a weight between those two vertices. otherwise, we represent that weight between those vertices with zero. Secondly, by using the Gaussian function: $w_{ij} = \exp(-(||x_i - x_j||^2)/(2\sigma^2))$, and thirdly, is by using the k-nearest neighbour graph method.

3.1.3 Locally Linear Embedding

Locally linear embedding (LLE) was proposed in [45] and consists of finding an embedding $\ominus$, which preserves neighbourhood relations. LLE consist of the following three steps in order to model the embedded space:

1. Calculate the weight matrix of the complex data set $\mathbf{X}$ by using one of the three methods from the last paragraph of Section 3.1.2.

2. For each data point X_i in the input dataset, find the weights w_{ij} which minimize the least square problem:

$$\mathbf{S}(\mathbf{W}) = \sum_i |X_i - \sum_j X_j w_{ij}|^2.$$

   Here, $\sum_j w_j = 1$ and $w_j > 0$ if and only if j is a neighbour.

3. Calculate the vectors $\mathbf{Y}$ of the lower dimensional space which are reconstructed by the weights of the step 2 by minimizing:

$$\mathbf{S}(\mathbf{Y}) = \sum_i |Y_i - \sum_j w_{ij} Y_j|^2.$$

Here, the weights w_{ij} are fixed and the embedded vectors $\mathbf{Y}$ are optimized based on minimising the locally linear reconstruction error. So, the geometry of nearby points is preserved. If the data points are weakly connected then the coupling between points which are far away is underestimated. This leads to points which are distant in the original high dimensional space $\mathbf{X}$, but nearby in the embedded lower dimensional space $\mathbf{Y}$. In contrast to LLE, which tries to preserve local geometry properties, the Isomap [47] method preserves the global geometry properties of the manifold.

3.1.4 Isomap

Isomap [47] is a non linear multidimensional scaling space method, where the similarity graph is calculated through paths along the manifold surface such as the geodesic distance. Multidimensional scaling (MDS) [46] is an algorithm aiming to find a low dimensional representation which preserves pairwise distances by finding the eigenvectors of the distance matrix. Initially, we have to compute the similarity graph then we have to calculate the shortest path for all pairs of points by using Dijkstra's algorithm. Now, we apply MDS to get a low dimensional space $\mathbf{Y}$, such that geodesic distances are preserved.

The low dimensional space preserves the distances between far away points, better than LLE. This happens because the Isomap algorithm is governed by the geodesic distances between distant points. However, Isomap takes more time to calculate the low dimensional space due to the complexity of MDS.

3.1.5 Laplacian Eigenmaps

Laplacian Eigenmaps [24] are similar to LLE in that they preserve distances. Additionally, they reflect the geometric structure of the manifold by approximating the Laplace-Beltrami operator using the weighted Laplacian of the similarity graph and it can only possible when the data on the manifold is uniform. We compute an embedding space $\ominus$ in the following steps:

1. Undirected similarity graph is calculated from the high dimensional complex data, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.

2. Define the weight matrix $\mathbf{W}$ either by setting $w_{ij} = 1$ for all connected vertices ($w_{ij} = 0$ when the vertices i and j are not connected) or using a heat kernel with parameter σ:
$$w_{ij} = \exp(-\frac{||x_i - x_j||^2}{\sigma}). \tag{3.1}$$

3. Graph Laplacians are the main tool of spectral graph theory [22]. We calculate unnormalized graph Laplacian by using the below equation:
$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{3.2}$$

   where, $\mathbf{L}$ is the graph Laplacian matrix and $\mathbf{D}$ is degree matrix which has entries on

its diagonal: $d_{ii} = \sum_{j=1}^{n} w_{ij}$ and $d_{ij} = 0 \; \forall i \neq j$.
We calculate the normalized graph Laplacian, which represents a random walk on the graph $\mathbf{G}$:

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{D}^{-1} * (\mathbf{D} - \mathbf{W}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}. \tag{3.3}$$

4. Find the eigenvalue λ and eigenvector $\mathbf{v}$ of $\mathbf{L}_{rw}$ if and only if λ and $\mathbf{v}$ solve the generalized eigen problem:

$$\mathbf{Lv} = \lambda \mathbf{Dv}. \tag{3.4}$$

   Here, $\mathbf{L}_{rw}$ is positive semi-definite with the first eigenvalue $\lambda_1 = 0$ and its corresponding eigenvector as the vector of all entries equal to 1. All eigenvalues are real and it holds that: $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Eigenvectors are represented as $v_1, v_2, ..., v_n$. Now, we define the embedding space:

$$\ominus : x_i \rightarrow (v_2(i), v_3(i)...v_d(i)).$$

Laplacian Eigenmaps handle only data on manifold which is sampled uniformly and which happens rarely in real machine learning tasks. The Laplacian only converges to the Laplace-Beltrami operator, if this condition met. From this, we can say that the Laplacian Eigenmaps are a special case of diffusion maps [1].

3.1.6 Diffusion Maps

Diffusion maps [1, 2, 3] are another non linear dimensionality reduction technique for finding the feature representation of the data sets even observed samples are non-uniformly distributed. Coifman and Lafon provide a new approach for normalized graph Laplacians by relating them to diffusion distances.

Diffusion maps achieve dimensionality reduction by re-organising data according to parameters of its underlying geometry. The connectivity of the data set, measured by using a local similarity measure, is used to create time dependent diffusion processes. As the diffusion progresses, it integrates the local data structure to reveal relational properties of the data set at different scales. Diffusion map embeds data into a low dimensional space, such that the Euclidean distance between points approximate the diffusion distance in the original high dimensional space.

For a data set $\mathbf{X} = \{x_1, ..., x_n\}$, a random walk is constructed by considering the probabilities of moving from x_i to another data point. The kernel k defines a local measure of similarity with in a certain neighbourhood. Outside the neighbourhood, the function quickly decreases to zero. We considered the popular Gaussian kernel with scale factor σ for measuring the similarity between data points x_i and x_j:

$$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{\sigma}\right). \tag{3.5}$$

For intricate, non-linear lower dimensional structures, a small neighbourhood is chosen. For sparse data, a larger neighbourhood is more appropriate. The diffusion kernel k satisfies the following two properties:

- k is symmetric: $k(x_i, x_j) = k(x_j, x_i)$.
- k is positivity preserving: $k(x_i, x_j) \geq 0$.

In order to calculate a normalized graph Laplacian, we divide the kernel $k(x_i, x_j)$ by the local measure of the degree in the graph, given by:

$$d(x_i) = \sum_{x_j \in X} k(x_i, x_j). \tag{3.6}$$

The similarity between x_i and x_j is defined as a probability of transition in matrix P with the entries given by:

$$p(x_i, x_j) = \frac{k(x_i, x_j)}{d(x_i)}. \tag{3.7}$$

Each entry in the probability transition matrix $\mathbf{P}$, provides the connectivity between two data points x_i and x_j, and encapsulates what is known locally. By analogy with a random walk, this matrix provides the probabilities for a single step taken from i to j, i.e $t = 1$. We consider probabilities of transition $t > 1$ by taking powers $\mathbf{P}^t$, and forming Markov chains. For higher the values of t, the probability of following a path along the underlying geometric structure of the data set increases. This happens because along the geometric structure points are dense and therefore highly connected. Pathways form along short, high probability jumps. On the other hand, paths that do not follow this structure include one or more long, low probability jumps, which lowers the path's overall probability. For higher values of t, kernel k propagates to the broader neighbourhood around the initial data location.

After applying the eigen decomposition of $\mathbf{P}$, we have $\mathbf{P}\mathbf{v}_l = \lambda_l \mathbf{v}_l$, where λ_l and $\mathbf{v}_l$ are the lth eigenvalue and its corresponding eigenvector respectively. The diffusion distance D_t metric between two data points is:

$$D_t(x_i, x_j) = \left(\sum_{l \geq 1} \lambda_l^{2t} (\mathbf{v}_l(x_i) - \mathbf{v}_l(x_j))^2 \right)^{\frac{1}{2}}. \tag{3.8}$$

$D_t(x_i, x_j)$ will be small, if there is a large number of short paths between x_i and x_j. All eigenvalues of $\mathbf{P}$ are real and it holds that: $1 = \lambda_0 > \lambda_1 > ....$ and their corresponding eigenvectors are represented as $\mathbf{v}_0, \mathbf{v}_1, ...$ We approximate the diffusion distances by considering only the most important s eigenvalues and their corresponding eigenvectors. Now, we can define the embedding space $\ominus$ for diffusion maps by using most important s

eigenvalues and their corresponding eigenvectors $\ominus_t(x_i) : \mathbf{X} \to \Re^s$ is defined by:

$$\ominus_t(x_i) \triangleq \begin{pmatrix} \lambda_1^t \mathbf{v}_1(x_i) \\ \lambda_2^t \mathbf{v}_2(x_i) \\ . \\ . \\ . \\ \lambda_s^t \mathbf{v}_s(x_i) \end{pmatrix}. \tag{3.9}$$

Compared to the Laplacian eigenmaps [24], each eigenvector is scaled by its corresponding eigenvalue in the final embedding of diffusion maps. This leads to a smoother mapping since higher eigenvectors are attenuated.

3.2 Diffusion Wavelets

Diffusion Wavelets introduce a multiresolution geometric construction for the efficient computation of high powers of local operators. Let us consider a matrix $\mathbf{T}$ representing a Markov transition matrix, which is a square matrix describing the probabilities of moving from one state to another in a dynamic system. This matrix $\mathbf{T}$ enables the fast computation of functions of the operator, notably the associated Green's function, in compressed form. Their construction can be viewed as an extension of Fast Multipole Methods [20], and the non-standard wavelet form for the Claderon-Zygmund integral operators as well as for the pseudo-differential operators of [21]. Unlike the integral equation approach, these start from the generator $\mathbf{T}$ of semi groups associated to a differential operator rather than from Green's operators. $\mathbf{T}$ was applied to a space of test functions at the finest scale, for compressing this range via local orthogonalization procedure representing $\mathbf{T}$ in the compressed range, compute $\mathbf{T}^2$, compress and again orthogonalize and so on. At scale j we obtain a compressed representation of $\mathbf{T}^{2^{j+1}}$, acting on the range of $\mathbf{T}^{2^{j+1}-1}$, for which we have a compressed form of orthonormal bases, then apply $\mathbf{T}^{2^{j+1}}$, locally orthogonalize and compress the result, thus getting the next coarser subspace. The computation of the inverse Laplacian $(\mathbf{I} - \mathbf{T})^{-1}$ in order to get the compressed form is done via the Schultz method [21].

$$(\mathbf{I} - \mathbf{T})^{-1} f = \sum_{k=1}^{+\infty} \mathbf{T}^k f. \tag{3.10}$$

and considering:

$$\mathbf{S}_K = \sum_{k=1}^{2^K} \mathbf{T}^k. \tag{3.11}$$

in above equation

$$\mathbf{S}_{K+1} = \mathbf{S}_K + \mathbf{T}^{2^K} \mathbf{S}_K = \prod_{k=0}^{K} (\mathbf{I} + \mathbf{T}^{2^k}) f. \tag{3.12}$$

From the above, we can calculate quickly $\mathbf{T}^{2^k}$ to any function f and hence the product $\mathbf{S}_{K+1}$ can apply $(\mathbf{I}-\mathbf{T})^{-1}$ to any function f fast, with the computational complexity of $O(n\log^2 n)$. This construction considers the columns of a matrix representing $\mathbf{T}$ as data points in the Euclidean space which are viewed as lying on a manifold.

3.2.1 The construction of Diffusion Wavelets

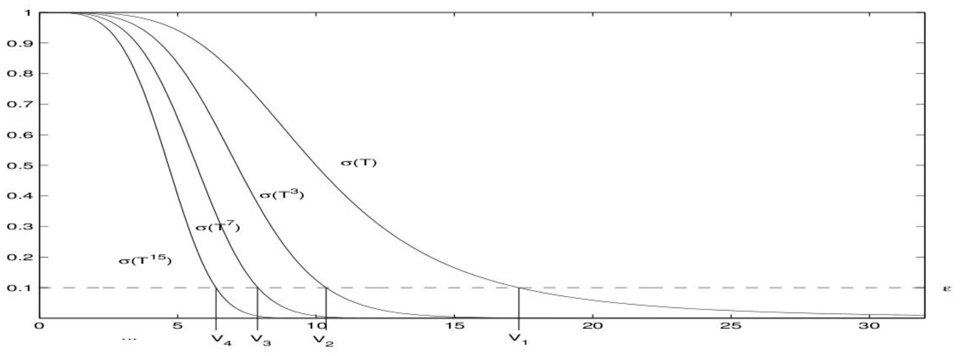


Figure 3.1: Spectral energy powers of $\mathbf{T}$ and their corresponding multiscale eigen-space decomposition(Fig 1 from [4])

In this section, we describe the construction of diffusion wavelets in a finite dimensional case, considering a purely discrete setting for which only finite dimensional linear algebra is needed.

Consider a finite graph $\mathbf{G}$ and a symmetric positive definite and positive diffusion operator $\mathbf{T}$ on $\mathbf{G}$. The graph could represent a metric space in which points are data and edges have weights, and $(\mathbf{I}-\mathbf{T})$ could be a Laplacian on $\mathbf{G}$, that induces a natural diffusion process. $\mathbf{T}$ is similar to a Markov matrix $\mathbf{P}$, representing the natural random walk on the graph $\mathbf{G}$, [22]. The main assumption on this method is that $\mathbf{T}$ is local, i.e. it has a small support and that high powers of $\mathbf{T}$ have low numerical rank. Fig 3.1, describes how the spectral powers of $\mathbf{T}$ relate with the multiscale eigen-space decomposition. Here the rank of $\mathbf{T}$ decreases when increasing the powers of $\mathbf{T}$, i.e rank($\mathbf{T}^{2^j}$) < rank($\mathbf{T}^{2^{j-1}}$) .

To describe the long term behaviour of the diffusion, we need to compute and describe the powers of space $\mathbf{T}^{2^j}$, for $j>0$ and this will allow the computation of functions of the operator in compressed form $(\mathbf{I}-\mathbf{T})^{-1}$, as well as the fast computation of the diffusion from any initial condition. This method is of interest in the solution of discretized partial differential equations of Markov chains. We assume that, high powers of $\mathbf{T}$ are of low rank. It is better to represent them on an appropriate bases at the appropriate resolution. From the analyst's perspective, high powers are smooth functions with small gradient, hence they are compressible, leading to data reduction.

A multiresolution decomposition of the functions on the graph is a family of nested subspaces $\mathcal{V}_0 \supseteq \mathcal{V}_1 \supseteq \mathcal{V}_2 \supseteq .... \supseteq \mathcal{V}_j \supseteq ...$ spanned by orthogonal bases of diffusion scaling function $\mathbf{\Phi}_j$. If $\mathbf{T}^t$ is an operator on functions on the graph $\mathbf{G}$, then the subspace $\mathcal{V}_j$ is defined as the numerical range up to the precision ϵ of $\mathbf{T}^{2^{j+1}-1}$ and the scaling functions are smooth bump functions with some oscillations, at a scale roughly 2^{j+1}. The orthogonal complement of subspace $\mathcal{V}_{j+1}$ into $\mathcal{V}_j$ is called $\mathcal{W}_j$ and is spanned by a family of orthogonal diffusion wavelets $\mathbf{\Psi}_j$, which are smooth and localized oscillatory functions at the same scale.

The input to the algorithm is a precision parameter $\epsilon > 0$ which controls the ampli-

$$\begin{tikzcd}
\Phi_0 \arrow[r, "{[T]_{\Phi_0}^{\Phi_1}}"] \arrow[d, "{[T]_{\Phi_0}^{\Phi_0}}"'] & \Phi_1 \arrow[r, "{[T^2]_{\Phi_1}^{\Phi_2}}"] \arrow[d, "{[T^2]_{\Phi_1}^{\Phi_1}}"'] & \cdots \arrow[r, "{[T^{2^j}]_{\Phi_j}^{\Phi_{j+1}}}"] & \Phi_{j+1} \arrow[r, "{[T^{2^{j+1}}]_{\Phi_{j+1}}^{\Phi_{j+2}}}"] \arrow[d, "{[T^{2^{j+1}}]_{\Phi_{j+1}}^{\Phi_{j+1}}}"'] & \cdots \\
\tilde{\Phi}_1 \arrow[ur, "{[\Phi_1]_{\Phi_0}}"] & \tilde{\Phi}_2 \arrow[ur, "{[\Phi_2]_{\Phi_1}}"] & \cdots \arrow[ur, "{[\Phi_{j+1}]_{\Phi_j}}"] & \tilde{\Phi}_{j+1} \arrow[ur, "{[\Phi_{j+2}]_{\Phi_{j+1}}}"] &
\end{tikzcd}$$

Figure 3.2: Diagram for downsampling, orthogonalization and operator compression (triangles are commutative by construction) (Fig 6 from [10])

tude of the vectors, $|\mathcal{V}_j| < \epsilon$ and a weighted graph $(\mathbf{G}, \mathbf{E}, \mathbf{K})$, where $\mathbf{G}$ is the graph, $\mathbf{E}$ are the edges and their weights are $\mathbf{K}$. We assume $\mathbf{G}$ is strongly connected and local, i.e each vertex is connected to a small number of vertices. The construction is based on using the natural random walk $\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}$, where $\mathbf{K}$ is the kernel function as in Equation (3.5) and $\mathbf{D}$ is the degree matrix as in Equation (3.6), i.e the sum of each row of elements in $\mathbf{K}$ representing the weights between a data and all other data, placed in the diagonal element of diagonal matrix $\mathbf{D}$. Here, the powers of $\mathbf{P}$ use to dilate or diffuse (corresponding to powers of t) functions on the graph and then define an associated coarse-graining of the graph as will be described in Section 3.3). In many cases of interest $\mathbf{P}$ is a sparse matrix, Usually normalized $\mathbf{P}$ is considered as $\mathbf{T}$ i.e $\mathbf{T} = \beta^{-1}\mathbf{P}\beta$, where β is the asymptotic distribution of $\mathbf{P}$. From the hypothesis on $\mathbf{P}$, β exists and is unique and strictly chosen to be positive as defined by the Perron-Fröbenius theorem. If graph $\mathbf{G}$ is undirected, $\mathbf{P}$ is reversible i.e $\beta = \mathbf{D}^{\frac{1}{2}}$ and $\mathbf{T}$ is symmetric. The powers of $\mathbf{T}$ are obtained as,

$$\mathbf{T}^t = (\beta^{-1}\mathbf{P}\beta)^t = (\mathbf{D}^{-\frac{1}{2}}\mathbf{P}\mathbf{D}^{-\frac{1}{2}})^t \quad here, (\beta = \mathbf{D}^{\frac{1}{2}}). \tag{3.13}$$

A diffusion wavelet tree consists of orthogonal diffusion scaling functions $\mathbf{\Phi}_j$ which are smoothly bumped functions with some oscillations at scale 2^j, roughly measured with respect to geodesic distance, and the orthogonal wavelets $\mathbf{\Psi}_j$ which are smoothly localized oscillatory functions at the same scale. The scaling functions $\mathbf{\Phi}_j$ span the subspace $\mathcal{V}_j$, which holds the property $\mathcal{V}_{j+1} \subseteq \mathcal{V}_j$ and the span of orthogonal bases wavelets $\mathbf{\Psi}_j$, while $\mathcal{W}_j$ is the orthogonal complement of $\mathcal{V}_j$ into $\mathcal{V}_{j+1}$ domain. A diffusion wavelets tree is achieved by using dyadic powers of $\mathbf{T}$ i.e $\mathbf{T}^{2^j}$, as dilations in order to create smoother and wider bump functions. Orthogonalizing and downsampling appropriately transforms these sets of bump functions into orthonormal scaling functions.

Fig 3.2 shows the construction of the multiscale extended bases functions in detail. $\mathbf{T}$, is initially represented on the bases $\mathbf{\Phi}_0 = \{\delta_k\}_{k\in\mathbf{G}}$. Consider the columns of $\mathbf{T}$ as the set of functions $\tilde{\mathbf{\Phi}_1} = \{\mathbf{T}\delta_k\}_{k\in\mathbf{G}}$ on $\mathbf{G}$. Local multiscale Gram-Schmidt orthogonalization procedure, which is the linear transformation represented by the matrix $[\mathbf{\Phi}_1]_{\mathbf{\Phi}_0}$, is used to orthonormalize these columns to get a bases $\mathbf{\Phi}_1 = \{\varphi_{1,k}\}_k \in \mathbf{G}_1$, where $\mathbf{G}_1$ is an index set written with respect to the bases $\mathbf{\Phi}_0$, for the range of $\mathbf{T}$ up to precision ϵ. This yields a subspace that we denote by $\mathbf{V}_1$. $\mathbf{\Phi}_1$ is a bases for the subspace which is ϵ-close to the range of $\mathbf{T}$ with bases elements that are well-localized. Moreover, the elements of the bases $\mathbf{\Phi}_1$ are coarser than the elements of $\mathbf{\Phi}_0$, since the dilation is the result of applying $\mathbf{T}$ once. Obviously, $|\mathbf{G}_1| \leq |\mathbf{G}|$ but the inequality may already be strict since part of the numerical range of $\mathbf{T}$ may be below the precision ϵ. Whether, this is the case or not, we have computed the matrix $[\mathbf{T}]_{\mathbf{\Phi}_0}^{\mathbf{\Phi}_1}$, the representation of an ϵ-approximation of $\mathbf{T}$ with respect to $\mathbf{\Phi}_0$ in the domain, and with respect to $\mathbf{\Phi}_1$ in the range. We can also represent $\mathbf{T}$ with respect to the bases $\mathbf{\Phi}_1$ with the notation in the upper left side of the matrix as $[\mathbf{T}]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_1}$. We compute $[\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_1} = [\mathbf{\Phi}_1]_{\mathbf{\Phi}_0}[\mathbf{T}^2]_{\mathbf{\Phi}_0}^{\mathbf{\Phi}_0}[\mathbf{\Phi}_1]_{\mathbf{\Phi}_0}^{\tau}$,where τ denotes the transpose of matrix. If $\mathbf{T}$ is self-adjoint, this is equal to $[\mathbf{T}]_{\mathbf{\Phi}_0}^{\mathbf{\Phi}_1}([\mathbf{T}]_{\mathbf{\Phi}_0}^{\mathbf{\Phi}_1})^{\tau}$, which has the advantage that numerical symmetry is forced upon $[\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_1}$.

We proceed now by looking at the columns of $[\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_1}$, which are $\tilde{\mathbf{\Phi}_2} = \{[\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_1}\delta_k\}_{k\in\mathbf{G}_1}$, i.e by unravelling the bases on which this is happening , $\{\mathbf{T}^2\varphi_{1,k}\}_{k\in\mathbf{G}_1}$ up to the precision ϵ. Once again we apply a local orthonormalization procedure to this set of functions and this yields a matrix $[\mathbf{\Phi}_2]_{\mathbf{\Phi}_1}$ and an orthonormal bases $\mathbf{\Phi}_2 = \{\varphi_{2,k}\}_{k\in\mathbf{G}_2}$ for the range of $\mathbf{T}_1^2$ up to precision ϵ and also for the range of $\mathbf{T}_0^3$ up to precision 2ϵ. Moreover, depending on the decay of the spectrum of $\mathbf{T}$, $|\mathbf{G}_2|$ is in general a fraction of $|\mathbf{G}_1|$. The matrix $[\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_2}$ is then of size $|\mathbf{G}_2|\times|\mathbf{G}_1|$ and the matrix for the following level is $[\mathbf{T}^4]_{\mathbf{\Phi}_2}^{\mathbf{\Phi}_2} = [\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_2}([\mathbf{T}^2]_{\mathbf{\Phi}_1}^{\mathbf{\Phi}_2})^{\tau}$.

After repeating for j steps in this manner we will have a representation of $\mathbf{T}^{2^j}$ onto a bases $\mathbf{\Phi}_j = \{\varphi_{j,k}\}_{k\in\mathbf{G}_j}$, encoded in a matrix $\mathbf{T}_j = [\mathbf{T}^{2^j}]_{\mathbf{\Phi}_j}^{\mathbf{\Phi}_j}$. The orthonormal bases $\mathbf{\Phi}_j$ is represented with respect to $\mathbf{\Phi}_{j-1}$, and encoded in a matrix $[\mathbf{\Phi}_j]_{\mathbf{\Phi}_{j-1}}$. Let $\tilde{\mathbf{\Phi}_j} = \mathbf{T}_j\mathbf{\Phi}_j$, we can represent the next dyadic power of $\mathbf{T}$ on $\mathbf{\Phi}_{j+1}$ on the range of $\mathbf{T}^{2^j}$. Depending on the decay of the spectrum of $\mathbf{T}$, we expect to have $|\mathbf{G}_j| << |\mathbf{G}|$. In fact in the ideal situation, the spectrum of $\mathbf{T}$ decays fast enough so that there exists $\gamma < 1$ such that $|\mathbf{G}_j| < \gamma^{2^{j+1}-1}|\mathbf{G}|$. While the bases $\mathbf{\Phi}_j$ is naturally identified with the set of Dirac δ-functions on $\mathbf{G}_j$, we can extend these functions defined on the compressed or downsampled graph $\mathbf{G}_j$ to the whole initial graph $\mathbf{G}$ by writing, [10]:

$$[\mathbf{\Phi}_j]_{\mathbf{\Phi}_0} = [\mathbf{\Phi}_j]_{\mathbf{\Phi}_{j-1}}[\mathbf{\Phi}_{j-1}]_{\mathbf{\Phi}_0} = [\mathbf{\Phi}_j]_{\mathbf{\Phi}_{j-1}}[\mathbf{\Phi}_{j-1}]_{\mathbf{\Phi}_{j-2}}....[\mathbf{\Phi}_1]_{\mathbf{\Phi}_0}[\mathbf{\Phi}_0]_{\mathbf{\Phi}_0}. \tag{3.14}$$

Every function in $\mathbf{\Phi}_0$ is defined on $\mathbf{G}$, and consequently every function in $\mathbf{\Phi}_j$ is defined on $\mathbf{G}$ too. Hence any function on the compressed space $\mathbf{G}_j$ can be extended naturally to the whole $\mathbf{G}$. The elements in $\mathbf{\Phi}_j$ are at scale $\mathbf{T}^{2^{j+1}-1}$ and are much coarser and smoother than the initial elements in $\mathbf{\Phi}_0$, which is the way they can be represented in the compressed or downsampled form. The projection of a function onto the subspace spanned by $\mathbf{\Phi}_j$ will be by definition an approximation of that function at that particular scale j.

There are three steps to construct a wavelet at each scale: downsampling, orthogonalization, and operator compression (dilation/diffusion). The dyadic powers of $\mathbf{T}^{2^j}$ correspond to "dilations", and can be used to create smoother and wider "bump" functions. Orthogonalizing and downsampling appropriately we transform sets of "bump functions" into orthonormal scaling functions.

In the Algorithm 3.1 is explained how diffusion wavelets are used in order to con-

Algorithm 3.1 Multiscale representation at different scales using diffusion wavelets construction

```
$\{[\mathbf{\Phi}_j]_{\mathbf{\Phi}_0}, [\mathbf{\Psi}_j]_{\mathbf{\Psi}_0}\} = DiffusionWavelets(\mathbf{T}, \epsilon, F_\theta, R_\theta, J, \kappa)$
//INPUT:
//$\mathbf{T}$: The Input Matrix
//$\epsilon$ :Desired Precision for modified Gram-Schmidt
//$F_\theta$:Threshold for two column inner product in modified Gram-Schmidt
Orthogonalization.
//$R_\theta$:Threshold for $R$ component, which is obtained from modified
Gram-Schmidt Orthogonalization.
//$J$:Desired levels for scaling, Program will stop at this level.
//$\kappa$:Program stops when columns are less or equal to this in extended
diffusion scaling function.
//OUTPUT:
//$[\mathbf{\Phi_j}]_{\mathbf{\Phi}_0}$ : Extended diffusion scaling functions at scale $j$
//$[\mathbf{\Psi}_j]_{\mathbf{\Psi}_0}$ : Extended diffusion wavelet functions at scale $j$
$\mathbf{\Phi}_0 = \mathbf{I}$; // $\mathbf{I}$ is Unit Vector
for $j$=0 to $J$-1 do
  $([\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_j}, [\mathbf{T}^{2^j}]_{\mathbf{\Phi}_j}^{\mathbf{\Phi}_{j+1}}) = QRgramschmidt([\mathbf{T}^{2^j}]_{\mathbf{\Phi}_j}^{\mathbf{\Phi}_j}, \epsilon, F_\theta, R_\theta)$
  $[\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_0} = [\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_j}[\mathbf{\Phi}_j]_{\mathbf{\Phi}_0}$
  $[\mathbf{\Psi}_j]_{\mathbf{\Phi}_j} = QRgramschmidt(\mathbf{I}_{\mathbf{\Phi}_j} - [\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_j}[\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_j}^{\mathbf{T}}, \epsilon, F_\theta, R_\theta)$
  $[\mathbf{\Psi}_{j+1}]_{\mathbf{\Phi}_0} = [\mathbf{\Psi}_{j+1}]_{\mathbf{\Phi}_j}[\mathbf{\Phi}_j]_{\mathbf{\Phi}_0}$
  $[\mathbf{T}^{2^{j+1}}]_{\mathbf{\Phi}_{j+1}}^{\mathbf{\Phi}_{j+1}} = ([\mathbf{T}^{2^j}]_{\mathbf{\Phi}_j}^{\mathbf{\Phi}_{j+1}}[\mathbf{\Phi}_{j+1}]_{\mathbf{\Phi}_j})^2$ //If Columns in $\mathbf{T}^{2^{j+1}}$ at this scale
  below or equal to $\kappa$ then break this loop else continue.
end for
```

structs multiscale representations [17]. Diffusion wavelets construct a compressed form of representation of the dyadic powers of a symmetric or non-symmetric square matrix by representing the associated matrices at each scale. Given a matrix $\mathbf{T}$, the modified Gram-Schmidt with pivoting the columns algorithm, called QRgramscmidt in the pseudo-code from Algorithm 3.2, decomposes $\mathbf{T}$ into an orthogonal matrix $\mathbf{Q}$ and a triangular matrix $\mathbf{R}$ such that $\mathbf{T}$ is similar to the product of $\mathbf{Q}$ and $\mathbf{R}$ components, $\mathbf{T} = \mathbf{QR}$. Columns in $\mathbf{Q}$ are orthonormal bases functions spanning the column space of $\mathbf{T}$ at the finest scale. From the invariant subspace theory, $\mathbf{RQ}$ (product of $\mathbf{R}$ and $\mathbf{Q}$) is the new representation of $\mathbf{T}$ with respect to the space spanned by the columns of $\mathbf{Q}$. At scale j, diffusion wavelets learn the bases functions from $\mathbf{T}^{2^j}$ using the procedure from Algorithm 3.2. Compared to

Algorithm 3.2 `Modified Gram-Schmidt with pivoting columns`

```
$[\mathbf{Q},\mathbf{R}]$ = QRgramschmidt($\mathbf{T}$,$\epsilon$,$F_\theta$,$R_\theta$)
[$rows$,$colns$]=size($\mathbf{T}$);
for $i$=1 to $colns$ do
  $fNorms$(1,$i$)=norm($\mathbf{T}$($all$,$i$));//eachclumnnrm
end for
$nLFcn$ = $colns$; $nFcnsChosen$ = 0;
for $i$=1 to $colns$ do
  [$fNorms$,$srtcln$] = SORT($fNorms$,"$ascend$")//if $nLFcn$>1 do
  $MaxNorm$ = $fNorms$(1,$nLFcn$);
  if($MaxNorm$<$\epsilon$ )AND($nFcnsChosen \geq$ 1) break;end if
  $ChosenNorm$=$fNorms$(1,$nLFcn$);$ChosenCol$=$srtcln$(1,$nLFcn$);
  $\mathbf{T}$($all$,$ChosenCol$) = $\mathbf{T}$($all$,$ChosenCol$)/$ChosenNorm$;//Normalizing
  //"Orthogonalize all other columns"
  for $j$=1 to $nLFcn$-1 do
    $sc$=$srtcln$(1,$j$);$ip$ = $\mathbf{T}$($all$,$sc$)$^\tau$ * $\mathbf{T}$($all$,$ChosenCol$);
    if abs($ip$) > $F_\theta$ then
      $\mathbf{R}$($sc$,$i$) = $ip$;
      $\mathbf{T}$($all$,$sc$)=$\mathbf{T}$($all$,$sc$)-($ip$*$\mathbf{T}$($all$,$ChosenCol$));
      $fNorms$(1,$j$) = norm($\mathbf{T}$($all$,$sc$));
    end if
  end for
  $nLFcn$ = $nLFcn$-1;$nFcnsChosen$ = $nFcnsChosen$ + 1;
end for
$\mathbf{R}$ = $\mathbf{R}$($all$, 1 to $nFcnsChosen$)$^\tau$; $\mathbf{R}$ = $\mathbf{R}$.*(abs($\mathbf{R}$)>$R_\theta$);
for $i$=1 to $nFcnsChosen$ do
  $\mathbf{Q}$($all$,$i$)=$\mathbf{T}$($all$,$srtcln$(1,$colns$+1-$i$));
end for
```

the number of bases functions spanning the original column space of $\mathbf{T}^{2^j}$, this process will result into fewer bases functions, since some high frequency information (corresponding to the noise or to small features at that scale) will be filtered out. Diffusion wavelets method then computes the bases functions $\mathbf{T}^{2^{j+1}}$ using the low frequency representation functions of $\mathbf{T}^{2^j}$ and this procedure repeats itself until it reaches the last j level. Sometimes we will get the required number of functions reduction before the end of this level. In order to overcome this case, we will check whether the number of columns, each representing a bases function in $\mathbf{T}^{2^j}$, are less than or equal to the minimal number of the columns κ and if this is true then the loop will break before the end of level j.

In Algorithm 3.2, we outline the procedure for modified Gram-Schmidt with pivoting the columns. Here, we used two thresholds and a desired precision ϵ. The precision ϵ will decide which column has to be removed from the orthogonalization projections $\mathbf{Q}$. The threshold F_θ decides whether the column should be included or not when calculating projections by verifying whether the dot product of two vectors is greater than threshold or not , see in Algorithm 3.2. R_θ is a threshold for elements in the triangular matrix $\mathbf{R}$. If an entry in this matrix $\mathbf{R}$ is less than the R_θ threshold then we force it to be zero, leading to a sparse matrix. In order to reduce the computation time, here we calculate projections for the columns which are significant, i.e for which the norm of the columns is greater than the precision ϵ value. This algorithm was used by Maggioni [16].

Running diffusion wavelets algorithm is equivalent to running a Markov chain on the input data forward in time, integrating the local geometry and therefore revealing the relevant geometric structure of the data at different scales. Here, two sets of bases functions are constructed, one represents the scaling functions which span the column space of the input matrix at a given level and the other bases set corresponds to the wavelet functions which spans the orthogonal component of the matrix column space. In our case, only the extended bases scaling functions are considered to represent the features from images at each scale.

3.3 Multiscale dimensionality reduction using diffusion wavelets

In this section we describe the diffusion wavelet algorithm and how this can be applied for multiscale dimensionality reduction. This research is applied in the experiment results chapter to the face and fingerprint recognition, as well as to the optical flow estimation from image sequences.

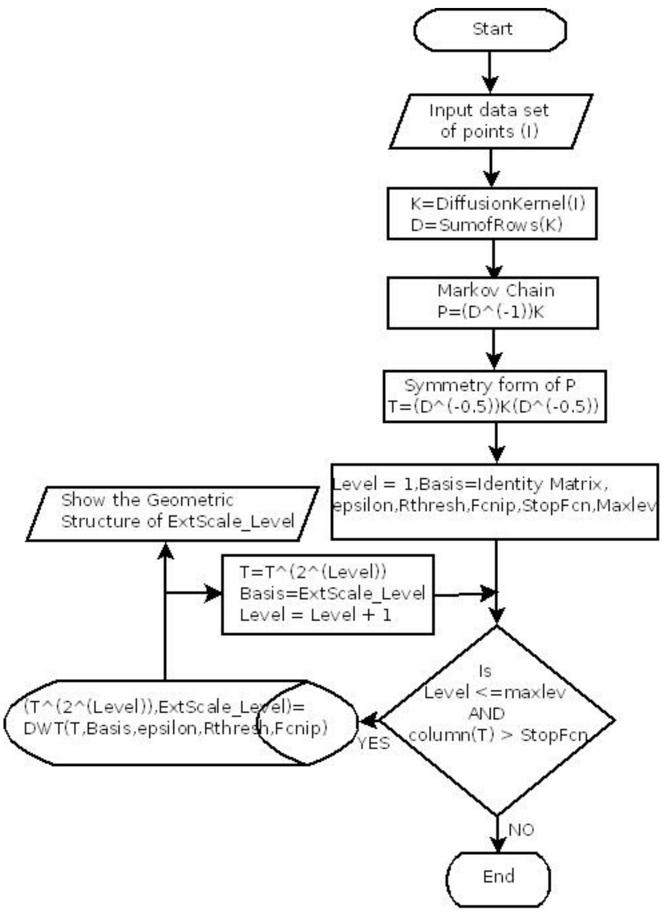
3.3.1 The main algorithm

For a set of points $\mathbf{X} = \{x_1, ..., x_n\}$, a random walk is constructed by considering the probabilities of moving from x_i to the other points (here,we considered that every point is neighbour to all the other points), denoted as $\{x_{j1}, ..., x_{jk}\}$. Probabilities of similarity are modelled by the kernel function, $K(x, y)$ that defines the similarity between two points, x

and y. In the present application we use the kernel function:

$$K(x, y) = e^{-||x-y||^2/\sigma}. \tag{3.15}$$

For two points x and y, and a scale factor σ. The kernel function guarantees the symmetry



Start
Input data set of points (I)
K=DiffusionKernel(I)
D=SumofRows(K)
Markov Chain
P=(D^(-1))K
Symmetry form of P
T=(D^(-0.5))K(D^(-0.5))
Level = 1,Basis=Identity Matrix, epsilon,Rthresh,Fcnip,StopFcn,Maxlev
Show the Geometric Structure of ExtScale_Level
T=T^(2^(Level))
Basis=ExtScale_Level
Level = Level + 1
Is Level <=maxlev AND column(T) > StopFcn
(T^(2^(Level)),ExtScale_Level)= DWT(T,Basis,epsilon,Rthresh,Fcnip)
YES
NO
End

Figure 3.3: Multi scale dimensionality reduction flowchart

of the adjacency matrix and yields non-negative entries representing the similarity among the data. In order to construct a normalized graph Laplacian using the kernel function, $K(x, y)$, we can normalize the kernel by the local measure of the degree in the graph

$$D(x) = \sum_{z \in \mathbf{X}} K(x, z). \tag{3.16}$$

and define the similarity of the pairs of points as a probability.

$$p(x, y) = \frac{K(x, y)}{D(x)}. \tag{3.17}$$

Generally, the probability of transition from x to y ($\mathbf{K}$ is symmetric but p is not) can be thought of occurring in one time step. If we define an adjacency matrix $\mathbf{P}$ by using these probabilities, we can consider the probabilities of transition, $p_t(x, y)$ for more than one

Algorithm 3.3 Multi scale dimensionality reduction by using Diffusion Wavelets

```
[$ExtBas$,$DFR$] = MSDRUDW($\mathbf{X}$,$\epsilon$,$\sigma$,$F_\theta$,$R_\theta$,$J$,$\kappa$)
//INPUT://$\mathbf{X}$:  The Input Matrix
//OUTPUT://$ExtBas$:Extend bases scaling at each scale j
//$DFR$:  Feature Representations at each scale $j$
[$\mathbf{T}$,$row$,$col$,$numPoints$] = MarkovTransmatrx($\mathbf{X}$,$\sigma$)
for $lev$=1 to $J$ do
  if size($\mathbf{T}$,2)<=$\kappa$ then
    if $lev$>1 then
      $ExtBas$ = $ExtBas$(1 to ($lev$-1),$all$)
    end if
    break
  end if
  [$\mathbf{Q}$,$\mathbf{R}$]=QRgramschmidt($\mathbf{T}$,$\epsilon$,$F_\theta$,$R_\theta$) ; $\mathbf{T} = (\mathbf{R} * \mathbf{Q})^2$;
  if $lev$ ==1 then
    $ExtBas\{lev,1\}$ = $\mathbf{Q}$
  else
    $ExtBas\{lev,1\}$ = $ExtBas\{lev,1\}$*$\mathbf{Q}$
  end if
  for $i$=1 to $numPoints$ do
    for $j$=1 to $numPoints$ do
      $DFdist(i,j) = (sum((ExtBas\{lev,1\}(i,all) - ExtBas\{lev,1\}(j,all)).^2))^{0.5}$
    end for
  end for
  $DFR\{lev,1\} = sum(DFdist,2)$
end for
```

time step by taking the higher powers of $\mathbf{P}$ forming Markov chains. The result from the Markov chain embeds the feature information from the data set $\mathbf{X}$, while higher values of t increase the diffusion of this information to the broader neighbourhood around the point of origin x. However, in our case we initially considered that every point in a data set is neighbour to the remaining data in a Markov chain and consequently $\mathbf{P}$ has a maximum possible broader neighbourhood ($n-1$ neighbours in n data set points) around the point of origin, x.

The scale factor σ, from the kernel function from Equation (3.15) is the other factor determining the extent of the Markov process from a starting point x. Lower values of σ inhibit the propagation of information across data features. On the other hand higher values of σ may fail to record the meaningful variations in the data resulting in poor boundaries in image data for example.

From equation (3.17),we understood that kernel function $\mathbf{K}$ is symmetric but the Markov chains matrix P is not. In order to achieve symmetry in equation (3.16) we use the local measure of the degree in graph, i.e $D(x)$ [3]

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{K}. \tag{3.18}$$

$$\mathbf{T} = \mathbf{D}^{\frac{1}{2}}\mathbf{P}\mathbf{D}^{-\frac{1}{2}}. \tag{3.19}$$

$$\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}. \tag{3.20}$$

Here, $\mathbf{P}$ is the Markov chain matrix, which is not symmetrical and $\mathbf{K}$ is a symmetrical form of the kernel function or adjacency matrix of the data set. $\mathbf{T}$ is the symmetrical form of the Markov chain and we use this symmetrical matrix as input to the diffusion wavelet Algorithm 3.3 to get the multiscale dimensional reduction representation. The diffusion wavelets can take both the symmetrical and non symmetrical form of Markov chains but here we considered the self-adjoint symmetry of Markov chains $\mathbf{T}$.

The multiscale analysis generates a sequence of Markov chains, each corresponding to a different time scale, i.e the dyadic power of the original Markov chain, represented on a set of scaling functions in compressed form. We apply $\mathbf{T}$ to $\mathbf{\Phi}_0$ and then orthonormalize to get $\mathbf{\Phi}_1$. Each function in $\mathbf{\Phi}_1$ is a linear combination of the original states $\mathbf{\Phi}_0$. Then represent $\mathbf{T}^2$ on $\mathbf{\Phi}_1$, to get a matrix $\mathbf{T}^4$, apply to $\mathbf{\Phi}_1$ and orthonormalize, and so on. This procedure follows that from Section 3.2.1.

3.4 Summary

In this chapter, we described how the construction of wavelets is used for multiscale representation. Dilation, orthogonalization and downsampling are the steps involved in the wavelet construction. Dilation produces a compressed form of the Markov transition matrix at each level with dyadic powers of time scale and orthogonalization projections of the Markov transition matrix. This is calculated with the help of modified Gram-Schmidt with pivoting columns algorithm, while downsampling is used in order to normalize the diffusion bases scale domain. This procedure repeats itself for the following representation levels. At the last level the extended bases is smoother than that of the previous level, because we would usually have fewer bases functions when compared to those from the previous level. Some high frequency information is filtered out and the current Markov transition matrix is calculated with the help of low frequency representations of the previous level and this procedure repeats itself. Multiscale dimensionality reduction is also similar to this concept, which was described in Section 3.3 .

Chapter 4

Face Recognition using Eigendiffusion Faces

In this chapter after describing various related face recognition approaches such as face correlation, eigenfaces, fisherfaces methods, we describe how eigendiffusion faces are used for face recognition and fingerprint authentication.

4.1 Face Recognition using Correlation

This is the simplest classification for face recognition, based on the nearest neighbour classifier in the image space. An image in the test set is classified by assigning to the label of the closest image in the training set, where distances are measured as Euclidean distances between their corresponding pixel values in the image space. If both training and testing set images are normalized to zero mean and unit variance then this is equivalent to choosing the image in the training set that best correlates with the test image. Images are normalized problems that occur are due to illumination variation, noise, face orientation, human motions etc. Two main drawbacks occur in this approach. One is when the images in the training set and testing set images are gathered under varying lighting conditions, then the pixels corresponding to the same face regions in the image space are no longer similar in their values. So, in order for this method to work reliably under variations in lighting, we would need a training set which would densely sample the continuum of all possible lighting conditions. Correlation is computationally expensive as well, [29].

4.2 Face Recognition using Eigenfaces

The information theory approach for representing face images may provide the insight into the information content of face images, modelling significant local and global features according to their significance for face identity. Such features may or may not be directly related to specific face features such as eyes, nose, lips or hair.

The eigenvectors of the covariance matrix (calculated from the set of normalized face images) are ordered and each one accounts for various amounts of variation among the face images. These eigenvectors represent a set of features that together characterize the variation among the face images from the given set. Each image contributes more or less to each eigenvector. We can represent each eigenvector as a sort of ghost face which is called an eigenface. Each eigenface models certain facial features characteristic to the set of training faces.

Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. Each face can be approximated using the most important eigenvectors or the best eigenfaces, those having the largest eigenvalues, and which therefore account for the largest variation within the set of face images. The best M eigenfaces span an M-dimensional subspace, i.e the face space of all possible images [28].

4.2.1 Calculating Eigenfaces

Let us consider a face image $I(x, y)$ as a two dimensional $M \times N$ array of 8 bit intensity values which can be represented as a vector of dimension MN . Images of faces, being similar in their overall configuration can be described by a relatively low dimensional subspace. Principal component analysis is used to find the most important eigenvectors that account for the distribution of face images within the entire image space. These vectors define a subspace of face images, which is called the face space. Each eigenvector of the covariance matrix corresponding to the original set of face images of size MN describes a linear combination of the original face images.

Let us consider the training set $\mathbf{S}$, of m face images $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2 .....\mathbf{\Gamma}_m$ and each image is transformed into a vector of size MN

$$\mathbf{S} = \{\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, ..., \mathbf{\Gamma}_m\}. \tag{4.1}$$

The Mean face of the set is defined as

$$\Psi = \frac{1}{m} \sum_{i=1}^{m} \mathbf{\Gamma}_i. \tag{4.2}$$

Each face differs from the mean face by,

$$\Phi_{fi} = \mathbf{\Gamma}_i - \Psi. \tag{4.3}$$

We apply principal component analysis (PCA) to this set of vectors, which gives the set of m orthonormal vectors, $\mathbf{u}_n$, n=1,2,...,m, which best describes the distribution of the data.

The kth vector, $\mathbf{u}_k$ is chosen such that,

$$\lambda_k = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{u}_k^{\tau}\Phi_{fn})^2. \tag{4.4}$$

is maximum, subject to

$$\mathbf{u}_l^{\tau}\mathbf{u}_k = \delta_{lk} = \begin{cases} 1 & \text{if l = k} \\ 0 & \text{Otherwise .} \end{cases} \tag{4.5}$$

The vectors $\mathbf{u}_k$ and scalars λ_k are the eigenvectors and eigenvalues of the covariance matrix.

$$\mathbf{C} = \frac{1}{m}\sum_{i=1}^{m}\Phi_{fi}\Phi_{fi}^{\tau} = \mathbf{A}\mathbf{A}^{\tau}. \tag{4.6}$$

Where the matrix $\mathbf{A} = [\Phi_{f1}, \Phi_{f2}, ..., \Phi_{fm}]$, Φ_{fi} is given by Equation (4.3) . The matrix C, is of size $MN \times MN$ and is characterized by MN eigenvectors and eigenvalues. To find these eigenvectors, we construct the matrix $\mathbf{L} = \mathbf{A}^{\tau}\mathbf{A}$, of size $m \times m$, where $L_{ij} = \Phi_{fi}^{\tau}\Phi_{fj}$ and let us find its m eigenvectors, denoted by $\mathbf{v}_l$. These vectors determine linear combinations of the m training set face images to form the eigenfaces $\mathbf{u}_l$:

$$\mathbf{u}_l = \sum_{k=1}^{m} v_{lk}\Phi_{fk}, l = 1, ..., m. \tag{4.7}$$

The calculations are greatly reduced from the order of the number of pixels, MN to the order of the number of images m in the training set. The training set of face images will be relatively small, i.e $m << MN$ and this will reduce the computational complexity.

4.3 Face Recognition using Fisherfaces

Both correlation in the gray level domain and the eigenface methods are expected to suffer from noise, face orientation, human expression, variation as well as illumination variation among the faces. To overcome this problem, the linear subspace scheme was prepared for face recognition [30]. This method is based on the fact that under idealized conditions, the variation within each class lies in a linear subspace of the image space. Hence the face classes are convex and therefore, linearly separable. One can perform dimensionality reduction using linear projections and still preserve the linear separability. The linear methods can recognise faces efficiently when the data set is affected by changes of insensitivity to lighting conditions.

Fisherfaces is a linear discriminant analysis method related to the eigenfaces method. In the case of Fisherfaces we have two covariance matrices, one is the within class covariance matrix $\mathbf{C}_W$, and the second is the between classes covariance matrix $\mathbf{C}_B$. The $\mathbf{X}_{ij}$

is represents for one particular face image as:

$$\mathbf{X}_{ij} = \{x_{ij}, y_{ij}, z_{ij}...\}. \tag{4.8}$$

We calculate the mean pixel values Ψ_i of subject i like below and here n_i represents the number of face images of subject i,

$$\Psi_i = \frac{1}{n_i}\sum_{j=1}^{n_i}\mathbf{X}_{ij}. \tag{4.9}$$

We calculate the mean pixel values of the whole testing data set Ψ like below where N represents the total number of images in the gallery, and c is the number of subjects, i.e the number of classes.

$$\Psi = \frac{1}{N}\sum_{i=1}^{c} n_i\Psi_i. \tag{4.10}$$

Afterwards, we calculate the covariance matrix of between classes as

$$\mathbf{C}_B = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{n_i}(\mathbf{X}_{ij} - \Psi_i)^{\tau}(\mathbf{X}_{ij} - \Psi_i). \tag{4.11}$$

We calculate the covariance matrix of within classes as

$$\mathbf{C}_W = \frac{1}{N}\sum_{i=1}^{c} n_i(\Psi_i - \Psi)^{\tau}(\Psi_i - \Psi). \tag{4.12}$$

In practice, these equations would create the covariance matrix of $MN \times MN$ dimensions. Calculating the eigenvectors for the above covariance matrix is computationally expensive, and we can use the same procedure as for the calculations of the eigenfaces in order to reduce the number of calculations.

$$\mathbf{L}_B = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{n_i}(\mathbf{X}_{ij} - \Psi_i)(\mathbf{X}_{ij} - \Psi_i)^{\tau}. \tag{4.13}$$

$$\mathbf{L}_W = \frac{1}{N}\sum_{i=1}^{c} n_i(\Psi_i - \Psi)(\Psi_i - \Psi)^{\tau}. \tag{4.14}$$

Calculating the eigenvectors for this matrix is less computationally expensive when compared to the approach from Equations (4.11) and (4.12) because $c << MN$. If $\mathbf{C}_W$ is non-singular, the optimal projection $\mho$ is chosen as the matrix with orthonormal columns which maximizes the ratio of the determinant of the between class scatter matrix of the projected samples to the determinant of the within class scatter matrix of the projected samples, i.e

$$\mho = \arg\max_x \frac{|x^{\tau}\mathbf{C}_B x|}{|x^{\tau}\mathbf{C}_W x|} = [\mho_1\mho_2...\mho_m]. \tag{4.15}$$

where, $\{\mho_i | i = 1, 2, ...m\}$ is the set of generalized eigenvectors of $\mathbf{C}_B$ and $\mathbf{C}_W$ corresponding to the m largest generalized eigenvalues $\{\lambda_i | i = 1, 2, 3...m\}$, i.e,

$$\mathbf{C}_B \mho_i = \lambda_i \mathbf{C}_W \mho_i. \tag{4.16}$$

$$(\mathbf{C}_W^{-1} \mathbf{C}_B) \mho_i = \lambda_i \mho_i. \tag{4.17}$$

The idea behind the Fisherfaces method is that it uses the knowledge of which image belongs to which subject. The definition of $\mho$ from equation (4.15) demonstrates the use of this knowledge as it finds the eigenvectors that maximize the ratio of between class covariance matrix to the within class covariance matrix. This means that the bases given by the span of these eigenvectors will enhance the variance between images of different subjects with respect to the variance among the images within those subjects. The above equation gives at most $(c - 1)$ non-zero eigenvalues and so this can represent an upper bound for the number of eigenvectors which can be used.

For the face recognition problem, one is confronted with the difficulty that the within class scatter matrix $\mathbf{C}_W$ is always singular (that means the determinant of $\mathbf{C}_W$ is zero). This comes from the fact that the rank of $\mathbf{C}_W$ is at most $N - c$ and in general $N << MN$, (the number of images in training set N is much smaller than the number of pixels in each image MN). The problem of singular matrix is solved by projecting the image set to a lower dimensional space so that the resulting within class scatter matrix $\mathbf{C}_W$ is non-singular. This is achieved by using the principal component analysis to reduce the dimensions of the space of size $N - c$, and then apply the standard Fishers discriminant analysis [31], to reduce the space dimension to c-1. Now, $\mho$ is given by

$$\mho^\tau = \mho_{fld}^\tau \mho_{pca}^\tau.$$

where,

$$\mho_{pca} = \arg\max_x |x^\tau \mathbf{C} x|.$$

$$\mho_{fld} = \arg\max_x \frac{|x^\tau \mho_{pca}^\tau \mathbf{C}_B \mho_{pca} x|}{|x^\tau \mho_{pca}^\tau \mathbf{C}_W \mho_{pca} x|}.$$

Where, $\mathbf{C}$ is the covariance matrix which is calculated like in the eigenfaces method, Equation (4.6).

4.4 Face Recognition using Eigendiffusion faces

Diffusion wavelets analysis is used to find the most important orthogonal projections at scale j (i.e the extended bases scale space) that best account for the distribution of face images within the entire image space. These vectors define the subspace of the given set of face images, which is called diffusion face space at scale j. Each vector of length MN, describes an $M \times N$ the image and is a linear combination of the original face images.

These vectors are the orthonormalization projections of the covariance matrix from Equation (4.6) corresponding to the original face images.

Let us consider the training set S, of m face images $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2.....\mathbf{\Gamma}_m$ and each image is transformed into a vector of size MN.

$$\mathbf{S} = \{\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, ..., \mathbf{\Gamma}_m\}. \tag{4.18}$$

where,

$$\mathbf{\Gamma}_m = \begin{pmatrix} \Gamma_{1,m} \\ \Gamma_{2,m} \\ . \\ . \\ \Gamma_{MN,m} \end{pmatrix}. \tag{4.19}$$

The mean face of the set is defined by

$$\Psi = \frac{1}{m} \sum_{i=1}^{m} \mathbf{\Gamma}_i. \tag{4.20}$$

Each face differs from the mean face and that vector represents as,

$$\Phi_{fi} = \mathbf{\Gamma}_i - \Psi. \tag{4.21}$$

In the following we calculate the covariance matrix by using the above vector matrix Φ_f, which has the columns as the difference from the mean face from every training set face. The covariance matrix gives the spread of the face variation within the the training set .

$$\mathbf{C} = \frac{1}{m} \sum_{i=1}^{m} \Phi_{fi} \Phi_{fi}^{\tau} = \mathbf{A}\mathbf{A}^{\tau}. \tag{4.22}$$

Where the matrix $\mathbf{A}$:

$$\mathbf{A} = [\Phi_{f1}, \Phi_{f2}, ..., \Phi_{fm}]. \tag{4.23}$$

Φ_{fi} is given by equation (4.21). The matrix $\mathbf{C}$, is of size $MN \times MN$. We apply the diffusion wavelet analysis to the covariance matrix $\mathbf{C}$. This is a multiple scale representation method using the modified Gram-Schmidt with pivoting at every level as described in Chapter 3. Initially we decide number of levels and the number of extended bases κ. Our algorithm verify the condition number of extended bases are less than or equal to κ at every level j and if this condition is not true then the algorithm proceeds to the next level and again calculates the orthogonal projection for that sub graph. At the last level, the extended scale space is coarser when compared to the initial level extended scale space because at every level we keep only the lower frequency information. The extended bases scale space at level j is known as the eigendiffusion face space. Each column in this extended bases scale space at level j represents the eigendiffusion face at level j.

$$\boldsymbol{\Upsilon} = DWT(C, \epsilon, \kappa, R_\theta, F_\theta, j). \tag{4.24}$$

Where the matrix $\boldsymbol{\Upsilon} = [\boldsymbol{\Upsilon}_1, \boldsymbol{\Upsilon}_2, \boldsymbol{\Upsilon}_3, ...]$. DWT is an algorithm for getting the output of eigendiffusion faces $\boldsymbol{\Upsilon}_i$ and $\{i = 1, 2, ..m\}$ at level j. ϵ is used for pivoting the columns in the Gram-Schmidt orthogonalization, if the norm of the column is less than ϵ then we can remove that column due to its neglectable contribution to data representation. F_θ is the threshold used for reducing the computational time. If the product of two columns is less than this threshold then we don't calculate the orthogonal projection for that column. R_θ is the threshold used to speed up the calculations. If elements in the triangular matrix of the Gram-Schmidt algorithm is below R_θ then we will force that element to be zero. κ is for limiting the extended bases functions. See the diffusion wavelets Chapter 3 to get better idea about the multiscale representation.

In the following we calculate weight matrix of the training set, ω represents the space of

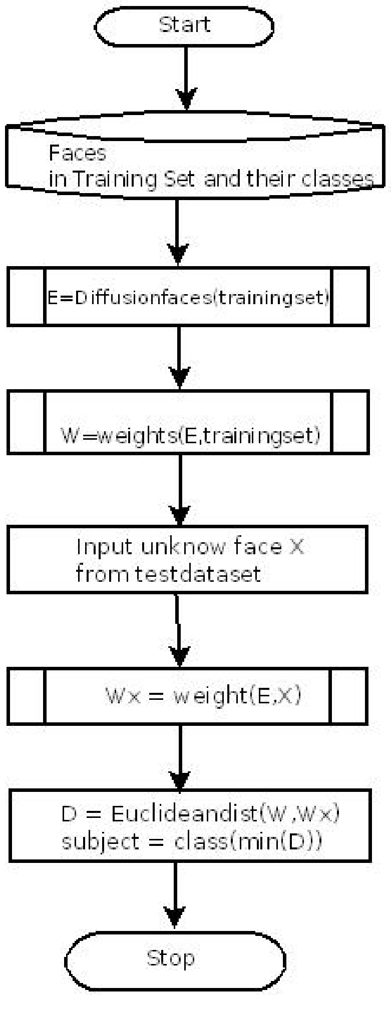


Figure 4.1: Face recognition using the diffusion wavelets flowchart

Algorithm 4.1 Face recognition using the diffusion wavelets

```
$\zeta_c$ = FRUDW($\mathbf{S}, S_c, \zeta, \epsilon, \sigma, \kappa, R_\theta, F_\theta, J, rows, colns$)
//INPUT:
//$\mathbf{S}$: Matrix,each column represents face
//$S_c$: Class for each face in train data(i.e column)
//$\zeta$:face for testing.(i.e column)
//OUTPUT:
//$\zeta_c$:Class for test face
$rc = rows * colns$; $\Psi$ = mean($\mathbf{S}$,2);
for $i$=1 to size($\mathbf{S}$,2) do
  $\Phi_f(all, i) = \mathbf{S}(all, i) - \Psi$;
end for
$\mathbf{C} = \Phi_f * \Phi_f^\tau$;
$\boldsymbol{\Upsilon} = DWT(\mathbf{C}, \epsilon, \sigma, \kappa, R_\theta, F_\theta, J)$;
for $h$=1 to size($\Phi_f$,2) do
  for $i$=1 to size($\boldsymbol{\Upsilon}$,2) do
    $\omega(i, h)$ = dot($\Phi_f(all, h), \boldsymbol{\Upsilon}(all, i)$);
  end for
end for
for $i$=1 to size($\boldsymbol{\Upsilon}$,2) do
  $\varphi(i, 1)$ = dot($(\zeta - \Psi), \boldsymbol{\Upsilon}(all, i)$);
end for
for $i$=1 to size($\boldsymbol{\Upsilon}$,2) do
  $p(i, 1)$ = dot($\zeta, \boldsymbol{\Upsilon}(all, i)$);
end for
$\hat{\mathbf{I}} = \Psi + \boldsymbol{\Upsilon} * \mathbf{p}$;
$\zeta_c$ = class($MinEuc(\omega, \varphi)$);
```

training face difference from the average face Φ_{fi} transformed into eigendiffusion faces,

$$\omega = [\omega_1, \omega_2, ...]. \tag{4.25}$$

$$\omega_i = \mathbf{\Upsilon}^{\tau} * \Phi_{fi}. \tag{4.26}$$

The testface ζ is transformed into its eigendiffusion faces components. First we compare the input test image with the mean face image and multiply their difference with each vector from the eigendiffusion faces. Each value would represent the weight of the test face in the space of eigendiffusion faces.

$$\varphi = \mathbf{\Upsilon}^{\tau} * (\zeta - \Psi). \tag{4.27}$$

We now determine which training face class provides the best description for the input test face image. This is done by minimizing the Euclidean distance,as

$$\Lambda_k = ||\varphi - \omega_k||. \tag{4.28}$$

We can reconstruct our test face by using the eigendiffusion faces. This is done as:

$$\mathbf{p} = \mathbf{\Upsilon}^{\tau} * \zeta. \tag{4.29}$$

$$\hat{\mathbf{I}} = \Psi + \mathbf{\Upsilon} * \mathbf{p}. \tag{4.30}$$

Where, $\hat{\mathbf{I}}$ is the reconstructed image and p represents the input test face ζ depending on the space of eigendiffusion faces $\mathbf{\Upsilon}$. Now, we multiply the vector **p** with eigendiffusion faces $\mathbf{\Upsilon}$ and then add the mean face Ψ to reconstruct the test face ζ and the reason to add mean face is because, initially we normalized each training face as described in Algorithm 4.1.

In the flow chart from Fig 4.1 , we describe the face recognition algorithm by using eigendiffusion faces. In step 1, we prepare the training face data set and assign them labels according to their classes. In step 2, we calculate the eigendiffusion faces from the training data set. Here, **E** represents a matrix containing all the eigendiffusion faces. In order to calculate the eigendiffusion face we follow the above described procedure, see. Equations (4.20 , 4.21 , 4.22 , 4.24) . In step 3, we calculate the weight of each face with the help of the eigendiffusion faces as given by Equation (4.26) . In step 4, we are processing test face and in step 5 we calculate test weight vector with the help of eigendiffusion faces is given by Equation (4.27) . In step 6 we recognise the test face class by identifying the minimum Euclidean distance between test weight vector and one of the train face weight vector, as in Equation (4.28). The eigendiffusion faces are applied to the face recognition and fingerprint authentication in Chapter 6, Experimental results.

4.5 Summary

In this chapter,we described the face recognition using correlation, eigenface, fisherfaces and eigendiffusion faces. We apply eigendiffusion faces approach to recognise the faces and authentication of fingerprint as will be described in Chapter 6. Face recognition using eigendiffusion faces comprises of the following steps: first we calculate the average face of all the training set of faces and then calculate difference of each face and the average face, then calculate the covariance matrix, which gives the variance among the training set. We use diffusion wavelets method to calculate the multiscale representation of extended bases at each level. Then we calculate the training weight matrix with respect to the diffusion faces at the maximum level. In the following we process the test faces and calculate the weight of test face depending on the eigendiffusion faces. The class face is identified which is related to the training face by using the minimum Euclidean distance between weight vectors.

This method is similar to Eigenfaces Section 4.2.1 but here, we have a multi scale representation orthonormal so we can recognise the faces using every level of the extended diffusion bases vectors. The same method is applied for fingerprint authentication.

Chapter 5

Optical Flow Estimation using Diffusion Wavelets

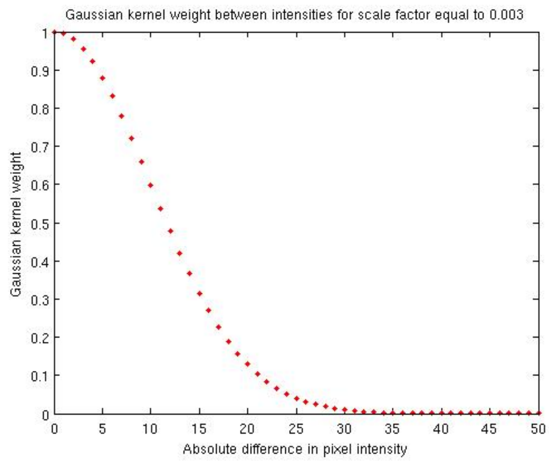
In this chapter, we detail the theoretical framework of applying diffusion wavelets to dense optical flow as well as to sparse motion estimation. The first step to achieve this consists of the feature extraction using diffusion wavelets and then the estimation of the optical flow using matching in the diffusion wavelets space.

5.1 The Diffusion Kernel

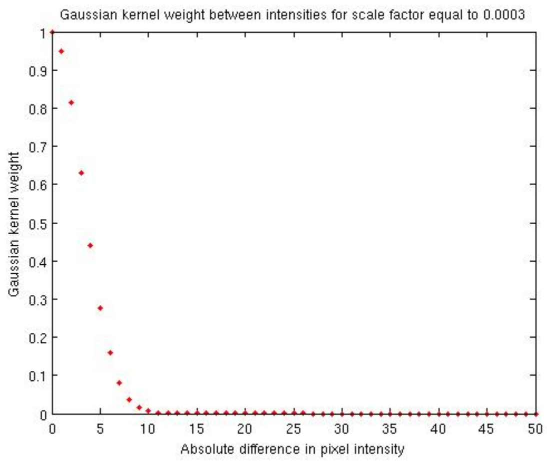
Let us consider a data set which represents a pixel block from an image, $\mathbf{X}_B = \{x_1, ..., x_n\}$, a random walk is constructed by considering the probabilities of moving from x_i to the other points. We consider that every point in the data set is neighbour to all remaining data. Probabilities arise from the kernel function, $K(x_i, x_k)$ that defines the similarity between two data, x_i and x_k. In the present application to feature detection we use the kernel function $K(x_i, x_k) = e^{-||x_i - x_k||^2/\sigma}$. Let us consider the kernel weight matrix $\mathbf{K}$ and $K(x_i, x_k)$ by K_{ik}:

$$\mathbf{K} = \begin{pmatrix} K_{11} & K_{12} & ... & K_{1n} \\ K_{21} & K_{22} & ... & K_{2n} \\ . & . & ... & . \\ . & . & ... & . \\ K_{n1} & K_{n2} & ... & K_{nn} \end{pmatrix}. \tag{5.1}$$

For two data x_i and x_k, which represent normalized image gray scale values in the range [0,1) and a scale factor σ is the scale of the diffusion kernel. The negative exponential function and parameters used yield high degrees of similarity for pixels of close brightness to each other, while yielding lower similarity scores for pixels further away on the grey scale. The scaling function influence in showed in Figure 5.1 . This allows an automatic



(a)



(b)

Figure 5.1: Gaussian functions of the normalized differences in pixel intensity for two scale values that were used in our experiments,(a)$\sigma = 0.003$ and (b)$\sigma = 0.0003$.

mechanism for detecting image features, as well as a good sensitivity to boundaries in the subsequent extended bases feature representation.

5.2 Markov Process and Diffusion Extended Bases

We described in Chapter 3 , how to compute the diffusion kernel and the corresponding Markov chain process. The degree of similarity defined by the Markov transition matrix is $\mathbf{P}$. We can consider the probabilities of transition, $p_t(x_i, x_k)$ for more than one time step by taking higher powers of $\mathbf{P}$ and forming Markov chains. Let us consider the Markov transition matrix $\mathbf{P}$, where $P(x_i, x_k)$) is denoted by P_{ik}.

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & ... & P_{1n} \\ P_{21} & P_{22} & ... & P_{2n} \\ . & . & ... & . \\ . & . & ... & . \\ P_{n1} & P_{n2} & ... & P_{nn} \end{pmatrix}. \tag{5.2}$$

The result from the Markov chain embeds the feature information about the data $\mathbf{X}_B$, while higher values of t increase the propagation of this information to the broader neighbourhood by calculating p^t around the point of origin, x_i. We initially considered that every data is a neighbour to all remaining data. Markov chains $\mathbf{P}$ has maximum possible broader neighbourhood (i.e $n-1$ neighbours in n data set points) around the point of origin, x_i. In order to achieve symmetry, we use the local measure of the degree in graph i.e $D(X_B)$ Equation (3.16) on the diffusion kernel $\mathbf{K}$ Equation (3.15) then we define matrix $\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{K}\mathbf{D}^{-\frac{1}{2}}$ Equation (3.20) . Let us consider the symmetric form of Markov transition matrix $\mathbf{T}$, where we denote $T(x_i, x_k)$ by T_{ik}.

$$\mathbf{T} = \begin{pmatrix} T_{11} & T_{12} & ... & T_{1n} \\ T_{21} & T_{22} & ... & T_{2n} \\ . & . & ... & . \\ . & . & ... & . \\ T_{n1} & T_{n2} & ... & T_{nn} \end{pmatrix}. \tag{5.3}$$

Here, $\mathbf{P}$ is the Markov chain matrix, which is not symmetrical and $\mathbf{K}$ from Equation (3.15) is the symmetric form of the kernel function of the adjacency matrix of the given data set. $\mathbf{T}$ is the symmetrical form of Markov chains and we use this symmetrical matrix as input to diffusion wavelet algorithm in order to achieve multiscale representation. In the diffusion wavelet we can take both symmetrical and non symmetrical forms of Markov chains but here we considered self-adjoint (symmetrical) Markov chains $\mathbf{T}$.

The multiscale analysis generates a sequence of Markov chains, each corresponding to a different time scale, i.e the dyadic power of the original Markov chain, represented on a set of scaling functions in compressed form. We apply $\mathbf{T}$ to $\mathbf{\Phi}_0$ and orthonormalize to get $\mathbf{\Phi}_1$. Each function in $\mathbf{\Phi}_1$ is a linear combination of the original states. Then represent $\mathbf{T}^2$ on $\mathbf{\Phi}_1$, to get a matrix $\mathbf{T}^4$, apply to $\mathbf{\Phi}_1$ and orthonormalize, and continue this procedure until a set of conditions are fulfilled. The diffusion wavelets extended bases at each level embed the feature representation of the given data.

In diffusion maps, the diffusion distances from Equation (3.8) are used for feature representation. We also use this approach for feature representation based on the extended bases functions, at level j. Let us consider $\mathbf{\Phi}_j$ at level j is,

$$\mathbf{\Phi}_j = \begin{pmatrix} \Phi_{j1}(x_1) & \Phi_{j2}(x_1) & ... & \Phi_{jm}(x_1) \\ \Phi_{j1}(x_2) & \Phi_{j2}(x_2) & ... & \Phi_{jm}(x_2) \\ . & . & ... & . \\ . & . & ... & . \\ \Phi_{j1}(x_n) & \Phi_{j2}(x_n) & ... & \Phi_{jm}(x_n) \end{pmatrix}. \tag{5.4}$$

Where, m is the number of extended bases functions at level j as in the formula:

$$D_{\Phi_j}(x_i, x_k) = \Big(\sum_{l=1}^{m} (\Phi_{jl}(x_i) - \Phi_{jl}(x_k))^2 \Big)^{\frac{1}{2}}. \tag{5.5}$$

Let us consider the whole extended bases distance matrix at level j, which is given by the following:

$$\mathbf{D}_{\mathbf{\Phi}_j} = \begin{pmatrix} D_{\Phi_j}(x_1, x_1) & D_{\Phi_j}(x_1, x_2) & ... & D_{\Phi_j}(x_1, x_n) \\ D_{\Phi_j}(x_2, x_1) & D_{\Phi_j}(x_2, x_2) & ... & D_{\Phi_j}(x_2, x_n) \\ . & . & ... & . \\ . & . & ... & . \\ D_{\Phi_j}(x_n, x_1) & D_{\Phi_j}(x_n, x_2) & ... & D_{\Phi_j}(x_n, x_n) \end{pmatrix}. \tag{5.6}$$

Now, we calculate feature representation of the x_i by summing up the rows of extended bases distance matrix and we represent it as $\mathbf{Y}_{B_i}$, which is given by the following:

$$\mathbf{Y}_{B_i} = \sum_{k=1}^{n} D_{\Phi_j}(x_i, x_k). \tag{5.7}$$

This proposed method provide an efficient representation of image feature orientation around a given pixel as it relates pairs of points based on all possible paths of length 2^j that connect the pixels. This also makes the representation robust in the presence of noise as the distances are less influenced by changes to individual pixels. We give several examples of features represented by diffusion extended bases distances from the experimental results Section 6.1 .

5.3 Estimating Dense Optical flow

Let, $\mathbf{I}_1$ and $\mathbf{I}_2$ are the two successive frames in an image sequence. We define the diffusion window with l pixel translation on both x and y directions. Then we calculate the extended bases at the maximum level for all of the blocks in both images. Now, we have the extended bases of the blocks. Consider each block extended bases in $\mathbf{I}_2$ and calculate the Euclidean distances corresponding to a search window in $\mathbf{I}_1$. We calculate the displacement of a diffusion window in x and y directions by identifying which the minimum Euclidean distance extended bases representation of block in search window. Repeat this steps for all the blocks in $\mathbf{I}_2$.

The diffusion wavelet framework to model the extended bases at the last level, which is smoother and coarser extended bases function. Let us consider, $\mathbf{\Phi}_p$ as data representation of block in frame p, which is also known as diffusion extended bases for particular block in frame p and $\mathbf{\Phi}_{p+1}$ as data representation of block in frame $p+1$. To estimate the optical flow we use matching of the extended bases functions corresponding to two blocks of pixels from two different image frames. The match corresponds to the the minimum Euclidean distance between two extended bases of blocks each corresponding to image frame from an image sequence, and this indicates the blocks are similar in structure. We have two windows for constructing this procedure, the diffusion window and the search window. The diffusion window defines the image region as a block of pixels for extracting extended bases at maximum level. The search window defines the image region and by one pixel translation we can possible to have multiple block of pixels with the same size of diffusion window. Then we calculate extended bases functions for all possible blocks in search window. After, that we calculate maximum correlated block of pixels corresponding to the diffusion window block of pixels. We can achieve this by following equation:

$$\mathsf{E} = \arg\min_{k,l} \sum_{b \in \mathbf{X}_B} (\Phi_p(i,j) - \Phi_{p+1}(i+k, j+l))^2. \tag{5.8}$$

where, p is the image frame in sequences, b is the number of the pixels in diffusion window $\mathbf{X}_B$, k and l are the offsets in search window. Algorithm 5.1, describes our proposed algorithm to estimate optical flow. Mainly it has two stages, which corresponds to feature representation from the diffusion extended bases at last level and then calculate Euclidean distances of extended bases of possible blocks to estimate optical flow. In flow chart from the Figure 5.2, we present our proposed dense optical flow estimation. In steps 1 and 2 , we calculate the diffusion wavelet extended bases of each block in both frames at last level. Here, F_θ is a threshold to decide the column whether to calculate orthogonal projections or not in modified Gram-Schmidt algorithm. If the two columns product is above threshold then we consider those projections otherwise we discard the column, R_θ is the threshold for elements in R component in modified Gram-Schmidt with pivoting column algorithm. If element falls below threshold then we force that element to zero, and κ is threshold for number of bases functions in the diffusion extended bases, J is represents the number of levels, and ϵ is the precision for pivoting the column in modified Gram-Schmidt with pivoting column algorithm. Now, we have extended bases function representation for each defined block in both frames.

In step 3, we take a block from frame $\mathbf{I}_2$ and it is named as diffusion window then we define the search window in frame $\mathbf{I}_1$ corresponding to the diffusion window with offsets $\pm k$. In step 4, we calculate Euclidean distance of all diffusion extended basis blocks in search window with respect to the diffusion extended functions of diffusion window block. Now, we identify which block in search window has minimum Euclidean distance and store those displacement values. The algorithm repeats for all the diffusion windows in frame $\mathbf{I}_2$. After calculating the optical flow for every block in frame $\mathbf{I}_2$ then we calculate the we calculate the optical flow which corresponds to the displacement of the diffusion wavelet representation. We provide dense optical flow results on various image sequences in Chapter 6.

5.4 Estimating sparse optical flow from the Euclidean distances

In this section, we detail our algorithm for sparse optical flow estimation using the diffusion wavelets. This consists of three steps, first we locate the key points using Scale Invariant Feature Transform (SIFT) [32], then we find the extended basis of surround blocks by taking the key points as pixel block centres and we consider the corresponding search

Algorithm 5.1 Dense optical flow using Diffusion Wavelets

```
$[\mathbf{U},\mathbf{V}] = DOFEUDW(\mathbf{I}_1,\mathbf{I}_2,\epsilon,\sigma,F_\theta,R_\theta,J,\kappa,dimx,dimy,srx,sry)$
//INPUT:$\mathbf{I}_1$ and $\mathbf{I}_2$ are Two Input Matrix(i.e images)
//OUTPUT:$\mathbf{U}$ and $\mathbf{V}$ are Optical flow in X and Y-direction.
for $i$ = 1 to ($dimy$-$blocky$+1) do
  for $j$ = 1 to ($dimx$-$blockx$+1) do
    $\mathbf{T}$ = MT($I_2$($i$ to ($i$+$blocky$-1),j to ($j$+$blockx$-1)),$\sigma$)
    $ExtBasfr2${$i$,1}($all,j$)=DWT($\mathbf{T},\kappa,F_\theta,R_\theta,J,\epsilon$)
    $\mathbf{T}$ = MT($\mathbf{I}_1$($i$ to ($i$+$blocky$-1),$j$ to ($j$+$blockx$-1)),$\sigma$)
    $ExtBasfr1${$i$,1}($all,j$)=DWT($\mathbf{T},\kappa,F_\theta,R_\theta,J,\epsilon$)
  end for
end for
for $i$ = 1 to size($ExtBasfr2${1,1},2) do
  for $j$ = 1 to size($ExtBasfr2${1,1},2) do
    $block$ = $ExtBasfr2${$i$,1}($all,j$);$leuc = inf$;
    for $y$ = -$sry$ to $sry$ do
      for $x$ = -$srx$ to $srx$ do
        $cureuc$ = sum$((block - ExtBasfr1\{i,1\}(all,j)).^2)$;
        if $cureuc < leuc$ then
          $leuc = cureuc$;
          $U(i,j) = -x$;
          $V(i,j) = -y$;
        end if
      end for
    end for
  end for
end for
```

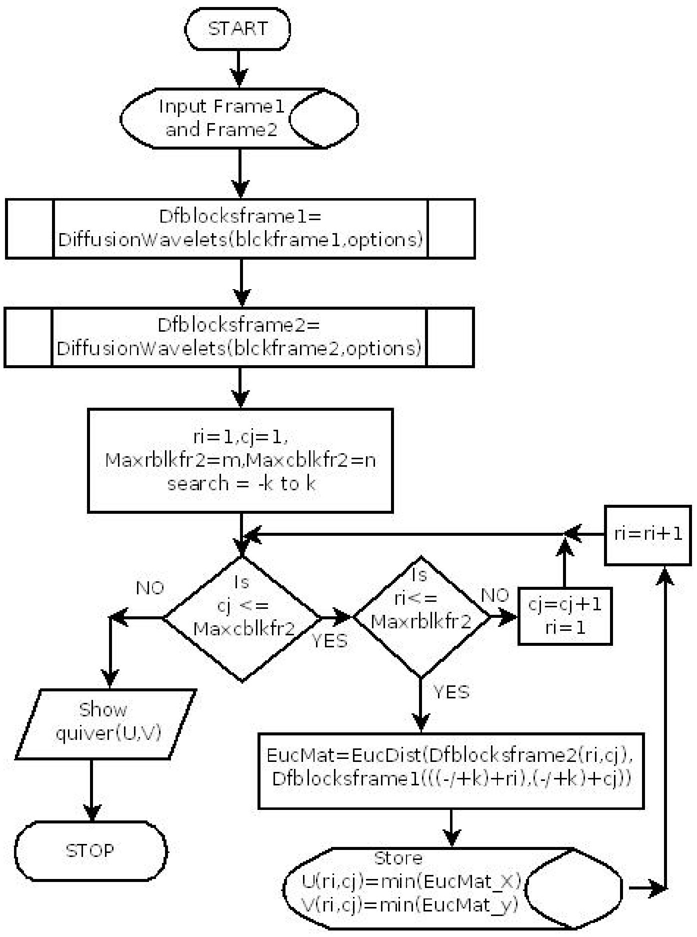

START
Input Frame1 and Frame2
Dfblocksframe1= DiffusionWavelets(blckframe1,options)
Dfblocksframe2= DiffusionWavelets(blckframe2,options)
ri=1,cj=1, Maxrblkfr2=m,Maxcblkfr2=n search = -k to k
ri=ri+1
Is cj <= Maxcblkfr2
NO
YES
Is ri<= Maxrblkfr2
NO
cj=cj+1 ri=1
YES
Show quiver(U,V)
STOP
EucMat=EucDist(Dfblocksframe2(ri,cj), Dfblocksframe1(((-/+k)+ri),(-/+k)+cj))
Store U(ri,cj)=min(EucMat_X) V(ri,cj)=min(EucMat_y)

Figure 5.2: Dense optical flow estimation using Diffusion Wavelets flowchart

window blocks in the other image, repeat this step for the remaining locations. The third step consists in finding the x and y direction displacement of each key point location blocks.

5.4.1 Scale Invariant Feature Transform

SIFT [32] algorithm is used to find the scale invariant feature locations of the image. Simple corner detector are used for matching features between images when all images have similar scale, orientation, illumination. Usually images do not have the same properties and in such case we cannot match features accurately using the simple corner detection method. To overcome this problem, Scale Invariant Feature Transform (SIFT), transforms image data into scale-invariant coordinates relative to local features [32]. SIFT is not only used for scale invariant but also for accounting to the variation in the illumination, rotation and viewpoint variants. SIFT algorithm has four steps to generate features of an image and these steps are explained in below.

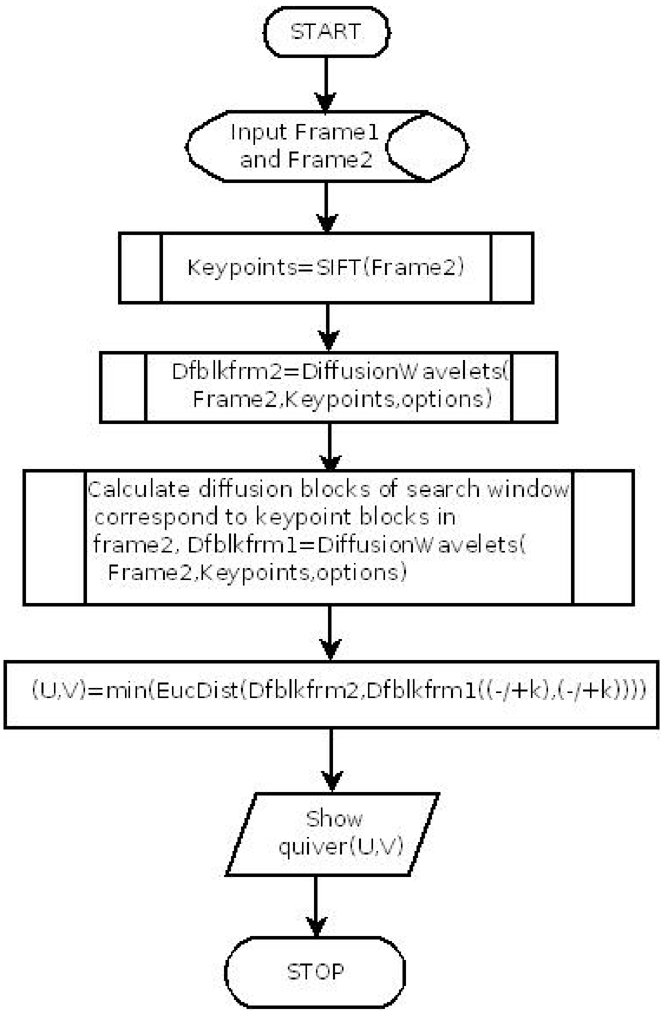

START
Input Frame1 and Frame2
Keypoints=SIFT(Frame2)
Dfblkfrm2=DiffusionWavelets(Frame2,Keypoints,options)
Calculate diffusion blocks of search window correspond to keypoint blocks in frame2, Dfblkfrm1=DiffusionWavelets(Frame2,Keypoints,options)
(U,V)=min(EucDist(Dfblkfrm2,Dfblkfrm1((-/+k),(-/+k))))
Show quiver(U,V)
STOP

Figure 5.3: Sparse optical flow using Diffusion Wavelets flow chart

Scale Space Extrema detection

The scale space of an image is defined as a function, $L(x, y, \sigma)$, that is produced from the convolution of a variable scale Gaussian $G(x, y, \sigma)$ with an input image, I(x,y).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y). \tag{5.9}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-(x^2+y^2)/2\sigma^2}. \tag{5.10}$$

To detect stable key point locations in the scale space,$D(x, y, \sigma)$,computed from the the difference of two nearby scales the separated by a constant multiplicative factor k as in the following equation:

$$D(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma). \tag{5.11}$$

An efficient approach to the construction of $D(x, y, \sigma)$, is that the initial image is incrementally convolved with Gaussian functions to produce blurred images separated by a

Algorithm 5.2 sparse optical flow using Diffusion Wavelets

```
$[\mathbf{U},\mathbf{V}] = SOFEUDW(\mathbf{I}_1, \mathbf{I}_2, \epsilon, \sigma, F_\theta, R_\theta, J, \kappa, dimx, dimy, srx, sry))$
//INPUT:$\mathbf{I}_1$ and $\mathbf{I}_2$ are Two Input Matrix(i.e images)
//OUTPUT:$\mathbf{U}$ and $\mathbf{V}$ are Optical flow in X and Y-direction.
$KeyLoc$ = SIFT($\mathbf{I}_2$);
$ofblkx$ = floor($blockx$/2);$ofblky$ = floor($blocky$/2);$lpi$=0;
for $i$ = 1 to size($KeyLoc$,1) do
  $lpi = lpi + 1; cnt1 = 0; leuc = inf; y1 = -ofblky + KeyLoc(i,1)$
  $y2 = ofblky + KeyLoc(i,1); x1 = ofblkx + KeyLoc(i,2); x2 = ofblkx + KeyLoc(i,1)$;
  $\mathbf{T}$ = MT($frame2(y1$ to $y2, x1$ to $x2), \sigma$)
  $ExtBasfr2\{lpi,1\}(all,j)$=DifWav($\mathbf{T}, \kappa, F_\theta, R_\theta, J, \epsilon$)
  for $j = -sry$ to $sry$ do
    $cnt1 = cnt1 + 1; cnt2 = 0$;
    for $k = -srx$ to $srx$ do
      $cnt2$=$cnt2$+1;$\mathbf{T}$ = MT($frame1(y1$ to $y2$,$x1$ to $x2$),$\sigma$);
      $ExtBasfr1\{lpi,1\}cnt1, cnt2$=DifWav($\mathbf{T}, \kappa, F_\theta, R_\theta, J, \epsilon$)
      $cureuc$ = sum(($ExtBasfr2\{lpi,1\}(all,1) - ExtBasfr1\{lpi,1\}\{cnt1,cnt2\}(all,1)).^2$);
      if $cureuc < leuc$ then
        $leuc = cureuc$;
        $U(KeyLoc(i,1), KeyLoc(i,2)) = -k$;
        $V(KeyLoc(i,1), KeyLoc(i,2)) = -j$;
      end if
    end for
  end for
end for
```

constant factor k in the scale space and we choose to divide each octave of scale space into integer numbers, s of intervals $k = 2^{\frac{1}{s}}$. This resolution images in the stack of blurred images for each octave so that final extrema detection covers a complex octave.

Key point Localization

Now, we generate the Laplacian of Gaussians (LoG) by applying second order derivative or Laplacian on the above scale space to locate the edges and corners on the image. These edges and corners are good for finding key points on the image. Usually, the Laplacian is sensitive to noise and blur in scale space smooths it out the noise and stabilizes the Laplacian. The problem is calculating second order derivatives to all in scale space is computationally high. To over come this, for calculating Laplacian of Guassians quickly, we use the scale space. We calculate the difference between the images blurred with adjacent image scales in an octave, which is called as Difference of Gaussians (DoG). DoG is approximately similar to LoG and this process is reduces the computational intensity. This DoG process gives another benefit to us that the above approximations are scale invariant. LoG are not scale invariant, because the scale σ^2, depend on the amount of blur in Gaussian expression given in Equation (5.10). But this taken care of by the Difference of Gaussian operation. The resultant images after the DoG operation are already multiplied by the scale σ^2, and it has produces much better key points than which we get by LoG. The DoG result is also multiplied by a constant factor $(k - 1)$. But, it won't give any problem because we are looking for only the location of the maxima and minima in the images and never check the actual values at those locations. So, this additional factor won't be a problem and even multiply throughout by some constant, the maxima and minima stay at the same location. Maxima and minima in the DoG images are generated to comparing neighbouring pixel in the current scale, and the lower and higher scales. Then low contrast features edges are eliminated. Corners always gives better key points. After this step we have scale invariant key points.

Orientation Assignment

One or more orientations are assigned to each key point location based on local image gradient directions. All the following processing operations are performed on the image data that has been processed according to the assigned orientation, scale and location for each feature.

Key point descriptor

The local image gradients are measured at the selected scales in the region around each key point. These are transformed into a representation that allows for significant levels of local shape distortion and changes in illumination.

5.4.2 Calculating Diffusion Wavelet Extended Bases

$\mathbf{I}_1$ and $\mathbf{I}_2$ are two successive image frames. We apply the SIFT algorithm on frame $\mathbf{I}_2$ and calculate key point location on that image. We use those key points as central pixel to the blocks which represents the diffusion window center and then we define the search window on image $\mathbf{I}_1$ in an area around the diffusion window location. Calculate the extended bases at maximum level l of the QR orthogonalisation for each block in the diffusion window and blocks in the search window. We repeat the same step to remaining key locations.

5.4.3 Optical flow using Euclidean distance in the diffusion wavelet space

We evaluate the minimum Euclidean distance between the corresponding diffusion extended bases functions for each key point location and then we identify the displacement in both x and y directions. This step is repeated to all key point locations.

Our proposed sparse optical flow estimation algorithm is described in Algorithm 5.2 . In the flowchart 5.3 , we given high level functioning principal of our proposed sparse optical flow estimation. In step 1, we collect the image sequence from the data base and in step 2, we process frame $\mathbf{I}_2$ to identify key point locations. In step 3, we define diffusion window in frame $\mathbf{I}_2$ by considering key point locations as centres to the block and we also define corresponding search window in frame $\mathbf{I}_1$ with offset $\pm k$. Apply diffusion wavelet algorithm on each defined blocks and find their feature representations i.e extended bases. In step 4, we use the minimum Euclidean distance for each diffusion window to its search window for estimating optical flow by identifying the displacement of block in search window which is having minimum Euclidean distance from the diffusion extended bases functions.

5.5 Summary

In this chapter, we presented our algorithm for dense and sparse optical flow estimation. The first step involves calculating the extended bases functions at different levels. Here, we considered only the extended bases which corresponds the maximum level because this represents extended bases which are coarser than the earlier bases at the lower level. The second step consists of calculating Euclidean distances between the diffusion wavelet vectors and taking the minimum distance for calculating the displacement of the block in both x and y directions between the frame $\mathbf{I}_1$ and $\mathbf{I}_2$.

Sparse optical flow estimation in Subsection 5.4, comprises of steps: first calculating the features at scale invariant locations of the image, the second step, we calculate the extended diffusion wavelets, and third one we calculate Euclidean distance between the diffusion extended bases functions for each key point and we take the minimum Euclidean distance and we identify the displacement. The main advantage of this is that computationally it takes less time than dense optical flow estimation Subsection 5.3 because the displacements are calculated only for few locations in the image.

Chapter 6

Experimental Results

In the following we present the results when applying diffusion wavelets methodology. In the following experiments we consider different data sets such as:

1. To estimate optical flow we used following data sets such are,
   - Middlebury data set from which we use the Dimetrodon and Venus image sequences [34] .
   - Hamburg taxi sequence.
   - Andrea Hurricane image sequence.
   - Infra-red Meteosat image sequence of fluid motion from Irisa [37] .
2. Cornea image sequence for image Registration.
3. To face recognition we used following face databases such are,
   - ORL face database [35] .
   - Yale face database.
4. Fingerprint verification competition data set (FVC2000) for fingerprint authentication [36] .

We take various images and processed each image with the help of diffusion wavelets methodology for feature representation. In the following we show the feature representation at each level and mention about their details. After that we estimate sparse and dense optical flow of various image data sets by using the proposed methodology. There are two stages to the dense optical flow estimation, which is the feature representation with diffusion extended bases functions and estimation of the optical flow using the minimum Euclidean distance of diffusion extended bases functions. For sparse optical flow estimation we have three stages, which are applying the SIFT on image to get the key image point locations and feature extraction for the blocks which correspond to each block and estimation of the optical flow using the diffusion extended bases functions. We use various image sequence for our dense optical flow estimation algorithm and we also include quantitative results for two Middlebury image sequences.

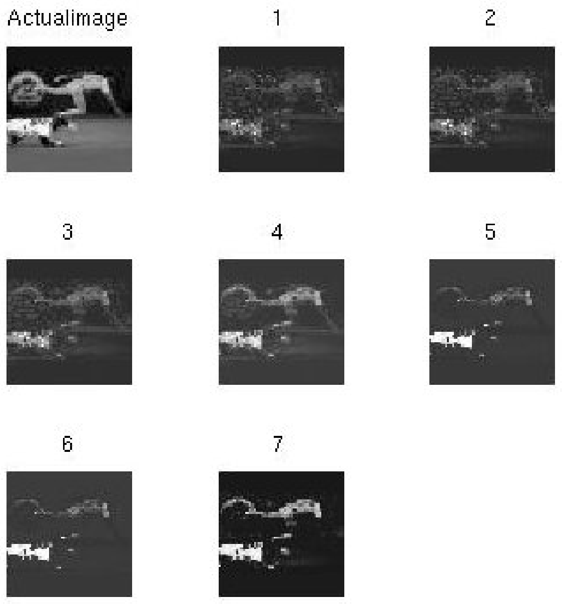


Figure 6.1: Football players image multiple scale feature representation

Finally, we show the experiment results of our proposed face recognition method on ORL face data base and we also include the experiment results of fingerprint authentication by using the same approach, which was used for face recognition.

6.1 Diffusion extended bases functions as feature descriptor in images

We present several examples of modelling features using diffusion extended bases functions from various images. In all the experiments we have chosen the scale parameter $\sigma = 0.03$, precision for pivoting the column in modified QR Gram-Schmidt $\epsilon = 10^{-6}$, threshold for the triangular matrix elements in modified QR Gram-Schmidt $R_\theta = 2.2204 \times 10^{-16}$, and the inner product column threshold for modified QR Gram-Schmidt $F_\theta = 10^{-12}$, required extended bases functions in diffusion wavelets algorithm is $\kappa = 1$ i.e our algorithm stops when we achieve the one extended bases function, and the required levels $J = 10$, but mostly we will get the required extended bases functions κ for less than 10 levels, our algorithm stops when we get our required extended bases functions κ See the convergence condition from Chapter 3.

Figure 6.1, shows the multi scale representation of the football players features by using the diffusion wavelets algorithm. Here, we considered the whole image as a kernel window of size 50×50. From the diffusion wavelet algorithm, we consider each pixel as a node and their intensity values are considered for the calculation of weights in the graph. We assume that our graph is undirected. We calculated Markov transition matrix $\mathbf{T}$ for this graph and it has the dimensionality 2500×2500. We applied multi scale dimensionality reduction algorithm on the Markov transition matrix. We took initial extended bases as identity matrix with the same size of Markov transition matrix i.e $\mathbf{\Phi}_0$ has 2500×2500 size of identity matrix. At the first level, Markov matrix $\mathbf{T}$ is reduced to 15×15 with time scale 2 (i.e $[\mathbf{T}]^2$) and extended bases scale functions $\mathbf{\Phi}_1$ dimensionality is reduced to 2500×15, i.e we represent the data by using only 15 orthonormal vectors. Now, we calculate the extended bases distance from $\mathbf{\Phi}_1$ by using the Equation (5.5) and representation of each pixel is calculated by using Equation (5.7) . We display this representation of image in Figure 6.1 as 1 and this feature representations is not accurate because it holds noise information more compared to the feature information. At second level, Markov matrix reduced to 11×11 with time scale 4 and extended bases functions dimensionality $\mathbf{\Phi}_2$ reduced to 2500×11 and we showed representation of image in Figure 6.1 as 2 , which is achieved by using the extended bases distances approach. This representation of image is better than earlier level but it still holds more noise information. So, this representation is not accurate. At third level, Markov matrix reduced to 7×7 with time scale 8 and extended bases functions $\mathbf{\Phi}_3$ dimensionality reduced to 2500×7 and we showed representation of image in Figure 6.1 as 3. At fourth level, Markov matrix reduced to 5×5 with time scale 16 and extended bases functions $\mathbf{\Phi}_4$ dimensionality reduced to 2500×5 and we showed representation of image in Figure 6.1 as 4. At fifth level, Markov matrix reduced to 3×3 with time scale 32 and extended bases functions $\mathbf{\Phi}_5$ dimensionality reduced to 2500×3 and we showed representation of image in Figure 6.1 as 5. At sixth level, we don't have any reduction in our Markov matrix and extended bases $\mathbf{\Phi}_6$ and the scale has increased to next dyadic power 64. The weights in the functions are reduced(i.e values of elements in the matrix are less than earlier) and the reason for this is all the columns are under threshold and we showed representation of image in Figure 6.1 as 6. At seventh level Markov matrix reduced to 1×1 with time scale 128 and extended bases functions $\mathbf{\Phi}_7$ dimensionality reduced to 2500×1 and we showed representation of image in Figure 6.1 as 7. We understood that this level representation is better, and still it excludes some feature information due to the precision for pivoting the columns in modified Gram-Schmidt i.e we have taken same precision for all experiment which is accurate to most of the cases. Earlier we mentioned about κ considered as 1 so, our algorithm terminated at end of the 7th level. From this we understood that, number of levels will vary from one graph to another. In the Table 6.1, we given multi scale reduction details for football players image.

Figure: 6.2, used for multi scale representation of an image of a deer. Kernel window

Table 6.1: Football players image multi scale reduction details

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	15×15	2500×15	2
2	11×11	2500×11	4
3	7×7	2500×7	8
4	5×5	2500×5	16
5	3×3	2500×3	32
6	3×3	2500×3	64
7	1×1	2500×1	128

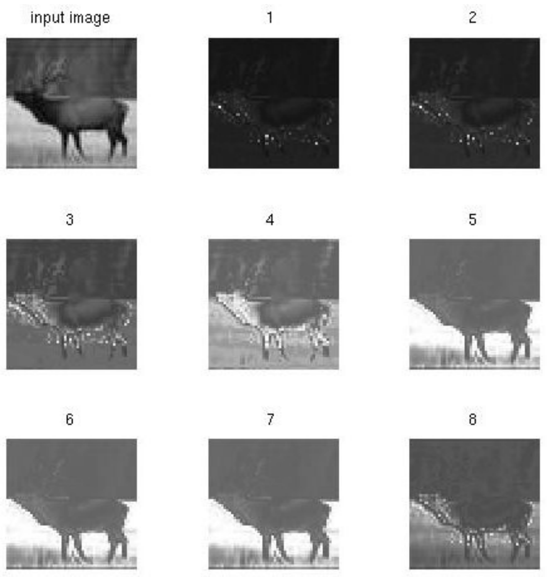


Figure 6.2: Deer image multiple scale feature representation

is 50×50. In the figure we showed feature representation from the image at each level. It took eight levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix $\mathbf{T}$ and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We have $\mathbf{T}$ with 2500×2500 and $\mathbf{\Phi}_J$ with 2500×2500 (i.e identity matrix of this size). In the Table 6.2, we given multi scale reduction details for deer image.

Table 6.2: Deer image multi scale reduction details

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	13×13	2500×13	2
2	10×10	2500×10	4
3	6×6	2500×6	8
4	4×4	2500×4	16
5	2×2	2500×2	32
6	2×2	2500×2	64
7	2×2	2500×2	128
8	1×1	2500×1	256

6.1.1 Multiscale feature representation of animal on tree branch image

Figure: 6.3, used for multi scale representation of an image of some animal on tree branch. Kernel window is 60×40. In the figure we showed feature representation from the image. It took 6 levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix $\mathbf{T}$ and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We have $\mathbf{T}$ with 2400×2400 and $\mathbf{\Phi}_J$ with 2400×2400 (i.e identity matrix of this size). In the Table 6.3, we given multi scale reduction details for an animal on tree

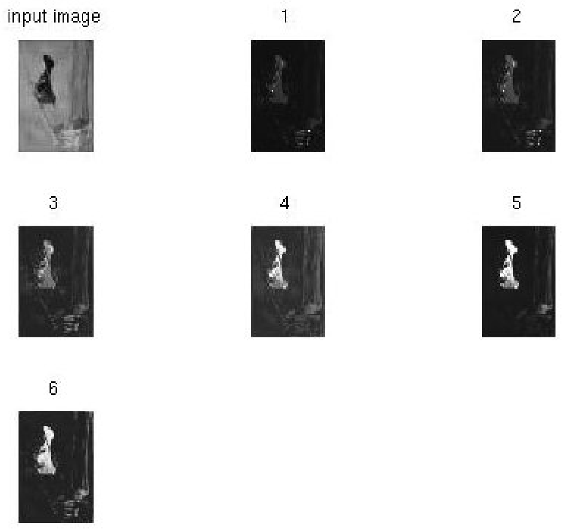


Figure 6.3: Multiple scale feature representation of animal on the tree branch

branch image.

Table 6.3: Animal on branch of tree feature multi scale reduction

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	12×12	2400×12	2
2	8×8	2400×8	4
3	5×5	2400×5	8
4	3×3	2400×3	16
5	2×2	2400×2	32
6	1×1	2400×1	64

6.1.2 Multiscale feature representation of skiing person image

Figure: 6.4, used for multi scale representation of an image of skiing person image. Kernel window is 47×60. In the figure we showed feature representation from the image. It took 7 levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix $\mathbf{T}$ and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We have $\mathbf{T}$ with 2820×2820 and $\mathbf{\Phi}_J$ with 2820×2820 (i.e identity matrix of this size). In the Table 6.4, we given multi scale reduction details for skiing person image.

6.1.3 Multiscale feature representation of hand image

Figure: 6.5, used for multi scale representation of an image of hand. Kernel window is 40×50. In the figure we showed feature representation from the image. It took 7 levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix $\mathbf{T}$ and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We

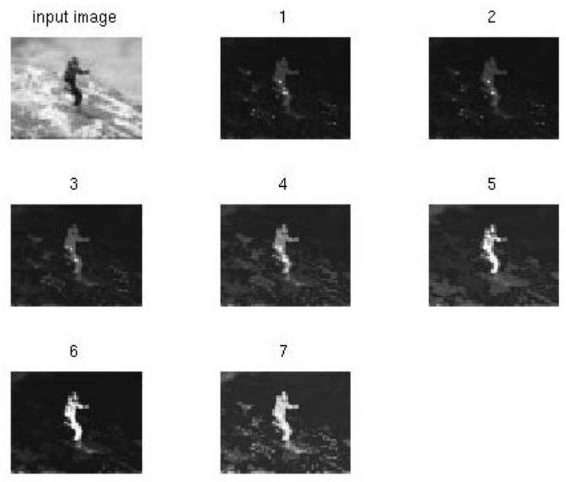


Figure 6.4: Skiing person feature multi scale reduction

Table 6.4: Skiing person feature multi scale reduction

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	15×15	2820×15	2
2	11×11	2820×11	4
3	7×7	2820×7	8
4	5×5	2820×5	16
5	3×3	2820×3	32
6	2×2	2820×2	64
7	1×1	2820×1	128

have $\mathbf{T}$ with 2000×2000 and $\mathbf{\Phi}_J$ with 2500×2500 (i.e identity matrix of this size). In the Table 6.5, we given multi scale reduction details for hand image.

Table 6.5: Hand feature multi scale reduction

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	8×8	2000×8	2
2	6×6	2000×6	4
3	3×3	2000×3	8
4	5×2	2000×2	16
5	3×2	2000×2	32
6	2×2	2000×2	64
7	1×1	2000×1	128

6.1.4 Multiscale feature representation of ballerina image

Figure: 6.6, used for multi scale representation of an image of ballerina. Kernel window is 50×50. In the figure we showed feature representation from the image. It took 6 levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix $\mathbf{T}$ and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We

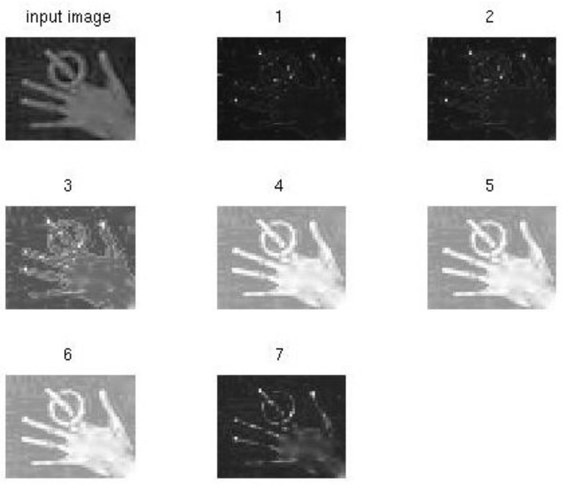


Figure 6.5: Multi scale feature representation of hand

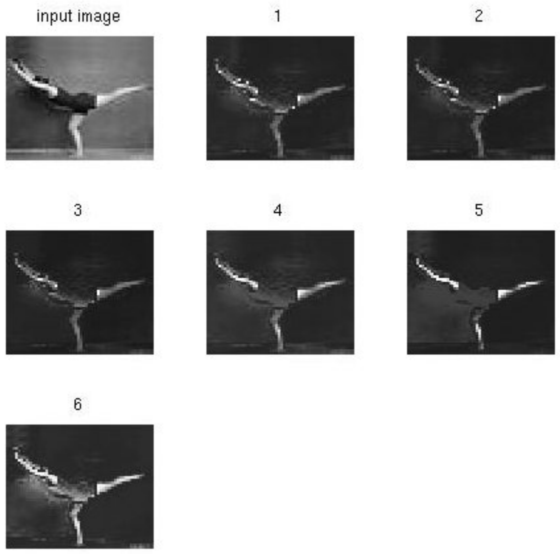


Figure 6.6: Multiple scale feature representation of a ballerina.

have **T** with 2500×2500 and $\mathbf{\Phi}_J$ with 2500×2500 (i.e identity matrix of this size). In the Table 6.6, we given multi scale reduction details for ballerina image.

6.1.5 Multiscale feature representation of face image

Figure: 6.7, used for multi scale representation of an image of face. Kernel window is 56×46. In the figure we showed feature representation from the image. It took 9 levels to get coarser extended bases scale function. As we discussed earlier, the last level functions have better feature representation than the previous levels. The Transition matrix **T** and extended bases function $\mathbf{\Phi}_J$ dimensionality reduced at level like below, Initially, We have **T** with 2576×2576 and $\mathbf{\Phi}_J$ with 2576×2576 (i.e identity matrix of this size). In the Table 6.7, we given multi scale reduction details for face image.

Table 6.6: Multiple scale dimensionality reduction of a ballerina

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extend Bas,$[\mathbf{\Phi}_J]$	time scale(2^J)
1	15×15	3000×15	2
2	11×11	3000×11	4
3	7×7	3000×7	8
4	4×4	3000×4	16
5	3×3	3000×3	32
6	1×1	3000×1	64

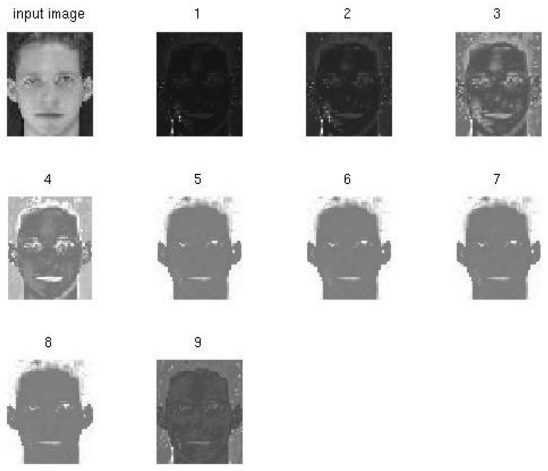


Figure 6.7: Multiple scale feature representation of a face.

6.2 Optical flow and Image Registration by using Extended diffusion bases functions

In this section we present the result when we apply our sparse and dense optical flow estimation on various data sets. As was described in Chapter 5, We consider 20×20 blocks for all the images used for optical flow and the image registration experiments. Initially, we tried for one pixel translation on both x and y direction but it is computationally intensive to find optical flow vectors. In order to reduce computational time, we considered the blocks in second frame with ten pixel translation and the blocks in first frame with one pixel translation on x and y directions. The next step is to calculate the extended bases for each defined block of both frames using the parameters which we discussed earlier. We calculate extended bases of each block by using multi scale dimensionality reduction using the diffusion wavelets. From the results the extended bases corresponding to the last level is a better representation than the previous levels because at each level that we reduce, we are recording the low frequency information and neglecting the noise content. In this way we will get a better representation at last level. So, we consider the last level extended bases for representing the each pixel block from the image. In Section 5.3, it was described a method to identify the displacement of maximum correlated block in the search window corresponding to the diffusion window in second frame.

The above methodology was used for dense optical flow estimation and the methodol-

Table 6.7: Multiple scale dimensionality reduction of a face

Level J	Markov,$[\mathbf{T}]^{2^J}$	Extended Bas,$[\mathbf{\Phi}_J]$	scale(2^J)
1	12×12	2576×12	2
2	8×8	2576×8	4
3	5×5	2576×5	8
4	3×5	2576×3	16
5	2×2	2576×2	32
6	2×2	2576×2	64
7	2×2	2576×2	128
8	2×2	2576×2	256
9	1×1	2576×1	512

ogy we can use for sparse optical flow estimation with some changes.

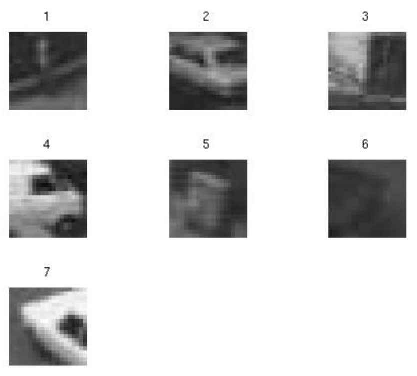
6.2.1 Dense Optical flow Estimation

We have experimented on the following database Hamburg taxi sequence, Andrea Hurricane,infrared Meteosat fluid flow, the artificial image sequences Venus and Dimetrodon from the Middlebury database and cornea image sequence showing cornea layers for Image registration.

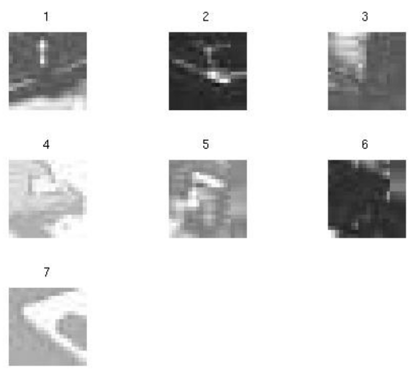
Hamburg Taxi Image Sequence

Hamburg taxi sequence is a scene involving four moving objects - a car on the left driving left to the right , a taxi near the centre turning the corner, a pedestrian in the upper left moving on the pavement and a van on the right moving to the left. The scene contains many other static objects too. In this section we describe the experimental results on the Hamburg taxi sequence by using our proposed dense optical flow estimation, which was introduced in Section 5.3 . We skipped one frame in between each two frames (i.e in immediate sequence of frames have too small optical flow and visually we can't see properly), which shows in Figure 6.9a and 6.9b , we describe the extended bases functions and applied our dense optical flow estimation process. Each image in this sequence has the size of 191 × 256 .

In Figure 6.8a, is showed different image blocks from the taxi sequence and we wish to describe feature representation of those blocks by using multi scale dimensionality reduction. We showed multi scale feature representation for various image features in Section 6.1 , and we explained dimensionality reduction of extended bases function at each level in this representation. From the experiments we understood that the last level extended bases are well defined (due to removing the high frequency information at each level) when compared the previous levels. So, for optical flow estimation we used the extended bases at the last level when representing each block in the image sequence. The number of levels vary from one block to another and depends on the thresholds to as explained in Section 5.3 even though we used the same parameters to calculate the extended bases



(a) 20 x 20 block parts



(b) Feature representation

Figure 6.8: Feature representation of 20 x 20 block parts in Hamburg taxi image

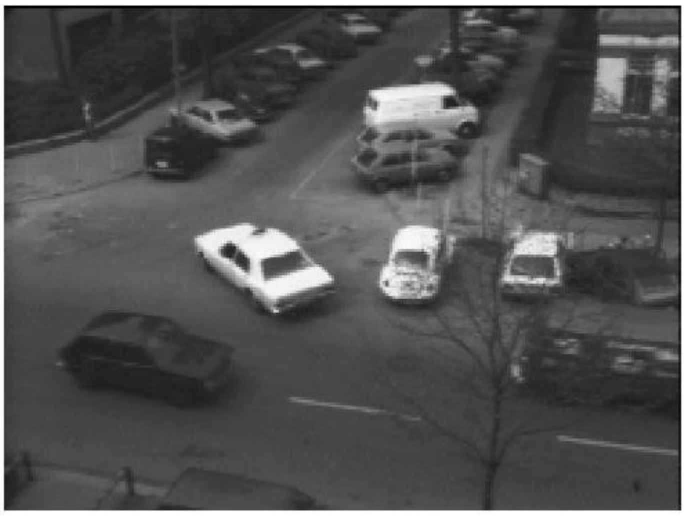
because it depends on the feature information of each block.

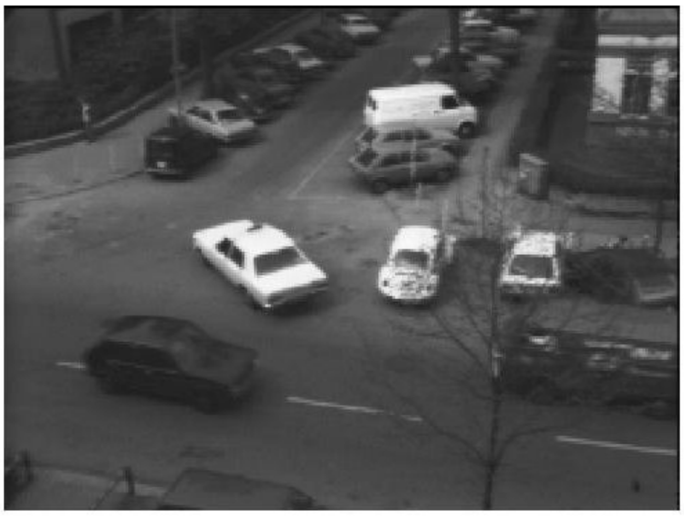
Figure 6.8b, showed extended bases representation of blocks in Figure 6.8a by using the proposed diffusion wavelets algorithm. Each block is represented with a number over the block. The first block from the Figure 6.8a-1 has 20×20 pixels size and shows the pedestrian on the upper left pavement. The pedestrian features intensity values are closer to the intensities of back ground from that block. Though this is not a clear feature, the diffusion wavelet algorithm represents well the pedestrian feature at fourth level, shows in Figure 6.8b-1.

The second block from the Figure 6.8a-2 has 20×20 pixels size showing a static car and we show the extended bases representation at level 7 Figure 6.8b-2. The third block from Figure 6.8a-3 has 20×20 pixels size showing a house window, and a wall with tree branch as foreground. Extended bases showed well defined features at level 8 as we can see in the Figure 6.8b-3. Pixel blocks fourth and seventh contain car features and their extended bases representation at fourth and seventh blocks is shown in Figure 6.8b

The fifth block has bin feature and it is well defined with extended bases representation at level 7 and the sixth block has black car features which have almost similar intensity values as the background but our extended bases has represented better these features. In Figure 6.8b-6, we can at least identify that there is a intensity values difference between background and foreground. We achieved a good representation at level 4. We understood from the above feature representation results, our extended bases representation is good when we process well defined features and if the processed pixels block is noisy. We understood that our extended bases representation is quite good for most of the pixels blocks from the Hamburg taxi sequence.

We took two frames from the Hamburg Image sequence $\mathbf{I}_1$, shows in Figure 6.9a and $\mathbf{I}_2$, shows in Figure 6.9b and they have the size of 191×256. The frame $\mathbf{I}_2$ is considered


(a) Taxi sequence image 1


(b) Taxi sequence image 2

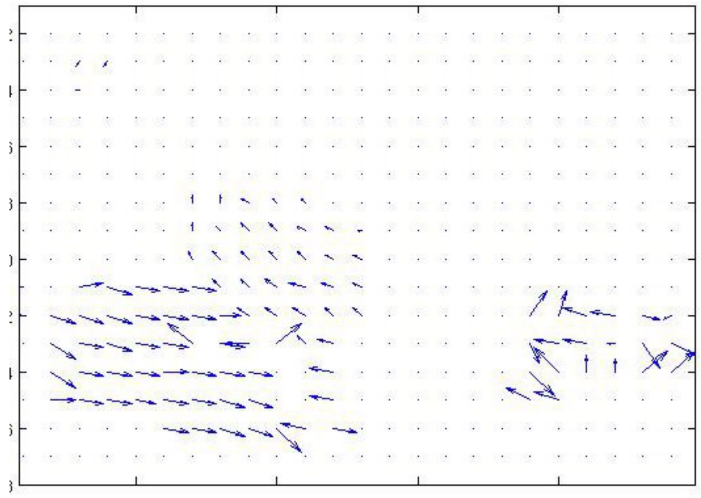

(c) Displacement fields estimated

Figure 6.9: Estimated optical flow in Hamburg taxi sequence

as the main frame that means diffusion windows are of 20×20 pixel block size are defined in this frame and frame $\mathbf{I}_1$ is considered as reference frame. In order to reduce the computational time we have taken ten pixels translation of the block along both x and y direction. The total number of pixel blocks is reduced to 18×24 that means in frame $\mathbf{I}_2$, we have 18 rows and each row contains 24 columns of blocks with size 20×20. The frame $\mathbf{I}_1$, has one pixel translation on both x and y direction. In our case we are not padding any zeros that means, we don't go out of the matrix/frame. So, we have 172×237 blocks in frame $\mathbf{I}_1$ with size 20×20 pixels. Then we calculated extended bases representation for each block in both frames.

Section 5.3 has described the methodology to calculate the Euclidean distance between the the diffusion window of size 20×20 in the frame $\mathbf{I}_2$ and the search window in the frame $\mathbf{I}_1$. We considered ± 6 offset of search window that means it has translation of 6 pixels on the four sides in the search window. That means we have a search window with the size of 32×32. In the search window, all possible 20×20 size pixel blocks are 169, i.e to identify the similar block in the next frame, we have to consider all the possible 20×20 size blocks. Here, we used minimum Euclidean distance to identify maximum likelihood block as described in Section 5.3. These displacement vectors on both x and y directions optical flow estimate.

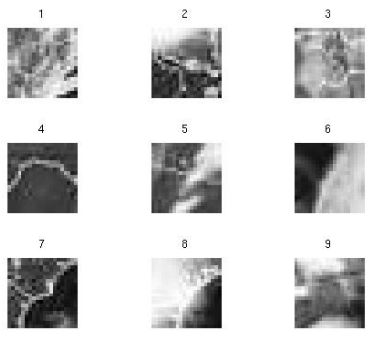
The estimated optical flow for the Hamburg taxi sequence is shown in Figure 6.9c. First shows moving object is a car on the left driving left to the right, Our estimated optical flow identified incorrect estimation of five displacement vectors with respect to this black car. This incorrect estimation of optical flow happened due to the confuse of car features and the background. The flow field is incorrect mostly at the edge of the car feature. The second moving object is a pedestrian and our algorithm has three flow fields with respect to the pedestrian and this motion flow is correct. The third moving object is a van, which is moving from the right to left. Many of the flow fields are incorrect due to confuse of the van object with a tree, which is overlapping in the foreground of this van object. Fourth moving object is a taxi and it is turning at the centre of the image. This has well defined features. So, our algorithm estimated its optical flow correctly.

Finally, we understood that our proposed dense optical flow estimation results are good for the Hamburg taxi sequence, and some incorrect estimations are due to the confusion in extended bases of pixel blocks.

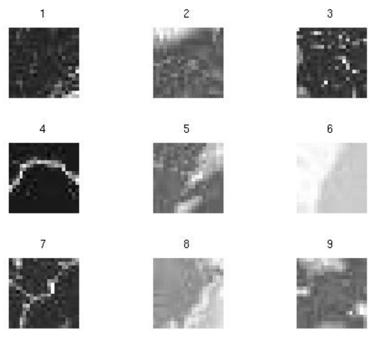
Optical flow for Andrea Hurricane image sequence

Andrea Hurricane is a satellite image sequence and which shows cloud movement and water vapours features. In this sequence all the features are very complex. Our algorithm is applied on this image sequence in order to estimate the displacement vectors. We skip one frame between each consecutive in order to estimate the optical flow.

In Figure 6.10a, is showed different image blocks from the Andrea Hurricane sequence



(a) 20 x 20 block parts



(b) Feature representation

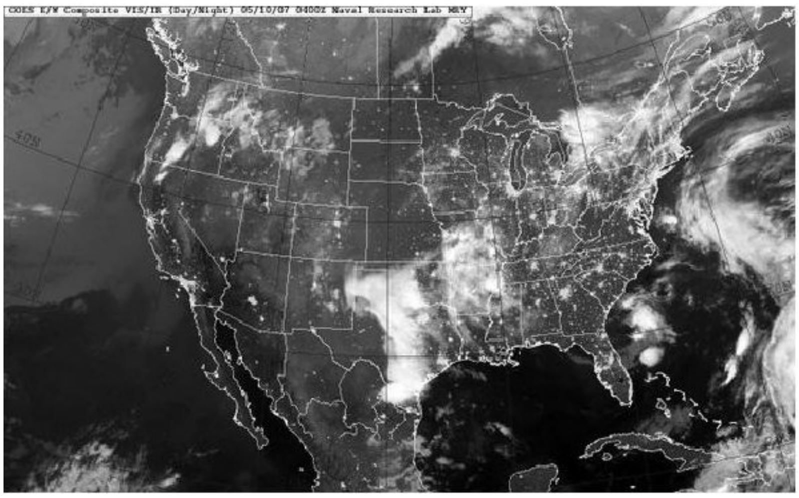
Figure 6.10: Feature representation of 20 x 20 block parts in Andrea hurricane image

and we describe feature representation of these blocks by using multi scale dimensionality reduction. We described the multi scale feature representation of features in Section 6.1, and we explained dimensionality reduction of extended bases function at each level of the representation.

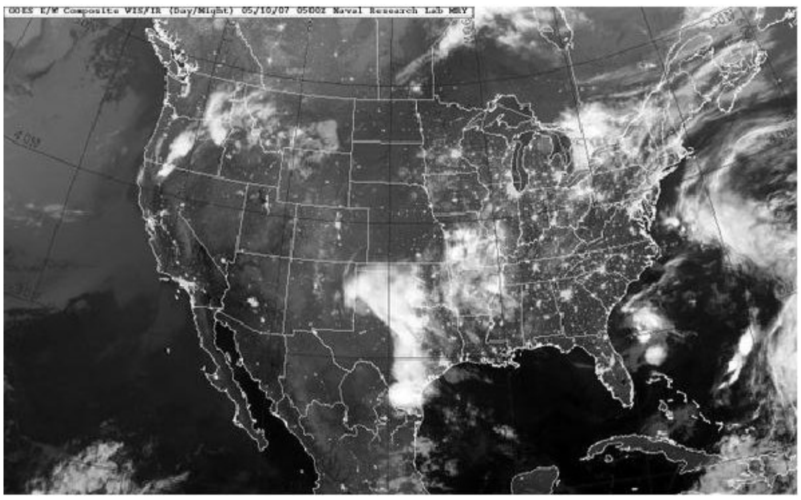
Figure 6.10b , shows the extended bases representation of blocks in Figure 6.10a by using the proposed diffusion wavelets algorithm. Each block is represented with a number over the block. The first block from Figure 6.10a-1 has 20×20 pixels size and the features of water vapour is closer in grey level to the intensities of the background. Though it is not well structured feature, our algorithm represents it quite well representation at the fourth level as shown in Figure 6.10b-1.

We consider two frames from the Andrea Image sequence $\mathbf{I}_1$, shows in Figure 6.11a and $\mathbf{I}_2$, shows in Figure 6.11b and they have the sizes of 338×550 pixels. Frame $\mathbf{I}_2$ is considered as main frame that means diffusion windows of 20×20 pixels are defined in this frame and frame $\mathbf{I}_1$ is considered as reference frame. In order to reduce the computational time we have taken ten pixels translation of the block on both x and y direction in frame $\mathbf{I}_2$. Because of this step all comparative blocks from frame $\mathbf{I}_2$ are are 31 rows and each row contains 53 columns of blocks with the size of 20×20 pixels. The frame $\mathbf{I}_1$, has one pixel translation on both x and y direction and it has 319 rows and each row contains 513 columns of blocks with the size of 20×20 pixels. Then we calculated extended bases representation of these blocks in both frames.

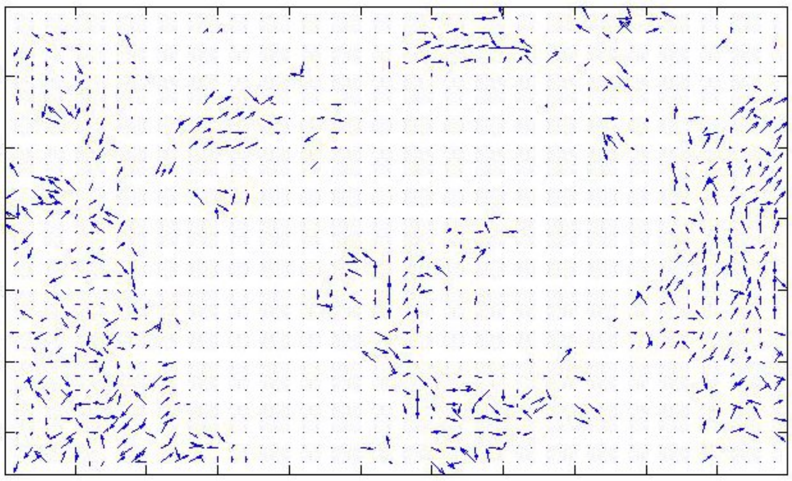
In Section 5.3 we described the methodology to calculate Euclidean distances between the diffusion window of size 20×20 pixels in the frame $\mathbf{I}_2$ and search window in the frame $\mathbf{I}_1$. We considered ± 6 offset for the search window. Here, we used minimum Euclidean distance between first extended bases vectors to identify the maximum likelihood block. These displacement vectors on both x and y directions form the optical flow. We used the this on the Andrea Hurricane image sequence and the estimated optical flow is shown in Figure 6.11c .


GOES E/W Composite VIS/IR (Day/Night) 05/10/07 0400Z Naval Research Lab MRY

(a) Andrea frame 1


GOES E/W Composite VIS/IR (Day/Night) 05/10/07 0500Z Naval Research Lab MRY

(b) Andrea frame 2



(c) Displacement fields estimated

Figure 6.11: Estimated optical flow in Andrea Hurricane image sequence

Finally, we understood that our proposed dense optical flow estimation results are good for the Andrea Hurricane satellite image sequence, and some incorrect estimations are due to the confusion in extended bases of pixel blocks (i.e similar features in neighbourhood blocks of search window).

Optical flow for Meteosat image sequence

Infra-red Meteosat image sequence is used for fluid optical flow estimation [37]. We used our dense optical flow proposed method to estimate the rotational motion in the center of the image, and a sort of divergence on the top right corner. The idea is to find the extended bases for every block in both frames and then we estimate the displacement vectors by using the Euclidean distance on diffusion wavelets in the two frames space method. In Figure 6.12, is showed three different image blocks from the Meteosat image sequence

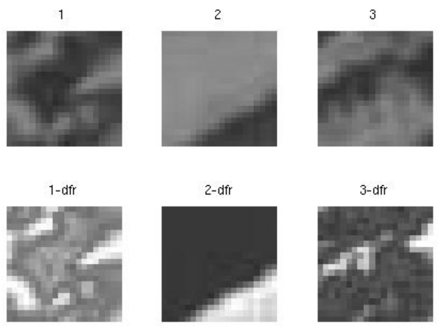
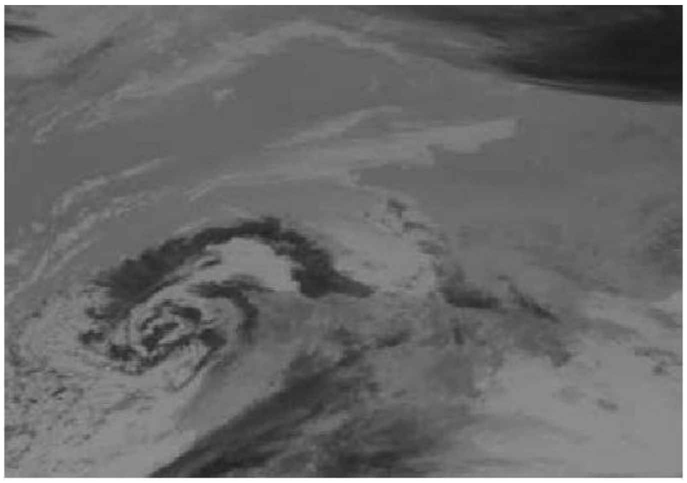
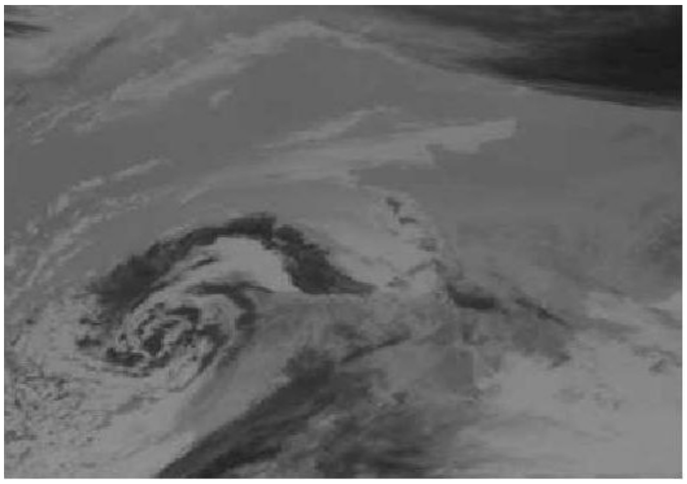


Figure 6.12: Feature representation of 20 x 20 block parts in Meteosat image

in first row and their corresponding feature representation is showed in the second row of the same figure. We describe feature representation of these three blocks by using multi scale dimensionality reduction with the help diffusion wavelets. We described the multi scale feature representation of features in Section 6.1, and we explained dimensionality reduction of extended bases function at each level of the representation. The Figure 6.12-1 has 20 $\times$ 20 pixels size and the features representation of this fluid features is shown as Figure 6.12-1-dfr. Though it is not well structured feature, our algorithm represents it quite well representation at the third level.

Initially, we consider two frames from the Meteosat Image sequences $\mathbf{I}_1$, as shown in Figure 6.13a and $\mathbf{I}_2$, as shown in Figure 6.13b . The size of each frame is of 367$\times$526 pixels. The frame $\mathbf{I}_2$ considered as main frame that means diffusion windows of the size of 20 $\times$ 20 pixels are defined and frame $\mathbf{I}_1$ is considered as reference frame. In order to reduce the computational time we have taken ten pixels translation of the block in both x and y directions. Because of this, we have 34 rows and each row contains 40 columns of blocks of size of 20 $\times$ 20 pixels. The frame $\mathbf{I}_1$, has one pixel translation in both x and y directions and we have 348 rows and each row contains 407 columns of blocks of size of 20 $\times$ 20 pixels. Then we calculated extended bases representation of each of these blocks


(a) Fluid image 1


(b) Fluid image 2

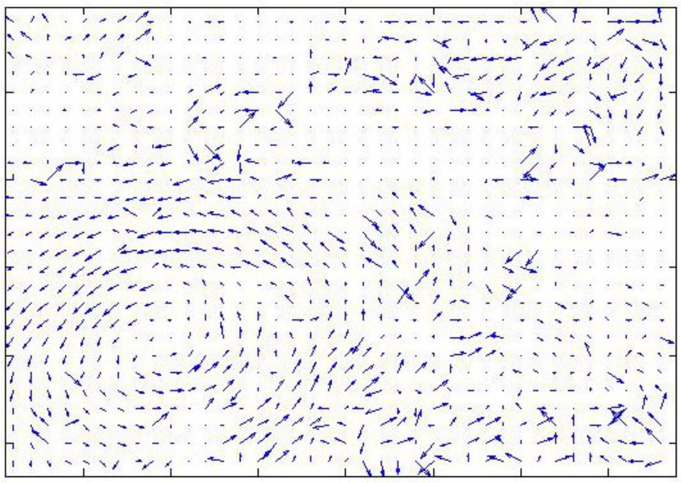

(c) Displacement fields estimated

Figure 6.13: Estimated fluid optical flow in Meteosat image sequence

in both frames by using the multi scale dimensionality reduction and we have chosen the first extended bases vector at last level because it has the better representation than all previous levels.

In Section 5.3 we described methodology to calculate euclidean distance between the the diffusion window of size 20×20 pixels in the frame $\mathbf{I}_2$ and the blocks from the search window in frame $\mathbf{I}_1$. We considered ± 6 offset of search window that means it has translation of 6 pixels on four sides in the search window . Here, we used the minimum Euclidean distance to identify the maximum likelihood matching block. Those displacement vectors resulting as best matching on both x and y directions from the optical flow. We used this concept on the Meteosat sequence and the estimated optical flow is shown at Figure 6.13c. In the Meteosat sequence we can observe a rotational motion in the center of the image, and a sort of divergence on the top right corner.

From our proposed algorithm we generated displacement vectors and Most of the vector fields are showing the correct direction. The optical flow at centre of the image has rotational structure. We understood that the Euclidean distance method estimates some wrong displacements due to the confusion in search window with a similar block. We estimate well the divergence on the top right with only few vectors of different direction than expected. The main reason for this wrong direction is that these blocks doesn't have a clear structure. So, without a proper structure the optical flow is wrongly estimated.

Dimetrodon and Venus Image Sequence from Middlebury data set

The standard data set for comparing optical flow estimation algorithms has been the Middlebury data set [33]. The data set has optical flow ground truth which can be used to evaluate the errors. We use the Dimetrodon and Venus image sequences from Middlebury. We show the arrow plot of optical flow and then calculate the color map displaying the velocity. Quantitative results are defined by two error measures which are commonly used for evaluating optical flow estimation algorithms [38, 33]. These measures are the average angular error defined by

$$AE = arccos\left(\frac{u_{gt}u_e + v_{gt}v_e + 1}{\sqrt{(u_{gt}^2 + v_{gt}^2 + 1)(u_e^2 + v_e^2 + 1)}}\right) \tag{6.1}$$

and the average flow error is defined by

$$FE = \sqrt{[(u_{gt} - u_e)^2 + (v_{gt} - v_e)^2]} \tag{6.2}$$

where (u_e, v_e) is the estimated flow and (u_{gt}, v_{gt}) is the ground truth flow. Errors for optical flows characterized by large vectors are smaller using AE measure, while the FE measure provides a less biased measure, especially for areas with vectors of near zero length.

The newer set of images use hidden texture and synthetic images that correspond to the Venus and Dimetrodon data sets respectively. A new class of images in a high-speed camera category include small regions in the image moving at high speeds, while much of the remainder of the scene remains stationary. This contrast with the other images in the sequence that usually contains motion of small magnitude throughout the entire image. It is difficult image sequence to estimate optical flow due to flow difference across object boundaries.

In this research, we use the Dimetrodon and Venus image sequences. The Dimetrodon sequence is a hidden fluorescent texture sequence, which is a real scene that has been scatter in drops with fluorescent paint and photographed. Ground truth motion is computed by tracking the fluorescent paint which is used as a marker. This approach allows for the computation of ground truth from the low texture data.

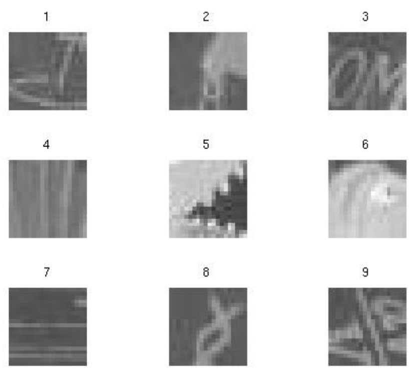
The Venus sequence is a synthetic scene generated using computer graphics. This method of generating images yields highly accurate ground truth and allows for the investigation of the accuracy of optical flow estimation.

Dimetrodon Image Sequence

Figure 6.14a shows different image blocks from the Dimetrodon sequence and we describe the feature representation of those blocks by using multi scale dimensionality reduction. We showed multi scale feature representation for various image features in Section 6.1 , and we explained dimensionality reduction of extended bases function at each level in this representation. From the experiments we understood that the last level extended bases are well defined (due to removing the high frequency information at each level) when compared the previous levels. So, for optical flow estimation we used first vector of the extended bases at the last level to represent each block in the image sequence. The number of levels vary from one block to another and depends on the thresholds to as explained in Section 5.3 even though we used the same parameters to calculate the extended bases because it depends on the feature information of each block.

Figure 6.14b, showed extended bases representation of blocks in Figure 6.14a by using the proposed diffusion wavelets algorithm. Each block is represented with a number over the block. The first block from the Figure 6.14a-1 has 20×20 pixels size and its feature representation at third level, shows in Figure 6.14b-1. The second block from the Figure 6.14a-2 has 20×20 pixels size and its representation at second level is shown in Figure 6.14b-2. The third block from the Figure 6.14a-3 represents at third level is shown in figure 6.14b-3. The blocks fourth,fifth,sixth,seventh,eight and ninth are sown in Figure 6.14a-4, 5, 6 ,7 , 8 ,9 and their corresponding feature representation at levels 3, 4, 2, 3 ,2 , 4 are shown in Figure 6.14b-4, 5, 6, 7, 8, 9 .

We took two frames from the Dimetrodon image sequence $\mathbf{I}_1$ is shows at Figure 6.15a and $\mathbf{I}_2$ with the size of 584×388 pixels. The frame $\mathbf{I}_2$ is considered as the main frame with diffusion windows of 20×20 pixels. Frame $\mathbf{I}_1$ considered as reference frame and search


(a) 20 x 20 block parts

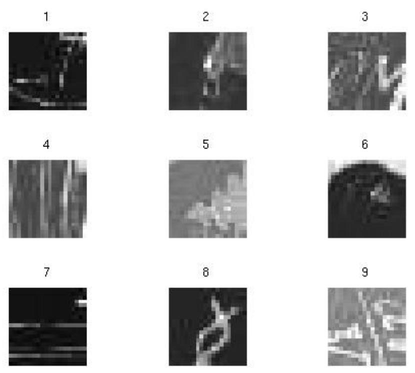
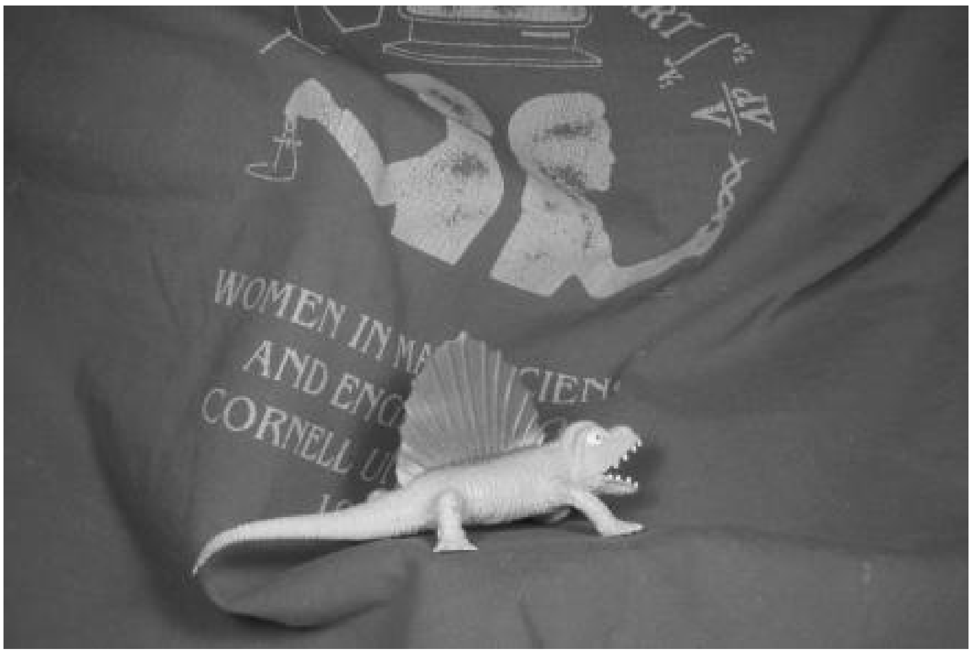
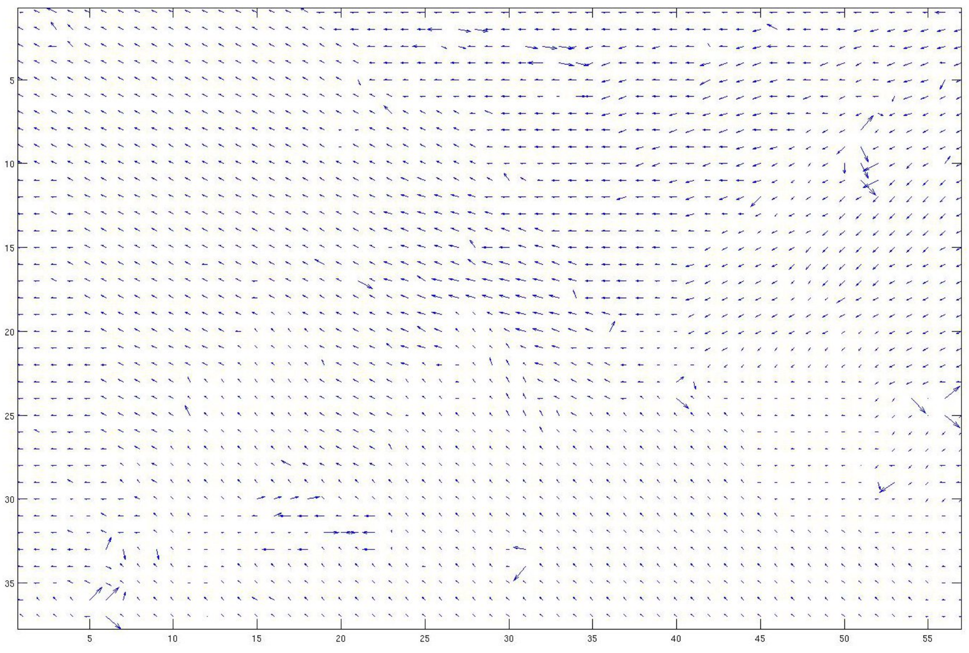

(b) Feature representation

Figure 6.14: Feature representation of 20 x 20 block parts in Dimetrodon image

window define from this frame with respect to pixels of block in $\mathbf{I}_2$. In our earlier optical flow experiments we took translations of ten pixel in frame $\mathbf{I}_2$ and translational of one pixel in frame $\mathbf{I}_1$ on both x and y directions in order to reduce the computational time. But, in this experiments we took one pixel translations on both frames in x and y directions, in order to estimate optical flow with equal size of ground truth. Then we calculated extended bases representation for each of these blocks in both frames. We calculate displacement vector with respect to the each block of pixels i.e diffusion window in frame $\mathbf{I}_2$. The displacement vectors on both x and y directions form the optical flow shown in Figure 6.15b and from it we understood that some of the displacement vectors are estimated incorrectly. Our algorithm estimated wrong displacement when the image has a folded structure and also estimated wrong displacements at the dinosaur's tail due to complex in flow structure.

Now we have both ground truth and estimated optical flow of the Dimetrodon sequence and using these optical flows we calculate angular error AE which is given in Equation (6.1) and flow error FE which is given in Equation (6.2). Results for our method and those tested in the original evaluation paper by Baker et al [33] are shown in Table 6.8 for the Dimetrodon sequence. We have also included the result from the paper [40] in this table. The calculated average angular error is 12.04 degrees and flow error is 0.51. From the methods which we included in table, our method has 5th rank on both errors. The estimated optical flow shown at Figure 6.17b and ground truth optical flow shown at Figure 6.17a are coded using the colour map shown in Figure 6.16 with the help of methodology in [33]. This colour map represents the optical flow in colour and the colour of the representation is changed with respect to the incorrect estimated optical flow. The top left side has a representation with various colour and it means that the estimated flow direction is incorrect and you can observe the same in the dense optical flow which is shown in Figure 6.15b. This colour representation is incorrect at folded structure of the features and it tries to represent dinosaur's feature but our approach failed to estimate optical flow at the tail of dinosaur's. From the results, our algorithm is good to estimate optical flow for this sort of sequence.


(a) dimetrodon image 1



(b) optical flow

Figure 6.15: Dimetrodon dense optical flow

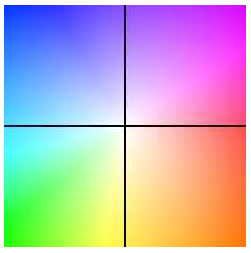


Figure 6.16: colormap

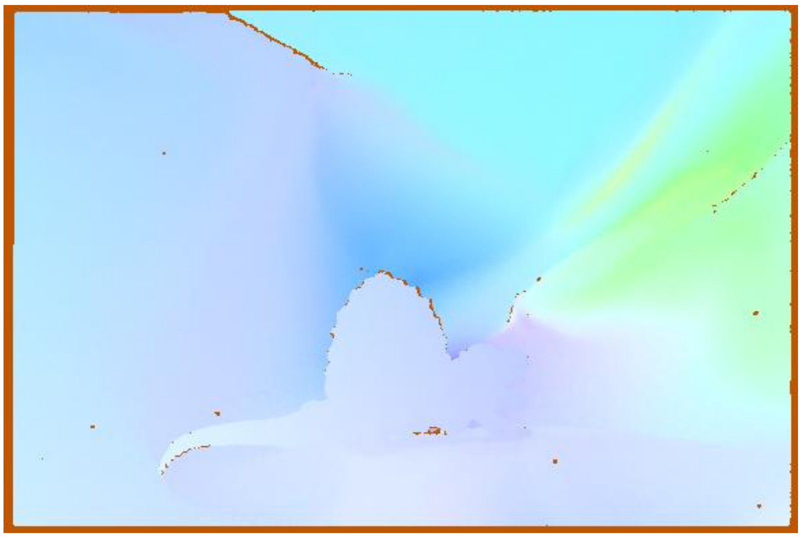
Venus Image Sequence

Figure 6.18a shows six different block of pixels with the size of 20×20 and most of the blocks are taken from news paper in the Venus image which is shown at Figure 6.19a. We calculate these blocks of pixels feature representation by using the multiscale representation of diffusion wavelets and those representations are shown in the Figure 6.18b.

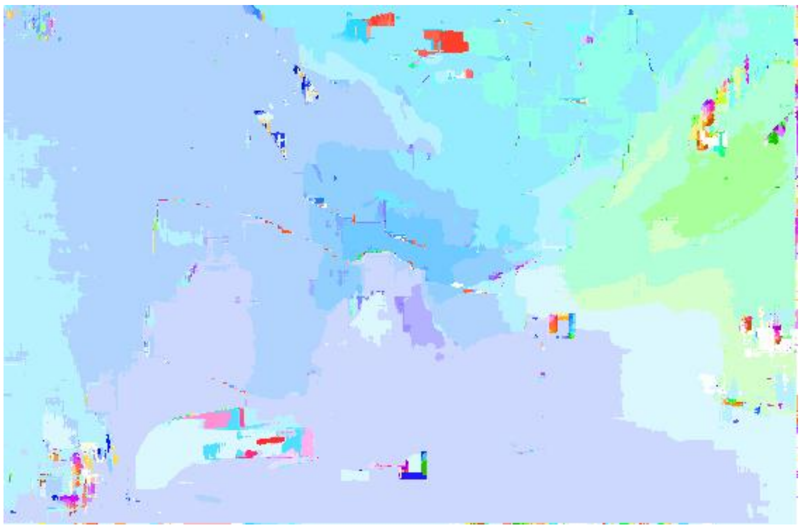
We consider two frames from the Venus image sequence $\mathbf{I}_1$ which is shown in Figure 6.19a and $\mathbf{I}_2$ and they have with the size of $420{\times}380$ pixels. The frame $\mathbf{I}_2$ considered as main frame with diffusion windows of size 20×20 block pixels. Frame $\mathbf{I}_1$ is considered as reference frame and this frame has search windows corresponding to each block in frame $\mathbf{I}_2$. We have taken one pixel translation of the block on both x and y direction in frame $\mathbf{I}_2$ for defining possible blocks of pixels to estimate optical flow. Then we calculate extended bases representation of these blocks in both frames.

In Section 5.3 we explained about dense optical flow estimation algorithm and here we used same procedure to calculate displacement vectors in both x and y directions corresponding to each block of pixels in frame $\mathbf{I}_2$ and we shown the estimated optical flow in Figure 6.19b, and top of the Venus image has motion to right direction and bottom of the image has motion to left direction. Our algorithm suffers to estimate optical flow at top left, right bottom edge and center of the image. Here, we consider second frame of the Venus sequence as a main frame and first frame as a reference frame. So, we estimate optical flow corresponding to the pixels of each block in the second frame but in this case the image features are not similar at the bottom right and top left edges i.e the pixels at top left edge in main frame are not present in the reference frame and same sort of problem occurring at right bottom edge and in between news paper feature. At the center of the image, our algorithm confuses with other similar blocks of pixels. So, some of estimated optical flow is incorrect.

Now we have both ground truth and estimated optical flow of the Venus sequence and using these optical flows we calculate angular error AE which is given in Equation (6.1)

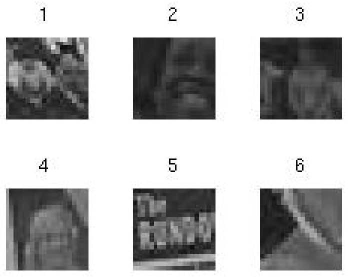


(a) Dimetrodon ground truth flow

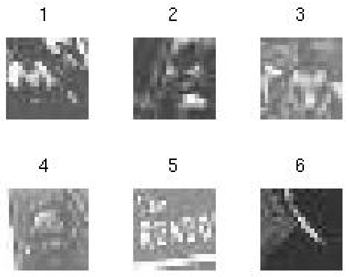


(b) Dimetrodon estimated flow

Figure 6.17: Dimetrodon estimated and ground truth flow

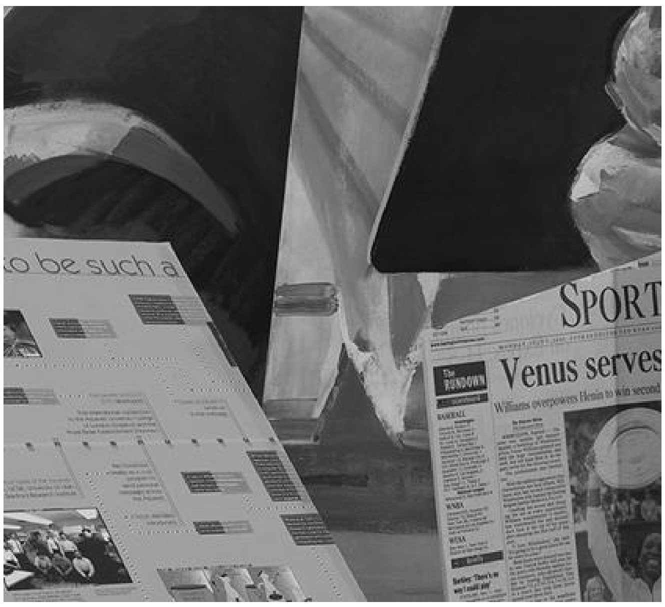


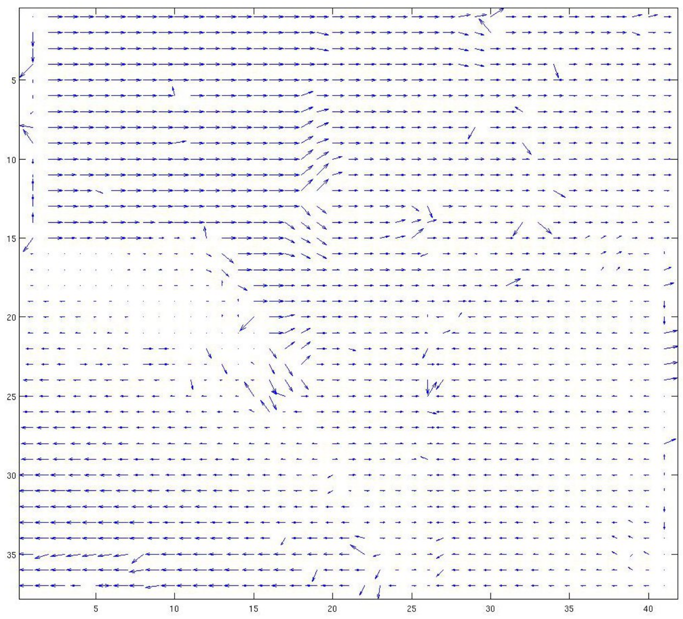
(a) 20 x 20 block of pixels



(b) Feature representation

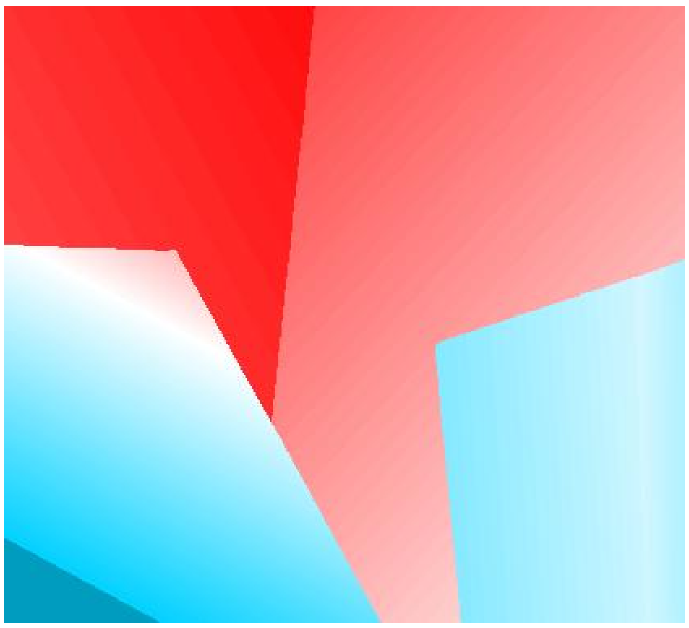
Figure 6.18: Feature representation of 20 x 20 block parts in Venus image


(a) Venus image 1



(b) optical flow

Figure 6.19: Venus Optical flow


(a) Venus ground truth flow

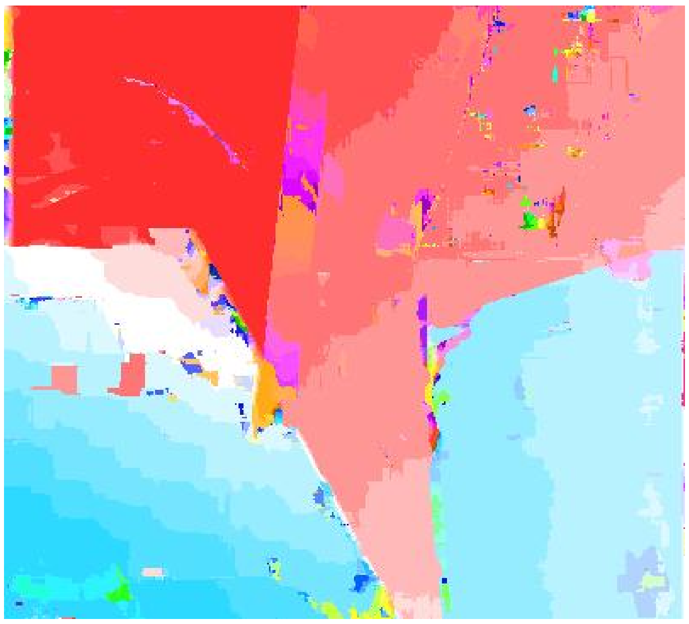

(b) Venus estimated flow

Figure 6.20: Venus estimated and ground truth flow

and flow error FE which is given in Equation (6.2). Results for our method and those tested in the original evaluation paper by Baker et al [33] are shown in Table 6.8 for the Venus sequence. We have also included the result from the paper [40] in this table. The calculated average angular error is 11.36 degrees and flow error is 0.83. From the methods which we included in table, our method has 5th rank for angular error and 4th rank for flow error. The estimated optical flow shown at Figure 6.20b and ground truth optical flow shown at Figure 6.20a are coded using the colour map shown in Figure 6.16 with the help of methodology in [33]. This colour map represents the optical flow in colour and the colour of the representation is changed with respect to the incorrect estimated optical flow. We already discussed about incorrect optical flow estimation in Venus sequence. Our colour representation shown the colour variation where the optical flow estimated wrongly. From the results, our algorithm is good to estimate optical flow for this sort of sequence.

Our quantitative results are similar to the results from the best algorithm which are

Table 6.8: Angular and flow error from the Dimetrodon and Venus sequences for the method proposed in this thesis, szymon paper [40] and method from Baker et.al [33]

	Average Angular Error		Average Flow Error	
	Dimetrodon	Venus	Dimetrodon	Venus
Black and Anandan	9.26	7.64	0.35	0.55
Bruhn et al.	10.99	8.73	0.43	0.51
Pyramid LK	10.27	14.61	0.37	1.03
Diff. distance	11.45	10.40	0.50	0.87
Proposed method.	12.04	11.36	0.51	0.83
Media Player	15.82	15.48	0.94	0.85
Zitnick et al.	30.10	11.42	0.55	1.08

mentioned in Table 6.8. Average angular error for Dimetrodon and Venus sequence are 12.04 degrees and 11.36 degrees respectively and the average flow error which is Euclidean distance, for the both sequences 0.51 and 0.83 units respectively, for our method and the best reported value in [33] which is from Black and Anandan's algorithm [39].

6.2.2 Sparse Optical flow Estimation for Hamburg Taxi Image Sequence

In sparse motion we don't consider the entire information in the frames when compared to dense optical flow. Initially we applied scale invariant feature transform on frame $\mathbf{I}_2$, as shown in Figure 6.9b. We get key point locations in the Hamburg taxi sequence. We understood that some key points are close to each other. So, we consider ± 5 pixels offset for each location neglecting the key points which are close to each other. Now we have taken each key point location as a center pixel and consider the block size of 21×21 pixels. We calculate extended bases functions for ever key point location block in frame $\mathbf{I}_2$ and we calculate extended bases functions of respective ± 6 pixels offset for the search window blocks in frame $\mathbf{I}_1$, shown in Figure 6.9a which means that we have 139 blocks extended functions for each key point locations and then we find minimum distance of those blocks then choose the displacement vector.We repeat this steps for remaining all key point locations. This algorithm explained at Section 5.4.

In this experiment we consider Hamburg taxi sequence is with the size of 191×256

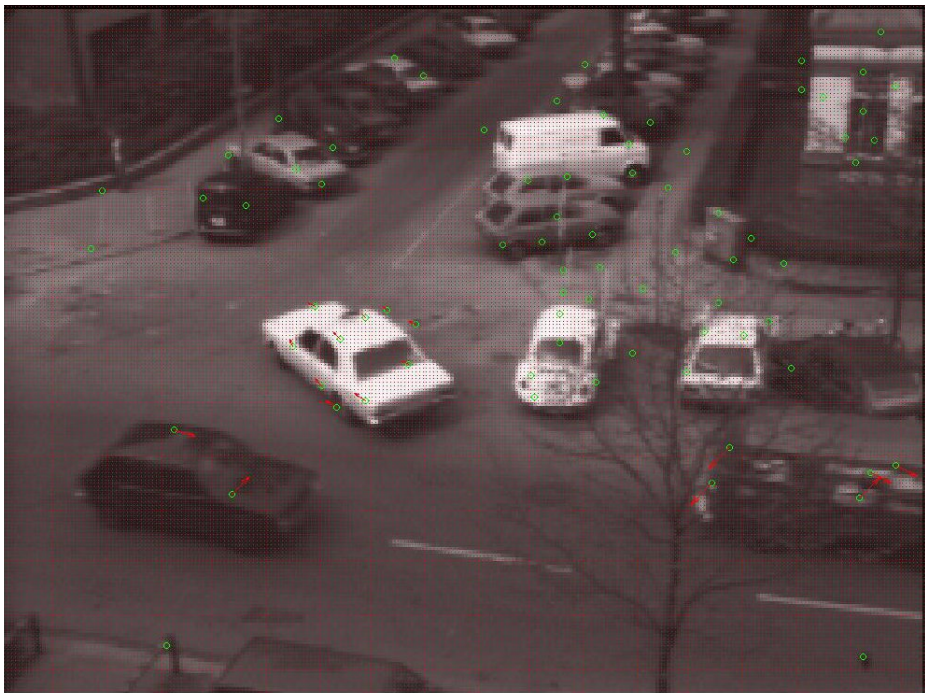


Figure 6.21: Sparse optical flow estimation in Hamburg taxi sequence

pixels and calculate key points for second frame by using SIFT algorithm. We have 258 key points for second frame but many of in it are close to each other. So, by using the offset we filtered the key points to 77. We used these key point blocks of pixels to estimate optical flow. The estimated sparse optical flow for the Hamburg taxi sequence is shown in Figure 6.21 and we scaled arrows 6 times in order to make visible. Generally, this sequence has four moving objects and they are car on the left driving left to the right, taxi near the center turning the corner, a pedestrian on the upper left moving on the pavement and van on the right moving to the left. Here, we don't have key point on pedestrian So, our algorithm neglect to calculate optical flow corresponding to this feature. A black car on the left driving left to the right has two key points and our algorithm estimated corresponding flow vectors, the first key points shows the correct direction and the second key point shows in correct direction confuses with search window blocks of pixels due to similar intensity blocks. A van moving from the right to left has 5 key points and all of these flow fields are incorrect due to confuse of the van object with a tree, which is overlapping in the foreground of this van object. A taxi which is turning at the centre of the image has 10 key points all of these flow fields are estimated correctly and it is well defined moving feature in this image. h is computational time is very less.

6.2.3 Image Registration

Registration is for using the feature representation of image in order to estimate the displacement between two images. For example, when we capture cornea layers by using microscope on the eye, there will be a loss of features due to small movements of the eye. So, the doctor cannot diagnose the eye disease correctly. To overcome this kind of problem we have the registration method. In our proposed algorithm we have estimated both sparse and dense optical flow applications by using diffusion wavelets.

Dense Optical flow on Cornea Image Sequence

We took two frames from the cornea layer image sequence $\mathbf{I}_1$, which is shown at Figure 6.22a and $\mathbf{I}_2$, which is shown at Figure 6.22b with the size of 477×477 pixels. The frame $\mathbf{I}_2$ considered as main frame and diffusion windows size of 20×20 block pixels are defined in this frame and frame $\mathbf{I}_1$ as reference frame and it has search windows. Then we calculate extended bases for the blocks in both frames.

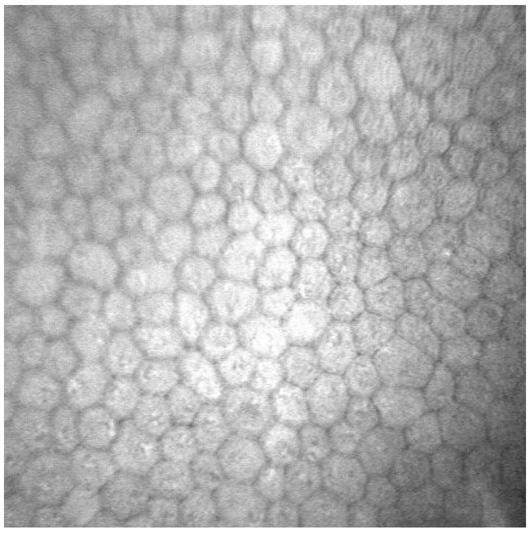
In Section 5.3 we have described the methodology for calculating euclidean distance between the the extended bases in the frame $\mathbf{I}_2$ and those in the frame $\mathbf{I}_1$. We considered ± 6 offset of search window that means it has translation of 6 pixels on four sides in the search window. Here, we used minimum euclidean distance to identify maximum likelihood block. The estimated optical flow is shown in Figure 6.22c. Our algorithm confuses at left edge and the right bottom edge due to unclear features.

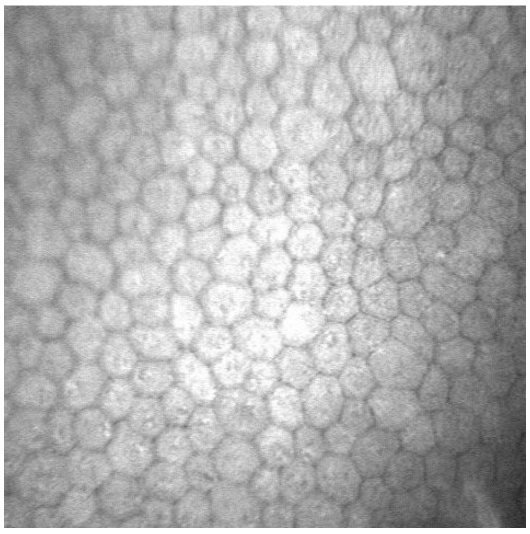
Sparse Optical flow on Cornea Image Sequence

In Sparse motion we do not consider the entire information in the frames like in the dense optical flow. Initially we have applied scale invariant feature transform on frame $\mathbf{I}_2$. We estimate 1329 key points in frame $\mathbf{I}_2$. We assume ± 5 offset from each other using SIFT location to neglect the key points which are close. Now we have 383 key points and taken each key point location as a center pixel and took the block size is 21×21. We calculate extended bases scaling function for ever key point blocks in frame $\mathbf{I}_2$ and we calculate the extended bases function for a ± 6 offset of search window blocks in frame $\mathbf{I}_1$, which means that we have the extended functions for each key point for 139 blocks and then we find the minimum distance of these blocks then decide the displacement vector. We repeat these steps for remaining all key points and the estimated optical flow shown in Figure 6.23. Here, we scaled 8 times each arrow to make it visible.

6.3 Human face recognition and fingerprint authentication using eigendiffusion faces

In the following section we present the results when applying eigendiffusion faces for face recognition and fingerprint authentication. In the following experiments we consider different data sets such as: ORL (Olivetti) face database [35], Yale face database and the set B of database DB1 from the FVC2000 [36].


(a) Cornea layer image 1


(b) Cornea layer image 2

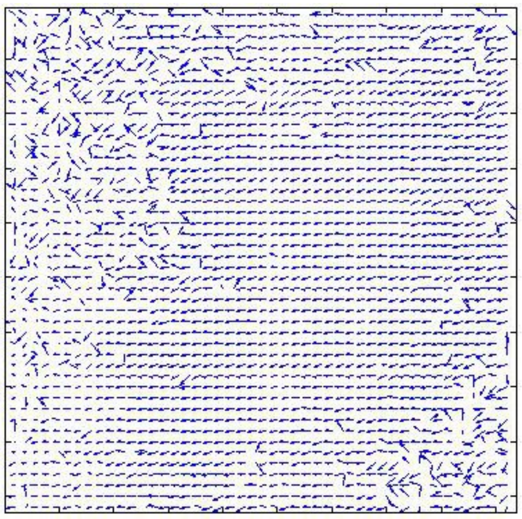

(c) Displacement fields estimated

Figure 6.22: Dense Optical flow estimation in Cornea layers image sequence

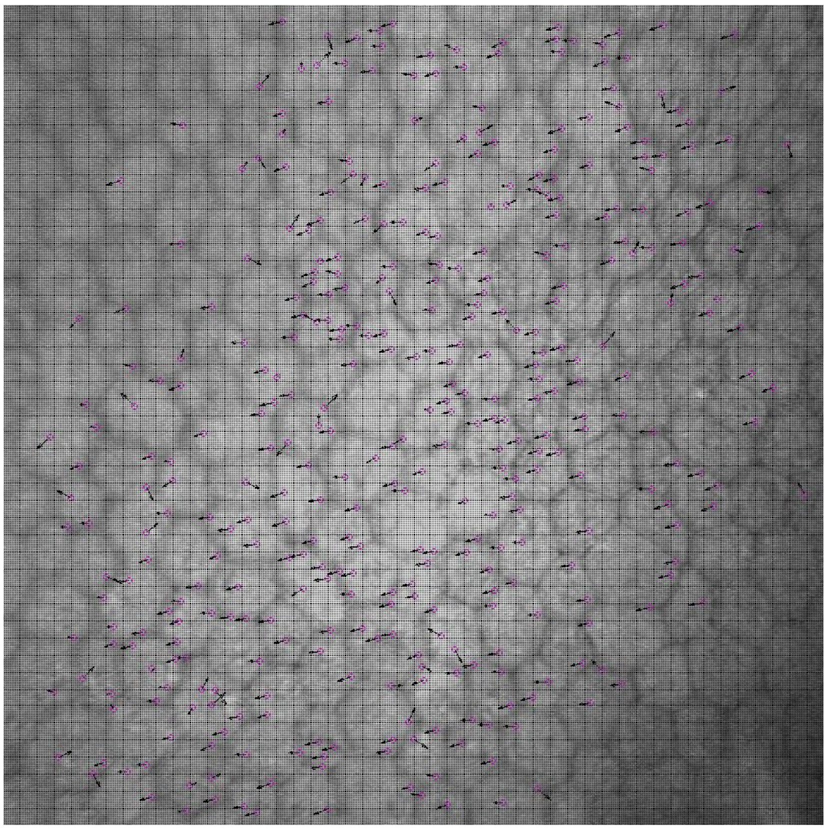


Figure 6.23: Sparse Optical flow estimation in Cornea layers image sequence

6.3.1 Face Recognition

In the following subsection we shown the results of face recognition on ORL and Yale face databases.

On ORL face database

The ORL database containing 40 subjects where each subject has 10 images of different orientations. For some subjects, the images were taken at different times, varying the lighting, facial expressions (i.e open or closed eyes, smiling or not smiling) and facial details (i.e glasses or no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The images are resized to 56×46 from their original size of 92×112 in order to reduce the complexity in computation. Figure 6.24 shows the ORL database. To calculate eigendiffusion faces (i.e extended bases of covariance of faces), we used above described parameters at Section 6.1 but here the required extended bases functions in diffusion wavelets algorithm is taken as $\kappa = 350$ i.e it will check at every level that whether we achieved required number of extended bases or not and to understand the algorithm of face recognition see Chapter 3 .

Section 4.4 was explained this algorithm step by step . Here, we have taken 5 faces from each subject as a training set. From the database, we have total 400 faces and we

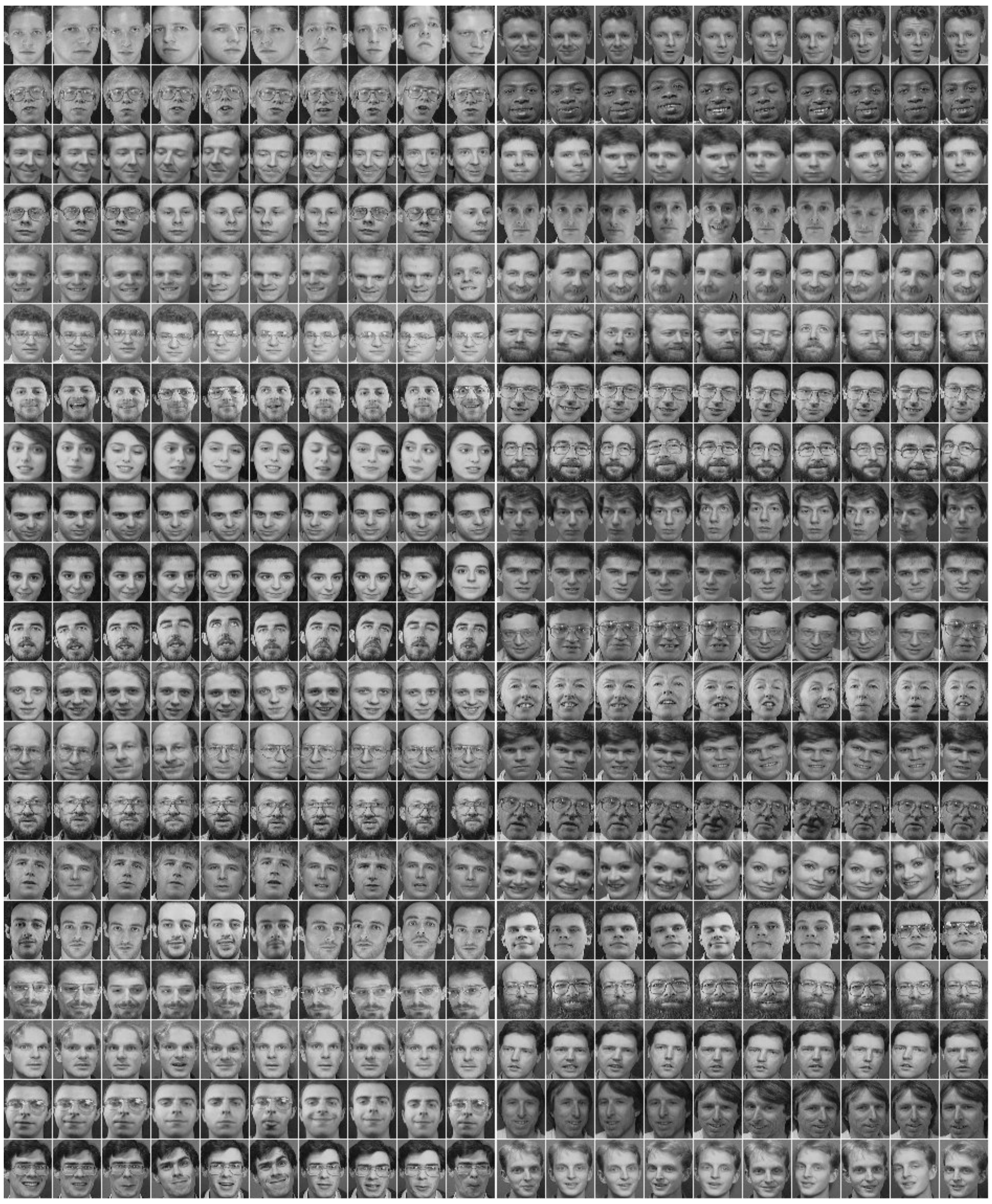


Figure 6.24: ORL database

are taking half of them(i.e 200) as a training set. We represent a matrix with information of training set that means, produce a column vector from each face and assign that column to the training set matrix, which was given in Equation (4.18) and it has the size of 2576×200 . We calculate mean face, which was given in Equation (4.20) and that we shows in Figure 6.25 . After that we normalize each training face by using the mean face which was given in Equation (4.3) .

Now, we calculate covariance matrix and which gives the variance among the faces, which was given by Equation (4.22) and it has size of 2576×2576 . In eigenface method, they used PCA method for dimensionality reduction but here, we used multi scale dimensionality reduction using diffusion wavelets method for representing low dimensional

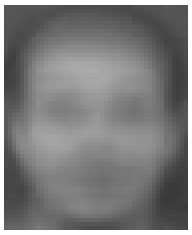


Figure 6.25: ORL Mean face

eigendiffusion face space and then we calculate the eigendiffusion face by using the diffusion wavelet algorithm, which is described in detail at Chapter 3. Now we have 198 eigendiffusion faces at level 1. We given the required extended bases is $\kappa = 350$ to our algorithm but it checks once finishing the level i.e after first level our algorithm checks the 198 extended bases are below 350 or not and the condition is satisfied So, our algorithm stopped at this level. Figure 6.26, shows initial 140 bases of 198 and we understood that the first eigendiffusion faces contained most of the faces information.

We calculated weight vectors of each face by using the eigendiffusion faces. These weights

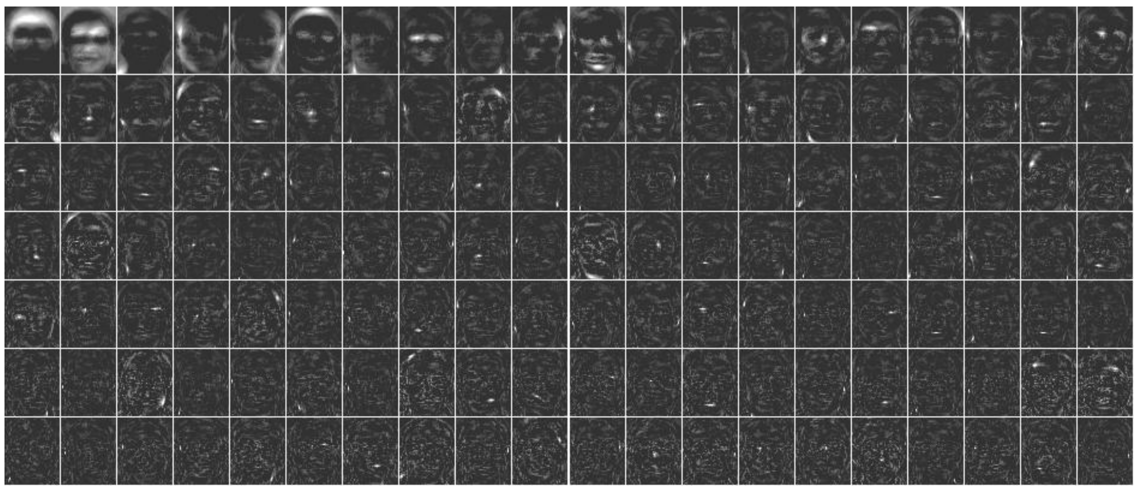


Figure 6.26: 140 out of a total 198 of eigendiffusion faces of ORL training faces

represent the each face in the eigendiffusion face space. We considered whole 400 images as the test data set and we normalize each face from the mean face and then calculate the weight vectors with the help of eigendiffusion faces. Each training face is associated to a class characterising a person. That means the whole 200 faces training set has 40 classes. We calculate the Euclidean distance from each test face to all the faces in the training set and then identify which training face has the minimum distance and assign that training face class to the class of the test face. This experiment repeats for the remaining test faces. We recognised 382 faces out of 400 i.e 95.5% face recognition. Our algorithm failed for 18 faces and those details are shown in Table 6.9 .

We calculated face recognition rate by varying the eigendiffusion faces and those results are shown in Table 6.10 . Initially, we have taken the 18 eigendiffusion faces to

Table 6.9: Faces which are not correctly recognised when the eigendiffusion faces are 198,train set as 50 % of ORL database and test set as 100 % ORL database

Subject-face no.	Train class	Predicted class
9 - 10	9	40
14 - 6	14	33
14 - 7	14	33
14 - 8	14	33
16 - 8	16	34
19 - 10	19	15
21 - 8	21	3
24 - 7	24	3
27 - 9	27	4
32 - 6	32	8
32 - 10	32	33
33 - 6	33	32
33 - 7	33	8
33 - 10	33	32
37 - 9	37	29
39 - 8	39	36
40 - 6	40	9
40 - 10	40	9

calculate weights of training and testing faces. Then we calculated Euclidean distance of these weights to predict the class of each test face and at this case we recognised 370 faces out of 400 i.e 92.50 % face recognition. The same experiment we repeated for 38, 58, 78, 98, 118, 138, 158, 178, 198 of eigendiffusion face and our algorithm recognised 377, 378, 380, 380, 381, 381, 382, 382 faces out of 400 faces respectively. We understood that our face recognition rate is poor when we consider less eigendiffusion faces and the maximum possible recognition is happened at highest possible eigendiffusion faces i.e 198 eigendiffusion faces and we also understood that the information at the last eigendiffusion face is too low when it compared to the first eigendiffusion face. The Plot 6.27, shows how the recognition rate increases with respect to taken eigendiffusion faces increases.

We reconstructed the whole database i.e 400 test faces by using the 198 eigendiffusion faces, which are shown in Figure 6.28. We calculate test face weights by using the 198 eigendiffusion faces and it is given by Equation (4.29). Then we calculate reconstruct of testing faces by using the following Equation (4.30), here we added mean face to reconstruct image because, initially we normalized each train face with mean face i.e each train face from mean face and those normalized faces used to calculate eigendiffusion faces.

Above all results obtained with training set 200 face (i.e 5 faces from each subject), testing set as 400 faces and their 198 eigendiffusion faces. Now, we observe how the face recognition rate changes with respect to various training faces but still testing faces are constant i.e 400 faces. Now, 80 faces (i.e 2 faces from each subject) are taken as training faces and calculated mean face, which is used for normalizing each training face and those

Table 6.10: Face recognition rate with respect to taken eigendiffusion faces from 198 where train set as 50 % of ORL database and test set as 100 % ORL database

eigendiffusion faces	Recognised/test faces	recognition (%)
18	370/400	92.50
38	377/400	94.25
58	378/400	94.50
78	380/400	95.00
98	380/400	95.00
118	381/400	95.25
138	381/400	95.25
158	382/400	95.50
178	382/400	95.50
198	382/400	95.50

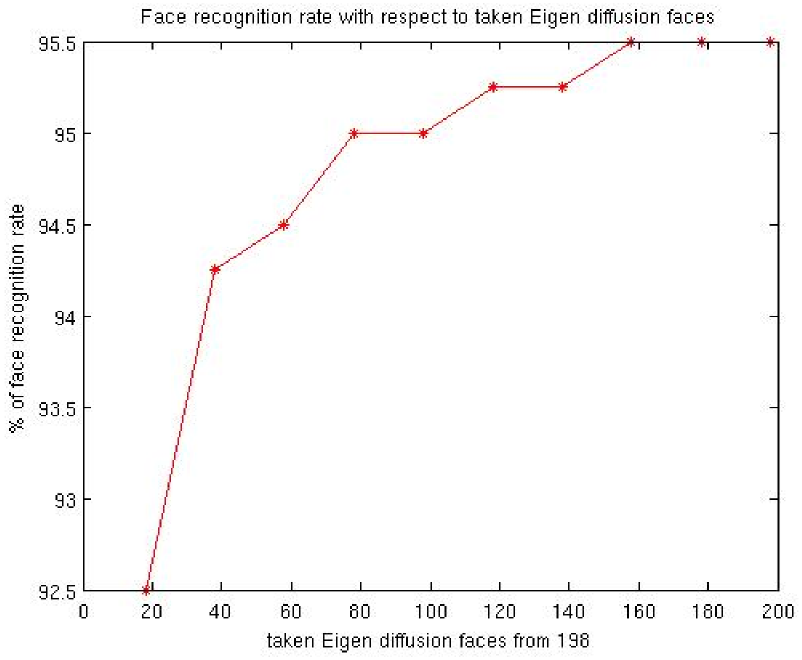


Figure 6.27: Face recognition rate with respect to taken eigendiffusion faces from 198 in ORL

normalized faces used to calculate covariance matrix. We applied diffusion wavelets algorithm on the covariance matrix, in order to calculate eigendiffusion faces. Here, we get 79 eigendiffusion faces. We added 50 to each pixel to make these eigendiffusion faces visible. We calculate weights of each training face by using the 79 eigendiffusion faces and then we calculate weights of 400 faces testing set. Now, by using both training and testing set weights we calculate the predict class for each face in testing set that is called recognition of face. For 80 training faces and 400 testing faces, our algorithm recognised 344 faces out of 400 i.e 86 % recognition rate. Now, the same approach is repeated for training faces of 120, 160, 200, 240, 280, 320 and their recognition rate is 90 %, 93.75 %, 95.00 %, 98.25 %, 98.75%, 99.25 % respectively, results are shown in Table 6.11 . From this experiment we understood that the rate of face recognition is increasing when increases the training set.

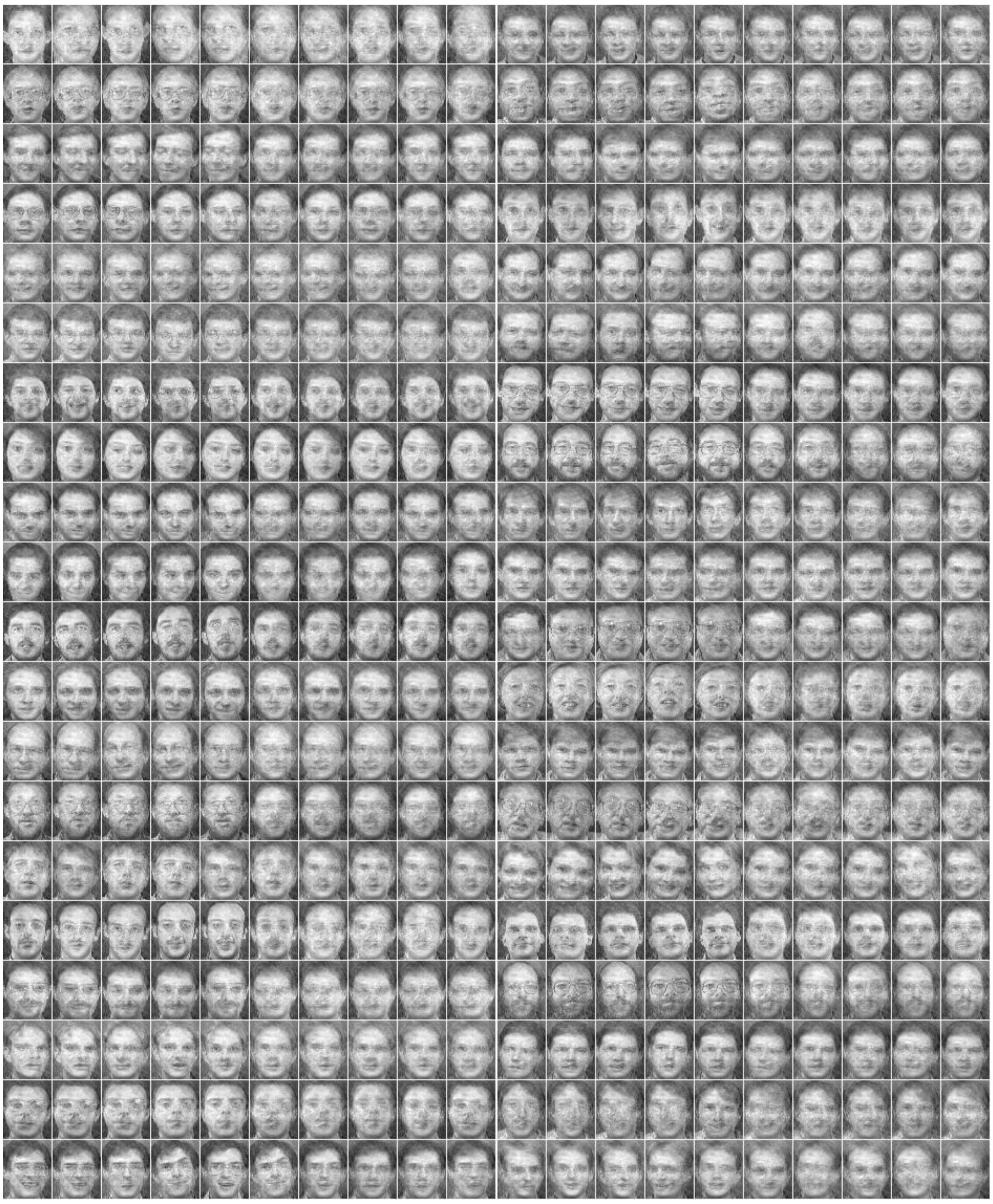


Figure 6.28: Reconstructed ORL database

The Plot 6.29, shows how the recognition rate increases with respect to training faces set increases.

We also experiment how the face recognition rate varies when training face set as n number of faces and testing face set as $400 - n$. Suppose, if we take training set as 80 faces then testing set taken as 320 faces. Now the above same procedure is applied to calculate face recognition. In this case, we have 79 eigendiffusion faces and 264 recognised faces out of 320 testing faces i.e 82.50 % face recognition rate. The same approach is repeat for various training - testing face sets such as, 120 - 280, 160 - 240, 200 - 200, 240 - 160, 280 - 120, 320 - 80 and their recognition rate is 85.71% , 89.58% , 91.00% , 95.63%

Table 6.11: Face recognition rate with respect to different ORL train set and test set as 100 % database i.e 400 faces

Training faces	eigendiffusion faces	Recognised/test faces	recognition(%)
80	79	344/400	86.00
120	119	360/400	90.00
160	159	375/400	93.75
200	198	382/400	95.00
240	238	393/400	98.25
280	278	395/400	98.75
320	318	397/400	99.25

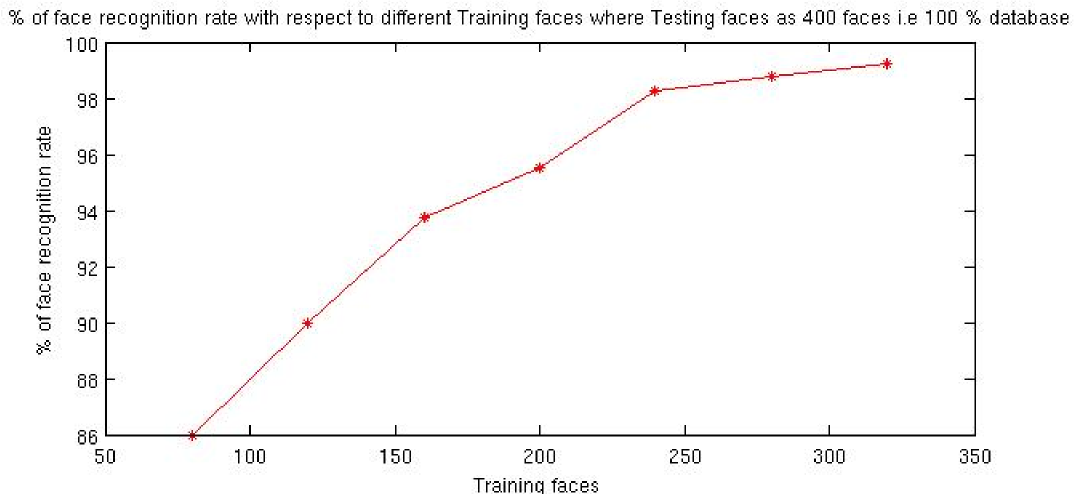


Figure 6.29: Face recognition rate with respect to different ORL train set and test set as 100 % database i.e 400 faces

, 95.83% , 96.25% respectively, results are shown in Table 6.12 . From this experiment we understood that the rate of face recognition is increasing when increases the training set but the rate of recognition is reduced in the case of varying testing face set when compared to constant whole database as testing face set . The Plot 6.30, shows how the recognition rate increases with respect to training faces set increases.

On Yale face database

The database contains 165 GIF images of 15 subjects. There are 11 images per subject, one for each of the following facial expressions or configurations: centre-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink. Here, the image "subject04.sad" has been corrupted and has been substituted by "subject04.normal". The each face image is resized to 56×46 and this whole database shows in Figure 6.31.

We consider first five faces from each subject which are centre light, glasses, happy, left-light, no glasses as training set. We calculate mean of the all training faces and which

Table 6.12: Face recognition rate with respect to different ORL train and test set cases

Training/Testing faces	eigendiffusion faces	Recognised/test faces	recognition(%)
80/320	79	264/320	82.50
120/280	119	240/280	85.71
160/240	159	215/240	89.58
200/200	198	182/200	91.00
240/160	238	153/160	95.63
280/120	278	115/120	95.83
320/80	318	77/80	96.25

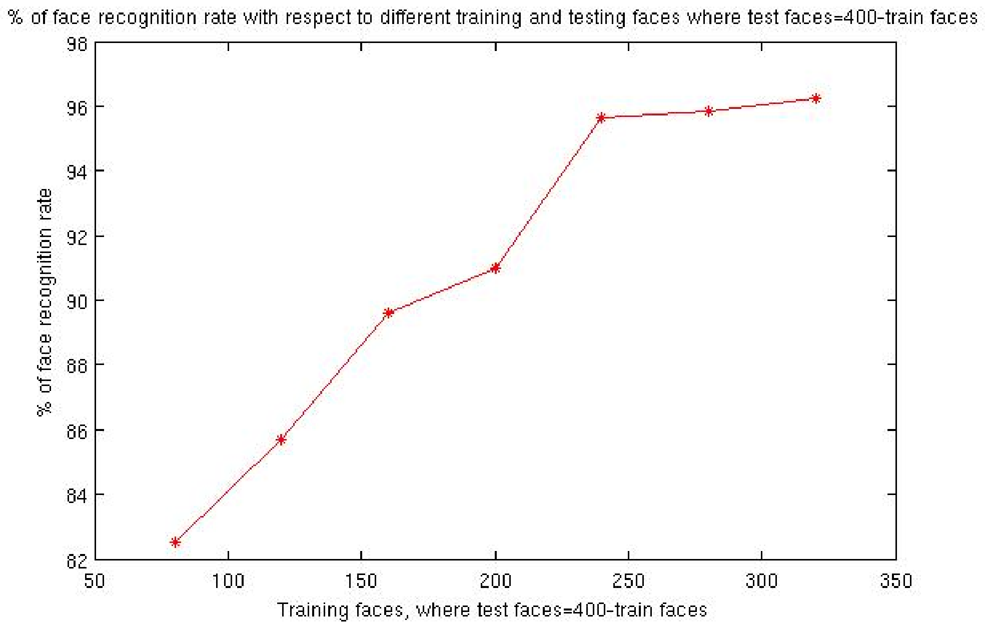


Figure 6.30: Face recognition rate with respect to different ORL train and test set cases

shows in Figure 6.32.

Now we calculate the covariance matrix from the training set and then calculate eigendiffusion faces by applying diffusion wavelets on covariance matrix i.e each orthogonal column vector represents a eigendiffusion face. The Figure 6.33 shows the 66 eigendiffusion faces out of a total 74 of eigendiffusion faces of training faces and these faces are obtained at first level of diffusion wavelets algorithm. We added 50 to each pixel to make these eigendiffusion faces visible.

We calculate weight vector for each face in the training set with the help of eigendiffusion faces. We consider whole Yale database as a test set and reconstructed each face in training set with the help of eigendiffusion faces and these reconstructed faces are shown in Figure 6.34. We recognised faces in two cases, firstly the training and test faces without histogram equalizer and secondly, the training and testing set with histogram equalizer. Histogram equalizer is a method to contrast adjustment using the image's histogram. In Yale face database 4th and 7th face of each subject has variation. So, in order to improve

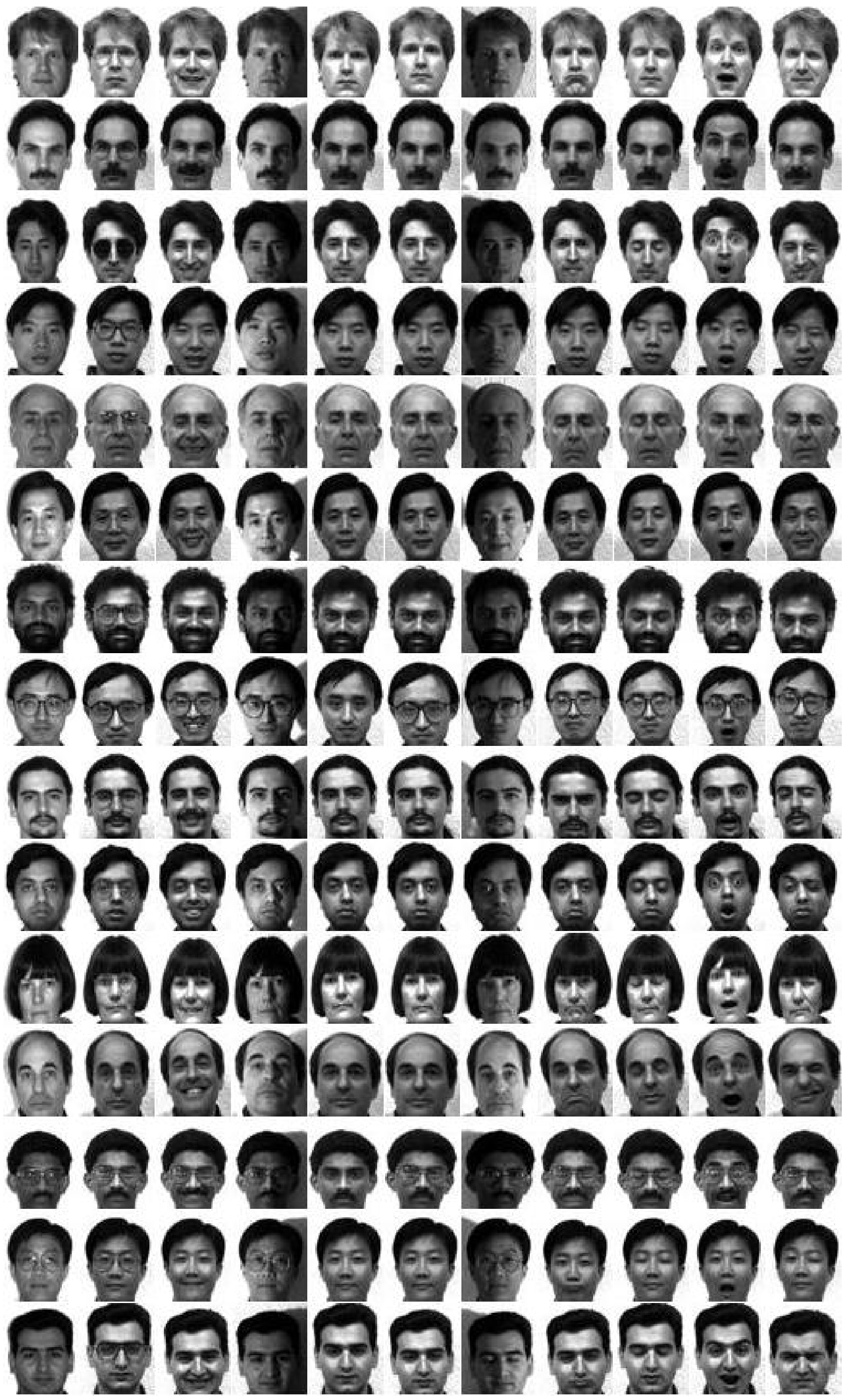


Figure 6.31: Yale Face Database B

the face recognition we tested this case. Our experimental results for face recognition at various training faces in both cases are shown in Table 6.13. Our face recognition rate at 45 training faces has difference and histogram equalizer method given better accuracy and same repeated till the training faces are 90. The training faces 105 and more has same recognition rate in both cases. Face recognition rate at various training and testing faces mention in the Table 6.14. Here also recognition rate same in both with or with-

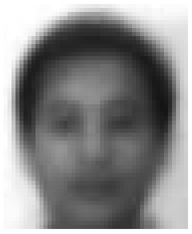


Figure 6.32: Yale Mean face

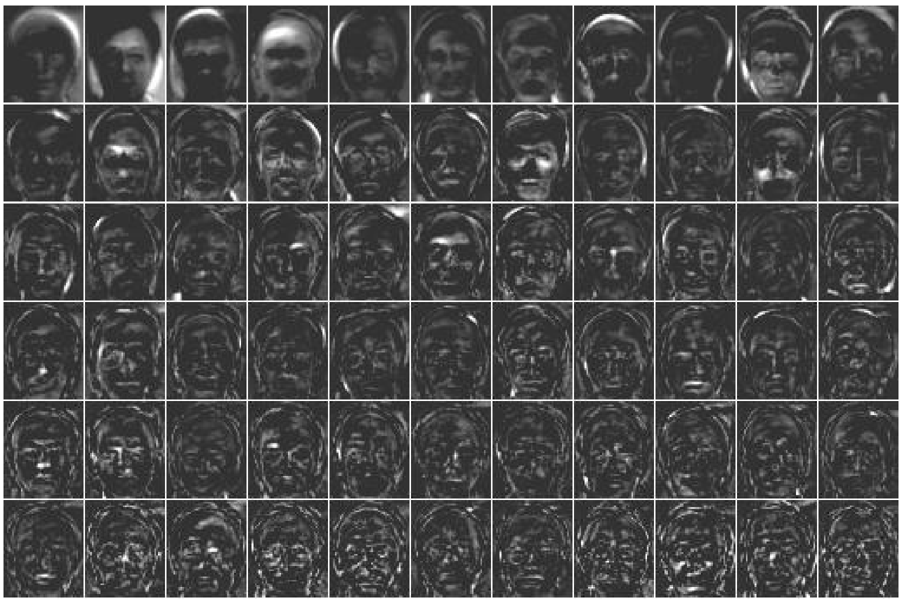


Figure 6.33: 66 out of a total 74 of eigendiffusion faces of Yale training faces

out histogram normalizer cases from the training faces 105 and more. Our algorithm is not suffering with illumination variation when we consider 64% or more of database as a training set.

6.3.2 Fingerprint Authentication

Fingerprint verification competition has different sets of databases. For our experiment we used the set B of the fingerprint database DB1 from FVC2000[36]. The fingerprints were acquired by using a low cost optical sensor with the of 300×300 and these fingerprints are mainly from 20 to 30 year-old students (about 50% male). The acquired fingerprints were manually analysed to assure that the maximum rotation is approximately in the range [-15, 15] and that each pair of impressions of the same finger have a non-null overlapping area. We used our proposed algorithm of eigendiffusion faces concept for authorizing finger prints and we explained in detail of our experiment in this section. We have 10 persons and each of them has 8 different orientations of their fingerprint image. The training set is derived from the database and we took 4 fingerprints from each subjects. We consider 40 fingerprint images as a training data. We described in detail about eigendiffusion faces at chapter 4. Each fingerprint image has a dimension 300×300 and if we use the same dimensionality for experimenting then we get 90000×90000 covariance matrix. So, in order to

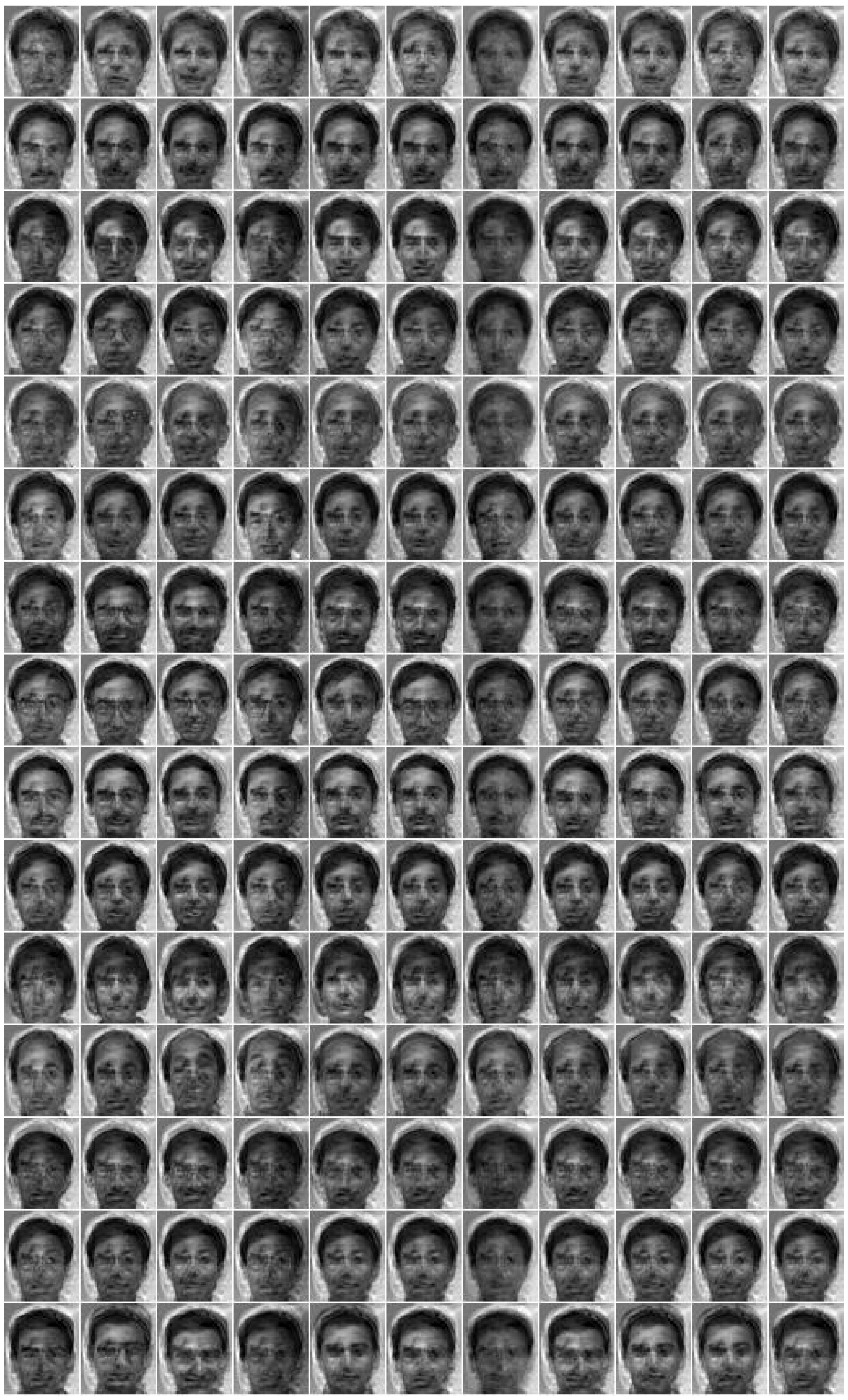


Figure 6.34: Reconstructed Yale database

reduce the computational complexity we sub sampled each fingerprint image to 100×100, shows in Figure 6.35 . Then we took this sub sampled image as an input to our fingerprint authentication algorithm. We formed training matrix with the size of 10000×40. Then we find the average of all the training fingerprints image, shows in Figure 6.36 to normalize every training fingerprint image. We calculate the covariance matrix by using the normalized fingerprint images.

Table 6.13: Yale face recognition rate with respect to different train set cases and and test set as 100 % database i.e 165 faces

Training faces	eigendiffusion faces	without histeq (%)	with histeq (%)
45	44	84.85	87.27
60	59	90.30	90.91
75	74	95.15	96.36
90	81	95.15	96.36
105	96	98.79	98.79
120	110	98.79	98.79
135	125	99.36	99.36
150	140	100	100

Table 6.14: Yale face recognition rate with respect to different train set and test set cases

Training/Testing faces	eigendiffusion faces	without histeq(%)	with histeq (%)
45/120	44	79.17	82.50
60/105	59	84.76	85.71
75/90	74	91.11	93.33
90/75	81	92.00	93.33
105/60	96	95.56	95.56
120/45	110	96.67	96.67
135/30	125	96.67	96.67
150/15	140	100	100

Now we have the covariance matrix with dimensions of 10000×10000. We applied our multi scale representation of diffusion wavelets to model low dimensional eigendiffusion faces from the higher dimensional finger print space. We achieve at the first level a reduction to 39 eigendiffusion faces, shows in Figure 6.37 Using these 39 eigendiffusion faces, we calculate weights of each training set of fingerprints. We already knew the classes for each training fingerprint. The fingerprint data which we use for authentication is the test set. We consider the whole database as a test set. Then we calculate the weights of each test fingerprint image by using the 39 eigendiffusion faces. We calculate Euclidean distance from each test data set to all the fingerprints in the training set and then identify which training fingerprint has the minimum distance and assign that class to test fingerprint i.e predicted class. This experiment repeats for the remaining test fingerprints. We recognised 50 fingerprints correctly i.e 62.50%. We understood that our method is authorising the fingerprints 100% which are considered in training set and it is not accurate when the test set is not include in training set.

we observe how the fingerprint rate changes with respect to various training fingerprints but still testing fingerprints are constant i.e 80 fingerprints. Now, 30 fingerprints (i.e 3 fingerprints from each person) are taken as training fingerprints and calculate the predicted class of each test fingerprint image by using eigendiffusion faces algorithm. Here,

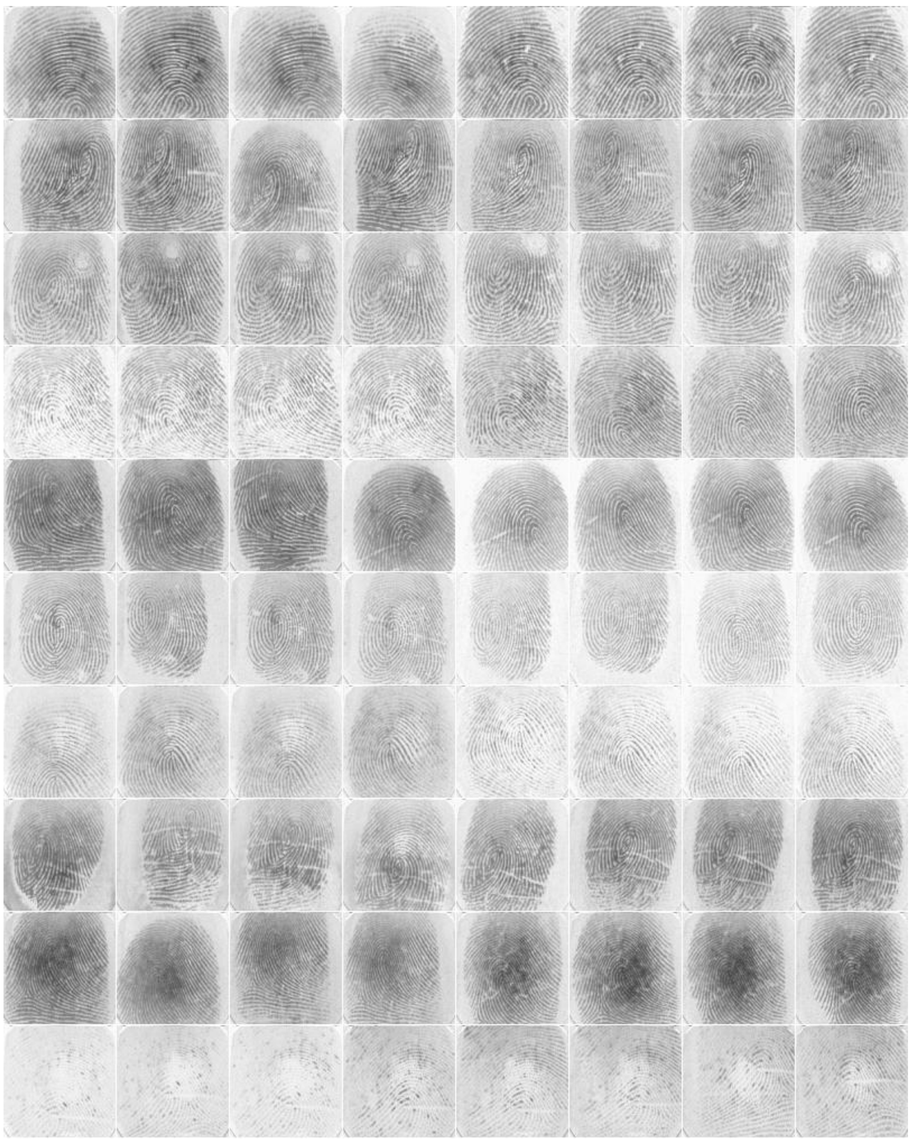


Figure 6.35: sub sampled set B of database DB1 from the FVC2000

Figure 6.36: Mean fingerprint

we get 29 eigendiffusion faces and 41 fingerprints are authenticated i.e 51.25%. Now, the same approach is to repeat for training fingerprints of 40, 50, 60, 70 and their recognition rate is 62.5%, 77.50%, 87.50%, 95.00% respectively, results are shown in Table 6.15 . From this experiment we understood that the rate of face recognition is increasing when increases the training set.

We also experiment how the fingerprint authentication rate varies when training fingerprint set as n number of faces and testing fingerprint set as $400 - n$. Suppose, if

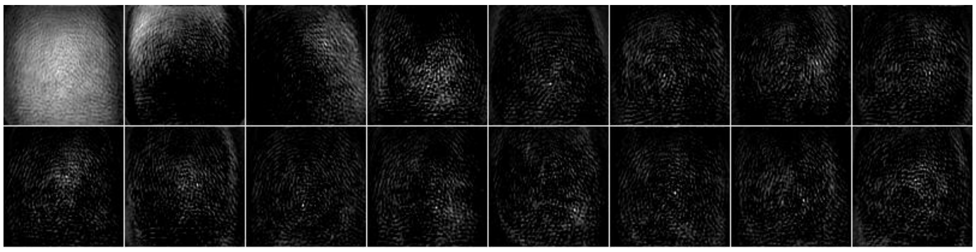


Figure 6.37: 16 out of a total 39 of eigendiffusion faces of fingerprints

Table 6.15: Fingerprint Authentication rate with respect to different train set, where test set as 100 % database i.e 80 fingerprint images

Training set	Eigendiffusion faces	Recognised/test	recognition (%)
30	29	41/80	51.25
40	39	50/80	62.50
50	49	62/80	77.50
60	59	70/80	87.50
70	69	76/80	95.00

we take training set as 30 faces then testing set taken as 50 faces. Now the above same procedure is applied to calculate fingerprint authentication. In this case, we have 29 eigendiffusion faces and 11 recognised fingerprints out of 50 testing faces i.e 22.00% fingerprint authentication rate. The same approach is repeat for various training - testing fingerprint sets such as, 40 - 40, 50 - 30, 60 - 20, 70 - 10 and their recognition rate is 25.00% , 40.00% , 50.00% , 60.00% respectively, results are shown in Table 6.16 . From this experiment we understood that the rate of fingerprint authentication is increasing when increases the training set but the rate of recognition is reduced in the case of varying testing fingerprints when compared to constant whole database as testing fingerprint set .

Table 6.16: Fingerprint Authentication rate with respect to different train and test set where test set = 80 - train set

Training/Test set	eigendiffusion faces	Recognised/test	recognition(%)
30/50	29	11/50	22.00
40/40	39	10/40	25.00
50/30	49	12/30	40.00
60/20	59	10/20	50.00
70/10	69	6/10	60.00

Chapter 7

Conclusion and Future Scope

7.1 Summary

In this thesis we have reviewed the diffusion wavelet algorithm in order to examine the possibility to estimate optical flow, to recognise faces and to authenticate fingerprint. We reviewed the various applications of diffusion wavelets. We also reviewed various algorithms for optical flow estimation and also reviewed about spectral graph dimensionality reduction methods and some of the face recognition algorithms. The diffusion wavelet provides a multiscale framework on high dimensional data. It extends the Fourier analysis on graphs and provides a wavelet approach to analyse the graphs. The multiscale property of the diffusion wavelet helps to analyse the data detailed from fine to coarser.

Our present results have shown how the diffusion wavelet can be used for dense and sparse optical flow estimation. An anisotropic kernel is used to define the similarity between pairs of pixels by considering all pixels as neighbourhood. A Markov chain is used in order to model the diffusion process. Finally, diffusion extended bases functions are computed at each level to calculate relation between pairs of pixels by considering the connectivity of the graph.

We experimented on various image for multiscale feature representation. We showed how the features are fine to coarse from starting level to end level. This representation is accurate when we have well defined feature. Some times even the processed image feature is not defined well, this algorithm tries to represent a feature better than processed image feature. Diffusion extended bases functions are coarser image features, i.e This algorithm removes the noise at each level and gives the coarser feature representation by removing the noise with respect to each level. Compared to raw pixels, diffusion extended bases have less noise. So, it gives the better results. Euclidean distances of locally defined diffusion extended bases functions are used to estimate the dense optical flow. The proposed methodology is applied on various image sequences. We have used this approach on Hamburg taxi sequence, Meteosat, Andrea Hurricane and Cornea layers. Our proposed dense optical flow algorithm is accurate and mostly the optical flow estimation fails where the features are not defined well. We used the same methodology on Dimetrodon and Venus image sequel to estimate dense optical flow and by using the colour code represented this colour map of optical flow. We calculated average angular error and average flow error with the help of our estimated optical flow and ground truth flows. For Dimetrodon AAE is 12.04 and AFE is 0.51 and for Venus AAE is 11.36 and AFE is 0.83. our results are quite comparable with other best methods.

Sparse optical flow estimation is also almost similar to above proposed methodology. Initially, we calculated image key points and then calculated diffusion extended bases for these key points. Again same steps to calculate optical flow. We used this methodology on Hamburg taxi sequence and Cornea layers. From the results we understood that the results are fine.

We proposed eigendiffusion faces methodology for face recognition and fingerprint authentication. We calculate the covariance of train set of faces and applied diffusion wavelet to calculate extended bases and these bases we called as eigendiffusion faces. By using this eigendiffusion faces we recognised faces and authenticated fingerprints. We applied two face data base, On ORL we have taken 200 faces as train set and remaining 200 as test set, Our algorithm recognised with 91% of accuracy. Yale data base has total 165 faces and in it 75 faces taken as train set and remaining faces took as test data. The percentage of recognition is 91.11% with out histogram normalizer. and with histogram normalizer 93.33%. The same methodology applied on fingerprints FVC2000 database but the results are not good. One reason for it, we subsampled the image before applying to our algorithm. I noticed, that our algorithm is recognising 100% if the test data is considered in train data. So, our algorithm will be good for feature matching.

7.2 Future Scope

Our proposed method for optical flow estimation is good. We just used only the Euclidean distance for estimating displacement. We will continue this work for better optical flow estimation by using the various similarity finding methods, example using the various types of correlation, various distance metrics. In our algorithm we are not used any smoothing algorithm. If we process our vector fields on smoothing algorithms then our estimation accuracy may increase.

For face recognition we just used histogram equalization to overcome illumination variation problem. But, compared this method there are many illumination normalization methods. So we will look for better illumination normalization method and will also for various classification methods. From our fingerprint result, we came to know that it may give best result for feature matching. We will continue our research on this issues.

Bibliography

[1] Ronald R.Coifman and Stéphane Lafon. Diffusion maps. *Appl.Comp.Harm.Anal.*, 21(1):5-30, 2006.

[2] Richard Socher, Matthias Hein. Manifold Learning and Dimensionality Reduction with Diffusion Maps. *Technical report*, July 20, 2008.

[3] J. de la Porte†, B.M.Herbst†, W.Hereman⋆, S.J.van der Walt†. An Introduction to Diffusion Maps. †*Applied Mathematics Division, Department of Mathematical Sciences, University of Stellenbosch, South Africa,* ⋆*Colorado School of Mines, United States of America*, 2008.

[4] Ronald R Coifman and Mauro Maggioni. Diffusion Wavelets. *Appl.Comp.Harm.Anal.*, 21:53-94, 2006.

[5] James C. Bremer, Ronald R. Coifman, Mauro Maggioni, Arthur D. szlam. Diffusion wavelet packets. *Appl.Comp.Harm.Anal.*, 21:95-112, 2006.

[6] Sridhar Mahadevan, Mauro Maggioni. Value Function Approximation with Diffusion Wavelets and Laplacian Eigenfunctions. *Technical report, Department of Computer Science, University of Massachusetts,* 2005-38, June, 2009.

[7] Arthur D. szlam, Mauro Maggioni, Ronald R. Coifman, James C. Bremer Jr. Diffusion-driven Multiscale Analysis on Manifolds and Graphs: top-down and bottom-up construction. *Technical report, Department of Mathematics, Yale University.*

[8] Mauro Maggioni, James C. Bremer Jr., Ronald R. Coifman, Arthur D. szlam. Biorthogonal Diffusion Wavelets for Multiscale Representation on Manifolds and Graphs. *Department of Mathematics, Program in Applied Mathematics, Yale University.*

[9] Mauro Maggioni, Sridhar Mahadevan. Fast Direct Policy Evaluation using Multiscale Analysis of Markov Diffusion Processes. *Appearing in Proceedings of* 23^{rd} *International Conference on Machine Learning, Pittsburgh, PA,* 2006.

[10] Mauro Maggioni, Sridhar Mahadevan. A Multiscale Framework for Markov Decision Process using Diffusion Wavelets. *Technical Report, University of Massachusetts, Department of Computer Science*, 2006-36.

[11] Mauro Maggioni, Ronald R Coifman. Multiscale Analysis of Data sets with Diffusion Wavelets. *Proc. Data Mining Biomed. Inf., Conf. presentation.*, April 28, 2007.

[12] Sridhar Mahadevan. Adaptive mesh compression in 3D computer graphics using multiscale manifold learning. *Appearing in Proceedings of* 24^{th} *International Conference on Machine Learning*, 2007.

[13] Marie Wild. Nonlinear approximation of spatiotemporal data using diffusion wavelets. *In Proc. CAIP*, 886-894, 2007.

[14] Mark Coates, Yvan Pointurier, and Michael Rabbat. Compressed Network Monitoring. *Technical report, Department of Electrical and Computer Engineering, McGill University*, 2007.

[15] K.P.Zhu, Y.S.Wong, W.F.Lu, J.Y.H.Fuh. A diffusion wavelet approach for 3D model matching. *Computer-Aided Design*, 41:28-36, 2009.

[16] Chang Wang, Sridhar Mahadevan. Multiscale Dimensionality Reduction Based on Diffusion Wavelets. *Technical Report, University of Massachusetts, Department of Computer Science*, June 29, 2009.

[17] Chang Wang, Sridhar Mahadevan. Multiscale Manifold Alignment. *Technical Report, University of Massachusetts, Department of Computer Science*, 2010.

[18] Salma Essafi, George Langs, Nikos Paragious. Hierarchical 3D diffusion wavelet shape priors. *IEEE 12th International Conference on Computer Vision*, 1717-1724, 2009.

[19] Ho Yan Suen, Wing Cheong Lau and OnChing Yue. Detecting anomalous web browsing via diffusion wavelets. *IEEE International Conference on Communications*, 1-6, 2010.

[20] L. Greengard, V.Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325-348, 1987.

[21] G. Beylkin, R. Coifman, V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Communications on Pure and Applied mathematics*, 141-183, 1991.

[22] F. Chung. Spectral Graph Theory. *CBMS-AMS*, 1997.

[23] James C Bremer Jr, Mauro Maggioni, Chang Wang, *Matlab code for Diffusion Wavelets*, available at `http://www.math.duke.edu/~mauro/code.html`, April,2011.

[24] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373-1396, 2003.

[25] Xiaofei He and Partha Niyogi. Locality preserving projections. NIPS 16. 2003.

[26] S.Mallat. A Wavelet Tour of Signal Processing. *Academic Press*, 1998.

[27] P.J.Burt, E.H.Adelson. The Laplacian Pyramid as a Compact Image code. *IEEE Trans. Commun.*, 31(4):532-540, 1983.

[28] Matthew Turk and Alex Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71-86, 1991.

[29] R. Brunelli and T. Poggio. Face Recognition: Feature vs Templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(10):1042-1052, 1993.

[30] Peter N. Belhumeur, João P. Hespanha and David J. Kriejman. Eigenfaces vs Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):711-720, July 1997.

[31] R. A. Fisher. The use of Multiple measures in Taxonomic Problems. *Ann. Eugenics*, 7:179-188, 1936.

[32] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91-110, 2004.

[33] S. Baker, D. Scharstein, JP Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *In Proc. ICCV*, 1-8 ,2007.

[34] Flow accuracy and interpolation evaluation. `http://vision.middlebury.edu/flow/eval`, April 2011.

[35] AT&T Laboratories Cambridge face database. `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`, April 2011.

[36] Fingerprint verification competition, FVC2000. `http://bias.csr.unibo.it/fvc2000/default.asp`, April 2011.

[37] Thomas Corpetti † , Étienne Mémin †, Patrick Pérez ⋆. Estimating Fluid Optical Flow. *Int. Conf. on Pattern Recognition*, 3:1033-1036, 2000.

[38] JL Barron, DJ Fleet, and SS Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43-77, 1994.

[39] M.J.Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75-104, 1996.

[40] Szymon Wartak , Adrian G. Bors . Optical Flow Estimation Using Diffusion Distances. *Int. Conf. on Pattern Recognition*, 189-192, 2010.

[41] M. Maggioni, H. Mhaskar. Diffusion polynomial frames on metric measure spaces. *Appl.Comp.Harm.Anal.*, 24(3):329-353, 2008.

[42] D. Geller, A. Mayeli. Continuous wavelets on compact manifolds. *Mathematishe Zeitshrift*, 262:895-927, 2009.

[43] David K. Hammond, Pierre Vandergheynst, Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Appl.Comp.Harm.Anal.*, 30:129-150, 2011.

[44] T. Hastie, R. Tibshirani, and J. H. Friedman. The elements of statistical learning. *Springer*, August 2001.

[45] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323-2326, 2000.

[46] T. F. Cox and M. A. A. Cox. Multidimensional scaling. *Chapman & Hall*, London, 1994.

[47] J.B.Tenenbaum, V.de silva, and J.C.Langford. A global geometric framework for non-linear dimensionality reduction. *Science*, 290:2319-2323, 2000.

[48] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185 203, 1981.

[49] T. Corpetti, E. Memin, and P. Perez. Dense estimation of fluid flows. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3):365 - 380, 2002.

[50] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *In Proc. of the Int. Joint Conf. on Artificial Intelligence*, 81:674-679, 1981.

[51] G. Le Besnerais and F. Champagnat. Dense optical flow by iterative local window registration. *IEEE Int. Conf. on Image Processing*, I - 137-40, 2005.

[52] C. Lawrence Zitnick, N. Jojic, and S.B. Kang. Consistent segmentation for optical flow estimation. *IEEE Int. Conf. on Computer Vision*, 2:1308-1315, 2005.

[53] Xiaofeng Ren. Local grouping for optical flow. *IEEE Conf. on Computer Vision and Pattern Recognition*, 1-8, 2008.

[54] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using PowerFactorization and GPCA. *Int. J. of Computer Vision*, 79(1):85-105, 2008.