AN INTERPOLATION BASED APPROACH FOR PATTERN RECOGNITION AND GENERATION

by

VISHNUKUMAR GALIGEKERE N

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
May 2013

This dissertation is dedicated to my mother and father, whose constant encouragement and support has been the driving force behind this endeavor.

ACKNOWLEDGEMENTS

This body of work would not have been possible without the support of many people who have provided immense help during various stages of my doctoral degree. First and foremost, I would like to express my deepest gratitude to my supervising professor and committee chair, Dr. Gutemberg Guerra-Filho, who continually and patiently conveyed words of encouragement with regards to research and scholarship. I am deeply appreciative of his constant guidance and persistent help, without which, this dissertation would not have been possible.

I would like to extend my sincere thanks to all my committee members Dr. Jean Gao, Dr. Heng Huang and Dr. Manfred Huber for showing interest in my research and serving on my committee. I am especially grateful to Dr. Manfred Huber for his support during the final stages of my research and engaging me in exciting research discussions. In addition, I would like to sincerely thank my graduate advisor Dr. Bahram Khalili for supporting and mentoring me throughout my career at UT Arlington. I am also deeply thankful to Dr. Nikola Stojanovic, without his support and encouragement, I would have not been able to pursue this endeavor.

I would also like to extend my gratitude to the department of Computer Science and Engineering at the University of Texas at Arlington and my advisor, Dr. Gutemberg Guerra-Filho for providing financial support and the necessary infrastructure required for my research.

A very special thanks to all my friends for being around like family during various stages of my tenure at UT Arlington. Last but certainly not the least, I would like to express my deepest gratitude to my parents Mrs. Sujaya N and Dr. GR Nagabhushana, my brother Dr. Veda Prakash and my loving wife Sridevi Bajgur for their unconditional love, constant support,

patience and encouragement, without which, I would have not been able to pursue this endeavor.

April 18, 2013

ABSTRACT

AN INTERPOLATION BASED APPROACH

FOR PATTERN RECOGNITION

AND GENERATION

Vishnukumar Galigekere N, PhD

The University of Texas at Arlington, 2013

Supervising Professor: Gutemberg Guerra-Filho

A large number of problems in computer vision and computer graphics can essentially be reduced to a pattern recognition problem. In this thesis, we explore a novel interpolation based framework to address some of the various recognition problems in these areas. Our interpolation based framework is a supervised learning algorithm that allows for both generation (synthesis) of new patterns as well as perception (analysis) of existing patterns. The method is simple to implement and yet, expects a very straightforward and intuitive set of parameters to model the complex nature of such recognition problems.

Specifically, given a set of training data along with their parameters, we can learn a model that is a compact representation of the set of all patterns defined in a parametric space. Having learnt such a model we are able to generate any new patterns defined within that parametric space. Moreover, as an inverse operation, we are also able to estimate the parameters of any existing pattern. Based on this 'synthesis-analysis' approach we propose a

method to recognize patterns and evaluate it in rather diverse areas such as recognition of objects/faces in varying illumination conditions and, human motion across different skeleton sizes. Using the same approach we demonstrate the methods application in the area of image based modeling and rendering, where, we are able to render ‘unknown’ objects into a scene provided we have at least one ‘known’ object in it. Another application is in the area of animation where, given a set of human motion data differing in skeleton size but for a specific action, we are able to re-target that specific action to an identical skeleton but of varying bone lengths.

Also, in this thesis, we explore a novel image feature descriptor built using a bank of Gabor filters and evaluate its effectiveness in an object recognition framework using synthetic and real data. We also describe our software tool that allows for automatic generation of ground-truth data for various computer vision problems such as camera calibration, feature matching, 3D reconstruction, object tracking and object recognition.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Background

A typical supervised learning algorithm involves a learning phase, where the algorithm learns a classification model using a set of training data. This learnt model is then used in the testing phase, to assign a class or label to unknown instances of the input pattern. The assigned label could be real-valued, integer-valued or even a name of a class such as Class-A or Class-B. Formally, consider a set S of n training data-label pairs $S = \{(d_1, l_1), (d_2, l_2), (d_3, l_3) \ldots\ldots (d_n, l_n)\}$, where $d_i \in D$ is the input data set and $l_i \in L$ its corresponding set of labels. Assuming, there exists an unknown function g that maps the data items to their corresponding labels given by $g: D \rightarrow L$, the goal of the learning algorithm is to be able to learn a function $f: D \rightarrow L$ which approximates g. An example application problem could be that of a face detection algorithm that uses several images of faces and non-faces/background images as training data with associated labels *face* and *non-face* respectively, to learn a model that can classify new (unknown) input images as a face-image or non-face-image. Currently many popular methods including Support Vector Machines [15], Naïve Bayes Classifiers [47] and Decision Trees [46] are used to design such learning algorithms and, over the years, several supervised learning algorithms have been designed and applied effectively to address various problems in diverse areas. However, the subject continues to be an active research topic in Machine Learning and Pattern Recognition circles.

In this document we present one such learning algorithm based on interpolation where, the training set comprises of patterns and a set of parameters associated with each pattern. The main novelty of this approach is that we are not only able to learn a model that can estimate the parameters of an input pattern, but also generate a synthetic pattern for a given set of

parameters. The parameter estimation part of the method can be termed as the *analysis* part while the pattern generation part can be defined as the *synthesis* part. As an example, if the patterns are images of an object taken in various illumination conditions, the associated parameters could be the intensity levels of the light sources or their positions with respect to the object or even a combination of the two. Having learnt the interpolation model for such a case, we are able to use the *analysis* part to estimate the illumination parameters of a query image and, as a reverse process, given a query set of parameters; we can generate a new image of the object with the illumination conditions defined by this query parameters, which is the *synthesis* part. Moreover, we use this *analysis-synthesis* approach to develop an effective pattern recognition framework and demonstrate its application to some of the popular problems in Computer Vision and Computer Graphics.

1.2 Motivation

As another example where science draws its inspiration from nature and imitates the same, this method of supervised learning is very much akin to us humans learning from our past experiences. Similarly, our approach here is motivated by the theory of *mirror neurons* [48] in Neuroscience. This theory essentially places perception and generation under the same foundation. The mirror neuron theory states that the same neurons fire in the brain when a person perceives a particular sensory-motor pattern and when the subject generates the same pattern. For an example in the visual domain, the theory suggests that a set of neurons in the brain will be active when a subject recognizes the image of an object such as an apple fruit. Similarly, the same set of neurons fire when the subject pictures the image of an apple fruit through imagination or dreaming. This theory indicates that both synthesis and analysis are performed according to the same fundamental framework. We propose here an interpolation based approach, inspired by this mirror neuron theory, to perform pattern recognition and generation using the synthesis-analysis framework.

The framework discussed in this work was essentially designed to specifically address the problem of object recognition. However, while furthering our research, we have had success in expanding the scope of the technique to address problems related to Image Based Lighting and human motion-based problems such as Motion Retargeting, in the domain of computer graphics.

1.3 Thesis Layout

Initially, our research focus was mainly on addressing the problems related to feature description for visual object recognition. In an attempt to address issues related to object recognition, we developed a novel image descriptor using a bank of Gabor filters. Chapter 2 is dedicated to this part of our work. However, the rest of the document is dedicated to elaborate on the interpolation based approach and its applications, which forms the crux of our research.

Chapter 3 formally introduces the interpolation approach and addresses the problem of object recognition and face recognition in varying illumination conditions. Further, in the same chapter, we discuss the possible expansion of the method's application to object recognition subjected to geometrical variations. In Chapter 4 we elaborate further and demonstrate the method's application in Image Based Lighting and Rendering. In chapter 5 we expand our approach to address the problem of Motion Retargeting and Motion Recognition. In chapter 6, we discuss the conclusion and future work. Moreover, to aid our research we developed a computational tool called CVPoV that allows one to automatically generate synthetic ground-truth data to test several computer vision applications. The description of the tool is made available in Appendix A.

CHAPTER 2

TOPOLOGICAL GABOR DESCRIPTORS: EXPLORING A FILTER BANK STRUCTURE FOR IMAGE FEATURE MATCHING

2.1 Introduction

In Computer Vision, salient features are associated with regions in an image which are visually more informative than others. Feature description is a fundamental and challenging problem in image processing and computer vision. It is the foundation to problems such as object or scene detection, recognition, categorization, and tracking, where robustness depends on the underlying feature and its descriptor. Information cues used to describe salient features are be color, texture, shape, structure, or a combination of two or more of such elementary measurements in a local neighborhood.

The major objective of a feature descriptor is to achieve the best compromise between invariance and distinctiveness. In other words, a good descriptor needs to allow the matching of similar regions through invariant properties while separating different patterns into distinct classes. This is a hard problem when real noisy images obtained under unconstrained illumination are considered. Moreover, other imaging variations such as camera viewpoint changes make it even harder to deal with. Hence, a good image feature descriptor should be invariant to these changes while still capable of discerning between different regions.

Image features constructed using Gabor filters are quite popular in the computer vision community [29, 37]. The main reason for this popularity is that the most important imaging variations such as scale and rotation are in fact parameters of Gabor filters themselves. In this chapter, we introduce a new feature descriptor based on a topologically structured bank of 2D Gabor filters. The construction of the feature descriptor from the filter bank responses allows us to compare different descriptors with a simple circular shift operation. In other words, the filter

bank is constructed in a way that rotation invariant search operations can be performed in the descriptor space by a simple circular shift of the descriptor vector. Scale invariance is achieved by searching in the scale space of the image pyramid. Moreover, our descriptors are obtained by computing a scalar for each filter in the bank by multiplying it with the image window. Although Gabor filters require higher computational resources, this method allows for the reconstruction of the image information from the descriptor space by computing the inverse of the filter bank.

Our main contribution here is the construction of a 2D Gabor filter bank with a topology that allows for a simple 1D circular shift in the descriptor space to search for matches that are rotationally invariant. The rotational invariance is robust to a complete 360 degree rotation in the z-axis. The descriptors are computed for a spatial location in a dense pyramid of images to achieve scale invariance. We also take advantage of our computational tool, CVPoV (details in Appendix A) that is used to generate synthetic data along with the ground truth to experiment and test our results in a controlled setting.

Our approach has given excellent results with synthetic data. For images of a synthetic scene that was subjected to 0 to 180 degree rotation about the z-axis in steps of 8.5 degrees, we obtained a 93.5% matching rate. In another experiment with the synthetic data where the camera was translated in the z-axis moving it toward the scene and thereby simulating true scale variation, we obtained a matching rate of 81.1%. While experimenting with real images of famous buildings, the average matching rate for three different sets of scenes was an encouraging 41.3%.

The outline of this chapter is as follows. In Section 2.2, we will briefly describe previous work on algorithms that address the problem of image or feature description. In Section 2.2, we will introduce the Gabor filter, describe the construction of the filter bank, and describe the topological Gabor descriptors in detail. In Section 2.4, we will discuss the methods for comparing descriptors using the shift operation and also the distance measures used to match

them. The Section 2.5 describes all the experiments in detail. In Section 2.6, we will discuss the conclusions we have arrived at with our experiments and go on to discuss the scope for improvement and some closing comments.

2.2 Previous Work

The Scale Invariant Feature Transform (SIFT) [38] concerns a descriptor for features which are reasonably invariant to image scaling, translation, rotation, illumination changes, and affine or 3D projection. The computation of SIFT features involves several stages. The initial step in the algorithm is the scale invariant feature detector based on the Difference of Gaussians (DoG). Once features are detected, the orientation of the feature is obtained by using a 36 bin histogram of gradient orientations within a Gaussian weighted circular window whose size is determined by the DoG feature detector. This region around the feature is divided into 4 x 4 squared areas associated with histograms of gradient orientations. The number of bins on these histograms is reduced from 36 to 8 bins. This results in a 4 x 4 x 8 = 128 dimensional feature vector which is then normalized to make it invariant to illumination changes. This resulting 128 dimensional vector represents the SIFT descriptor. The main disadvantage of SIFT features is that they are relatively expensive to compute. Furthermore, as a histogram-based method, it is impossible to reconstruct the original image region from the descriptor.

There are several algorithms that improve on SIFT features both in terms of computational speed and accuracy. In the PCA-SIFT algorithm [30], instead of the original 128 dimensional feature vector, a 41 x 41 patch around the interest point is extracted and then horizontal and vertical gradients are computed for the 39 x 39 interior patches. This leads to a 39 x 39 x 2 = 3042 dimensional vector. The dimensionality of this vector is then reduced to 20 using the Principal Component Analysis (PCA). The algorithm achieves a more distinctive representation of the image features coupled with a reduced feature dimension. Gradient location-orientation histogram (GLOH) [43] differs from SIFT at the sampling stage. In the

GLOH descriptor, 17 log-polar location bins and 16 orientation bins are considered. This leads to a 272 dimensional feature space which is reduced to 128 dimensions using PCA.

The Speeded-Up Robust Features (SURF) [6] approximates or even outperforms previously proposed schemes such as SIFT features with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster. SURF features are, like SIFT features, a histogram-based descriptor. The main advantage of SURF over SIFT is that it is significantly faster than the latter. The algorithm owes its computational speed advantage to the usage of integral images to avoid image convolutions. SURF relies on a Hessian-matrix approximation on integral images to compute interest points. This algorithm gains an even more significant advantage with respect to computational speed as integral images allows the up-scaling of these filters at constant cost instead of computing an image pyramid. Once the interest points are detected at the given scales, the algorithm computes a distribution of the intensity content within the interest point neighborhood, similar to the gradient information extracted by SIFT. First order Haar wavelet responses in x and y directions are computed in a circular region to arrive at the most suitable orientation. The SURF descriptors are then computed in the form of an 8 x 8 oriented grid where each cell in the grid is the response of the Haar wavelet.

Other feature descriptors such as Geometric Blur [8] and Jet Descriptors [31] do not perform as well as SIFT or SURF. A survey of such methods is found in [45, 54]. Descriptors based on the Geometric Blur compute the average of the edge signal response over small geometric transformations. Therefore, Geometric Blur is basically an average over geometric transformations of a signal. The Jet descriptor combines local derivatives into sets of differential operators which are rotationally invariant. An issue with these filters is that they are not scale invariant.

In contrast to histogram-based methods such as SIFT and SURF, a major advantage of our method is the ability to reconstruct the original image region from the descriptor as long as

the inverse of the filter bank can be obtained. An advantage over other filter-based approaches is that rotation and scale variations are directly handled as they form the parameters of the Gabor filter itself. Moreover, these filters allow us to construct filter banks with a topology such that rotation invariant matching of the descriptors can be achieved simply by circularly shifting the descriptor.

2.3 Topological Gabor Descriptor

2.3.1 Gabor Filters

Gabor filters have been extensively used in texture recognition [39] and dominated the area of iris recognition [16]. Gabor filters are linear filters that are sensitive to the local frequency and rotation of an image region. They are generated by modulating a 2D Gaussian function by a sinusoidal plane wave. In simple terms, the frequency of the complex sinusoidal carrier senses the spatial frequency information while the Gaussian envelope localizes this aspect. The rotation of the elliptical Gaussian envelope allows the filter to sense the rotation of the image information. Hence, a bank of such filters associated with a range of frequencies and a range of orientations may be used to construct a descriptor for an image region. In fact, it has been established by Daugman [17, 18] that such structures of 2D Gabor filters were very similar to the organization and the characteristics of the mammalian visual system.

There are a few different versions of Gabor filters that have been designed. In this chapter, we use the normalized 2D Gabor filter function in the continuous spatial domain defined below [14]: $\psi(x,y;f,\theta) = \frac{f^2}{\pi\gamma\eta}\, e^{-\left(\frac{f^2}{\gamma^2}x'^2+\frac{f^2}{\eta^2}y'^2\right)} e^{j2\pi f x'}$, where $x' = x\cos\theta + y\sin\theta$, $y' = -x\sin\theta + y\cos\theta$, f is the frequency of the 2D sinusoidal carrier, θ is the anti-clockwise angle of the major axis of the Gaussian envelope and the sinusoidal plane wave, γ is the sharpness (spatial width) along the major axis, and η is the sharpness along the minor axis (perpendicular to the wave). The aspect ratio of the Gaussian is therefore given by $\lambda = \eta/\gamma$. The choice of using the above normalized 2D Gabor filter is to guarantee that the filters of

different frequencies are scaled versions of each other [32]. The real part of such a filter is show in Figure 2.1. The imaginary part is a 90 degree phase shifted version of the real part.

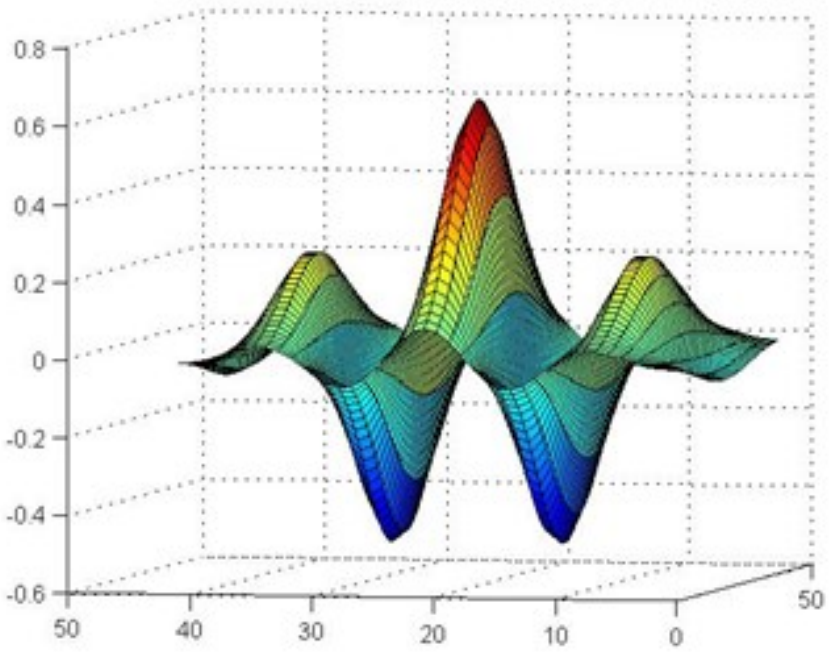


Figure 2.1: Real part of a Gabor filter, where f = 1.4142, $\boldsymbol{\theta} = \mathbf{0}$, and $\boldsymbol{\gamma} = \boldsymbol{\eta} = \mathbf{1}$.

2.3.2 2D Gabor Filter Bank

The response of 2D Gabor filters is invariant to changes in scale, rotation, and uniform illumination. However, construction of a filter bank that is optimal for a problem in a general setting is by no means a trivial task. Since our goal is to build a robust feature descriptor that should potentially address higher level problems such as object or scene recognition, we design our filter bank in such a way to cover all possible scales and all possible rotations about the axis perpendicular to the image plane with a certain granularity in the discrete domain. We also want the descriptor to tolerate minor changes in viewpoint and illumination. Given such objectives, we use a large filter bank to achieve a good degree of robustness to image variations. The orientations must be spaced uniformly to effectively cover a range of rotational angles about the z-axis [33]. Formally, the angles are defined as $\theta_k = \frac{k2\pi}{n}$ for $k = \{0,1, \ldots . n-1\}$. Since the

Gaussian envelope is symmetric about the axes, we chose to vary θ in the range [0, π) instead of the range [0, 2π) and we select *n* such that θ varies in steps of 1 degree. This results in a filter bank with 180 2D Gabor filters that should be sensitive to minor rotations such as a few degrees to a complete 180 degree out of phase about the z-axis. Furthermore, scale invariance is realized by spacing the frequency parameter in a logarithmic scale [11] as depicted in the following equation. $\mathrm{f_k} = \mathrm{a^{-k}f_{highest}}$ for $\mathrm{k} = \{0,1,\ldots.\}$ and $\mathrm{a} > 1$. The scaling factor $\mathrm{a} = 2$ or $\mathrm{a} = \sqrt{2}$ allows for octave spacing or half octave spacing, respectively. Figure 2.2 depicts a filter bank constructed using the above mentioned scheme for four orientations and four frequencies.

Since we are using normalized Gabor filters varying the filter frequency is equivalent to applying a single filter of a constant frequency to scaled images. We opted to construct dense image pyramids so that we can achieve a robust search through a high density scale space. For the rest of this document we will deal with a filter bank constructed with filters of a constant frequency but varying orientations.

For an image window with radius *w*, a 2D Gabor filter is a square matrix of size $(2\mathrm{w}+1)\mathrm{x}(2\mathrm{w}+1)$. We construct our filter bank by vectorizing each filter to a $(2\mathrm{w}+1)^2$ sized row vector and stacking them up to form a matrix of size $(2\mathrm{w}+1)^2 \times 180$.

2.3.3 Topological Gabor Descriptor

Traditionally, the approach to compute the response of a Gabor filter in the spatial domain on an image window is by performing a 2D convolution between the filter and the image. For a single image window, 2D convolution is the dot product between the filter vector in the bank and the vectorized image window. The convolution of the same image window with all filters in the bank results in a feature descriptor in the form of a vector of 180 real numbers associated with the filter responses. Each scalar in the descriptor is the response of a 2D Gabor filter for a particular orientation. Therefore, the Topological Gabor Descriptor (TGD) is basically a time series of length 180. Figure 2.3 shows the TGDs computed for a two sample image windows.

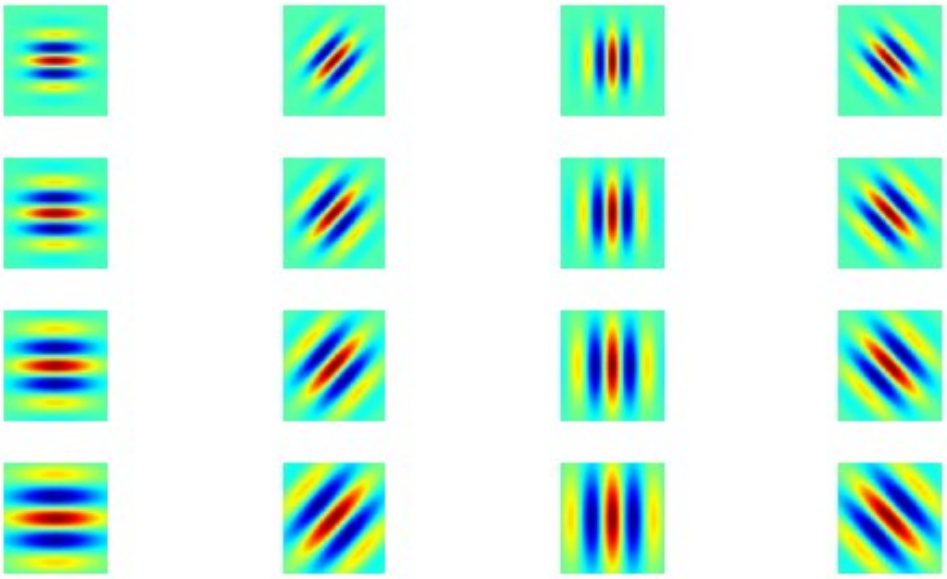


Figure 2.2: A 4 x 4 Gabor filter bank generated for angles from 0 to 180 degrees and for 4 different frequencies.

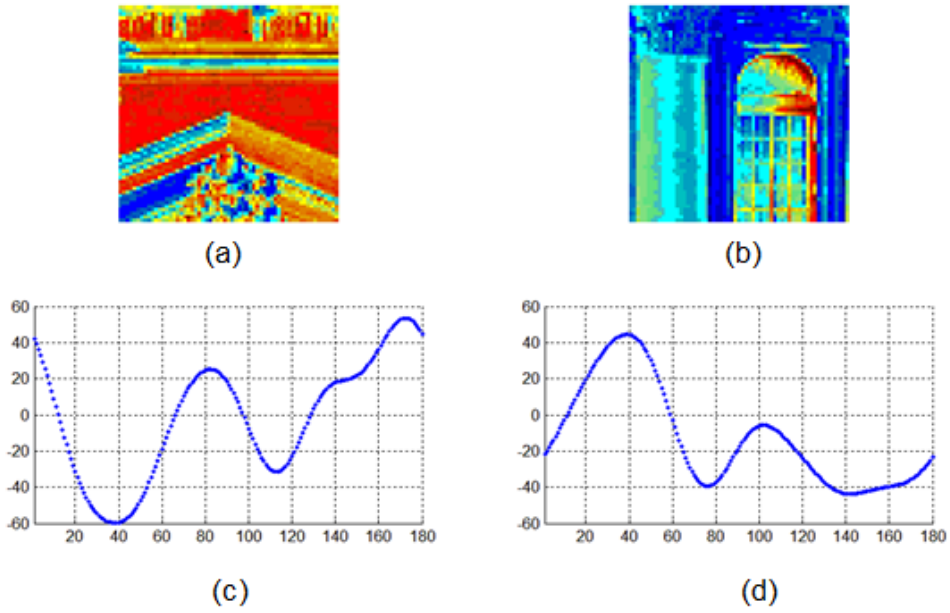


Figure 2.3: Original image windows (a) and (b) along with their corresponding 1D Topological Gabor Descriptors (c) and (d).

2.4 Circular Shift Similarity

The other significant contribution in this chapter is that the topology of the filter bank is such that matching different descriptors for rotated data is achieved by simply performing a circular shift on the computed descriptor. After the circular shift, a nearest neighbor search is executed.

The computed TGD is a sequence of 180 scalar values where each value is the dot product of the vectorized image window and the 180 Gabor filter vectors whose orientation changes according to the equation discussed in Section 2.3. As a result, TGDs are basically time series of size 180. The topology of the designed filter bank is such that, rotation invariance is achieved by simply performing an iterative circular shift in one of the two compared descriptors and measuring the similarity between descriptors for each shift. The shift yielding the most similarity is the best match in the rotation space. The same procedure is performed at all scales of the image pyramid such that the descriptors are invariant to changes in scale. For example, given an image window and a 45 degree rotated (anticlockwise) version of the same window, then the TGD for the original window and the TGD for the rotated window will be identical when the TGD of the rotated window is circularly shifted to 45 places to the left. However, in the case of image windows that are similar but extracted from different images, the TGDs should be similar in shape when compared to descriptors of dissimilar windows. See Figure 2.4 and 2.5 to visualize this example.

The comparison of two TGDs d_1 and d_2 (blue curves in figure 2.4 and 2.5 respectively) involves the iterative circular shifting of d_2 (blue curve in 5) into a shifted version d_2^s (green curve in 4 and 5) and the similarity measure between d_2^s and d_1, for all possible shifts s. Our experiments suggest that a simple similarity measure consisting of the combination of L_2 norm and correlation coefficient works better than other readily available techniques. We

experimented with several distance metrics ranging from the simple L_1 norm to techniques such as Dynamic Time Warping [49]. The details of which are described in the next section.

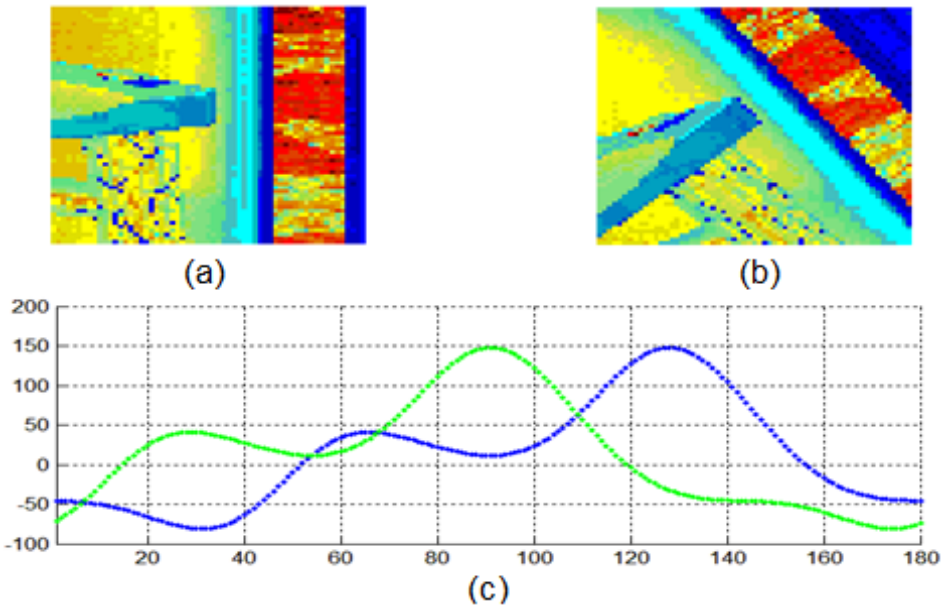


Figure 2.4: The TGD (c) of the rotated window (b) in blue. The shifted TGD of the same window (a) in green depicting that the shift is about 45 degrees to the left.

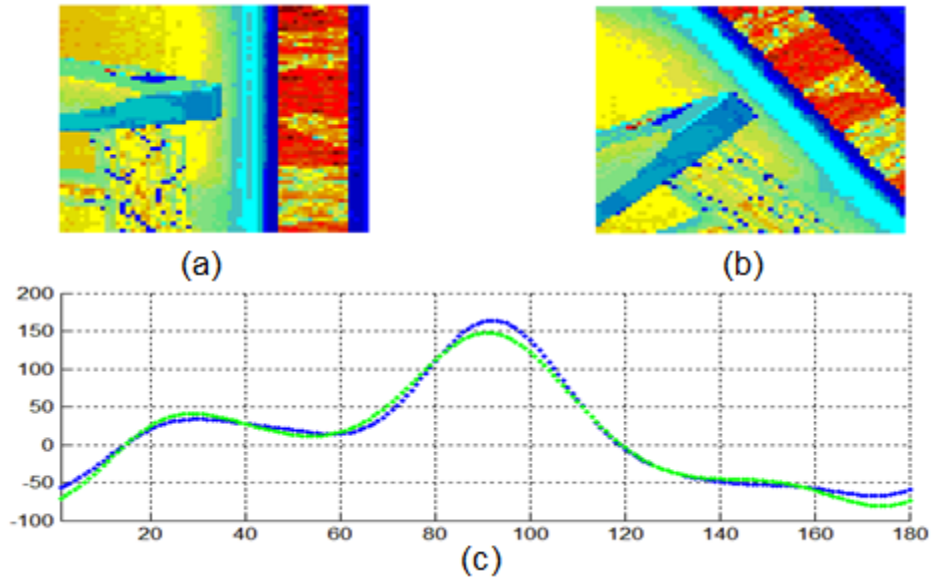


Figure 2.5: The TGD (c) of the original image window (a) in blue and the matched TGD of the rotated window (b) in green.

2.5 Experimental Results

All the experiments were performed separately on images of synthetic scenes and real scenes.

2.5.1 Evaluation Methods

The Gabor filters in the filter bank used in our experiments are obtained according to the following parameters. The frequency of the filters was set to 1.0 and the radius of the Gaussian envelope was fixed at 30 pixels giving rise to a filter of size 61 x 61 pixels. A total of 180 of such filters were generated for angles starting at 0 degrees and ending at 179 degrees. Each of these filters was vectorized to form a row vector of size 3721. The filter bank was then constructed as a matrix of 180 rows and 3721 columns. The scale space consisted of searching through an image pyramid constructed for each image. Each image was scaled by a scalar factor ranging from 0.5 to 2.0 in steps of 0.1. Hence, a rather dense pyramid consisting of 16 images was incorporated in our experiments.

With respect to the similarity measure of the descriptors, we considered several distance metrics such as correlation coefficient, L1 and L2 norms, and Dynamic Time Warping. We found that the best similarity measure is the Euclidian distance between the first derivatives of the compared descriptors divided by the correlation coefficient between the two derivatives of the compared descriptors. Formally, let $D_1$and D_2 be two descriptors computed for a pair of image windows, and their respective derivatives be represented by $D_1' = [d_1^1\ d_2^1\ \dots\ d_n^1]$ and $D_2' = [d_1^2\ d_2^2\ \dots\ d_n^2]$ where *n* = 179. Now, we compute the Euclidian distance between D_1' and D_2' using the equation, $L2\ (D_1', D_2') = \sqrt{\sum_{i=1}^{n}(d_i^1 - d_i^2)^2}$ and the correlation coefficient using the equation, $(D_1', D_2') = \frac{cov(D_1', D_2')}{\sigma_{D_1'}\sigma_{D_2'}}$, where the $cov(D_1', D_2')$ is the covariance between D_1' and D_2' and the $\sigma_{D_1'}$ and the $\sigma_{D_2'}$ are the standard deviations. We now compute the distance metric $\Delta(D_1, D_2) = \frac{L2\ (D_1', D_2')}{2+\rho(D_1', D_2')}$. Since, $\rho(D_1', D_2')$ yields a value between -1 and 1, we add 2 to the

denominator to make the metric positive. Using this metric and performing a nearest neighbor search will give us the best match between the image descriptors.

The performance of our approach is measured in terms of its efficiency in feature matching. We consider the rate of correct matches between various 2D points of an image with points of another image of the same scene. Here, a match is said to be correct if the nearest neighbor search results in the same point as that of the ground truth. We construct the confusion matrices for different image pairs in each dataset. We followed the same evaluation procedure for both synthetic and real data. Each dataset of a particular scene S consists of m images and n three-dimensional points in the scene. The 3D points correspond to 2D projected points in each image. The confusion matrix C_{ij} for any given image pair I_i and I_j in S is a square matrix of size n x n, where each row corresponds to points in I_i and each column corresponds to points in I_j. Each element $C_{ij}(r,c)$ of the confusion matrix represents the error value computed while matching the TGDs for the r^{th} point of I_i and the c^{th} point of I_j. Note that an ideal algorithm should generate a confusion matrix with a diagonal populated with errors that are much smaller than those in the remaining elements. By constructing confusion matrices for different image pairs of a scene S, we visualize the low level behavior of the algorithm across the whole dataset. While the confusion matrix provides a good tool to assess the performance in detail, the matching rate for each dataset and the overall average matching rate will provide a quantitative measure for the performance in terms of the percentage of correct matches. This gives us the overall general performance of the algorithm. The matching rate for a data set S is the ratio of the number of points that were matched correctly to the total number of points (m x n) in the data set S. Similarly, we compute the matching rate for all the scenes. We then compute the total average matching rate by taking the ratio of the sum of the individual scene matching rates and the total number of scenes considered.

2.5.2 Synthetic Data

The synthetic data used in our experiments was generated with CVPoV. CVPoV (presented in Appendix A) is a computational tool to generate synthetic data with the necessary ground-truth for various computer vision problems such as camera calibration, feature matching, 3D reconstruction, object recognition, and others. We use the depth map and the camera calibration data generated by MegaPoV [63] and VLPoV [68] to compute the motion field which accurately describes the horizontal and vertical displacement of each pixel in different images of the same scene. CVPoV allows the user to define different camera configurations and automatically render all the images along with a depth map for each of them. Once the images and the depth maps are generated, the motion field data and occlusion maps for different pairs of the rendered images are obtained. A detailed description of all functionalities of CVPoV and the source code are publicly available at [67].

The testing of our algorithm with synthetic data focused on two main transformations: rotation about the *z*-axis, the axis perpendicular to the image plane, and translation in the direction of the *z*-axis which results in scale variations. We used a realistic 3D model of a kitchen, designed by a ray tracing artist [58, 59], to render synthetic images. This particular scene was chosen as it had lots of objects and texture variations. To obtain the rotated data using CVPoV, we considered a total of 21 camera poses where in each pose the camera was in the same location but rotated in uniform steps about the *z*-axis from 0 to 180 degrees in the clockwise direction. The images rendered for each of these cameras are the rotated versions of-the image. Since CVPoV also generates a depth map for each image, we computed the motion field between the first image and the remaining 20 images.

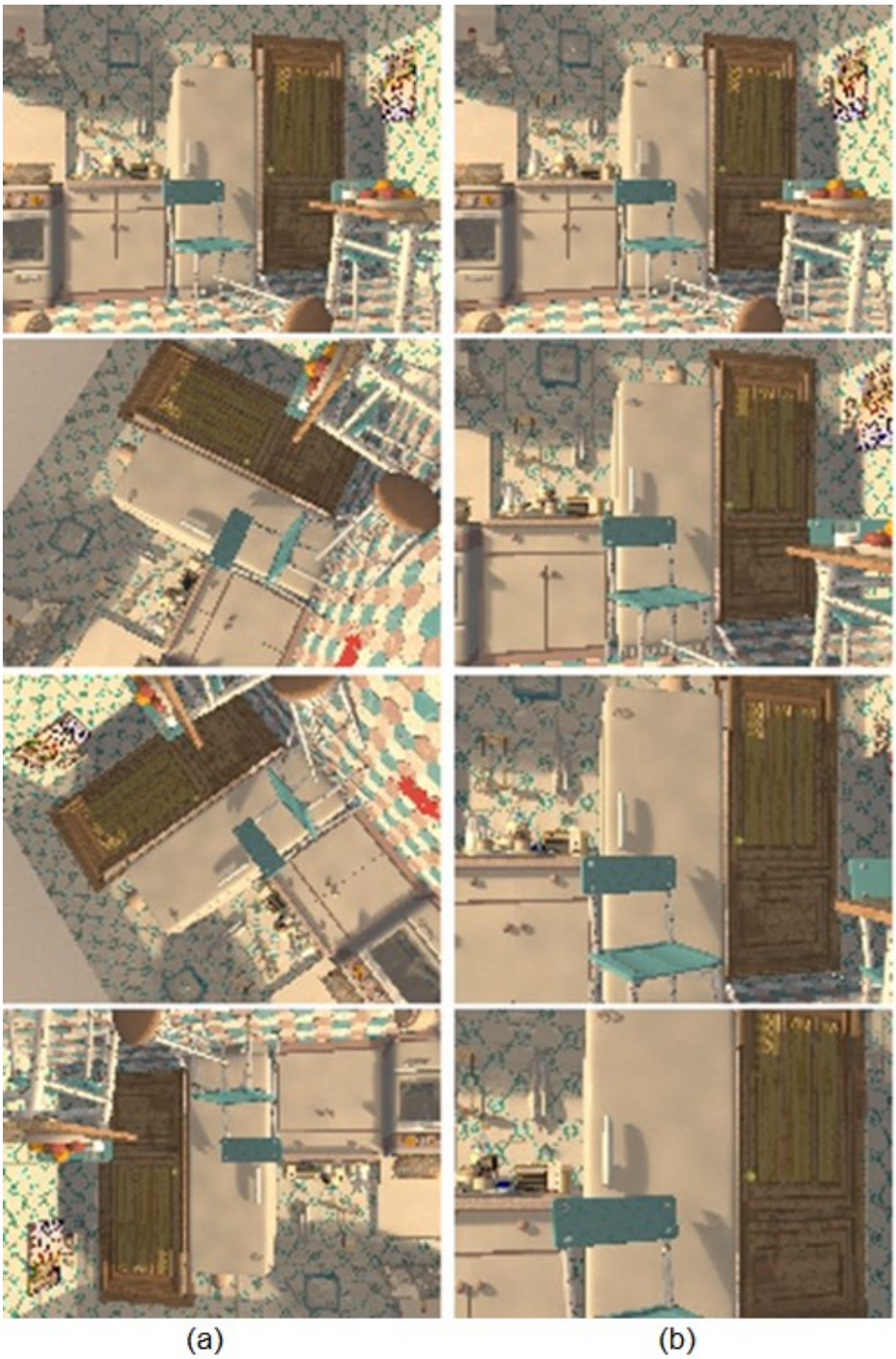


Figure 2.6: A sample of the synthetic images. The four images in column (a) are rotated data and the four images in column (b) depict the scale variation.

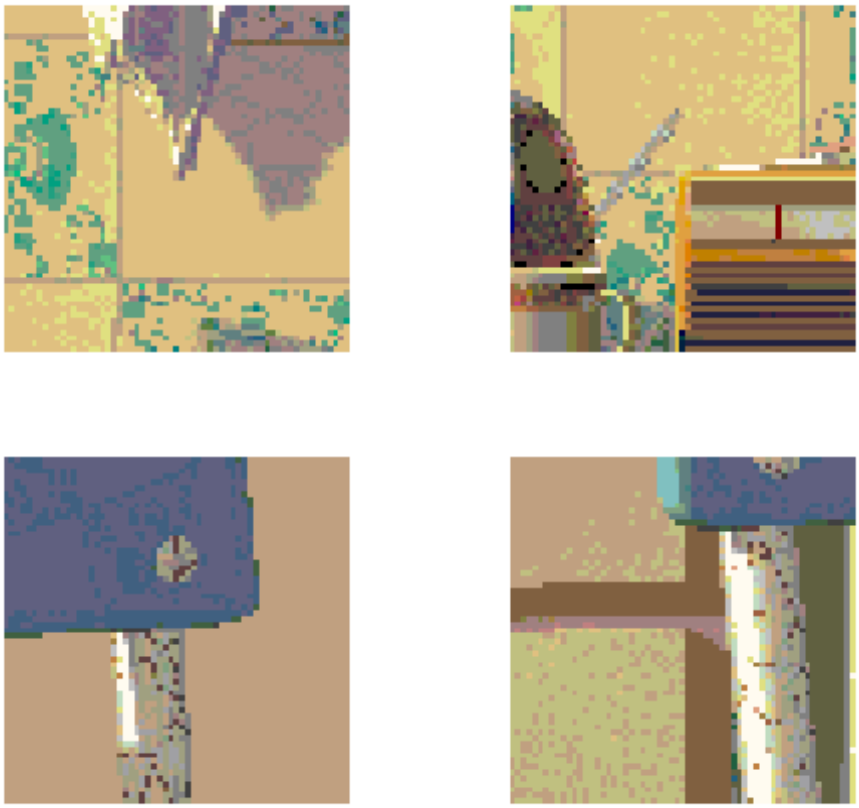


Figure 2.7: A sample of image windows used in the synthetic data experiment.

We selected 10 different points in the first image of the kitchen scene and, using the motion field data, computed the corresponding points in the remaining 20 images, thus, generating the ground truth for our problem. In the same fashion, we generated 19 images of the same scene where the camera moves along the *z*-axis towards the direction of the scene and hence generating images of uniformly increasing scale. Figure 2.6 shows some of these images. The column of images on the left is the rotated data starting from the first image at 0 degrees and ending with the last image at 180 degrees. Similarly, the column on the right of Figure 2.6 shows the synthetic images with increasing scale. The Figure 2.7 shows a window of 61 x 61 pixels centered at points selected by us to test our algorithm.

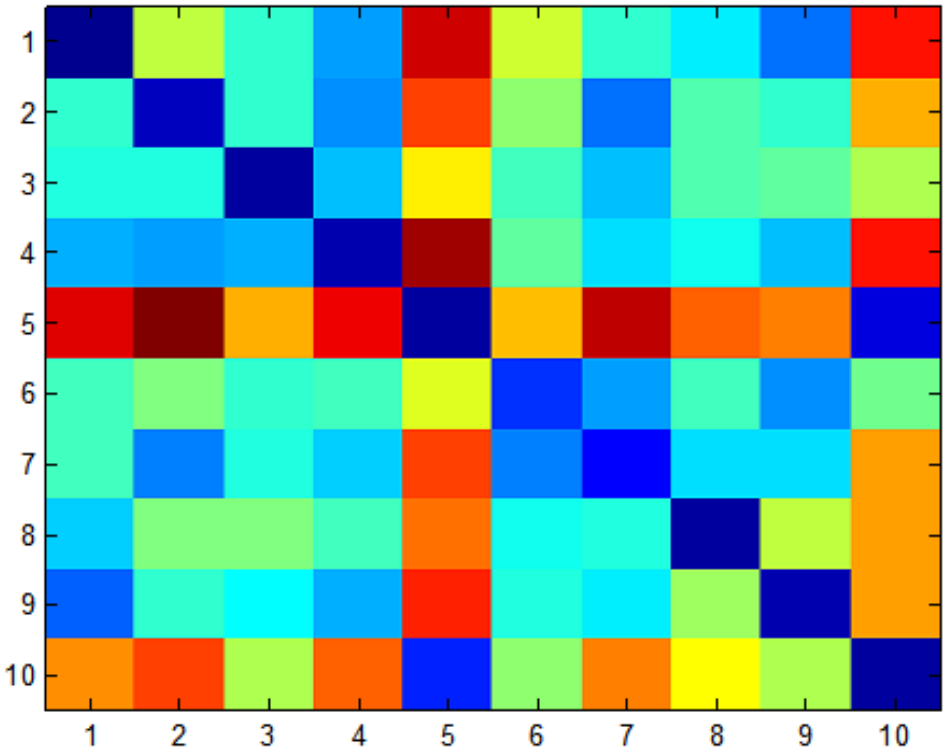


Figure 2.8: A confusion matrix for a pair of synthetic images.

For the rotated synthetic data, we tested the 10 points in the first image against the 10 corresponding points in the remaining 20 rotated images. We computed the confusion matrices and the matching rate as described in Section 2.4. Figure 2.8 shows a confusion matrix for the matching of the 10 points in two images. The algorithm managed to correctly match 187 points out of the 200 points. A matching rate of 93.50% for images between 0 and 180 degrees shows robustness to a high degree of image rotation. In a similar experiment with the scaled synthetic images, the algorithm matched 146 points out of 180 correctly. An 81.11% matching rate shows a good degree of scale invariance.

Figure 2.9: A few real images used in our experiments.

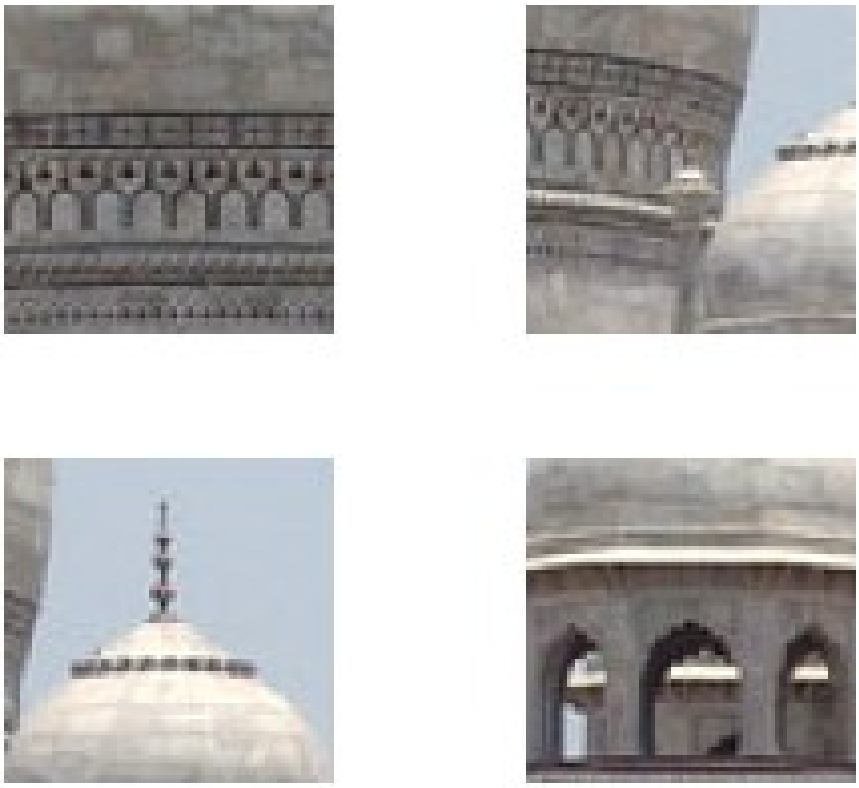


Figure 2.10: Sample of image windows used in the real data experiment.

2.5.3 Real Images

To evaluate our feature descriptor with real world images, we selected images of three different buildings: the south facade of the White House, the front view of St. Peter's Basilica, and the front view of the Taj Mahal. We used 11 different images of each of these scenes such that they are a good assortment of images of different scales, different rotations, varied illumination, and viewpoint changes. Figure 2.9 shows some of the real images used for testing our descriptors. For each of the three scenes, we manually annotated 15 different points in each of the 11 images. Figure 2.10 shows image windows around four of such points chosen in the Taj Mahal scene.

Figure 2.11 shows a confusion matrix for two images of the White House. The matching rates seemed to vary quite a bit for the three data sets. Given 15 points per image and 10 images per building, summing up to a total of 150 points, the algorithm matched 58, 38 and 28 points for the White House, the Taj Mahal and the St Peter's Bascilica respectively. That results in an individual matching rate of 53.33%, 34.44% and 26.11% and an overall matching rate of 41.33%.

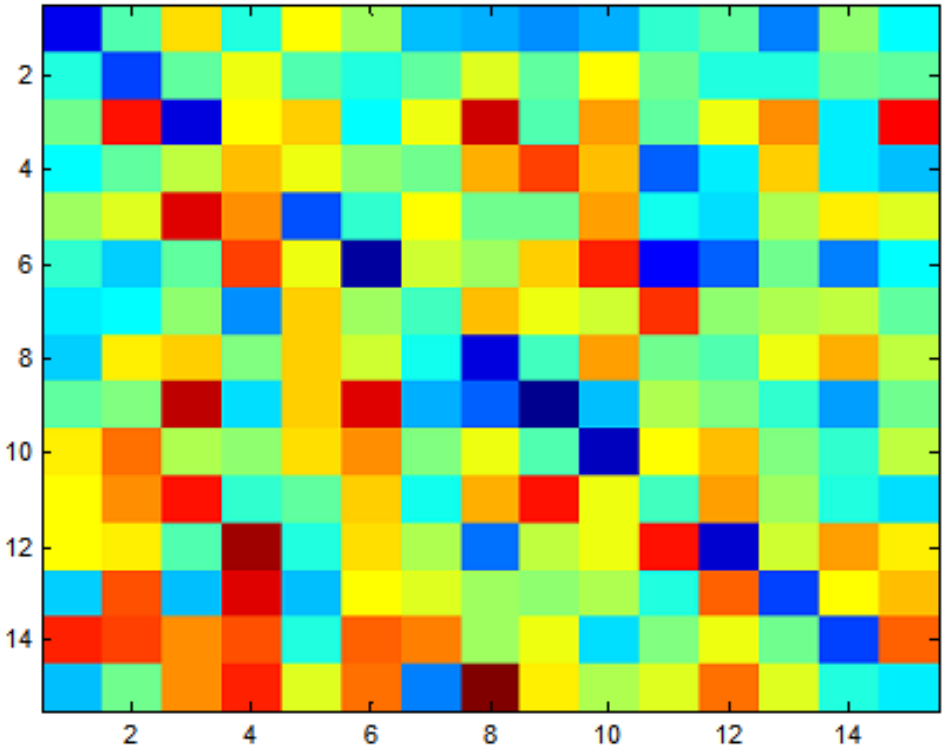


Figure 2.11: Confusion matrix for a pair of real images.

2.6 Conclusions

We proposed a novel approach of computing an invariant image descriptor based on a bank of Gabor filters, called Topological Gabor Descriptors. We take advantage of the simple shifting property of the descriptor to search for similarity in the rotation space. We obtained good results with the synthetic images, an 81.11% matching rate for an image subjected to a scale change of two times the original scale and 93.50% for an image rotated from 0 to 180

degrees. The results with respect to the real images are promising with an average matching rate of 41.33% for three different scenes.

The main bottleneck at the moment is the comparison of the high-dimensional descriptor in a way that reduces the rate of false positives. An area for further research is to design a more robust distance metric designed specifically for matching these time-series.

Another aspect of our approach is that, as long as the computation of the inverse of the filter bank is feasible, we will be able to regenerate the original image information. This way, we may compress a digital image with the aid of such a filter bank and be able to regenerate all the original image information using the inverse of the filter bank.

CHAPTER 3

AN INTERPOLATION BASED APPROACH FOR IMAGE RECOGNITION UNDER LIGHTING VARIATION

3.1 Introduction

Pattern recognition problems such as object recognition and face recognition are fundamental to image processing and computer vision. Over the years, a whole array of techniques and algorithms [11, 45] has surfaced to address the recognition problem. However, the performance of the state-of-the-art solutions fall way short of that of human vision. Many challenges inherent to the problem itself remain to be addressed. *Perceptual alias* refers to variation in pose, lighting, and other conditions [2, 7]. Among these challenges, the variation in illumination conditions, and its effects on the appearance of objects, is an important issue in object and face recognition. Many solutions have been proposed to address this problem alone, where a fixed pose and no occlusions are assumed, but different illumination settings are considered. Several methods address the problem of face recognition under varying illumination [2, 7, 27]. Some algorithms addressing the illumination issue include illumination cone methods [7, 24], spherical harmonic based methods [5, 35], and quotient-image representation [50, 51].

In this chapter, we propose a novel interpolation based approach to model the lighting conditions. This results into robust recognition methods under varying illumination conditions. Given a fixed object/scene in varying illumination conditions, where the parameters are the positions of the light sources and/or their intensity levels, we are able to generate new images within the parametric space and, also to determine the lighting parameters of a query image. Formally, given a set S of n training images and their corresponding parameters, $S = \{(p_1, i_1), (p_2, i_2), \dots (p_n, i_n)\}$, where $p_i \in P$, the parametric space; the interpolation model can be

used to synthesize any query image i_p for a query parameter $p_p \notin S$ within the parametric space, and also to estimate the parameters p_q of a query image $i_q \notin S$.

The main contributions of this chapter are (1) an interpolation based technique that effectively synthesizes an image given its lighting parameters (*e.g.*, pose or intensity of light sources), (2) estimates the lighting parameters for a query image, and (3) performs object recognition and face recognition in varying illumination conditions.

We demonstrate that our interpolation model performs well in object recognition and face recognition applications with fixed pose and varying illumination. We show the synthesis from the parametric space to the image space and analysis from the image space to the parametric space in both cases. We present the results of our experiments performed on two different datasets for face and object recognition, respectively.

For the face recognition problem, we use the Extended Yale B+ [34] face database. The results in face recognition are comparable to that of the top performing algorithms on the same database [35, 56]. Refer to [56] for a comparison of these methods in terms of recognition rate using the Yale face database. Many of these methods achieve a recognition rate of over 96% compared to our 91.92%. However, they all (except [56]) exclude the images under extreme lighting conditions. And the approach of [56] itself dips slightly below the 90% mark while considering the extreme lighting set. Most of the current approaches simply normalize images in terms of illumination to aid recognition. On the other hand, our approach not only models the lighting conditions but also allows going back and forth between the image space and the parametric space. In fact, the reconstructed images are so good that it is almost impossible for the human eye to differentiate between a real image and its regenerated version.

To summarize, with real images of objects shot under varying lighting conditions we obtained a recognition rate of 99.53%. And, with the Yale extended face database considered as the *de facto* standard benchmark [56], we showed a 91.92% recognition rate despite considering all the images under extreme lighting in our test set.

The remaining of this chapter is organized as follows. In Section 3.2, we formally describe our interpolation based approach to the problem. Section 3.3 describes our recognition methods and a detailed description of our experiments and their results. The chapter conclusion follows in Section 3.4.

3.2 Interpolation Approach

Since a set of sample images and their corresponding parameters are given, the core of our approach is to learn the kernel matrix K such that when *K* is multiplied by the parametric matrix *C* built using the parameters $(p_1, p_2, \dots, p_n)$ will result in the corresponding image matrix *I*, which is composed of the individual images $(i_1, i_2, \dots, i_n)$: $K \times C = I$. Essentially, for a particular image i_q associated with parameter p_q, K is a one-to-one mapping from the parametric space to the image space given by the equation $K \times f(p_q) = i_q$, where *f* is an *interpolation function* used to construct the parametric matrix from individual pairs of images and respective parameters. Here, the interpolation function f can in fact be polynomial, trigonometric, logarithmic, or even a combination of these functions.

As an example of an interpolation function, we have a third degree polynomial equation f_h in h variables, where h is the dimensionality of the parametric space. Consider images taken with four lighting parameters $p_i = (w_i, x_i, y_i, z_i)$, so a third degree polynomial equation in four variables is of the form: $f_4(p_i) = w_i^3 + x_i^3 + y_i^3 + z_i^3 + w_i^2 x_i + w_i^2 y_i + w_i^2 z_i + x_i^2 w_i + x_i^2 y_i + x_i^2 z_i + y_i^2 w_i + y_i^2 x_i + y_i^2 z_i + z_i^2 w_i + z_i^2 x_i + z_i^2 y_i + w_i x_i y_i + w_i x_i z_i + w_i y_i z_i + x_i y_i z_i + w_i^2 + x_i^2 + y_i^2 + z_i^2 + w_i x_i + w_i y_i + w_i z_i + x_i y_i + x_i z_i + y_i z_i + w_i + x_i + y_i + z_i + 1$, where $p_i = (w_i, x_i, y_i, z_i)$ is a particular lighting parameter such as the illumination level of four local light sources. The equation above has 34 components excluding the constant. We construct a *component matrix* $C_{l \times n}$ using any *l* individual components of the interpolation function given by $f_h(p_i)$ for $i = 1, \dots, n$. The number *l* of components should be smaller than the number *n* of training images to avoid a rank deficient system. Formally, if the rank of the component matrix C

is *r*, then *l* should be in the range [*r*, *n*]. For example, if we use the four components $\{w_i^3, x_i {y_i}^2, y_i z_i, w_i\}$ (*i.e.*, *l* = 4), we build the matrix $C_{4 \times n}$ of the form:

$$\begin{bmatrix} w_1^3 & w_2^3 & \dots & w_n^3 \\ x_1 y_1^2 & x_2 y_2^2 & \dots & x_n y_n^2 \\ y_1 z_1 & y_2 z_2 & \dots & y_n z_n \\ w_1 & w_2 & \dots & w_n \end{bmatrix}$$

To be able to learn the kernel matrix *K*, we use n training images $(i_1, i_2, i_3, \dots, i_n)$ and define an *image matrix* $I_{m \times n}$, where $m = r \times c \times d$ (r is the number of rows in the image, c is the number of columns, and d is the number of dimensions, which can be 1 or 3 depending whether the images are treated as grayscale or colored, respectively). Therefore, each image i_j is vectorized as a single column-vector of size $r \times c \times d$ and the image matrix *I* is the concatenation of *n* such column vectors.

Given the image matrix *I* and the component matrix *C*, we have a linear system of equations that represents the interpolation of n vectorized images of size m using l components of the interpolation function as $K_{m \times l} \times C_{l \times n} = I_{m \times n}$. Using linear algebra, we can compute the pseudo inverse of the component matrix *C* and multiply it to both sides of the equation to infer the kernel matrix K. Formally, the kernel matrix is obtained as $K_{m \times l} = I_{m \times n} \times C_{l \times n}^{-1}$.

Now that the kernel matrix is learnt, the problem of image synthesis reduces to a simple matrix multiplication. That is, for a given query parameter $p_q = (w_q, x_q, y_q, z_q)$ of an image that is not in the training set of the kernel matrix, we plug in the parameters into the interpolation function f_h to get a vector c^q of *l* components. From our previous example, $c^q = [w_q^3, x_q {y_q}^2, y_q z_q, w_q]^T$. Multiplying the kernel matrix *K* by the component vector c^q gives the synthesized image vector i_q. Formally, the synthesis equation is given by $i_{m \times 1}^q = K_{m \times l} \times c_{l \times 1}^q$.

The analysis part, being the inverse of the synthesis problem, consists of finding the lighting parameter p_q for a given query image i_q. The parameter is obtained by first computing the product of the inverse of the kernel matrix *K* with the query image vector. That is, the

components of the query image are obtained using the matrix equation $c^q_{l\times1} = K^{-1}_{l\times m} \times i^q_{m\times1}$. Having thus obtained the component vector c^q, we find the parameter by solving a linear system of equations. This system is constructed by equating the individual elements of the component vector c^q to the corresponding components in the interpolation function f_h. According to our example above, we have $w_q^3 = c^q(1)$, $x_q {y_q}^2 = c^q(2)$, $y_q z_q = c^q(3)$, and $w_q = c^q(4)$. We take the logarithm on both sides of these equations and solve for the linear system of equations which results in terms of the logarithm parameter of the query image. Considering the used components of the third degree polynomial in our example, the equations in this system are: $3\log w_q = \log c^q(1)$, $\log x_q + 2\log y_q = c^q(2)$, $\log y_q + \log z_q = \log c^q(3)$, and $\log w_q = \log c^q(4)$. Once this system is solved for the logarithm parameter $(\log w_q, \log x_q, \log y_q, \log z_q)$, the actual parameter is found using the exponential function: $(w_q, x_q, y_q, z_q) = (e^{\log w_q}, e^{\log x_q}, e^{\log y_q}, e^{\log z_q})$. Once again, it is important to note that the third degree polynomial equation described in this section is only an example to give a clear picture of the synthesis and analysis methods of our interpolation technique.

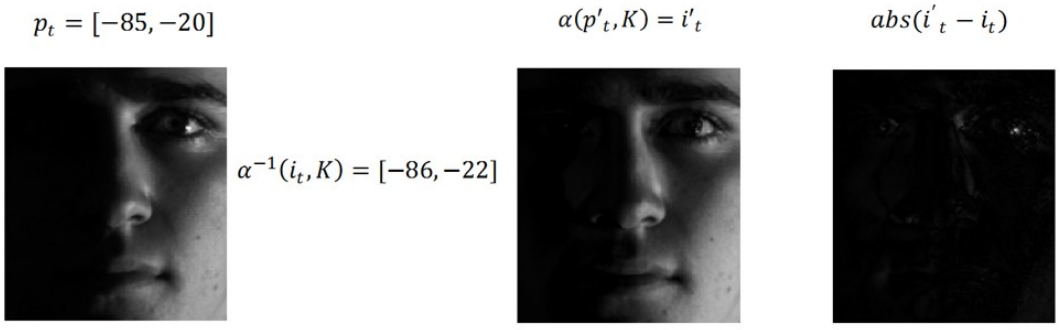


Figure 3.1: A sample recognition step where (a) represents a test face image, (b) is the synthesized image and (c) is the error image.

Figure 3.1 depicts a sample recognition step. For a test image i_t we can estimate its parameters p'_t using the analysis part represented by $p'_t = \alpha^{-1}(i_t, K)$. Once we analyze the test image, we can use the synthesis step to generate a new image with the estimated parameters. This synthesis step can be represented by $\alpha(p'_t, K) = i'_t$, where i'_t is the newly

synthesized image. We then compute the L1 norm between the two images as $abs(i_t' - i_t)$ to measure the image synthesis error. The parameter error is similarly computed as $abs(p_t - p_t')$, where p_t represents the ground truth parameters.

3.3 Recognition Methods

3.3.1 Image Datasets

We captured images of a complex scene with several objects in varying illumination conditions. The idea of having several objects was to create complex shadows on each other when subjected to different lighting situations.

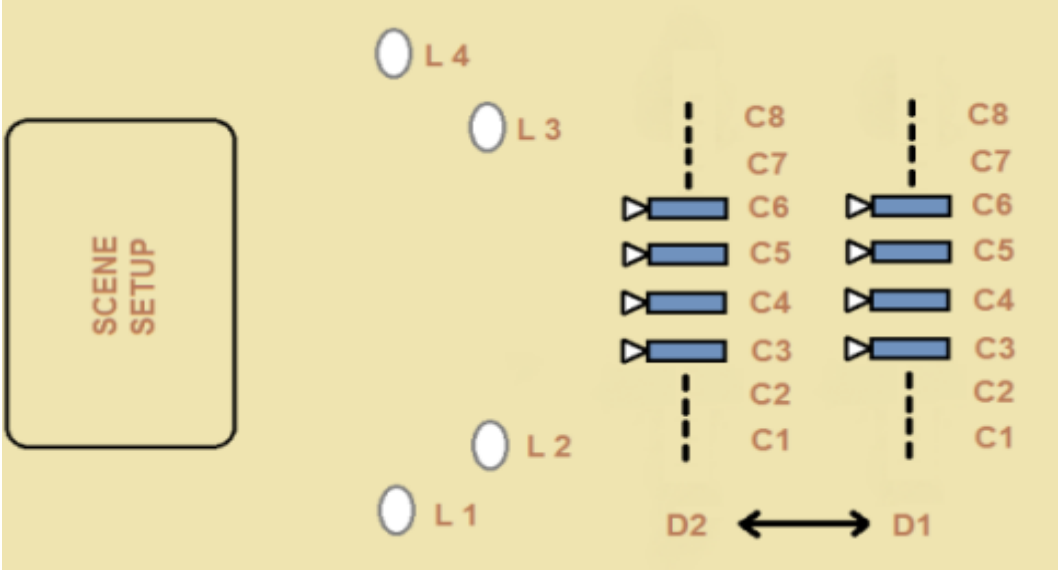


Figure 3.2: Camera setup for data capture.

The camera setup, as illustrated in Figure 3.2, involved eight CCD cameras (Point Grey Flea®2G) mounted with uniform spacing on a straight metal frame. The cameras labeled C1 through C8 were used to capture images of the scene at two depth levels D1 and D2, thereby generating images from sixteen different camera poses.

To generate different lighting conditions, we used four identical incandescent lamps with plain white shades in a fixed spatial configuration as shown in Figure 3.2. The entire setup was in a dark room to avoid outside lighting conditions from interfering with our lighting setup. Each lamp, namely L1, L2, L3 and L4, could be set at three intensity levels with level-1, level-2,

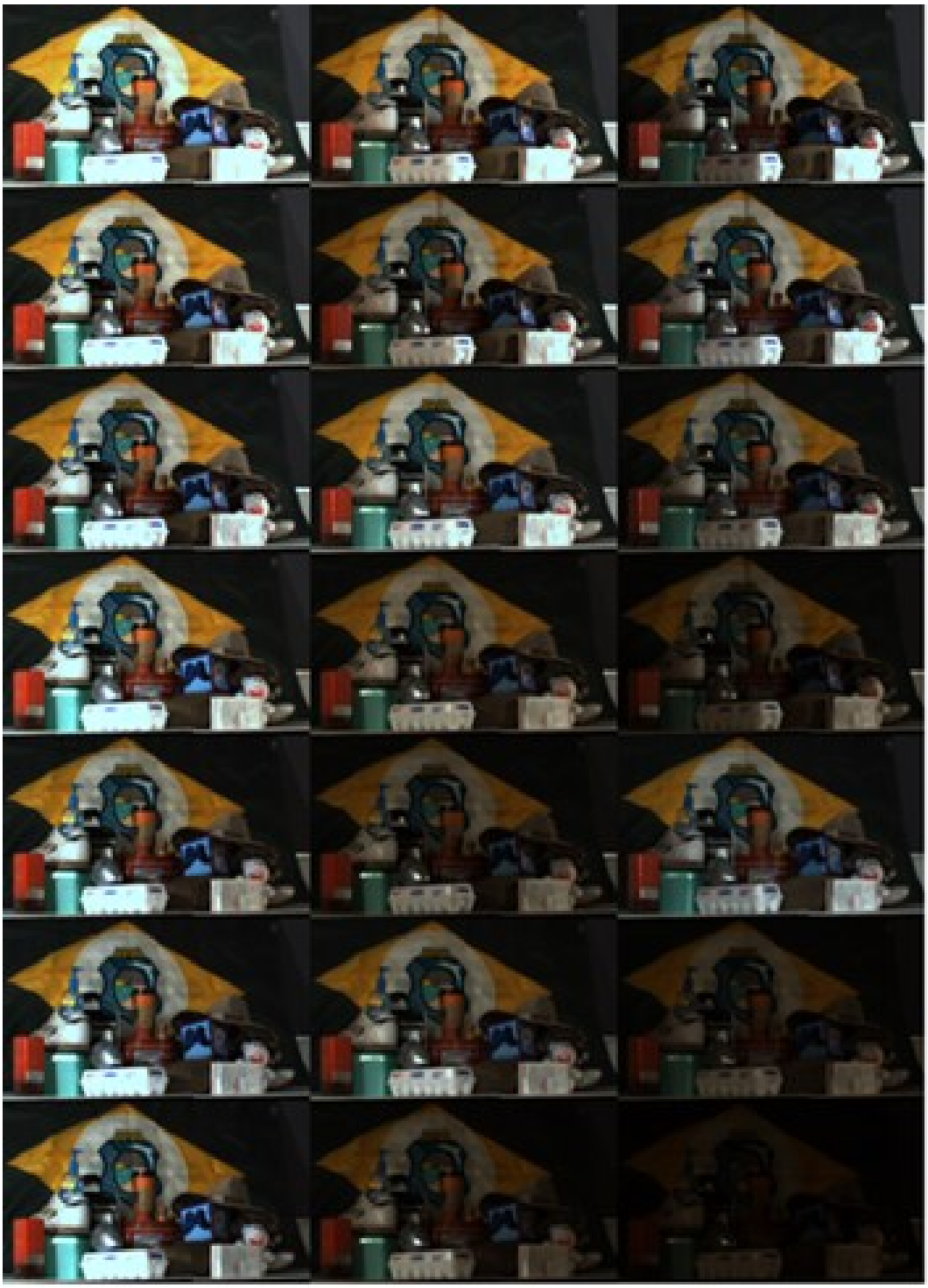


Figure 3.3: Sample images from our object image dataset depicting lighting intensity variation. The lighting parameters vary from (1,1,1,1) through (3,3,3,3) in uniform steps from top-left to bottom right.

and level-3 representing 100%, 75%, and 50% of the lamp total luminosity, respectively. This setup of four lamps with three levels of lighting leads to $3^4 = 81$ different lighting configurations. All images are in RGB color format with a 1280 x 960 resolution. In Figure 3.3, we show the lighting variations of the images taken from camera C1 at depth level D2 where the parameters of the lighting range from (1, 1, 1, 1) to (3, 3, 3, 3).

To consider the variation of light source position while the lighting intensity is constant, we used the Extended Yale Face Database B+ [34]. The database contains face images of 38 subjects in 64 different illumination conditions. The illumination variation was achieved by placing the light sources in a set of different azimuth and elevation angles with respect to the camera axis. The face image database [34] is divided into five subsets based on the position of the light source. The subsets 1, 2, 3, and 4 are comprised of images taken with the light source at an angle whose absolute value is less than 12°, between 12° and 25°, between 25° and 50°, and between 50° and 77°, respectively. Subset 5 consisted of images taken with the angle between the light source and the camera axis greater than 77°. The azimuth angles vary from -130° to 130° and the elevation angles from 0° to 90°. This makes subset 5 the most challenging. In our experiments, we have considered all five subsets for testing in order to subject our methodology to an evaluation under extreme lighting conditions. The only images we discarded were that of the eight subjects with corrupted images and, hence, testing was performed with 30 different subjects instead of 38. Moreover, since we are dealing with face recognition with illumination variations in this chapter, we consider only the frontal pose of the 30 subjects. A set of sample images of three subjects from the 5 subsets are shown in Figure 3.4. All the frontal face images are cropped and aligned as described in [34]. The images are of dimensions 168 x 192 and in grayscale.

Figure 3.4: Images from the face database depicting lighting pose variation. The azimuth and elevation pairs for images in column (a), (b), (c), (d) and (e) are (05, 10), (10, -20), (20, -40), (60, 20), (95, 0) respectively.

3.3.2 Object Recognition with Varying Lighting Intensity

In our evaluation of recognition methods, we use the leave one out strategy to train the kernel matrices. That is, excluding the testing image, all the remaining images in the dataset are used for training.

The images of a scene captured with one of the 16 different camera poses were divided into 15 windows with 15 different objects, respectively. The lighting parameters here are the intensity levels of the four lamps. Since we have 81 different lighting images of the same scene, we end up with 81 lighting images for each of the 15 sub-scenes (objects). Figure 3.4 shows five such windows (rows) under six different lighting parameters (columns). We set up an experiment to test our algorithm's performance in terms of recognizing objects in a fixed pose but varying lighting. We defined 12 neighborhoods with 27 training images each, uniformly distributed in the parametric space. We used only 14 independent components while computing

the kernel matrices. We tested 8 of the 15 objects with 27 of the 81 different lighting conditions each. We were able to accurately recognize 215 out of the 216 test images leading to a recognition rate of 99.53%.

Figure 3.5 Sample objects under six different lighting parameters.

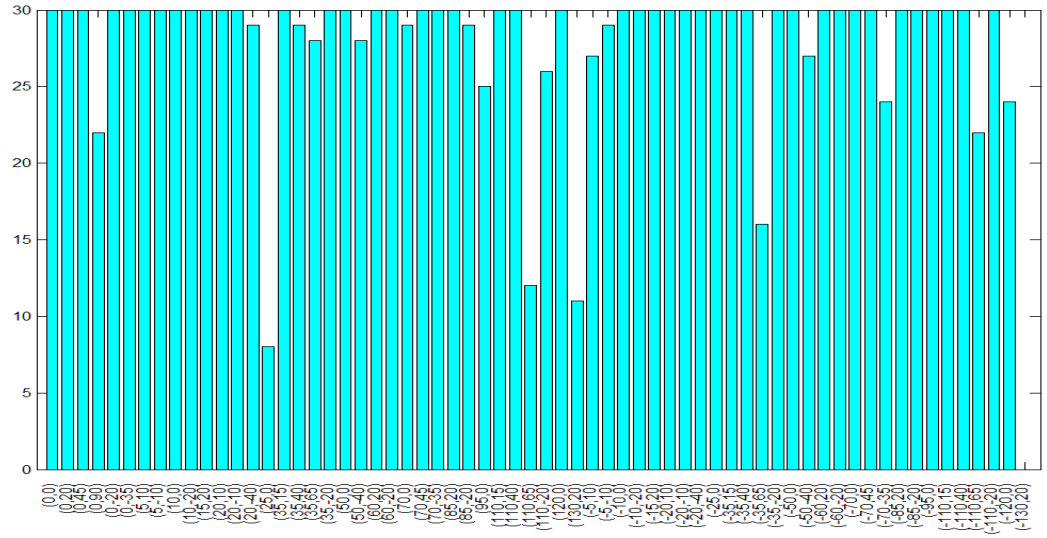


Figure 3.6 Recognition rate for each of the 64 lighting parameters using a total of 30 subjects. The vertical axis represents the number of subjects and the horizontal axis represents the lighting parameters.

3.3.3 Face Recognition with Varying Lighting Pose

For our face recognition experiment, we used the cropped images from the Extended Yale Face Database B+. Out of the 38 subjects we discarded the ones with corrupted images and ended up testing with 30 different subjects. We tested the recognition rate of frontal pose images with 64 different lighting conditions. The lighting parameters here were the azimuth and elevation angles of the light source with respect to the camera axis. We tested every single lighting condition of every single subject (*i.e.*, 30 subjects times 64 lighting conditions for a total of 1920 images). We are able to recognize 1765 images accurately, a recognition rate of 91.92%. Figure 3.4 shows how the algorithm performed individually for each of the 64 different illumination conditions.

3.4 Conclusion

We have presented a novel approach based on interpolation to model illumination conditions. The model allows for synthesizing a new image given the lighting parameters (next chapter describes the synthesis phase in more detail) and also analyzing the lighting parameters given a query image. We have also demonstrated its potential in recognizing objects in varying lighting conditions. The face recognition system based on our interpolation model has shown extremely competitive results in terms of recognition rate. The model is simple to implement and yet, at the same time, expects a very straightforward and intuitive set of parameters that captures the complex nature of lighting variations. In theory, the same approach can be extended to model geometric variations as well. With our object recognition experiment, we showed that a scene can be broken down into sub-scenes and the model can be applied individually to each of the windows. Furthering this technique, we can build on the model to handle occlusion by dividing images into windows and incorporate some kind of a voting scheme. Handling geometry and occlusions based on this approach is currently part of our ongoing research. In the following chapter we elaborate further on the interpolation

approach and use the synthesis-analysis scheme to demonstrate its application in an Image Based Lighting setup.

CHAPTER 4

A SYNTHESIS-AND-ANALYSIS APPROACH TO IMAGE BASED LIGHTING

4.1 Introduction

A usual way to synthesize images of a given scene is to programmatically consider the geometry and the photometric properties of the various virtual objects in the scene to render a synthetic image. This method involves generating a digital image based on a comprehensive model that comprises all the necessary information such as, object geometry, viewpoint, lighting, and texture. Recently, Image Based Modeling and Rendering (IBMR) approaches have been proposed to obtain depth maps from stereo images of a real scene and then to construct a corresponding textured 3D model [21]. The 3D model is re-projected into a different viewpoint as a synthetic 2D image. An important part of this methodology is referred to as Image Based Lighting (IBL) [13]. IBL considers different lighting conditions retrieved from sample images of an object under several local light sources and a lighting representation of a different location to synthesize new images of this object in the different location. Tunwattanapong *et al.* [53] introduced an IBL approach which allows for the editing of the lighting in terms of intensity and angular width of local light sources.

In addition to sample images of the target objects under different lighting conditions, a limitation of current IBL approaches is the requirement for a lighting representation of the specific environment to be considered in the synthesis of new images. This means that the generation of new images requires access to this specific location for the capture of data that leads to the lighting representation. An example of such a lighting representation is the environment map or incident light map. Therefore, IBL techniques cannot be applied to the modification of existing images, when access to the actual location where the images were

captured is not possible. Furthermore, lighting representations are specific to a particular environment and its corresponding lighting conditions at that time. These representations cannot account for the entire range of possible lighting conditions and, consequently, do not allow the generalization necessary to consider lighting variation.

In this chapter, we address the Image Based Lighting problem to advance the state-of-art towards these two directions: independence of lighting representations and generalization to lighting variation. We propose a novel interpolation based approach for the synthesis and analysis of images under lighting variation. Our approach generates images in different lighting conditions (synthesis) as well as infers the parameters of the scene by perceiving objects (analysis), where the lighting parameters are the positions of the light sources and/or their intensity levels. More specifically, given several sample images of a scene in varying illumination conditions, we are able to generate new images within the parametric space and to determine the lighting parameters of unknown query images. Formally, given a set S of n training images of an object under different lighting conditions and their corresponding parameters, $S = \{(p_1, i_1), (p_2, i_2), \dots, (p_n, i_n)\}$, where $p_j \in P$ is the parameter of image i_j in the parametric space *P* for $j = 1, \dots, n$; our interpolation based approach synthesizes any image i_k for a given parameter p_k within the parametric space such that $(p_k, i_k) \notin S$. Conversely, our approach is also able to estimate the parameter p_q of a given query image i_q such that $(p_q, i_q) \notin S$.

Our approach is motivated by the mirror neuron theory [48] in Neuroscience that essentially places perception and generation under the same foundation. The mirror neuron theory states that the same neurons fire when a person perceives a particular sensory-motor pattern and when the subject generates the same pattern. For an example in the visual domain, the theory claims that a set of neurons in the brain will be active when a subject recognizes the image of an object such as an apple fruit. Similarly, the same set of neurons fire when the subject pictures the image of an apple fruit through imagination or dreaming. This theory

indicates that both synthesis and analysis are performed according to the same fundamental framework. We propose here an IBL approach, inspired by the mirror neuron theory, to perform the synthesis and analysis of object images under general lighting conditions.

Without any previous knowledge on the lighting models, our approach relies on the lighting parameters of the target environment. These parameters are either given or estimated by using known objects in a target image of the environment. The analysis component of our approach detects known objects in this image and computes the parameters from the respective image regions associated with these objects. Once parameters are obtained, the synthesis of new object images for the target environment is performed according to these parameters. Using this approach, we are independent of lighting models by assuming our training sample images include images of at least one object in the target image. This way, instead of having access to that particular location, we only need to capture sample images of an object in the image. Most likely you may not be able to have access to the white house in 1945 but you may have access to the president's desk at the time. Our framework enables the editing of existing images and provides an alternative to image based lighting when the construction of lighting models is unfeasible.

In other words, we propose a framework where an object can be rendered in any scene (and respective lighting condition) as long as a known object (*i.e.*, an object whose interpolation model is obtained previously) exists in that particular scene. Formally, let us define a dictionary *D* of *k* pairs of objects in a known object set *O* and their corresponding interpolation models in a model set *M* as follows: $D = \{(o_1, m_1), (o_2, m_2), \dots, (o_k, m_k)\}$, where m_j is the interpolation model of a known object o_j for $j = 1, \dots, k$. Now, given an image i_q of an unknown scene (under an unknown illumination condition) that contains a known object $o_r \in O$, we can compute the parameter p_q using the model m_r of the reference object o_r. Note that the parameter estimation may use more than one known object in the scene for a more robust outcome. The parameter p_q represents the lighting conditions for the image i_q of the considered scene. This parameter is

used to render any known object according to the lighting conditions of the target image i_q. Figure 4.1 shows a schematic drawing of this analysis-and-synthesis framework.

Besides eliminating the dependence on lighting representations, our approach is able to generalize the given samples to generate any possible new image associated with a parameter in the parametric space. In this chapter, we consider sample images of objects under several local light sources. From the sample data, our method generalizes to light sources at different positions and with different intensities.

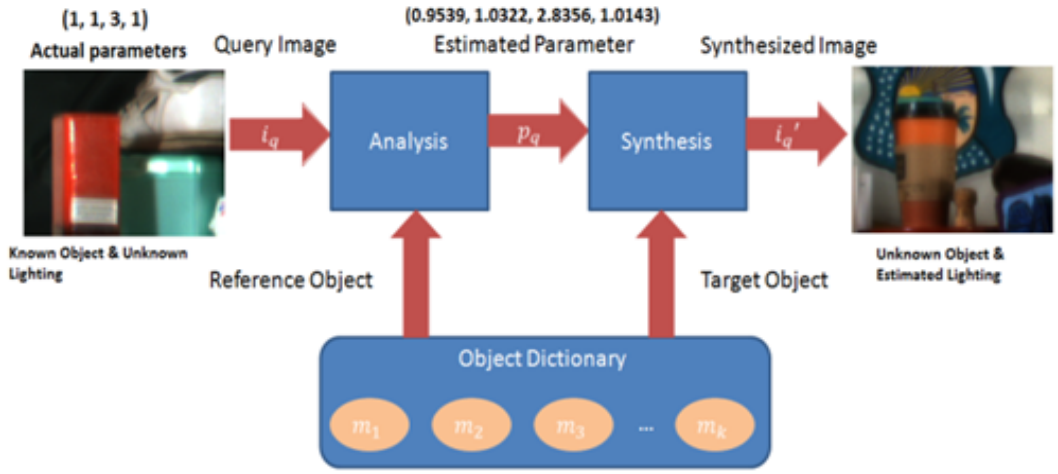


Figure 4.1: A general schematic of our interpolation based approach for the synthesis and analysis of images under lighting variation. The query image $\boldsymbol{i_q}$ has unknown parameters and at least one known object. This reference object is used in the analysis component to estimate the imaging parameter $\boldsymbol{p_q}$. Once the parameter is found, our synthesis component generates the output $\boldsymbol{i_q}'$ as the rendering of an unknown object in the query image according to the estimated lighting parameters.

We demonstrate that our interpolation based approach performs well in the synthesis from the parametric space to the image space and in the analysis from the image space to the parametric space with images taken under fixed viewpoint and varying illumination. We present experimental results with two image datasets: a *face image dataset* captured with same intensity light sources at different positions and an *object image dataset* captured with fixed pose light sources at different intensities. The synthesized images are compared to ground-truth images using a leave-one-out testing methodology. The average image synthesis errors

computed as the sum of the absolute difference were 1.156 and 8.188 for the object image dataset and the face image dataset, respectively. The average parameter estimation error for the object image dataset was 0.101 for intensity parameters varying between 1.0 and 3.0. For the face dataset, the average estimation error was 8.836 for a parameter range of [-130°, 130°] in azimuth and [-40°, 90°] in elevation. It should be noted that the higher errors with the face image dataset are due to several images with extreme lighting conditions.

The remaining of this chapter is organized as follows. In section 4.2, we describe previous work in areas related to image synthesis. Section 4.3 introduces our interpolation based approach for image generation and image perception. A detailed description of our experiments and their results are presented in Section 4.4 and our conclusions follow in Section 4.5.

4.2 Previous Work

In computer graphics, photorealistic images have been traditionally created using ray tracing techniques. These techniques rely on artificial scenes created with an exhaustive specification of the physics of the scene such as object geometry, lighting conditions, camera model, and surface texture. In this case, changing the lighting is as trivial as changing the illumination setup in the scene description file. However, this cannot be applied to real images of real objects and scenes. Moreover, the rendering part of the ray tracing engine is time consuming, especially with complex scenes. The scene description itself can get extremely complex and demand a significant amount of artistic skills [13, 53].

Most of the issues related to ray tracing techniques are overcome by Image Based Rendering (IBR) techniques. In IBR techniques, the complex task of scene/geometry description is bypassed. The scene itself is learned either by completely ignoring geometry and computing the light field with the plenoptic function [36, 34, 40] or by constructing the 3D geometry of the scene from a sparse set of images [55]. Shum *et al.* [52] presented a survey of the IBR techniques. The problem with the light field approach is that it requires many images of the

scene acquired by calibrated camera arrays. On the other hand, the issue with the geometry based approaches is the need for computation of depth information. A major issue in most of these techniques is that the rendering is possible only for a fixed viewpoint or a single illumination condition. The geometry based approach is slightly more flexible in terms of rendering images with slightly different views or illumination settings.

Over the years, several image based algorithms have surfaced to manipulate the lighting conditions of an existing real or synthetic image. These techniques, commonly known as Image Based Relighting (IBRL), address the problem of relighting scenes or objects with complex real world illumination. These methods have met with a lot of success and have gained popularity by producing visually stunning scenes. IBRL techniques can be broadly classified into plenoptic function-based, basis function-based, and reflectance-based categories [13]. Most of these approaches use a High Dynamic Range (HDR) image as a light source for rendering different scenes. These HDR images are usually termed as incident light map or environment map.

A plenoptic function is a 7D function that models the 3D dynamic environment by observing the light rays at all possible spatial locations [1]. Usually, the time component of the function is ignored to reduce the dimensions of the function. Light sources with known plenoptic functions can simply be added or subtracted to achieve effective relighting. However, this method is data intensive and requires many images acquired from calibrated camera arrays.

Basis function-based relighting techniques render new images by computing a linear combination of a set of pre-rendered images called basis images. The basis images are chosen in such a way that the combination of a small number of them can simulate images under varying illumination conditions. Nimeroff *et al.* [44] demonstrated this concept of generating new images simulating different illuminations using a linear combination of weighted basis images derived using steerable functions.

Debevec *et al.* [19, 20] acquire the reflectance field of a human face using a setup called ‘light stage’. The light stage allows them to capture still facial data under a small set of viewpoints and dense (2048) incident illumination directions. The reflectance function is constructed for each pixel over space of incident illumination directions. Relighting is achieved using mirrored ball images as light sources. This method achieves realistic and visually appealing results. However, the approach requires a light stage setup for data capture and huge amounts of data to be captured and stored. Moreover, inclusion of new data or editing existing data requires a significant effort.

Most of the previous approaches, as discussed above, are data intensive. Tunwattanapong *et al.* [53] significantly reduce the number of images required to achieve good quality relighting. They use a combination of low frequency spherical harmonics lighting to simulate real world lighting and a set of local lighting to compute a suboptimal residual environment map. They then subject the map to an optimization procedure to generate a close approximation to a given environmental illumination. This method produces visually appealing results despite significantly reducing the data requirements. However, rendering new lighting requires access to the very location. Moreover, having to render a scene with a different lighting setup would require a new environment map. Although changes can be made to an existing image based on the local lights, a capability to render all possible illumination settings given a set of lighting parameters is still missing. Our approach, on the other hand, is capable of rendering any possible illumination settings in the parametric space. The interpolation based model discussed here can work as a standalone application as long as we can have the lighting parameters. If not, it can certainly aid other techniques such as that of [53] to broaden its scope of problems they can address.

Another approach addressing image relighting or retexturing is known as image decomposition [4]. This approach decomposes an image into an illumination component (shading) and a reflectance component (albedo). Once reflectance and illumination have been

decoupled, the scene can simply be relighted with a new illumination map. Bousseau *et al*. [10] decompose a single image into intrinsic images with the aid of user inputs to disambiguate illumination from reflectance. Their approach mostly addresses issues related to user based image editing.

Malzbender *et al*. [41] proposed an image based approach that builds polynomial texture maps based on a quadratic polynomial. They consider images of a static object with a static camera under varying lighting conditions and use a quadratic polynomial to interpolate between these images. They construct a texture map that reproduces the effects of variations in the illumination direction relative to the object. With regards to the synthesis part of our algorithm, the polynomial texture maps are constrained to a quadratic model whereas our approach considers any component function (*e.g.*, polynomials, exponentials, logarithmic, trigonometric) as long as it is invertible to be used in the analysis part of our method. Furthermore, our approach may use any number of component functions. Another major difference is that our method employs the subdivision of the parametric space into neighborhoods. This not only allows for a better modeling of local features but can also significantly speed up the synthesis process. In addition to the image based synthesis, another original contribution of our method concerns the inverse problem where the lighting parameters are inferred from images. This analysis part of our method is what enables the computation of the illumination condition from a sample image. As a consequence, our interpolation based approach may infer the lighting parameters of an unknown scene to render new objects according to the same lighting parameters of this scene.

Drew *et al*. [22] introduced a more robust version of the polynomial texture maps described in [41]. This method also models the contribution of specular and shadow pixels using a radial basis function based interpolation. The method although much more robust to specular and shadow outliers needs to employ a minimum number of light sources that is more than twice as many observations as the number of variables. To be more specific, if they employ 6-D

regression coefficients, a minimum of 13 light sources are required to build the model. On the other hand, the image synthesis part of our approach is a more general model that is versatile in terms of the input parameters and does not mandate a specific number of light sources or illumination levels for that matter.

Matsushita *et al*. [42] interpolate lighting appearance of a scene with sparsely sampled lighting conditions. They use a number of lightfields, each captured under different illumination conditions. Depth maps are computed using a multi-view stereo algorithm on these lightfield images. In addition to the depth maps, the lightfields are decomposed to intrinsic images. This enables them to use both geometry and intrinsic images for view reconstruction and hence synthesize with more accurate lighting interpolation. The major advantage of this method is the ability to synthesize images with significant realism using a sparse dataset. However, the method requires a precision controlled camera grid to effectively capture the lightfields. Moreover, the final synthesized image relies on the choice and performance of the multiview stereo algorithm and the image decomposition method. In contrast, our image synthesis method requires only the position of the light source and/or its illumination level. We are also able to demonstrate good quality image synthesis with a reasonable sampling.

4.3 Interpolation Based Approach

Given a set of n sample images of an object under different given lighting conditions described in terms of a parametric space, we build an interpolation model based on the decomposition of an image matrix I into a kernel matrix K and a parametric matrix C, where K represents the intrinsic features of the object (independent of imaging parameters such as lighting conditions) and C fully embeds the influence of the imaging parameters such as the position of local light sources and their respective intensities. Different from dimensionality reduction techniques such as Component Analysis (*e.g.*, PCA [28], ICA [14]), our decomposition explicitly models the imaging conditions in terms of parameters and, consequently, represents

these conditions in a manner that leads to a structure modeling the imaging process more closely. A detailed description of the interpolation based approach can be found in Section 3.2.

4.4 Experimental Results

In this section, we used to demonstrate the image synthesis and analysis capabilities of our interpolation based approach. We use the object image dataset described in Section 3.3.1 to consider the variation of light intensity when local light sources are fixed. We also use the face image dataset described in Section 3.3.1 to consider the variation of the position of light sources under constant intensity. Besides evaluating our approach with regards to the variation of lighting intensity and light source pose, we will also discuss the impact of the number of components and the neighborhood size on the performance of our method. Finally, we demonstrate our approach with regards to synthesis-and-analysis and discuss how they can be used together to render objects into an unknown scene with different lighting given one known object in the scene.

4.4.1 Image Synthesis and Analysis

For the image synthesis problem, we construct the kernel matrix *K* by using only the images associated with nearest neighbors (in parametric space) to the query parameter p_q. We also select only a fraction of the original components of the interpolation function that are independent of each other to learn our model.

The performance of the kernel matrix in terms of accurate image synthesis or analysis depends on the following: (a) the way in which neighborhoods are constructed in the parametric space, (b) the size *k* of the neighborhood and, (c) the number *l* of components in the interpolation function used to learn the kernel matrix. To investigate the effects of neighborhood construction to our approach, we performed a series of experiments using the k nearest neighbors of the query image parameters according to Euclidian distance in the parametric space. To address the behavior of our approach as the number of used components varies, we consider 27 different numbers of components (values of l) from 1 to 79 incrementing in steps of

3. For each l value, we varied neighborhood size k starting from $l + 1$ to 80, again incrementing in steps of 3. Note that when *l* > *k*, we have an undetermined system and, hence, we avoid computing the errors in that region. For each different *k* and *l* values, we performed the synthesis of 15 random test images from the total of 81 images corresponding to different illumination parameters. We used the leave- one-out strategy for training, where the test image is left out of the training set used to build the interpolation model.

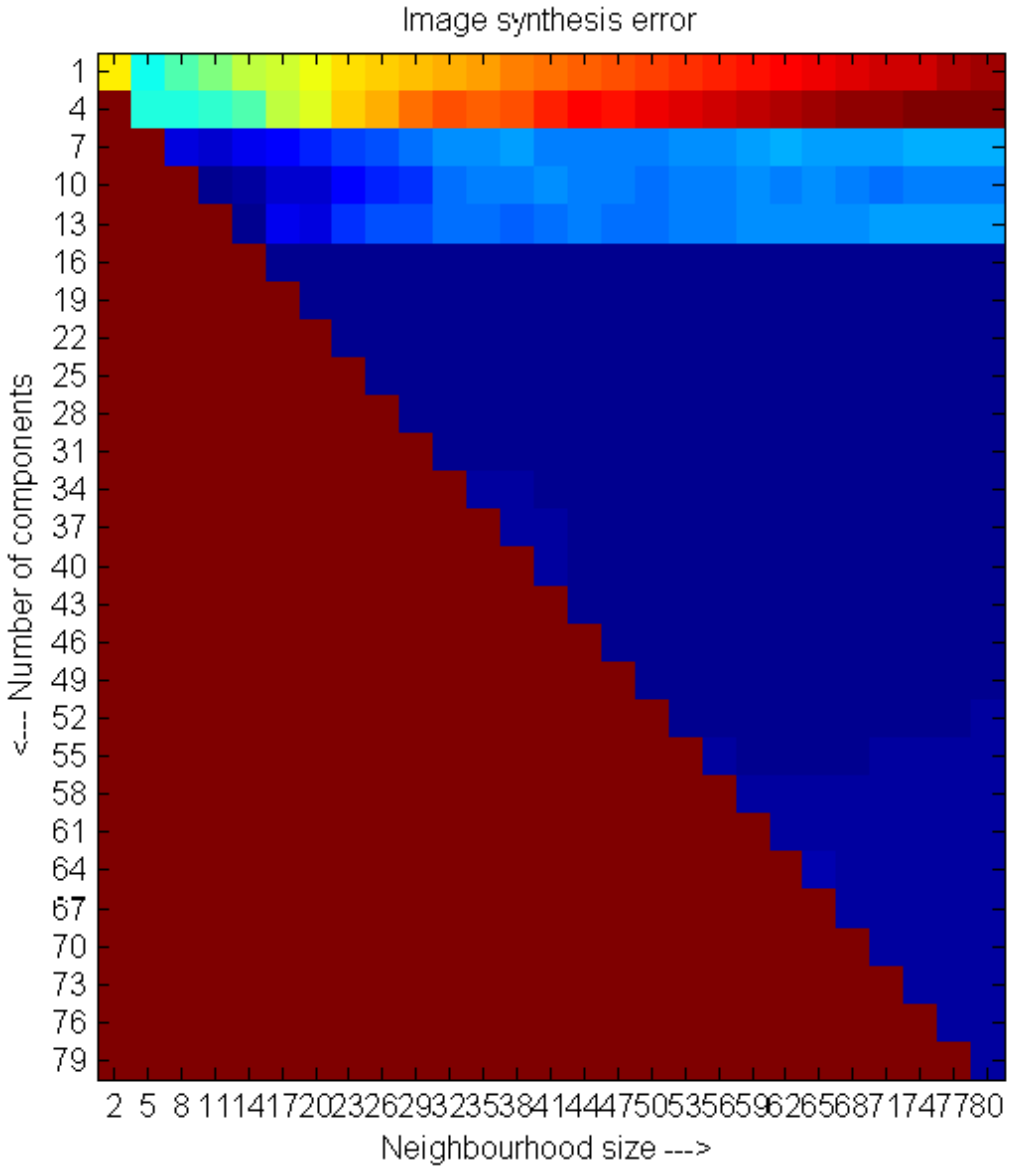


Figure 4.2: The average synthesis error for all possible neighborhood sizes and number of components.

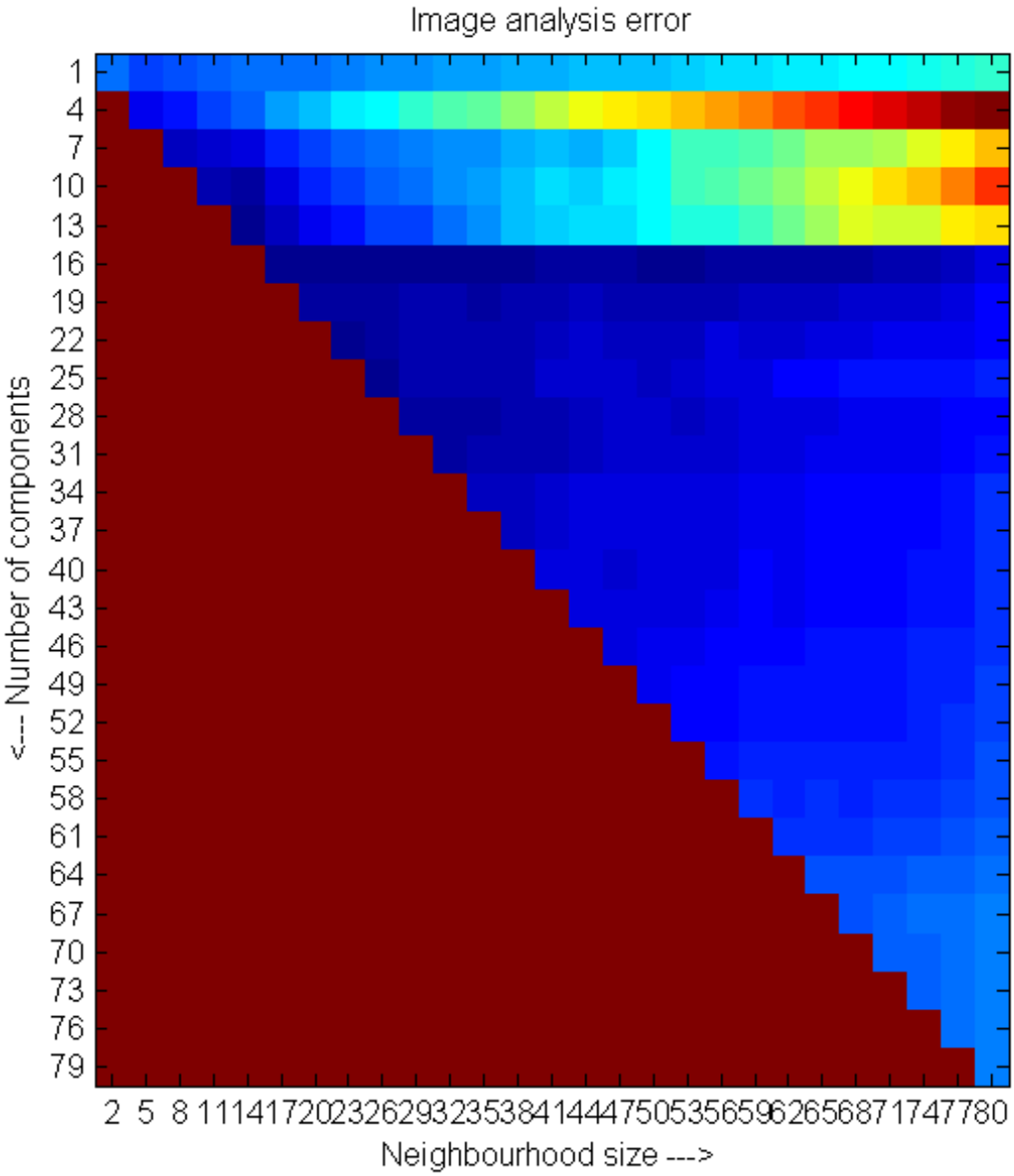


Figure 4.3: The average analysis error for all possible neighborhood sizes and number of components.

For each of the 15 test images, the synthesis error was computed as the sum of absolute differences between the synthesized image and the test image. The errors thus computed are divided by the dimensions of the image to get the error value in the pixel range [0, 255]. The overall synthesis error for a particular neighborhood size *k* and number *l* of used components is the average error for all 15 test images.

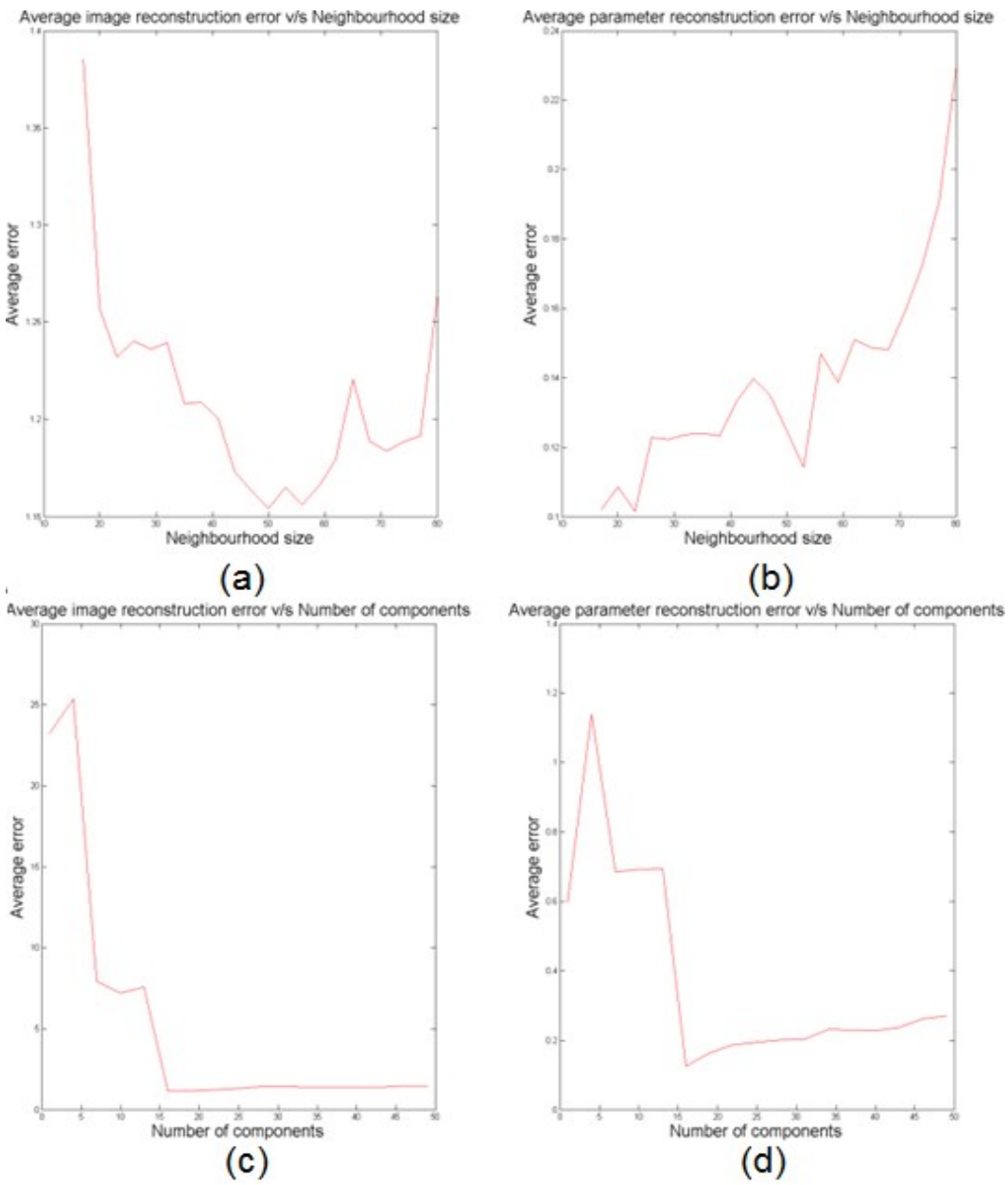


Figure 4.4: Average synthesis error and analysis error for constant number of components with increasing neighborhood size are shown in (a) and (b) respectively and the average synthesis error and analysis error for a constant neighborhood size with increasing number of components are shown in (c) and (d) respectively.

The two error matrices for synthesis and analysis are shown in Figure 4.2 and 4.3. The error values are depicted using a color scheme where smaller values correspond to colder

colors (*i.e.*, blue) and larger values are associated with warmer colors (*i.e.*, red). From this experiment, we learnt that the image synthesis error ranges from 1.154 to 28.673 for all neighborhood size and all possible numbers of components. The image synthesis error is a minimum at 1.154 for a neighborhood size k = 50 and number of components l = 16. Similarly, the analysis error is computed for the same set of k and l values and for the same 15 testing images. The analysis error is computed as the sum of the absolute differences between the estimated parameters and the known test parameters. The minimum parameter estimation error was a 0.101 for k = 23 and l = 16. The maximum average error was 1.694. From Figure 4.2 and 4.3, we can easily infer that the image synthesis error is better behaved in comparison to the parameter estimation error. However, the parameter estimation error was found to be close to a minimum for most values of k when l = 16. For example, the error is 0.124 for l = 16 and k = 50. Given this situation, we would prefer to set a larger k for a given l to achieve a more compressed model. In other words, choosing k = 50 rather than k = 23 for l = 16 will result in a more compact representation and yet not having compromised much in terms of the analysis accuracy.

The plots in Figure 4.4(a) and Figure 4.4(b) respectively show the average synthesis error and average analysis error for a constant number of components but increasing neighborhood size. The number of components was fixed at 16 for this experiment. In the synthesis error, a clear valley was observed. This shows that over-fitting occurs for smaller neighborhood sizes and then the average error decreases to a minimum of 1.154. After that, the average error increases suggesting the generalization phenomenon. The image analysis error shows a different trend where the average error starts at a small value but increases almost exponentially with increasing neighborhood size.

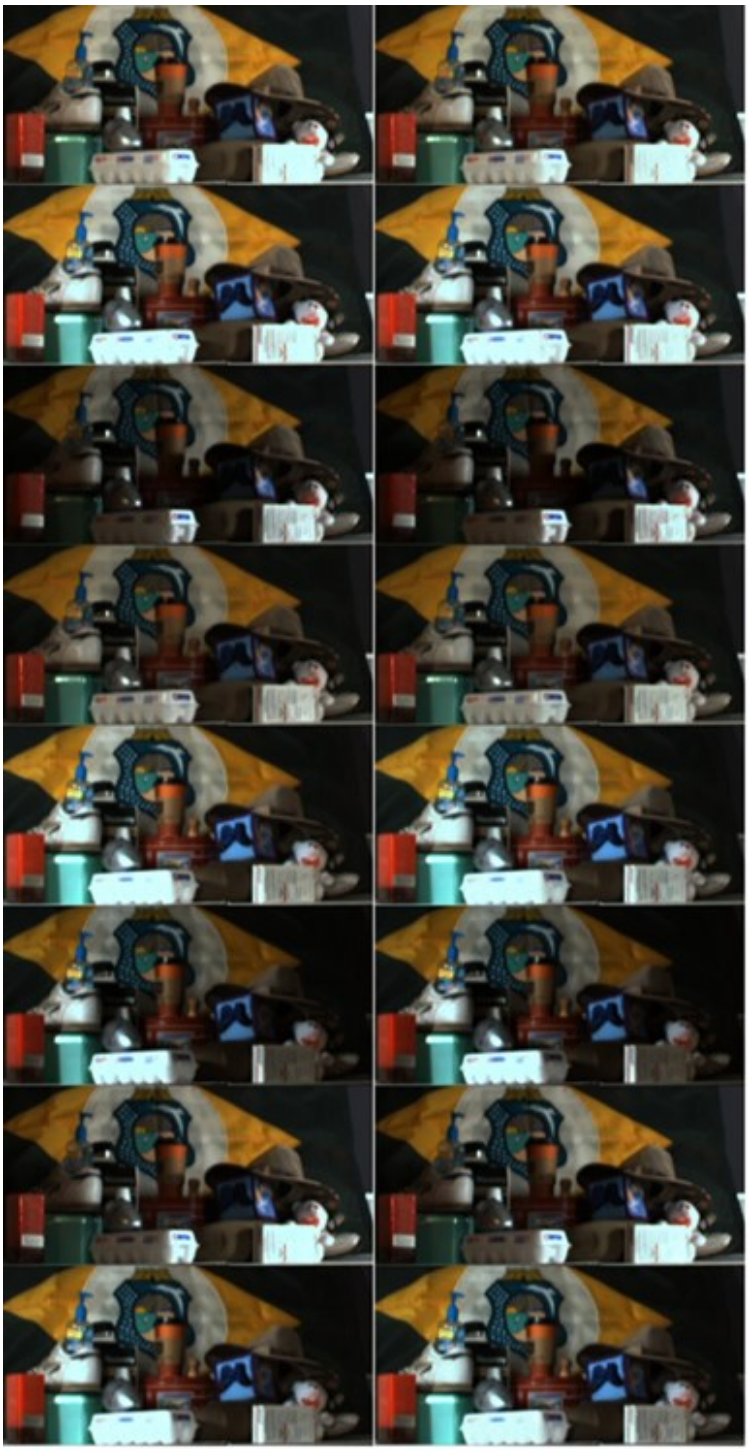
Similarly, the plots in Figure 4.4(c) and Figure 4.4(d) in show the behavior of the average error while increasing the number of components with a constant neighborhood size respectively. The neighborhood size in this case was fixed at 50. From the plots for both

analysis and synthesis, the average error improves with the increase in the number of components. The error reaches a minimum at 16 components and then shows a saturation which suggests that adding more components further will not produce significant improvement in error. These experiments form the basis for choosing and fine tuning the neighborhood size and an appropriate number of components to address the synthesis and analysis parts for a given dataset.

Figure 4.5(b) shows eight synthesized images with different lighting parameters compared to the corresponding ground-truth images in Figure 4.5(a). The synthesized images are almost identical to that of the ground truth. Moreover, on keen observation, one can see that the intricate details of the shadow formations are very well preserved in the generated images. This experiment using the object image dataset was set up with k = 50 and l = 16 for which the average image synthesis error was 1.154 and the average analysis error was 0.124.

To evaluate our approach with regards to the variation of light source position with constant intensity, we tested our interpolation based approach on the Extended Yale Face Database B+ images [34]. We used the front pose face images with 64 different illumination settings, including some extreme light source angles. We have used our method to address the face recognition problem with an illumination invariant approach. We obtained results with an overall recognition rate of 91.92%. In this chapter, we demonstrate the capabilities of our interpolation based technique as a synthesis and analysis tool.

We used 10 independent components of the interpolation function for the generation of face images. A sample set of face images generated from models built based on this dataset are shown in Figure 4.6. The average image synthesis error was 8.188 for the face image dataset. The face images are reproduced well in terms of shape. However, the specular reflections seem softened. This is the result of the interpolation function being only an approximate fit to the sample data. Since specularity is a highly localized lighting effect, a dense sampling is necessary to address this issue while modeling specular objects. However, using a



(a) (b)

Figure 4.5: Image synthesis results with 16 components and neighborhoods of size 50. The column (a) is the original images from camera C4 and depth D2. The column (b) comprises of the corresponding generated images.

Figure 4.6: Real face images (a) and (c) compared to synthesized images (b) and (d).

reasonable sampling level, our method performs remarkably well in reproducing the images under different illumination conditions along with the complex self-shadows.

As an inverse process, we demonstrate the models ability in finding the locations of the light sources. Figure 4.7 shows the estimated lighting parameters for face images of a single subject for all the 64 light sources. The blue and red dots show the actual light source positions and the black asterisk marks show the estimated positions. Red dots mean estimated parameters whose distance to the actual positions is greater than 2.5 degrees while the blue

ones indicate that our method estimated the lighting parameters with an error less than the threshold of 2.5 degrees. The average parameter estimation error was 8.836 for a parameter range of [-130°, 130°] in azimuth and [-40°, 90°] in elevation

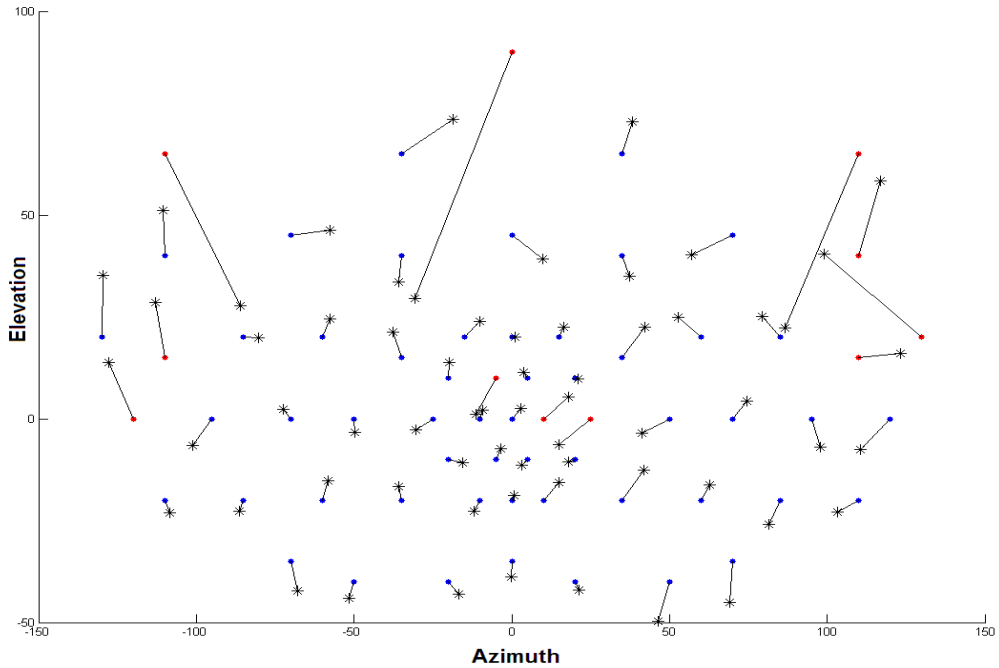


Figure 4.7: Analyzing face images for their illumination parameters.

4.4.2 Rendering a Known Object in an Unknown Scene

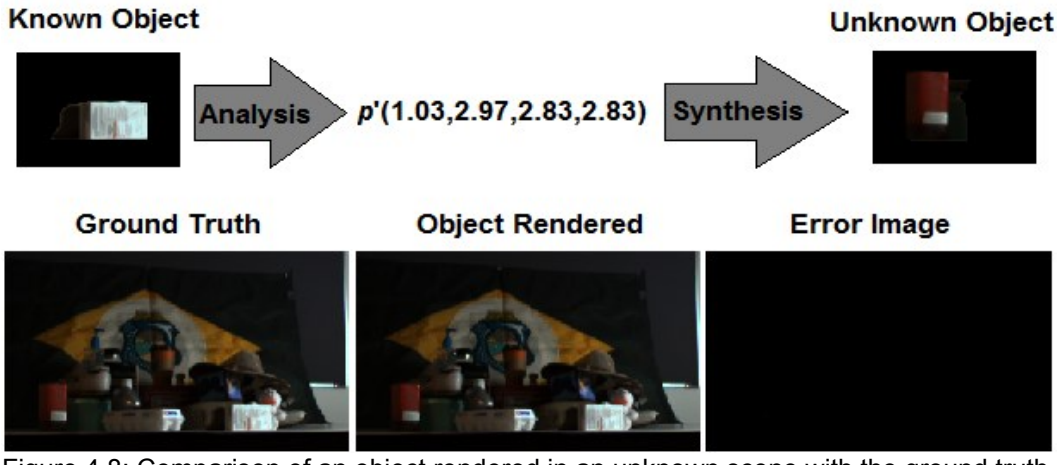


Figure 4.8: Comparison of an object rendered in an unknown scene with the ground truth.

To demonstrate our method's capability of rendering an object in an unknown scene, we infer the lighting parameters of the unknown scene using the pixels associated with a known object in the scene. Once the lighting parameters are inferred, we can now render a new object according to the illumination conditions of the unknown scene by using the interpolation model of the new object. Figure 4.8 depicts the actual rendering of an unknown object rendered into the scene.

Figure 4.9 shows a few sample renderings to demonstrate the progression of the object with different lighting parameters being rendered into a scene. A red bounding box is drawn when the parameters of the rendered object matches the parameters of the scene.

4.5 Conclusion

We have presented a novel application of our interpolation based approach which allows for synthesizing a new image given the lighting parameters and also analyzing the lighting-parameters given a query image. The model is simple to implement and yet, at the same time, expects a very straightforward and intuitive set of parameters that captures the complex nature of lighting variations. The model is not only a compact representation of all possible images in the given parametric space but it is also capable of reconstructing extremely realistic images. Moreover, the same approach can be extended to model geometric variations and hence allowing the synthesis and analysis of objects in different poses. Handling geometric variations of the cameras and/or objects based on this approach is currently part of our ongoing research. In the next chapter, we present our method applied to address the problems pertaining to human/character motion and recognition.

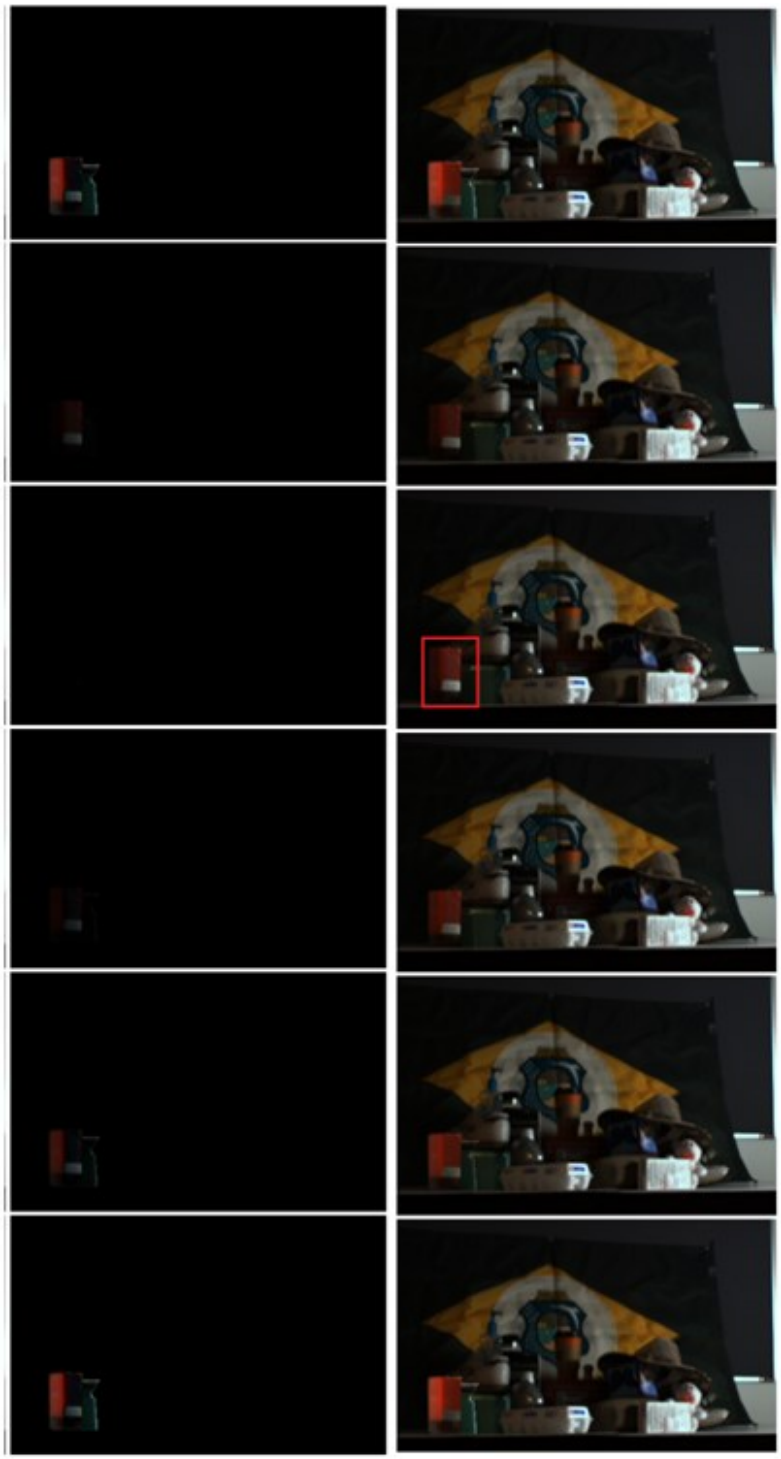


Figure 4.9: The column (a) is the error when compared with the ground truth and column (b) is the actual rendering.

CHAPTER 5
AN INTERPOLATION BASED APPROACH FOR MOTION SYNTHESIS AND ANALYSIS

5.1 Introduction

Motion-capture is a technique to record sequences of motion of a particular character using specialized sensors and software, and saving them in digital format. A character here could be a person, an animal, a robot or even just a body part like a human hand/face making different gestures. A motion sequence could range from something as simple as waving one's hand to a complex sequence of a set of martial arts steps or dance steps. Motion capture techniques are usually categorized based on the type of sensor used to capture the data. Optical motion capture, a technique currently very popular and also a subject of interest here in this chapter involves cameras to track and capture motion data. Current optical motion capture techniques use special cameras that are built to track a set of markers strategically placed on an actor. Specialized software is used to track the trajectories of these markers and compute and save the action sequence of the actor. In many cases, human intervention is needed to post-process or clean the captured data.

Currently, character animation for most applications such as film making, video games and virtual reality rely heavily on motion capture data. The biggest advantage of motion capture techniques is that it yields high quality and realistic animated motion. Some of the major disadvantages however, is that the data capture requires a lot of time, resources, expensive equipment (hardware, software, cameras, and lighting) and sometimes actors capable of performing specialized action sequences and of course trained engineers and personnel. Together, the process of capturing motion data can be prohibitively expensive. One way to bring down the overall cost of data capture would be to reuse existing data, which, is the main

motivation behind motion retargeting. Motion retargeting is the process of transferring a motion sequence of one character to another character such that the new character performs the same action sequence realistically. The main issue here is that different characters have different skeleton structures. The variation in skeleton structures could mainly be of two types; (a) skeletons are topologically different as in one skeleton is that of a human while the other is that of a cat or a dog (b) the topology is the same but the proportions of different segments (bones) are different as in one character is a six foot tall person of average built and the other character is a little kid who is three foot tall. It should be noted that topologically identical skeletons varying in height does not imply that all the bone segments scale proportionally; in fact it is rarely the case. This makes motion retargeting a hard problem to address.

In this chapter, we propose a novel interpolation based approach to mathematically model animated motion using a set of motion capture data as training data. The model represents a particular action for a set of all possible skeletons/characters defined in a parametric space. These skeletons need to be topologically identical *i.e.*, they all have the same number of bones, joints and the same degrees of freedom but can vary in the proportion of the bone segments. The parameters in this case are the lengths of a few selected bone segments.

This model serves as a compact representation of a particular animated motion sequence for a set of all possible skeletons within the parametric space. Once such a model is learnt, we can generate the motion data for a new skeleton which can be termed as motion synthesis. Also, as an inverse process, we can estimate the parameters of a skeleton when given its motion sequence, which is the analysis part. As a consequence of learning such a model, motion retargeting becomes a straightforward application of the synthesis part as long as the skeleton, into which the motion has to be retargeted to, falls within the parametric space. Moreover, our method allows for the inverse computation of the interpolation as well, *i.e.*, given a motion sequence, we are able to estimate the lengths of a set of bones of the skeleton pre-

chosen as the parameters. This sets up a framework to recognize the skeleton's identity given its motion sequence.

Formally, the motion retargeting problem in this context can be defined as follows; Consider ground-truth motion sequence for a test skeleton S^0 and action a represented as: ${S_a}^o(t)$. With a model M_a learnt using a set of n training skeletons; $T_a = \{{S_a}^1,\ {S_a}^2, \ldots\ldots,\ {S_a}^n\}$ for action a and compute: ${S_a}^r(t) = f\big(M_a,\ p(S_a)\big)$ for the new skeleton $S^r \notin T_a$ where $p(S_a)$ are the lengths of a set of bone segments chosen as parameters.

The rest of the chapter is organized as follows. In Section 5.1 we discuss the related work and draw comparisons to our method. Section 5.2 describes the interpolation approach specific to motion retargeting and motion recognition. In Section 5.3 we describe the dataset used for our experiments and will elaborate on preprocessing the motion data. Section 5.4 is dedicated to all the experiments we performed and finally, Section 5.5 discusses the conclusion and future work.

5.2 Previous work

Gleicher [25] designed a space-time constraints solver to address the problem of motion retargeting. The approach here is to optimize this solver by considering all the geometric constraints. This way he computes the retargeted motion along with the constraints, and hence preserving the frequency characteristics of the original motion.

Bindiganavale *et al.* [9] computed the zero-crossings of the second derivative of the motion signal to detect significant changes in the motion. Further, they applied inverse kinematics to enforce these detected constraints. This technique is specifically useful for actions involving interactions with external objects or with the subject itself.

Arikan *et al.* [3] designed a user interactive framework that allows the user to interactively choose certain actions by annotating them. The final motion then performs the specified actions at specified times. The user interactive synthesis process requires the user to first annotate the motion database with the same vocabulary. However, the user needs to

annotate only a portion of the database. The system then uses an interactive Support Vector Machine to generalize the user annotations across the database. Once the annotations are ready, the synthesis algorithm tailors the pieces of motion from the database according to the user's specification and successively optimizes the motion sequence using a dynamic programming algorithm. The final motion is then available immediately after the optimization.

Choi *et al.* [12] present an online motion retargeting algorithm to retarget the motion of a character to another in real time. The technique is based on inverse rate control, which computes the changes in joint angles with respect to that of the end-effector position. This is essentially implementing inverse kinematics using Jacobians. The retargeting algorithm tracks the trajectory of multiple end-effectors and imitates the joint motion of the original character by exploiting the kinematic redundancies of the animated model. This method is also effective in maintaining the high frequency details of the original motion.

5.3 Interpolation Model for Human Motion

The interpolation approach involves learning a kernel matrix K such that when K is multiplied by a parameter matrix built using a parameter p_q will result in the corresponding motion sequence S_q. Essentially, K is a one-to-one mapping from the parametric space to the motion (of a skeleton) space given by the equation $K \times f(p_q) = S_q$. Here, the motion for a particular skeleton performing an action q is basically the variation of the joint angles from frame-to-frame. Clearly, for a given joint, the parameters that affect its rate of change the most is the length of the two bones that make up the joint. This data is usually available with motion capture data and hence, it is a rather intuitive choice to use the lengths of bones of a skeleton as parameters to learn our interpolation model. In the following sections we present a more formal description for learning an interpolation model for optical motion capture data.

Let us now consider an optical motion capture data for a particular action a. The motion capture data is basically the rate of change of joint angles in each degree of freedom with respect to frame numbers, for a particular skeleton structure. In our case the skeleton design

involves 22 joints with 3 degrees of freedom per joint giving us 66 different angle variations from frame-to-frame. Therefore, if the captured data is n frames long, and the number of joint angles is j, 66 in this case, the motion data for a skeleton s is a matrix represented by ${m^s}_{n\times j}$. For our method we need a set of such motion data for different skeletons but for the same action sequence a. If we consider t training samples to learn the kernel matrix for an action w, we build the training motion matrix by vectorizing each of the t training motions into column vectors and arranging them as a matrix given by; ${M^w}_{nj\times t} = [{m_{nj\times 1}}^{s_1}, {m_{nj\times 1}}^{s_2}, {m_{nj\times 1}}^{s_3}, \ldots, {m_{nj\times 1}}^{s_t}]$, where $S = \{s_1, s_2, s_3, \ldots, s_t\}$ are the training skeletons. This gives us the matrix of training data that allows us to learn the interpolation model.

To build the component matrix, we use the lengths of the bones as parameters and subject them to the interpolation function $f(p)$, where f is the interpolation function and p is the set of parameters used to learn the model. Our skeleton comprises of 20 bones (A detailed description of the skeleton will be made in Section 6.3.). However, we don't have to use as many parameters for learning the model. Therefore, if we choose h parameters, we will have an h dimensional parametric space. Now, as an instance of an interpolation function, let us consider a second degree polynomial equation f_h in $h = 4$ variables as the interpolation model. Let the chosen parameters for a specific skeleton i be represented as $p_i = (w_i, x_i, y_i, z_i)$, where w_i, x_i, y_i and z_i are the lengths of the bones w, x, y and z of the i^{th} skeleton. Let us now consider a second degree equation in four variables as an example model given by; $f_4(p_i) = w_i^2 + x_i^2 + y_i^2 + z_i^2 + w_i x_i + w_i y_i + w_i z_i + x_i y_i + x_i z_i + y_i z_i + w_i + x_i + y_i + z_i + 1$. It must be noted that the above equation has 14 components excluding the constant. Building these equations for each of the t skeletons in the training set and arranging them together as columns vectors, we construct a component matrix for action a given by, ${C^a}_{l\times t}$, where l is the number of components in each of these equations.

Given the motion matrix ${M^a}_{nj\times t}$ and the component matrix ${C^a}_{l\times t}$ for a particular action sequence a, we construct a linear system that performs the interpolation with t vectorized

training motion data m, and l components of the interpolation model as $K_{nj\times l} \times {C^a}_{l\times t} = {M^a}_{nj\times t}$. We now compute the pseudo inverse of the component matrix *C* and multiply it to both sides of the equation to infer the kernel matrix K. Formally, we obtain $K_{nj\times l} = {M^a}_{nj\times t} \times (C^a_{l\times t})^{-1}$.

5.4 Motion Capture Data

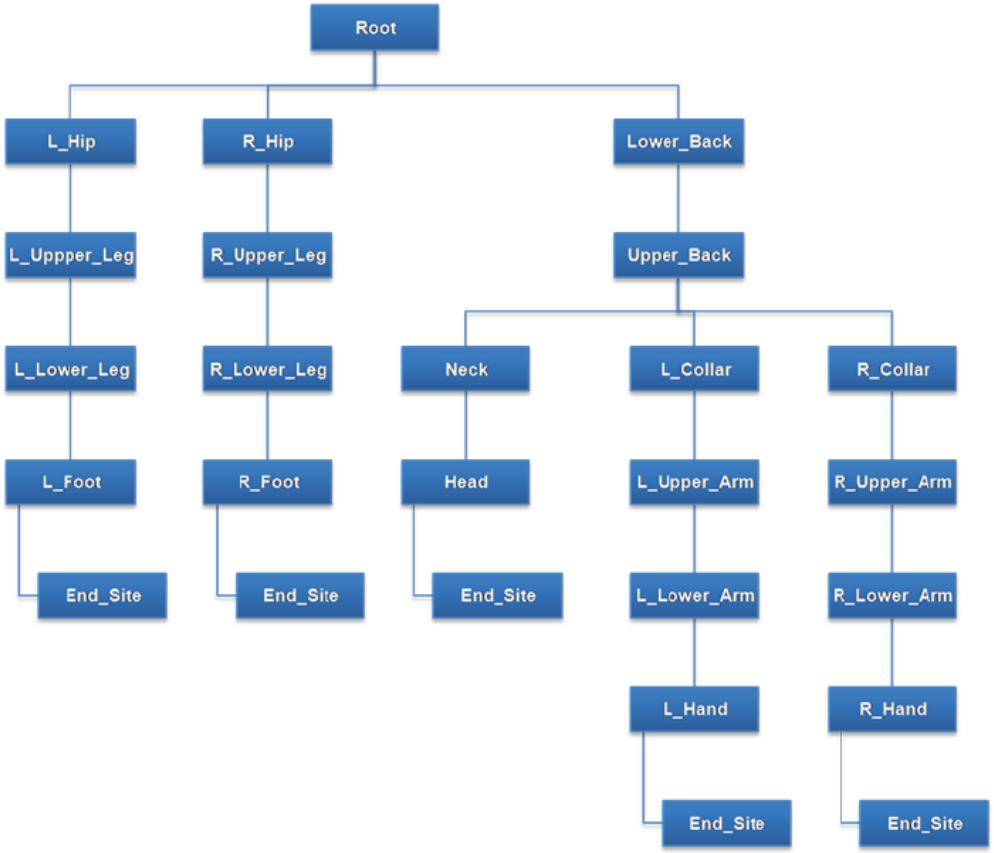


Figure 5.1: Hierarchy of the skeleton used for our experiments.

In this section, we discuss the motion capture data used for our experiments. Our data is taken from the Human Motion Database [26] where motion capture data is made available for several actions and for several people varying in age, height and weight. This database uses the Biovision BVH file format. This format mainly has two parts to it, a header section and a motion section. The header defines the skeleton and its initial pose. The motion section is basically values of angles of each joint for each degree of freedom for each frame. The

skeletons in this data comprises of 20 different bones and 22 joints, with the *root* being an imaginary node at the *hip*. The 22 joints have 3 degrees of freedom each which makes a total of 66 different joint angles. The hierarchy of the skeleton is shown in Figure 5.1.

For our experiments we use the cross-validation dataset from the Human Motion Database. The cross-validation dataset comprises of 70 different actions such as *clap*, *bounce*, *bang-door* and *walk*. These actions are performed by about 50 different subjects in a wide range of skeletal structures distributed over height, weight, gender, and age of the subjects. We performed our experiments for the *walk* and *jog* datasets. The main reason for choosing these two is that they have a lot more frames than more trivial actions such as *clap* or *jump-in-place*.

5.4.1 Data Pre-processing

The *walk* and *jog* datasets comprises of 49 subjects walking and jogging up and down a few times respectively. From here on, we will discuss about these two datasets particularly and describe all the pre-processing steps taken before learning the interpolation model.

5.4.1.1 Manual selection

Let us consider the *walk* dataset first. In this dataset, the subjects start from a still pose and starts walking in one direction for a few steps and walks back to the starting point. This cycle is repeated a few times. Since our method expects the training actions to be the same, we encounter a problem here, which is; not all the subjects turn at the same time and more importantly some turn around in the clockwise direction and some other in an anticlockwise fashion. This cannot be defined as the same action unless if all the subjects were enforced to turn in one fashion and at certain pre-decided points. Besides, turning around can be defined as an action in itself that is very different from the walk action. Moreover, the walk action for all the subjects needs to be in the same direction. Hence, in order to make the data a homogenously pure walk-action data, we have to select the frames of the subjects walking in one particular direction where the start and end pose of all subjects are respectively similar. Doing this manually seemed a simpler option considering the size of the dataset. We picked the sequence

starting with the frame where the right heel makes the first contact with the floor while walking in the forward direction. Further, instead of similarly choosing the end frame, we simply got 279 frames from the start frame, making it 280 frames pure walk data. As a result, the end frames of the different subjects are far less similar compared to that of the start frame. The model was able to handle it since the variation of speed-of-walking over this small segment was not very pronounced. If that was the case, then we could have chosen the end frame as well and normalize the data across all the subjects. The number 280 was chosen because it was approximately the average number of frames for all the subjects to start with the right heel and approximately end with the same. This action is basically the movement of a subject from placing her right heel on the floor, and walk forwards until the next contact of the right heel is made with the floor. This can be defined as one *cycle* of walk for our purpose. Similarly, we extracted a *cycle* of 120 frames for the jog dataset as well. For a frame rate of 120 frames-per-second (fps), the walk action takes 2.3 seconds and the jog action take 1.0 second. It should be noted that if for a particular action by a particular subject, we are not able to find such a cycle, then we omit that data from the training set.

5.4.1.2 Motion data smoothing

Motion capture data is usually quite noisy and can have a lot of undesirable high frequency components. These high frequency components are sometime hard to perceive by watching the animation, however, if we plot the angles they become more apparent. This is especially bad for interpolation techniques. Moreover, certain subjects have the tendency to make other random unnecessary/unusual movements while performing an action like walk. The reasons could be many, such as a nervous tick, injury, high caffeine intake or just plain habit. This causes outliers in the motion capture data, which again is not great for interpolation techniques. To get rid of these outlier movements and high frequency components of the data we us a robust spline smoother [23] to preprocess the manually segmented data. Figure 5.2

shows the plots of four random joint angles plotted with respect to frame numbers for a sample *walk* dataset.

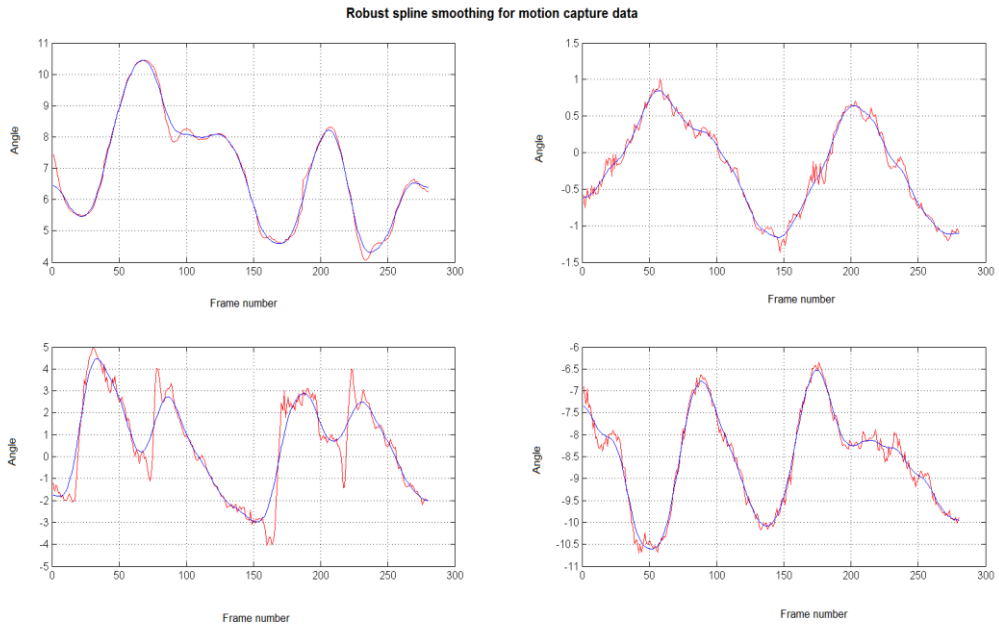


Figure 5.2: Robust spline smoothing for 4 sample joint-angles of a subject performing the walk action.

5.4.2 Parameter selection

Another important requirement for learning these interpolation models is the selection of appropriate parameters. In our case here, using all the 20 bones as will lead to a high dimensional parametric space. Moreover, using all the parameters will be redundant simply because not all parameters have the same level of influence for a particular action. In other words, some bones play a more important role for a specific action than others. As an example, for the walk action, the neck does not have the same influence as the right foot or left upper leg. Therefore, for a given action we can use only a subset of the parameter set. Consequently, for a specific action we compute the zero-crossings of all the 66 joint angles and use the top h unique bones that constitute these joints. In our case we used $h = 8$ parameters for both *walk* and *jog* experiments and, aligned with our intuition, the 8 bones were the two feet, two lower legs, two upper legs and two upper arms.

5.5 Experiments

We now describe the experiments performed on the walk and jog datasets. First, we demonstrate the interpolation model used in a motion retargeting setup and then move on to motion recognition.

5.5.1 Motion Retargeting and Recognition

For the motion retargeting problem we use the synthesis part of our method. To achieve good quality retargeting we need to minimize the average synthesis error. The synthesis error here is the sum of absolute difference between the synthesized motion and ground-truth motion. The average synthesis error is computed by running the synthesis experiment for a set of test data (data excluded from training) and taking the average of the errors. As we know from our earlier experiments with the images, the synthesis error depends on the number of training data and the number of components used to learn the model. Figure 5.3 and 5.4 shows the plots of the average synthesis error for the walk dataset.

For motion recognition, we use the analysis part of our method to perceive the parameters of a query motion, compute the motion based on the perceived parameters and compute the sum of the absolute difference between the synthetic query motion and all the subjects. We then conclude that the subject for which, the analysis error is the minimum is what the system recognized. Here it is obvious that the performance of the analysis part is central to achieving good recognition rates. Here again, the average analysis error has to be minimized and the parameters it depends on are the number of training samples and the number of independent components. Figure 5.5 and 5.6 show the plots of the average analysis error for the walk dataset.

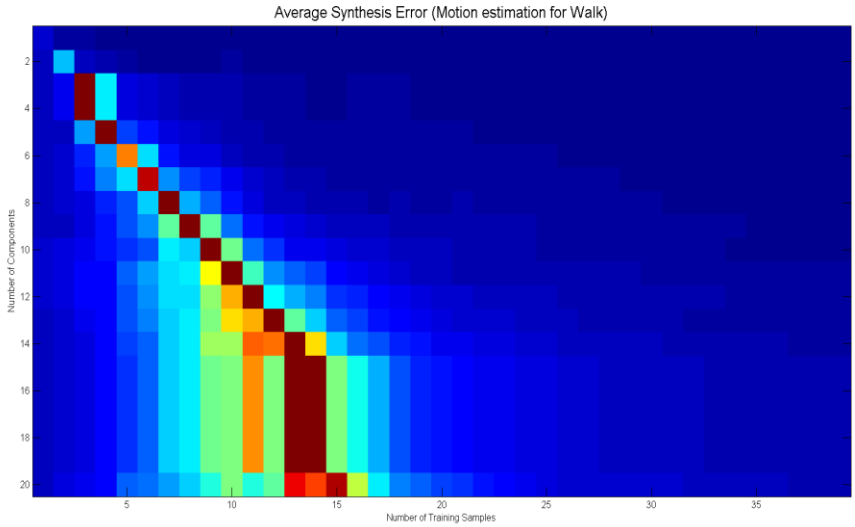


Figure 5.3: Average synthesis error computed for the walk dataset, with the number of components varying from 1 to 20 and the number of training samples, varying from 1 to 40. Dark blue indicates lesser error and dark red is the other extreme.

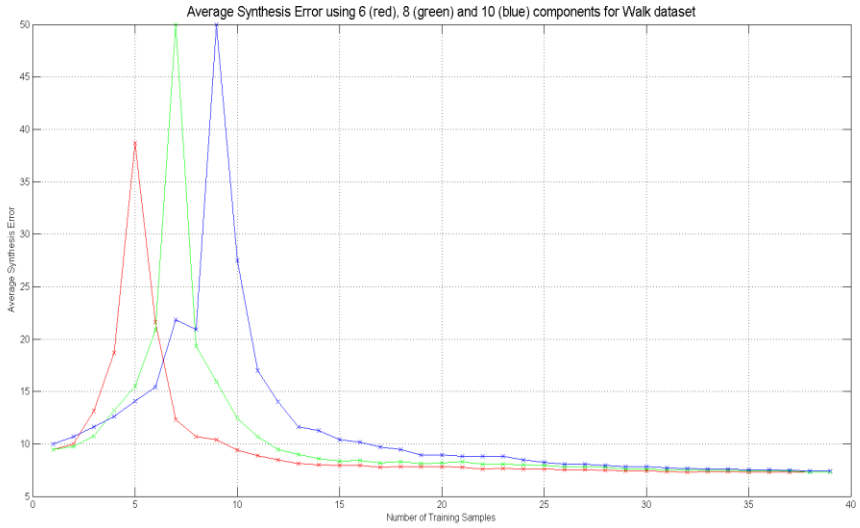


Figure 5.4 Average synthesis error computed for the walk dataset, for 6(red), 8(green) and 10(blue) components with number of training samples varying from 1 to 40.

From the graphs (Figures 5.3 to 5.6) of the walk dataset with 40 subjects, we can see that, to learn a model that can both perform synthesis and analysis the optimum number of components used should be 10 while the number of training samples should be 30. These

experiments were performed with 8 parameters namely, right foot, right lower leg, right upper leg, right upper arm, left foot, left lower leg, left upper leg and left upper arm. With this set up we get the average synthesis error: 6.9052 and the average analysis error: 0.7586, and the recognition rate of 67.5%, where 27 out of the 40 were recognized correctly.

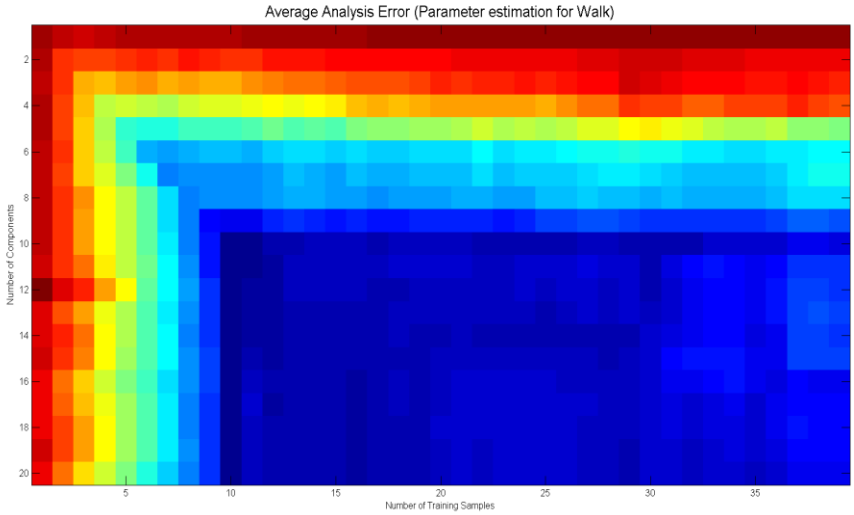


Figure 5.5: Average analysis error computed for the walk dataset, with the number of components varying from 1 to 20 and the number of training samples, varying from 1 to 40. Dark blue indicates lesser error and dark red is the other extreme.

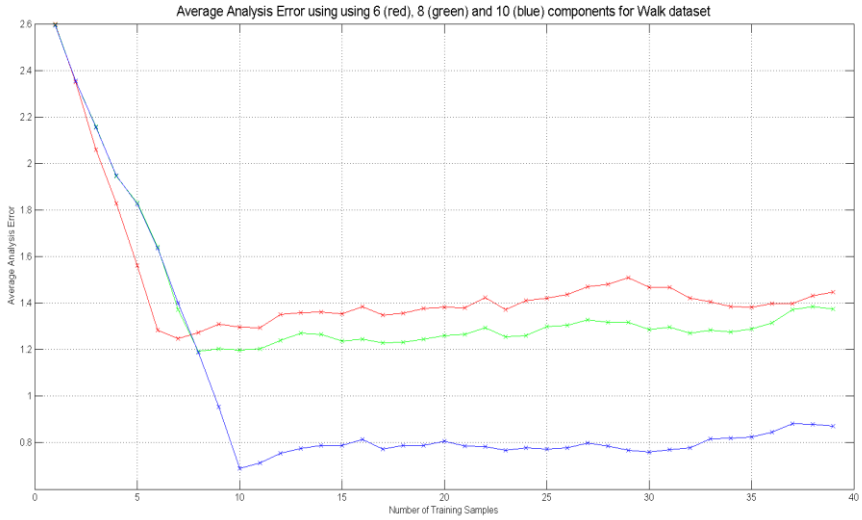


Figure 5.6: Average analysis error computed for the walk dataset, for 6(red), 8(green) and 10(blue) components with number of training samples varying from 1 to 40.

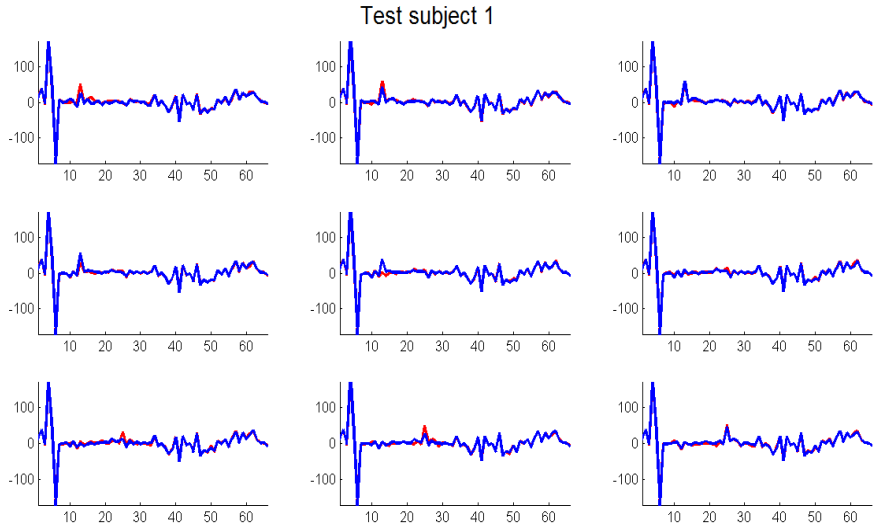


Figure 5.7: Nine random frames depicting the actual joint angles in blue and the retargeted joint angles in red for test subject-1 doing the walk action.

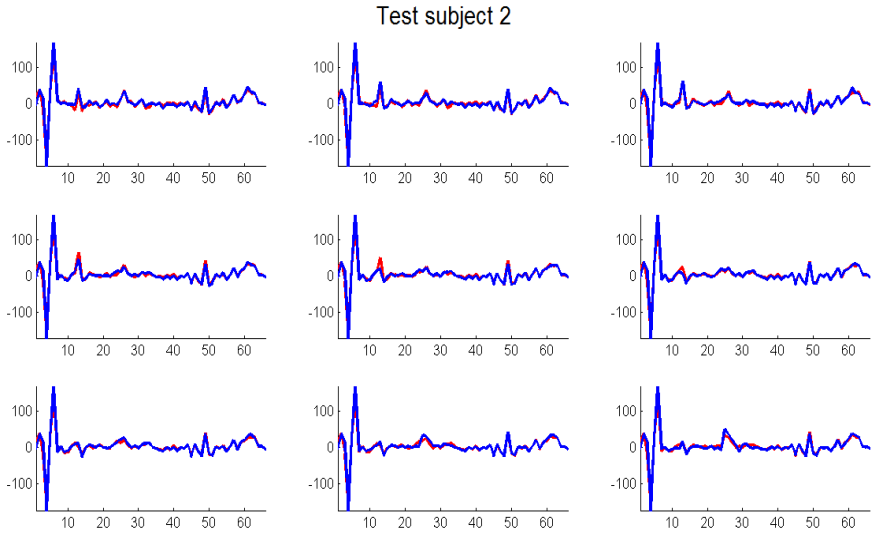


Figure 5.8: Nine random frames depicting the actual joint angles in blue and the retargeted joint angles in red for test subject-2 doing the walk action.

Similarly for the jog dataset with 47 subjects and the same set of parameters as walk, we found out that the combination of 17 components and 46 training samples worked best. We obtained the average synthesis error: 7.8581 and the average analysis error: 1.4013. With this

average analysis error, we observed a recognition rate of 53%, where 25 of the 47 subjects were identified.

To demonstrate the performance of motion retargeting we plot nine random frames out of the 280 frames of motion depicting the variations of the 66 joint angles. The plot in blue is the ground-truth motion while the overlaid red plot is the retargeted motion. Figure 5.7 and 5.8 show this for two subjects doing the walk action.

5.6 Conclusion and Future work

In this chapter we have demonstrated the application of our interpolation based approach for two problems namely, motion retargeting and motion recognition. With the former, we have met with good results and the retargeted motion look pretty realistic and visually appealing. However, the method, at the moment does not handle the issues related to interaction of the skeleton with the environment. This can be easily extended by including such parameters into the interpolation model, which is part of our ongoing research. On the other hand, the recognition-rate was not up to the mark; however, the model certainly demonstrates its potential for application in this area. One way to improve this could be to use more parameters and then project the analyzed parameters to a lower dimensional space using a technique like Principal Component Analysis. This too, is a part of our current research.

APPENDIX A

CVPOV: AN AUTOMATED TOOL FOR GENERATING SYNTHETIC GROUND TRUTH

Introduction

It is common knowledge that capturing data, be it images or videos for most experiments in the areas of computer vision or image processing is time consuming and many times requires a lot of resources and expensive equipment. Although many datasets are easily available today, and many of them are designed to test the current state-of-the-art methods, it usually is very difficult to have the data to specifically meet the requirements of users other than that of the designers of the dataset. With an aim to address these issues, we present CVPoV, a tool built over a popular ray tracing engine, POV-Ray [65] along with VLPov [68], a patch designed to save the depth information. CVPoV allows the user to specifically generate images/videos of simple objects or complex scenes involving several different objects. Moreover, the user can specify different poses, scale and lighting conditions based on her requirements.

A fundamental and very important feature of CVPoV is that it finds point correspondence between synthetic images of the same scene under different viewing configurations. The point correspondence problem can be posed as; given a point in an image of a particular scene, find the corresponding point in another image of the same scene. This is a very challenging problem while dealing with real images under unconstrained illumination and noise. Furthermore, the camera calibration and the depth information are not available. However, in case of synthetically generated images such as those rendered using the POV-Ray ray tracer, we have all the necessary information to calculate the disparity map and hence compute the point correspondences between pixels of two different images of a scene. Specifically, we use the depth information and the camera parameters to compute the disparity map. Once we have the disparity map we can also compute an occlusion map. The biggest advantage of using this tool is that we can obtain all the information namely, calibration, depth and disparity maps for realistic scenes in user defined lighting conditions and camera poses, something which is practically impossible with real world data.

Background

We now describe the building blocks of CVPoV, namely, POV-Ray [65], MegaPOV [63] and VLPov. It should be noted that CVPoV is built over these three tools and uses several functions from all of them, the details of which are as follows.

The Persistence of Vision™ Ray-Tracer [65] is a powerful ray tracing software package. POV-Ray is perhaps the most popular ray-tracing software package to date. Its popularity is mainly due to its high quality scene rendering capabilities, easy to use scene description language, and the availability of a large library of example scene files. Moreover, POV-Ray binaries as well as the source code are freely available for the PC, Macintosh, and UNIX platforms. POV-Ray is essentially used as the ray tracing engine. The user of CVPoV will have to design their scene/object or use existing files in accordance with this platform.

MegaPOV is a set of custom and unofficial patches that are built over the POV-Ray ray tracer. These patches provide additional features to the existing POV-Ray package. One of the additional features provided by MegaPOV is the post processing patch. The MegaPOV post processing patch not only allows manipulation of the color of the pixels after the rendering step is completed, but also provides access to the content of the rendered image through internal functions for the user. This data contains several components such as: color of the pixel, intersection point, and the depth information. In other words, we gain access to all the information the ray tracer receives for each of the pixels in the rendered scene.VLPov is an annotation patch developed by Vedaldi [68] for MegaPOV. This annotation patch extends MegaPOV to export camera and depth information for the rendered scenes. The VLPov patch will save by default, along with any output image, an annotation file with the camera calibration. VLPov can also export an accurate depth map of the scene. VLPov comes with MATLAB functions for reading camera calibrations and processing the depth map data along with the synthetic images generated by POV-Ray.

CVPoV is a set of functions built over these tools to allow the user to automatically generate synthetic ground-truth data with user specified view points along with motion field data and occlusion maps. The user simply has to choose a suitable synthetic object/scene and design an input file that describes the locations and orientations of the cameras. The tool then generates the synthetic images/videos accordingly. This setup allows the users to specifically design and generate datasets to test their algorithms.

Figure A1: Example scenes designed by [57]; (a) The Patio, (b) Urban Tree (c) The office and (d) Travieso.

POV Scenes

We will now introduce some sample POV scenes rendered by CVPoV for our demonstration purposes. We chose four different scenes created by Jaime Vives Piqueres [57], shown in Figure A1. Two of them namely, Patio [61] and the Urban Tree [60] are outdoor scenes (1st and 2nd pictures from top in Figure A1) and the remaining two, the Office [62] and Travieso [59] are indoor scenes (3rd and 4th pictures from top in Figure A1). It is probably worth noting that the scenes are pretty realistic and extremely detailed. In the following sections we

will describe the usage of CVPoV along with example outputs generated using these four scenes.

CVPoV Input Files

Besides the POV-Ray scene description file, CVPoV reads in a user defined camera description file as an input file. The camera file needs to be designed by the user for a specific POV scene/object and saved with a '.cam' extension. Each line in the '.cam' file corresponds to a particular camera at a specific frame. Each line has an integer number representing a time frame, followed by another integer number representing camera identification, followed by the complete camera pose specification in the scene to be rendered. The camera description includes any transformations applied to the camera such as translation and/or rotation. The camera pose specification uses the same syntax as the camera structure in POV-Ray files. We will now describe a POV camera model followed by the design of the camera-description files required for CVPoV. We then elaborate on the functions along with example output images.

CVPoV Camera Description File

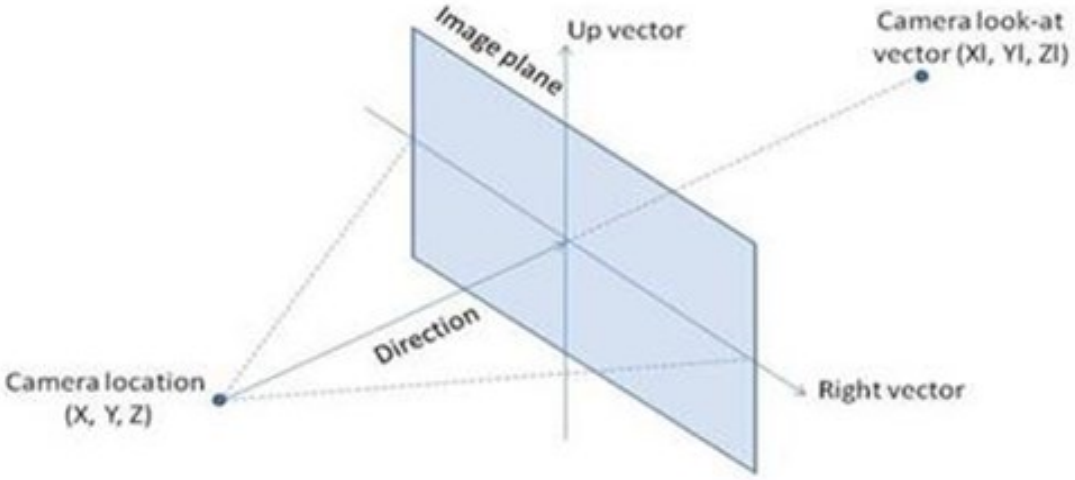


Figure A2: POV-Ray camera model [66]

It is important to understand the camera models used by POV-Ray to be able to design and build camera description files for CVPoV. The POV-Ray camera model [66] uses the following notations to setup a camera in the ray-tracing world: (a) location: a point that defines

the *x*, *y*, *z* coordinates of the camera in the ray-tracing world coordinate system, (b) direction: a vector that describes the initial direction to point the camera before any other transformations, (c) look_at: a point that specifies *x*, *y*, *z* coordinates for the camera to look at after pan and tilt transformations and (d) up and right: vectors that give the relative height and width of the view screen. Figure A2 illustrates these vectors used in the camera description. The POV camera description usually consists of the location, the direction vector and the look_at vector. The translate and rotate vectors transform the camera from its original pose according to the values specified and the order of transformation. The numbers from left to right in the angle brackets represent the values in the *x*, *y* and *z* axes, respectively. For example, location<10,20,30> specifies that the camera is located at the three dimensional point (10,20,30) and translate<-10,0,0> moves the camera 10 units in the negative *x*-direction. The modifier translate will translate the camera along the *x*, *y* and *z* axes by the amount specified in the vector <TX, TY, TZ>. Similarly, the modifier rotate will rotate the camera about the *x*, *y* and *z* axes by the degrees specified. Figure A3 depicts the sample camera description file-1 and Figure A4 is the output images generated for the entries in this camera description file.

```
0    0    camera{ location <10,70,-159> direction <0,0,1.3> look_at <85,76,170> translate<0,0,0> rotate<-2.5,0,0> }
1    0    camera{ location <10,70,-159> direction <0,0,1.3> look_at <85,76,170> translate<0,0,15> rotate<-10,0,0> }
2    0    camera{ location <10,70,-159> direction <0,0,1.3> look_at <85,76,170> translate<0,10,35> rotate<-20,-5,0> }
3    0    camera{ location <10,70,-159> direction <0,0,1.3> look_at <85,76,170> translate<0,30,55> rotate<-25,-15,0> }
```

Figure A3: Sample camera-description file-1

The sample camera-description file-1 basically depicts- the trajectory of camera-0. The translation and rotation vectors are varied by the user for frame 0 to 3. This essentially renders four images as shown in the output in Figure A4. In another example we show a trajectory of two frames (frame-0 and frame-1) of a grid of 4 cameras (camera-0 through camera-3). In this case we vary the look_at vector instead of using the translate and rotate vectors. Figure A5 and A6 shows the cam file and its corresponding output. For each line in the input camera file, the associated rendered bitmap is saved with the naming scheme scene_camera_frame_id.bmp, where scene is obtained from the POV-Ray scene file name scene.pov, camera is obtained

from the input camera file name camera.cam, frame is the time frame in the current line of the camera file, and id is the camera identification in the current line of the camera file. Similarly, Figure A7 shows a sample object [64] rendered with rotation about the z-axis. As demonstrated with these sample camera-description files, CVPoV can be used to render synthetic data as per the specific requirements of the application.

CVPoV Functions

The ray-tracing and rendering part of CVPoV is essentially MegaPOV v1.2.1, C source code with an active VLPov annotation patch used to save the bitmap images, the depth maps, and the camera calibration files. The functionality built over this to be able to generate the renderings according to a camera-description file and compute the motion field and occlusion maps are our MATLAB functions. We now describe these functions and provide some sample output for visualization.

```
0    0    camera{ location<10.000000,70.000000,-159.000000> direction <0,0,1.3> look_at<68.228956,92.084505,182.522482> }
0    1    camera{ location<19.748330,70.000000,-161.222264> direction <0,0,1.3> look_at<78.133137,92.160454,181.155064> }
0    2    camera{ location<10.000000,60.000000,-159.000000> direction <0,0,1.3> look_at<68.422670,82.122558,181.672725> }
0    3    camera{ location<19.748330,60.000000,-161.222264> direction <0,0,1.3> look_at<78.326851,82.198507,180.305307> }
1    0    camera{ location<10.000000,70.000000,-159.000000> direction <0,0,1.3> look_at<13.420717,136.033265,203.054149> }
1    1    camera{ location<19.748330,70.000000,-161.222264> direction <0,0,1.3> look_at<23.295372,137.202858,204.098957> }
1    2    camera{ location<10.000000,60.000000,-159.000000> direction <0,0,1.3> look_at<14.180896,126.636339,199.719496> }
1    3    camera{ location<19.748330,60.000000,-161.222264> direction <0,0,1.3> look_at<24.055551,127.805932,200.764304> }
```

Figure A4: Sample camera-description file-2

Depth and Camera Calibration

The depth information and the camera calibration files are generated and saved using the generateDepthFiles function. For each entry in the camera-description file, the function renders a bitmap image, saves its depth map, and exports the camera intrinsic and extrinsic parameters into a calibration file. The function also takes in the scene width and height in pixels as parameters so that the user can choose the resolution of the rendered data. Figure A8 shows a sample depth map for the Patio scene.

Motion Field

The function generateDepthFiles computes the motion field from the geometry of the cameras and the depth of each pixel for pairs of images. The motion field is computed for all

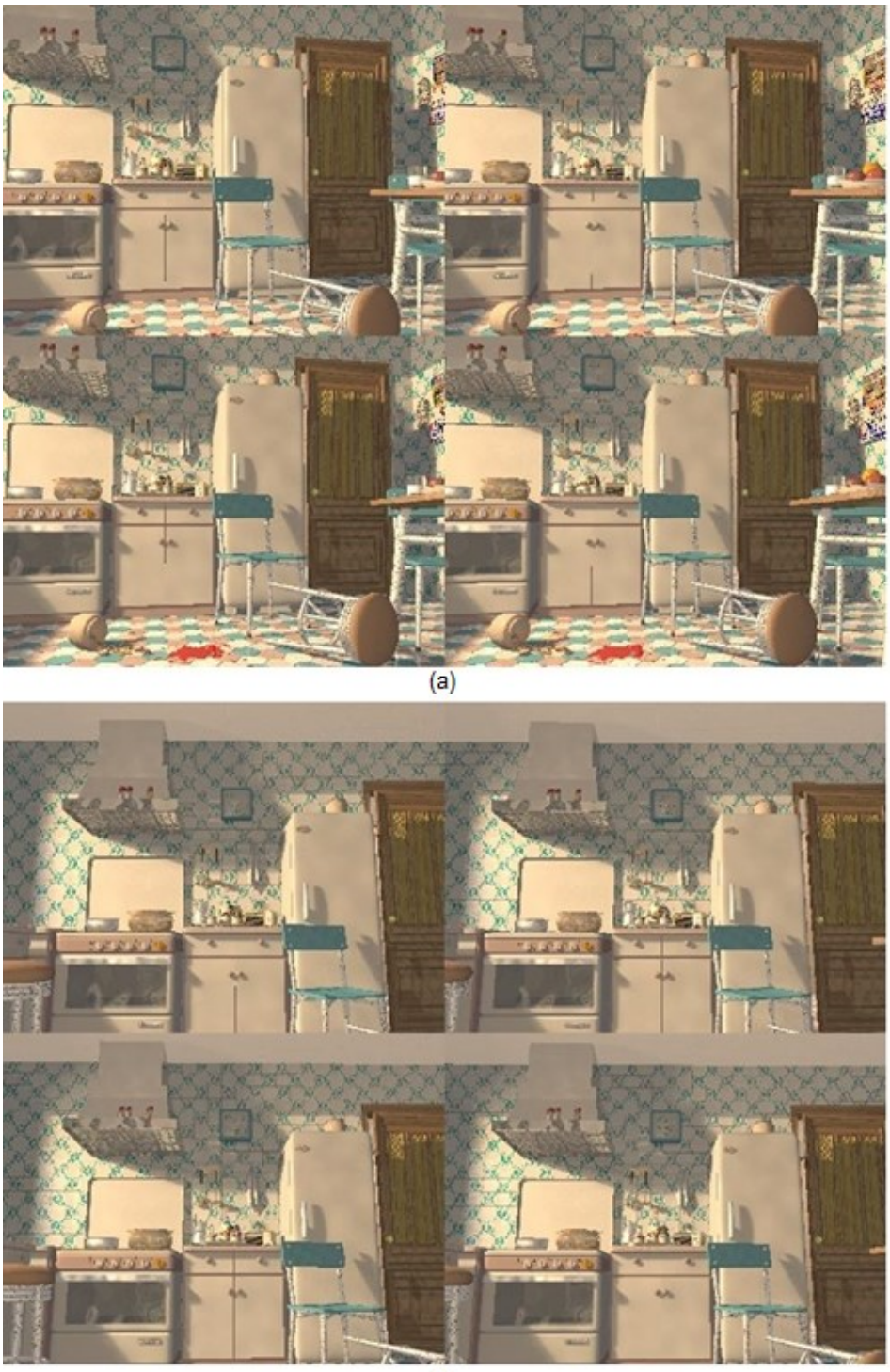
pairs of different cameras that do exist in the same time frame and for consecutive time frames of the same camera. The motion field data is essentially two matrices that give the relative horizontal (x-axis) and vertical (y-axis) displacement of each pixel between the two images. Given this information, we find the point correspondence $p'$of a pixel $p = (p_x, p_y)$ by adding the horizontal displacement dx and the vertical displacement dy such that $p' = (p_x + dx, p_y + dy)$.

Given the depth information for each pixel of a rendered scene, the scene is a three dimensional cloud of points. Consider a point $P(X, Y, Z)$ in the scene, where (X, Y, Z) are in world coordinates. The projection of P on the image plane of camera-1 is the two-dimensional point $p_1(x_1, y_1)$ and the projection of P on the image plane of camera-2 is the two-dimensional point $p_2(x_2, y_2)$, where projections p_1 and p_2 are computed as: $p_1(x_1, y_1) = K_1 \times P(X, Y, Z)$ and $p_2(x_2, y_2) = K_2 \times P(X, Y, Z)$ and K_1 and K_2 are the intrinsic calibration matrices of camera-1 and camera-2, respectively. More specifically, the projection equation is described as: $\begin{bmatrix} x_{img} \\ y_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} -f/S_x & 0 & O_x \\ 0 & -f/S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$, where, f is the camera focal length, S_x is the pixel size in the x-axis, S_y is the pixel size in the y-axis. (O_x, O_y) is the principal point, and if the image width is *w* and height is *h* then $O_x = (w + 1)/2$ and $O_y = (h + 1)/2$ respectively. In our case, the principal point is the center of the image plane. For each rendered image, generateMotionField retrieves this intrinsic calibration matrix of the camera to be able to compute the projections of the points from the 3D space onto the image planes.

For each camera the generateMotionField function also retrieves its pose in terms of translation vector T and rotation matrix R to compute the extrinsic parameters. Therefore, for a given pair of cameras, we have the intrinsic matrix K_1 of the first camera, the rotation matrices R_1 and R_2 of both cameras, and the translation vectors T_1 and T_2 of both cameras. Now, for every 3D point of the second image, the function computes the corresponding 2D point in the first image using the equation; $p = K_1 \times (((R'_1) \times R_2 \times (P + T_2)) - T_1)$, where R'_1 is the

transpose of the rotation matrix R_1. The displacement of each pixel *i.e.*, the motion field, is then computed by taking the difference between the corresponding points in the two images.

Figure A5: Output for the sample camera-description file-1



(a)

(b)

Figure A6: Frame-0 (a) and frame-1 (b) of a grid of 4 cameras as described in sample camera-description file-2

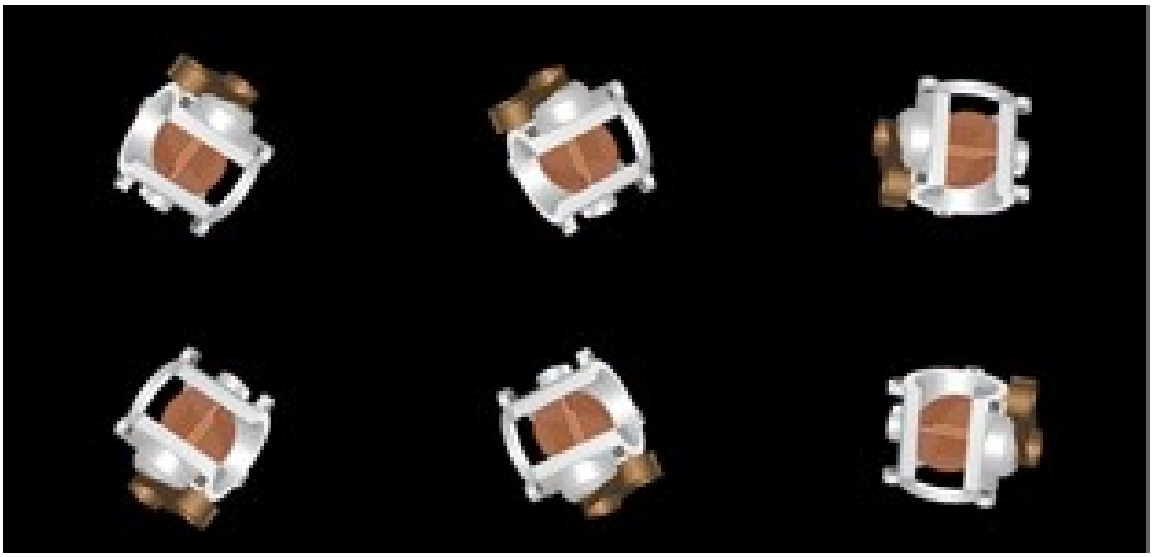


Figure A7: Sample object rendered with rotation about the Z axis

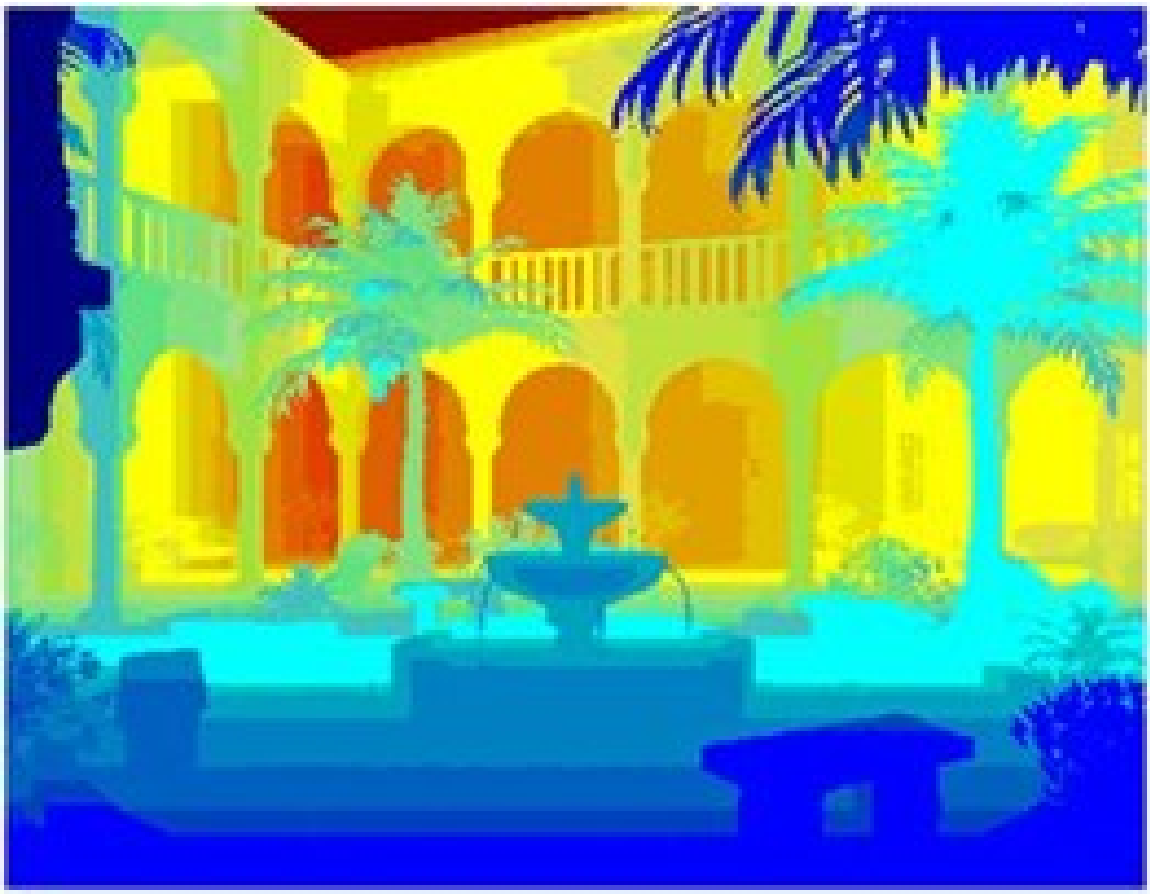
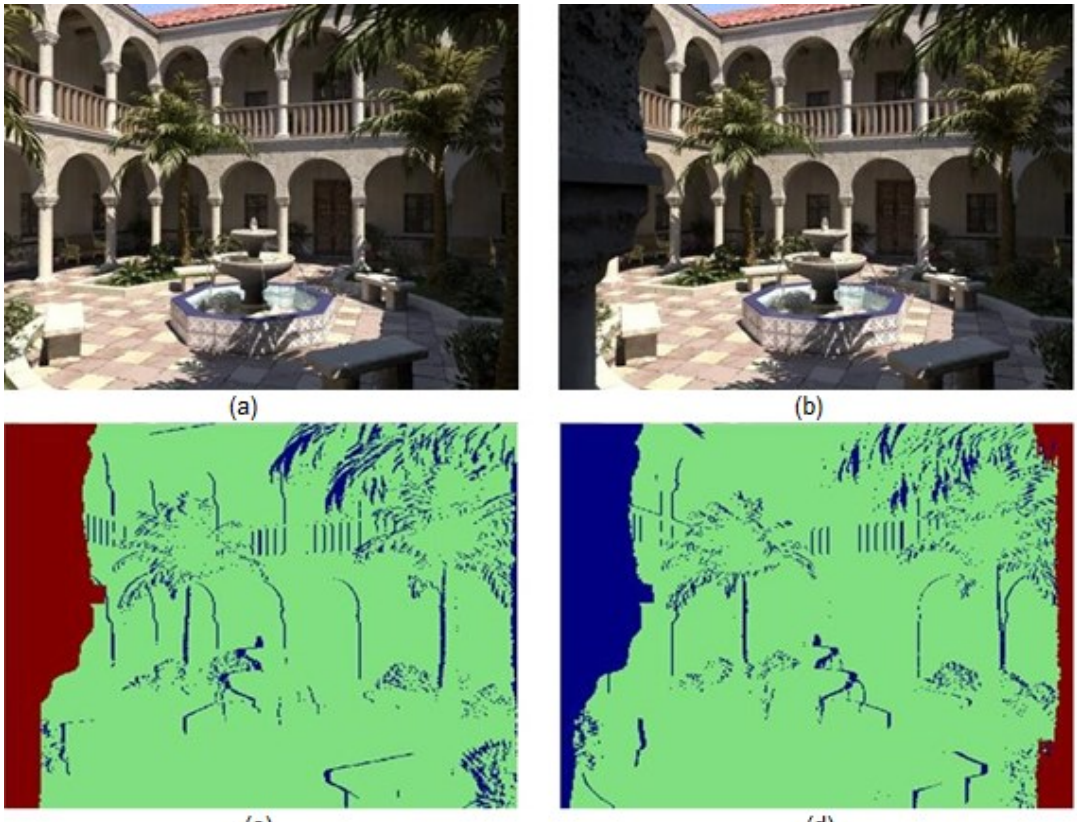


Figure A8: Sample depth map for the Patio scene.

(a) (b)

Figure A9: The point correspondence displayed by the demoMotionField function. The two images are from the Urban Tree scene with a translation in the x and y directions. The red crosshair on the image (b) is selected by the user while the one on the image (a) is computed using the motion field data.



(a) (b)

(c) (d)

Figure A10: The occlusion maps of the Patio scene with a translation along the x axis. The image (c) is the occlusion map of the first image (a) with respect to the second image (b). The image (d) is the occlusion map of the second image (b) with respect to the first image (a).

We also have a function called demoMotionField that uses the motion-field data to compute and plot the corresponding point on the second image for a user selected point on the first image. Figure A9 shows a screenshot depicting point correspondence between pairs of images of the same scene.

Occlusion maps

Given two images, this function determines the pixels of the images that are not visible from both viewpoints. For each pixel p in one image, the function checks if the corresponding pixel p' is visible from the other viewpoint. If the pixel p' is not visible, then it is occluded in the other image. The usage of this function is occlusionMap(sceneName, cameraName, frame1, camer1, frame2, camer2). The function returns an occlusion map occMap1 of the first image with respect to the second image and an occlusion map occMap2 for the second image with respect to the first image. Figure A10 depicts the occlusion maps for two images of the Patio scene that are translated in the x direction. The pixels in red are out of the field of view with respect to the camera view point, the pixels in green are the ones that are visible from both viewpoints, and the blue pixels are occluded.

REFERENCES

[1] Adelson, E. H., Bergen, J. R.: "The Plenoptic Function and the Elements of Early Vision", Computational Models of Visual Processing (1991), 3-20

[2] Adini, Y., Moses, Y. and Ullman, S.: "Face Recognition: The Problem of Compensating for Changes in Illumination Direction", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 721-732, July 1997.

[3] Arikan, Okan., Forsyth, David A. and O'Brien, James F.: "Motion Synthesis from Anotations". In Proceedings of ACM SIGGRAPH 2003, pages 402–408, August 2003.

[4] Barrow, H., Tenenbaum, J.: "Recovering Intrinsic Scene Characteristics from Images", Computer Vision Systems, (1978).

[5] Basri, R., and Jacobs, D.W.: "Lambertian reflectance and linear subspaces", Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.25, no.2, pp. 218- 233, Feb 2003.

[6] Bay, H., Tuytelaars, T., & Van Gool, L.: "SURF: Speeded Up Robust Features", 9th European Conference on Computer Vision.

[7] Belhumeur,P.N. and Kriegman, D.J.: "What is the Set of Images of an Object Under All Possible Lighting Conditions?", Computer Vision and Pattern Recognition, 1996. Proceedings CVPR, IEEE Computer Society Conference on, vol., no., pp.270-277, 18-20 Jun 1996.

[8] Berg, A. C., Berg, T. L., Malik, J.: "Shape matching and object recognition using low distortion correspondences", CVPR. 2005.

[9] Bindiganavale, Rama., and Norman, I., Badler.: "Motion abstraction and mapping with spatial constraints", Modeling and Motion Capture Techniques for Virtual Environments,

Lecture Notes in Artificial Intelligence, pages 70–82. Springer, November 1998. held in Geneva, Switerland, November 1998.

[10] Bousseau, A., Paris, S., Durand, F.: "User-Assisted Intrinsic Images", Proc. ACM SIGGRAPH, Asia, (2009), 16-19.

[11] Chellappa, R., Wilson, C., and Sirohey, S.: "Human and Machine Recognition of Faces: A Survey", Proc. IEEE, vol. 83, no. 5, pp. 705-740, 1995.

[12] Choi, K. J. and Ko, H.-S.: "On-line motion retargeting", J. Visual. Comput. Animation 11, 2000, 223–243.

[13] Choudhury, B., Chandran, S. and Herder, J.: "A Survey of Image-based Relighting Techniques", Proceedings of the First International Conference on Computer Graphics Theory and Applications, (2006), 176-183.

[14] Comon, P.: "Independent Component Analysis, a New Concept?", Signal Processing, (1994), 36(3), 287-314.

[15] Cortes, Corinna., and Vapnik, Vladimir N.: "Support-Vector Networks", Machine Learning, 20, 1995.

[16] Daugman, J. G.: "How Iris Recognition Works", IEEE Trans. Circuits Syst. Video Technol., vol. 14, pp. 21 2004.

[17] Daugman, J. G.: "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters", Journal of the Optical Society of America A 2, 7 (1985), 1160-1169.

[18] Daugman, J. G.: "Complete Discrete 2-d Gabor Transform by Neural Networks for Image Analysis and Compression", IEEE Transactions on Acoustics, Speech, and Signal Processing 36, 7 (1988), 1169-1179.

[19] Debevec, P.: "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography", Proc. ACM SIGGRAPH, (1998), 199-198.

[20] Debevec, P., Hawkins, T., Tchou, C., Duiker, H. P., Sarokin, W. and Sagar M.: "Acquiring the Reflectance Field of a Human Face", Proc. ACM SIGGRAPH, (2000), 145-156.

[21] Debevec, P., Taylor, C. J., Malik, J.: "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry-and Image-based Approach", Proc. ACM SIGGRAPH, (1996), 11-20.

[22] Drew, M.S., Hel-or, Y., Malzbender, T., Hajari, N.: "Robust Estimation of Surface Properties and Interpolation of Shadow/Specularity Components", Image and Vision Computing, (2012).

[23] Garcia, D.: (2010b) Robust Smoothing of Gridded Data in One and Higher Dimensions with Missing Values. Computational Statistics and Data Analysis 54:1167–1178

[24] Georghiades, A.S., Belhumeur, P.N. and Kriegman, D.J.: "From Few to Many: Illumination Cone Models for Face Recognition Under Variable Lighting and Pose", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.23, no.6, pp.643-660, Jun 2001.

[25] Gleicher, Michael.: "Retargeting Motion to New Characters", SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pages 33–42. ACM SIGGRAPH, AddisonWesley, July 1998. ISBN 0-89791-999-8.

[26] Guerra-Filho, Gutemberg and Biswas, Arnab.: "The Human Motion Database: A Cognitive and Parametric Sampling of Human Motion", In Proc. of the 9th IEEE Conference on Automatic Face and Gesture Recognition (FG), 2011.

[27] Hallinan, P.W.: "A Low-Dimensional Representation of Human Faces for Arbitrary Lighting Conditions", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol., no., pp.995-999, 21-23 Jun 1994.

[28] Hotelling, H.: "Analysis of a Complex of Statistical Variables into Principal Components", Journal of Educational Psychology, (1933), 24(6), 417-441.

[29] Kamarainen, J.K., Kyrki, V., Kalviainen, H.: "Invariance properties of Gabor filter-based features - overview and applications", IEEE Trans. on Image Process. 15 (5) (2006), pp. 1088–1099.

[30] Ke, Y. and Sukthankar, R.: "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors", Proc. Conf. Computer Vision and Pattern Recognition, 2004.

[31] Koenderink, J. and Doorn, A. van.: "Representation of Local Geometry in the Visual System", Biological Cybernetics, vol. 55, pp. 367-375, 1987.

[32] Kyrki, V.: "Local and global feature extraction for invariant object recognition", Ph.D. thesis, Lappeenranta University of Technology. 2002

[33] Kyrki, V., Kamarainen, J.K., Kalviainen, H.: "Content-based image matching using Gabor filtering", In: Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems Theory and Applications. Baden-Baden, Germany, pp. 45–49.

[34] Lee, K.C., Ho, J. and Kriegman, D.: "Acquiring Linear Subspaces for Face Recognition Under Variable Lighting", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.27, no.5, pp.684-698, May 2005.

[35] Lie, Zhang and Samaras, D.: "Face Recognition Under Variable Lighting Using Harmonic Image Exemplars", Computer Vision and Pattern Recognition, 2003.

Proceedings. 2003 IEEE Computer Society Conference on, vol.1, no., pp. I-19- I-25 vol.1, 18-20 June 2003.

[36] Levoy, M., Hanrahan, P.: "Light Field Rendering", Proc. ACM SIGGRAPH, (1996), 31-42.

[37] Liu, C. and Wechsler, H.: "A Gabor Feature Classifier for Face Recognition", Eighth IEEE Internat. Conf. on Computer Vision (2001), pp. 270–275.

[38] Lowe, D.G.: "Object Recognition from Local Scale-Invariant Features", Proc. Seventh Int'l Conf. Computer Vision, pp. 1150-1157, 1999.

[39] Manjunath, B.S. and Ma, W.Y.: "Texture Features for Browsing and Retrieval of Image Data", CIPR TR 95-06, July 1995.

[40] Mcmillan, L., Bishop, G.: "Plenoptic Modeling: An Image-based Rendering System", Proc. ACM SIGGRAPH, (1995), 39-46.

[41] Malzbender, T., Gelb, D., Wolters, H.: "Polynomial Texture Maps", Proc. ACM SIGGRAPH, (2001), 519-528.

[42] Matsushita, Y., Kang, S. B., Lin, S., Shum, H. Y., Tong, X.: "Lighting Interpolation by Shadow Morphing Using Intrinsic Lumigraphs", 10th Pacific Conf. on Computer Graphics and Applications, (2002), 58-65.

[43] Mikolajczyk, K. and Schmid, C. A.: "Performance Evaluation of Local Descriptors", PAMI, 2004.

[44] Nimeroff, J. S., Simoncelli, E., Dorsey, J.: "Efficient Re-rendering of Naturally Illuminated Environments", Proc. 5fth Eurographics Workshop Rendering, (1994), 359-373.

[45] Peter, M. Roth and Winter, Martin.: "Survey of Appearance-based Methods for Object Recognition", Technical Report ICG-TR-01/08, Graz University of Technology, Institute for Computer Graphics and Vision, 2008.

[46] Quinlan, J. R.: "Induction of Decision Trees", Machine Learning 1: 81-106.

[47] Rish, Irina.: "An empirical study of the naive Bayes classifier", IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence.

[48] Rizzolatti G., Craighero L.: "The Mirror-Neuron System", Annu. Rev. Neuroscience, (2004), 169-192.

[49] Sakoe, H. and Chiba, S.: "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, 26(1) pp. 43- 49, 1978, ISSN: 0096-3518.

[50] Shan, S., Gao, W., Cao, B., and Zhao, D.: "Illumination Normalization for Robust Face Recognition against Varying Lighting Conditions," Proc. Int'l Workshop Analysis and Modeling of Faces and Gestures, 2003.

[51] Shashua, A. and Riklin-Raviv, T.: "The Quotient Image: Class-Based Re-Rendering and Recognition with Varying Illuminations," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 129-139, Feb. 2001.

[52] Shum, H. Y., Kang, S. B.: "A Review of Image-based Rendering Techniques", Proc. Int'l Conf. Visual Comm. and Image Processing, vol. 4067, (2000), 2-13.

[53] Tunwattanapong, B., Ghosh, A., Debevec, P.: "Practical Image-Based Relighting and Editing with Spherical-Harmonics and Local Lights", CVMP, (2011), 138-147.

[54] Tuytelaars, T. and Mikolajczyk, K.: "Local Invariant Feature Detectors: A Survey", Foundations and Trends in Computer Graphics and Vision, vol. 3, no. 3, pp. 177-280, 2008.

[55] Werner, T., Hersch, R. D., Hlavac, V.: "Rendering Real-world Objects Using View Interpolation", ICCV, (1995), 957-962.

[56] Xiaoyang, Tan., and Triggs, B.: "Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions", Image Processing, IEEE Transactions on , vol.19, no.6, pp.1635-1650, June 2010.

[57] http://www.ignorancia.org/en/

[58] http://www.ignorancia.org/en/index.php?page=About_me

[59] http://www.ignorancia.org/en/index.php?page=Childhood

[60] http://www.ignorancia.org/en/index.php?page=Gardens

[61] http://www.ignorancia.org/en/index.php?page=Patio

[62] http://www.ignorancia.org/en/index.php?page=The_office

[63] http://megapov.inetart.net/

[64] http://objects.povworld.org/objects/cat/Engineering/

[65] http://www.povray.org/

[66] http://www.povray.org/documentation/

[67] http://ranger.uta.edu/~guerra/CVPoV.html

[68] http://www.vlfeat.org/~vedaldi/code/vlpovy.html

BIOGRAPHICAL INFORMATION

Vishnukumar Galigekere N was born in Bangalore, India, in 1979. He received his B.Sc. (Electronics) from Bangalore University, Bangalore, in 2001 and his Masters in Computer Applications from Visvesvaraya Technological University, Bangalore, in 2004 and M.S. (Computer Science) from The University of Texas at Arlington in 2008.

His current research interest is in Object Recognition, Action/Gesture Recognition, Image Descriptors, and Machine Learning methods for Computer Vision.