THEORY AND APPLICATIONS OF STEERABLE FUNCTIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Patrick Cheng-San Teo
March 1998

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. David J. Heeger
(Principal Adviser)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Carlo Tomasi

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Guillermo Sapiro
(University of Minnesota)

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies & Research

Preface

A function is called *steerable* if transformed versions of the function can be expressed using linear combinations of a fixed set of basis functions. For example, translated versions of a sinusoid can always be expressed as linear combinations of a sinusoid and a cosinusoid of the same frequency. Likewise, rotated versions of the first derivative of a two-dimensional Gaussian can always be expressed as linear combinations of the first derivatives of the Gaussian along the x and y axes. Steerability is a very general and often desirable property. As a result, steerable functions have recently been applied to an assortment of problems in image processing, computer vision, and computer graphics. Over the years, a moderate amount of work has been done both in increasing our understanding of steerable functions as well as in developing algorithms to construct them. Nevertheless, many of these earlier works usually confine their investigations to functions that are steerable only under specific transformations.

In this thesis, we propose a framework, based on Lie group theory, for studying and constructing functions steerable under any smooth transformation group. We argue that Lie theory is the appropriate mathematical tool for analyzing the properties of these steerable functions. This position is supported by the fact that all existing analytical approaches to steerability can be consistently explained within the framework. The framework provides a general technique for identifying and constructing functions steerable under any transformation group. In particular, the framework was used to derive a canonical form for all functions steerable under any one-parameter or any Abelian multi-parameter transformation group.

The design of a suitable set of basis functions given any arbitrary steerable function is one of the main problems concerning steerable functions. To this end, we have developed two very different algorithms. The first algorithm is a symbolic method that can be implemented in any symbolic package. Typically, the basis functions of a steerable function are derived by inspection; this algorithm derives the minimal set of basis functions automatically given an arbitrary steerable function. The second algorithm addresses two practical considerations: approximate steerability and local steerability. In practice, functions that need to be steered might not be steerable with a finite number of basis functions. Moreover, it is often the case that only a small subset of transformations within the group of transformations needs to be considered. In response to these two concerns, the second algorithm computes the optimal set of k basis functions to steer an arbitrary function under a subset of the group of transformations.

Lastly, we demonstrate the usefulness of steerable functions in a variety of applications. In particular, we present five applications that use steerable functions: (1) continuum approximation in vision modeling (a method of approximating an infinite number of interacting mechanisms in a model), (2) the design of optimal steerable filters for gradient-based motion estimation, (3) efficient linear re-rendering of synthetic scenes under changes in illumination, (4) the construction of invariants from steerable filters, and (5) the application of steerable functions to discrete sets of points and lines.

Acknowledgements

I am extremely grateful to my advisor, David Heeger, who had confidence in me from the very beginning, gave me this wonderful opportunity, and supported my research over the years. David has been an excellent academic advisor, a patient and understanding counselor, and above all, a generous friend. I would also like to thank Brian Wandell for being my unofficial second advisor and mentor, and for making me feel so welcomed in his lab. I will always treasure his advice, guidance, and friendship.

My graduate life would have been very different if not for my friend and colleague, Yacov (Toky) Hel-Or. It has been an absolute pleasure working with him: I will miss the energetic discussions we used to have so regularly and his addictive passion for research. My heartfelt thanks to Toky for his friendship and for so many other things. I would also like to thank my friend and colleague, Guillermo Sapiro, for his generosity, abundant concern for my welfare, and unfailing friendship. My gratitude to him for a very memorable working experience and for sharing his insight and enthusiasm for research with me. I am also thankful for the many opportunities I have had that would not have been possible if not for him.

I would like to thank the members of my reading committee: David Heeger, Carlo Tomasi, and Guillermo Sapiro. I would also like to take this opportunity to thank the members of my orals committee: Brian Wandell, David Heeger, Carlo Tomasi, Leonidas Guibas, and Guillermo Sapiro.

I value the many summers I have spent working in industry. My thanks to my colleagues at the SRI Computer Science Laboratory; in particular, Caveh Jalali and Chris Dodd. My thanks to my colleagues at Apple Computer ATG: Lance Williams, Michael Chen, and, in particular, Eric Chen. To my colleagues at Xerox PARC, Eric

Saund, David Marimont and Michael Black, thanks for all the interesting discussions we had. I especially enjoyed my summer at Hewlett Packard Laboratories; many thanks to Dan Lee, Tom Malzbender, Poorvi Vora, Daniel Keren, and especially, Joyce Farrell. My thanks to many others whom I have had the pleasure of knowing: Al Ahumada, Andrew Watson, Brent Beutter, Preeti Verghese, Tjeerd Dijkstra, David Fleet and Leif Haglund. My special thanks to Eero Simoncelli for all the numerous interesting discussions we had and for simply taking the time to talk and share his insights with me.

Many thanks to my friends in the lab, Hagit Hel-Or, Xue-Mei Zhang, Geoff Boynton, David Hoffman, Sean Stromsen, and Tina Tian for making the lab such a fun place to be. To Chase Garfinkle, Chye-Hwang Yan, Maria Tovar, and Anuchit Anuchitanukul, my fellow graduate students, my thanks for the numerous interesting discussions we had. My heartfelt thanks to my good friends Jean Wang, Christine Nguyen, Sumalee (Jam) Kiattinant, Sharon Perlmutter, and Keren Perlmutter for their invaluable friendships and their tireless support throughout the years upon which I have depended. I am also deeply grateful to my dear friend, Teoh Cher Sun, for her support and encouragement, and especially for her patience and dedication in our friendship all these years.

Finally, I would like to express my immense gratitude to my family, my brother, Ivan Teo Cheng Tien, my mother, Doris Tan, and my father, Allen Teo Lian Hin, for their unwavering faith in me and their innumerable sacrifices, without which I would not have had this great opportunity. They have always been my rock during turbulent times and my compass during times of despair. I am indeed fortunate to have such a wonderful family and I thank God for this blessing.

Dedicated to my parents.

Contents

Preface iv

Acknowledgements vi

1 Introduction 1

1.1 Applications of Steerable Functions 4

1.1.1 Image Analysis 4

1.1.2 Motion Estimation 5

1.1.3 Invariant Pattern Recognition 7

1.1.4 Computer Graphics 8

1.2 Survey of Existing Research 10

1.2.1 The Numerical Approach 10

1.2.2 The Analytical Approach 12

1.3 Contributions of this Thesis 15

1.4 Lie Theory in Related Areas 17

2 Concepts and Mathematical Preliminaries 19

2.1 Lie Groups 20

2.2 Lie Algebras 23

2.2.1 Tangent Space 23

2.2.2 Vector Fields 28

2.2.3 Lie Bracket 30

2.3 Steerable Functions 33

2.3.1 Definition 33

2.3.2 Construction of Equivariant Function Spaces 36

3 Canonical Decomposition **37**

3.1 Construction of Equivariant Function Spaces 38

3.2 Function Spaces for One-Parameter Groups 40

3.2.1 The Translation Group 41

3.2.2 The Rotation Group 42

3.2.3 Canonical Coordinates of One-Parameter Transformation Groups 44

3.2.4 Canonical Decomposition of One-Parameter Equivariant Spaces 45

3.3 Function Spaces for Multi-Parameter Groups 51

3.4 Summary 55

4 Generator Trees **57**

4.1 Generator Chains 58

4.2 Generator Trees 63

4.3 Simulations 67

4.3.1 Steering Polynomials 67

4.3.2 Numerical Simulations 69

4.4 Discussion 71

4.5 Summary 73

5 Cascade Basis Reduction **77**

5.1 Local Steerability 79

5.2 Cascade Basis Reduction 82

5.2.1 Singular Value Decomposition 82

5.2.2 Basis Reduction 83

5.2.3 Basis Reduction using Equivariant Function Spaces 84

5.2.4 Analytic Form of Basis and Steering Functions 85

5.3 Results 86

5.3.1 Comparison with Conventional SVD 86

5.3.2 Steering a Gabor Function under General Linear Transformation 88

5.4 Summary 89

6 Applications **93**

6.1 Continuum Approximation in Modeling Vision 94

6.2 Optimal Filters for Motion Estimation 100

6.3 Steerable Light Sources 104

6.4 Invariants from Equivariant Function Spaces 109

6.5 Equivariance of Points and Lines 117

7 Conclusions **120**

Bibliography **123**

List of Tables

1 Several examples of one-parameter transformation groups and their infinitesimal generators. 25

2 Several examples of one-parameter transformation groups and their equivariant function spaces. The parameter k is any integer while the parameter α is any complex number. The function h is any arbitrary function. The variables r, θ refer to polar coordinates. The set notation describes a set of functions indexed by p for $0 \leq p \leq k$. 50

3 Several examples of multi-parameter groups and their equivariant measuring spaces. The parameters $p, q, m, l, k \in \mathbf{Z}$ and $\alpha, \beta \in \mathbf{C}$. 54

List of Figures

1 The relationship between the Lie group and its tangent space (Lie algebra) is due to the exponential map, which maps an element in the tangent space onto an element of the group in the neighborhood of the identity. 23

2 The vector field representing the infinitesimal generator of the one-parameter group of rotations in the plane, the infinitesimal generator being $L_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$. 29

3 The vector fields associated with the infinitesimal generators of the two-parameter group of translations in the plane, the infinitesimal generators being $L_{t_x} = \frac{\partial}{\partial x}$ and $L_{t_y} = \frac{\partial}{\partial y}$. 29

4 The vector field associated with the element $(\tau_0\frac{\partial}{\partial x} + \tau_1\frac{\partial}{\partial y})$ of the tangent space of the two-parameter group of translations in the plane. The vector field is shown as a linear combination of the vector fields associated with the infinitesimal generators. 30

5 Closure of the equivariant function space with respect to the group transformation is equivalent to closure of the function space with respect to the infinitesimal generators of the group. 38

6 Recipe for verifying that the function space of span(Φ) is equivariant. If so, the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$ is also derived. 40

7 Two equivariant function spaces span the same function space if and only if their corresponding $\boldsymbol{B_i}$ matrices are similar. 46

8 Procedure for computing the basis functions to steer an arbitrary function f under a one-parameter group. 62

9 Procedure for computing the basis functions to steer an arbitrary function f under a k-parameter group. The nodes in the generator tree are tested in a breadth-first manner. 65

10 Basis functions that steer $(12x - 8x^3)\exp[-(x^2+y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure. . . 70

11 Orthogonal basis functions that steer $(12x - 8x^3)\exp[-(x^2+y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure. 70

12 Basis functions that steer $\sin(x)\exp[-(x^2+y^2)]$ under rotation. The leftmost image is the function that was used as input to the procedure. 71

13 Orthogonal basis functions that steer $\sin(x)\exp[-(x^2+y^2)]$ under rotation. The leftmost image is the function that was used as input to the procedure. 71

14 The 22 basis functions that steer $(4x^2-2)\exp[-(x^2+y^2)]$ under $x, y-$ translation and rotation. The image at the top left is the function that was used as input to the procedure. 74

15 The 22 orthogonal basis functions that steer $(4x^2-2)\exp[-(x^2+y^2)]$ under $x, y-$ translation and rotation. The image at the top left is the function that was used as input to the procedure. 75

16 Graph (a): Relative squared errors of the steered approximations to the actual functions over a range of x-translations. The graph describing the errors with respect to the y-translations is virtually identical. Graph (b): Relative squared errors of the steered approximations to the actual functions over the entire range of rotation angles. The actual function has been translated by 0.5 units in both the x and y dimensions; i.e., $(4(x-0.5)^2-2)\exp[-((x-0.5)^2+(y-0.5)^2)]$. The percentage errors for an untranslated function are negligible. 76

17 The support of the raised cosine is within the interval $[-1,1]$. If the function is to be steered in translation over the range $[-1,1]$, then the *integration region* corresponding to a steerable filter would be $[-2,2]$. The interval over which the raised cosine needs to be approximated, i.e. the *approximation region* is $[-3,3]$. 81

18 (a) Reconstructions of translated replicas of the original function using 10 basis functions. (b) Basis functions corresponding to the three largest singular values computed using the cascade basis reduction method. 86

19 Graph (a) plots the analytic form of the basis function with the largest singular value. The asterisks represent the corresponding discretely sampled basis function computed using the conventional SVD method. Graph (b) plots the analytic form of the steering function for the basis function with the largest singular value. The asterisks represent the corresponding discretely sampled steering function computed using the conventional SVD method. 87

20 Graph (a) plots the magnitude of the singular values for each singular vector. Each singular vector corresponds to a single basis function. A total of 231 singular vectors were present but only the largest 30 of them are plotted. Graph (b) plots the cummulative sum of the squared magnitudes of the singular values. The squared magnitudes of the singular values have been normalized so that their sum equals one. 89

21 (a) Ten out of the eleven basis functions (99.9% total squared norm) computed to steer the odd-phase Gabor under any local linear transformation. The basis functions are arranged in descending order of the magnitudes of their singular values from left to right and from top to bottom. (b) Image (i) shows a reconstruction of the original function. Image (ii) shows a reconstruction of the function rotated by 60 degrees. Image (iii) shows a reconstruction of the function scaled along the x-axis. Image (iv) shows a reconstruction of the function scaled along the y-axis. Image (v) shows a reconstruction of the function skewed along the x-axis and uniformly scaled. All of these functions were reconstructed using the 11 basis functions. 90

22 (a) Eight basis functions (99.9% total squared norm) computed to steer the even-phase Gabor under any local linear transformation. The basis functions are arranged in descending order of the magnitudes of their singular values from left to right and from top to bottom. (b) Images (i) through (v) show reconstructions of the original function under various linear transformations. See the caption of Figure 21 for further description. 91

23 Overview of the human spatial pattern detection model. The model consists of three major components: a retinal component, a cortical component and a decision mechanism. The model simulates a discrimination task; the output of the model is a number representing the discriminability between the two input images. 94

24 Schematic of contrast normalization in the cortical component of the model. Contrast normalization involves: (1) computing energy responses as the sum of the squared responses of a quadrature pair of linear responses, and (2) divisive normalization, that is, dividing each energy response by a sum of other energy responses. 95

25 Summary of model's fits to contrast masking data with maskers of different orientations. Empirical data from Foley and Boynton [Fol94, FB94] is represented by filled circles while the model's predictions are described by the solid lines. 98

26 (Top) Original Einstein image. (Middle-left) Image was distorted so as to minimize perceptual distortion (RMSE = 9.01, peak-SNR = 29.04 dB). (Middle-right) Image was distorted so as to maximize perceptual distortion (RMSE = 8.50, peak-SNR = 29.54 dB). While the left image looks generally less distorted than the right image, it has a larger root mean squared error. (Bottom-left) Perceptual distortion measured from the minimally distorted image. Darker regions correspond to areas of lower perceptual distortion while brighter regions indicate areas of greater perceptual distortion. (Bottom-right) Perceptual distortion measured from the maximally distorted image. 99

27 Continuous profiles $h(x), g_1(x), g_2(x)$ of the optimal set of 5-tap filters such that $g_1(x) = \frac{\partial}{\partial x} h(x)$ and $g_2(x) = x \frac{\partial}{\partial x} h(x)$. The continuous profiles are created by interpolating the digital filters using a narrow Gaussian interpolant. The optimal digital filters are $\boldsymbol{h}$ = (0.0167, 0.1847, 0.3738, 0.1847, 0.0167)^T, $\boldsymbol{g_1}$ = (0.0602, 0.2675, 0.0000, -0.2675, -0.0602)^T, and $\boldsymbol{g_2}$ = (0.2959, 0.3366, -0.4893, 0.3366, 0.2959)^T. 101

28 Images of a scene illuminated by a directional spot light. The angular radiance distribution of the light source is a fifth degree polynomial. Each of the images were re-rendered by linearly combining a set of 12 basis images. The left and middle images show the scene re-rendered with the spot light pointed in different directions. The right image shows the scene re-rendered with two spot lights in the same position, but pointing in different directions. 107

29 Images of a chess piece illuminated by an area light source. The left image shows the scene re-rendered with the area light source positioned to the front and left of the object. The middle image is a re-rendering of the scene with a broader area light source. The right image shows the object illuminated by three primary colored lights. A total of 20 basis images were used to re-render all three images. 107
30 Images of a single polygon illuminated by an area light source that has an anisotropic angular distribution. The left image shows a re-rendering with the light source pointing downwards, and positioned to the rear and left of the object. The middle image shows a re-rendering with the light source in the same position as before but pointing in a different direction. The right image is a re-rendering with the light source centered at a different position. A total of 50 basis images were used to re-render all three images. These basis images were computed using the singular value decomposition; the actual number of basis images required was 400. 112
31 Orbits of a two-dimensional function space equivariant under a one-parameter transformation group. Each curve (circle) is an orbit representing some fixed value of the invariant $h(f_1, f_2) = f_1^2 + f_2^2$. 112
32 Orbits of a three-dimensional function space equivariant under a one-parameter transformation group. The space of invariants is two-dimensional. Each one-dimensional space curve is an orbit representing some pair of fixed values of the invariants $h_1(f_1, f_2, f_3) = f_3$ and $h_2(f_1, f_2, f_3) = f1f3 - \frac{1}{2}f_2^2$. Each curve in (a) is specified by $h_1 \in [-1, 1]$ and $h_2 = 0.5$. Each curve in (b) is specified by $h1 = 0.5$ and $h2 \in [-1, 1]$. 119

Chapter 1

Introduction

The goal of image understanding is to develop methods that are capable of making inferences about the 3D scene from 2D images. These 2D images are typically made up of samples of irradiant intensity or depth. The first phase of any image understanding system, quite commonly, involves data reduction. The amount of information available in an image is often unmanageable directly; more importantly, task-specific information required by subsequent stages in an image understanding system is usually not explicitly available. As a result, the data reduction phase is responsible for converting the original data into a more computationally efficient and explicit representation, which may not be, and is often not, parsimonious. As far as the rest of the image understanding system is concerned, however, these new re-represented inputs are its sole measurements of the 3D scene.

The data reduction phase is often associated with feature detection and extraction. The most ubiquitous method of feature detection involves linear filtering; this method has its roots in matched filtering or template matching in 1D signal processing. A template of the feature to be detected is first constructed and correlated with the input image. The resulting image of correlation values is then examined for locations bearing high correlation. For example, if the feature to be detected is an intensity step edge, the partial derivative of a Gaussian would be an appropriate template. However, using only one partial derivative, say the partial derivative in the x-direction, is not enough. In this case, the correlation result only computes the regularized partial derivative of the input image in the x-direction, which only indicates the presence or absence of a vertical edge. This quantity gives no information about the presence or absence of a horizontal (or any off-vertical) edge. Furthermore, a detector, using only this quantity, would be unable to discriminate between the presence of a strong off-vertical edge and the presence of weak vertical edge.

A brute-force, and often computationally inefficient, method would be to use multiple templates, each being a partial derivative in a different direction; the template yielding the maximum response would then be considered as indicating the presence of an intensity edge in its corresponding orthogonal direction. However, elementary calculus informs us that the partial derivative along any direction of a two-dimensional function can always be written as a linear combination of the partial derivatives of the function along two linearly independent directions. Thus, it would be unnecessary to use more than two templates since each of the other templates could be written as a linear combination of the first two basis templates (that are linearly independent). More importantly, because correlation is a linear operation, the *result* of correlating any of the other templates could be expressed as a linear combination of the results of the basis templates [FA91].

Steerable functions are a special class of functions characterized by a generalization of the above property. Specifically, a function is said to be steerable if transformed versions of the function can always be expressed as a linear combination of a fixed, finite set of basis functions. The weights of the linear combinations are dependent on the parameters of the transformation.[1] In the previous example, the function in question would be the derivative of the Gaussian, and the transformation would be rotation; as such, two basis functions, corresponding to the partial derivatives along any two linearly independent directions, are sufficient. Mathematically, $G'_\theta(x,y) \equiv \cos(\theta)G'_x(x,y) + \sin(\theta)G'_y(x,y)$ where G'_θ, G'_x, G'_y denote the first partial derivatives in the θ-direction, in the x-direction, and in the y-direction respectively.

Transformations other than rotation are also considered in the definition. For

[1] Other names have been used to describe functions possessing this property with different restrictions. These other definitions will be highlighted when we review the existing work that has been done in this area.

example, a sinusoid is steerable with respect to translation since a translated sinusoid is expressible as a linear combination of a fixed sinusoid and cosinusoid. That is, $\sin(x + \delta x) \equiv \cos(\delta x)\sin(x) + \sin(\delta x)\cos(x)$ where the basis functions are $\sin(x)$ and $\cos(x)$, and the weighting or steering functions are $\cos(\delta x)$ and $\sin(\delta x)$ respectively. The two previous examples deal with one-parameter transformations; multi-parameter transformations like a combination of rotation and uniform scaling in the plane or Euclidean transformation (rotation plus $x-$ and $y-$ translation) are also included in the definition. One example of a function steerable under Euclidean transformation is the two-dimensional parabola, $(x/a)^2 + (y/b)^2$, because any transformed replica of it can always be expressed as a linear combination of the monomials $x^i y^j$ up to second degree.

The definition of a steerable function implies *exact* steerability; that is, the transformed function must be expressible as a linear combination of its basis functions in analytic form. In practice, steerable functions and their basis functions are almost always represented in sampled form, and this exact requirement is often relaxed and some error is tolerated. Thus, it is often sufficient that the function be only approximately steerable; that is, it can be described in terms of a linear combination of basis functions, allowing for some amount of error. Consequently, one might want to compute the best k basis functions to steer a given function under some transformation as opposed to finding all the basis functions when a large number or an infinite number of them might be needed. Although the definition suggests that the transformation applied to the function can be any transformation within the family of transformations, for example, any uniform scaling of the function, in practice, a more local requirement of steerability is often sufficient. Following the same example, it is possible that only a certain moderate amount of scaling is expected. Thus, in contrast to the original global definition of steerability, a more local version might be more useful in practice. These practical subtleties will be dealt with more precisely in subsequent chapters.

1.1 Applications of Steerable Functions

In this section, we present several different applications in which steerable functions have been used as motivation for the study of this special class of functions. These applications are from a wide variety of areas that include image analysis, motion estimation, invariant pattern recognition, and computer graphics. In several of these areas, the use of steerable functions has not traditionally been recognized. We discuss these examples in some detail and demonstrate that the steerability property is always implicitly assumed.

1.1.1 Image Analysis

Steerable functions have found application in numerous areas, the most popular of which is in image analysis. In [FA91, Per92, SF95a, Len90b], various methods are described for detecting local image features, like edges, junctions, and corners, at different orientations and scales. These methods employ *steerable filters*, which are simply linear filters whose kernels are steerable functions. Because correlation is linear, the outputs of these steerable filters are a linear function of the outputs of its basis filters. The transformation of the steerable filter yielding the maximum response is usually determined either analytically or via numerical optimization, and used as an indication of the presence or absence of the particular structure. Several others have employed steerable filters within the framework of a multi-resolution pyramid decomposition [SFAH92, SP94, GBG+94, MP95]. These representations are usually over-complete and invertible. In one such decomposition [SFAH92], the image is locally decomposed into multiple, octave-spaced, spatial frequency bands; each of these bands are then further decomposed using a set of basis filters corresponding to the set of basis filters of an orientation-steerable filter. If two such pyramids are constructed, one with even-phased filters and the other with odd-phased filters, then the decomposition analyzes the local image content into octave-spaced spatial-frequency bands selective to both orientation and phase. Such local image analysis is useful for a variety of applications including texture discrimination and segmentation [Sap96, GBG+94] and human vision modeling [SH98, TH94a].

1.1.2 Motion Estimation

Steerable filters have also been used to estimate optical flow in image motion sequences. Since image motion can be understood as orientation in the spatio-temporal domain [AB85, WA85], motion estimation can be implemented using local orientation estimators in 3D, thus, naturally extending the 2D image analysis techniques described above [Hee87, HS93]. Although in numerous other motion estimation techniques, the use of steerable filters is not explicitly noted, the requirement that the motion estimation filter be locally steerable is always implicitly assumed. This can be readily seen in the following common first-order linearization of the optical flow constraint: $I^{t+1}(\boldsymbol{x}) = I^t(\boldsymbol{x} + \triangle\boldsymbol{x}) \approx I^t(\boldsymbol{x}) + \triangle\boldsymbol{x} \cdot \nabla I^t(\boldsymbol{x})$, where I^t and I^{t+1} represent the images at time t and $t+1$ and ∇I^t denotes the spatial gradient of image I^t. Thus, a translation of image I^t is described as a linear combination of a set of three basis functions comprising itself and its partial derivatives in the $x-$ and $y-$ directions. In practice, this constraint is typically implemented using filters. One can show that an equivalent condition on the filters needs to be satisfied.

Recent research in the design of accurate optical flow algorithms stress the importance of choosing appropriate basis functions [Sim94]. In particular, [XS95b, XS95a] suggest the use of moment and hypergeometric filters as basis filters; also, [LHHC94] describes an accurate optical flow technique using Hermite polynomials. When the motion estimation filter is quite oscillatory, i.e., it has a narrow pass-band, like a Gabor filter, for instance, then translating it by a small amount gives rise to a function that can be approximated by a linear combination of an odd-phased and an even-phased version of it. As a result, translation is also sometimes described as phase (relative to the dominant frequency of the filter being used, or more accurately, to the dominant frequency of the signal present within the pass-band). Since these filters are band-pass in nature, they have an added advantage of being insensitive to illumination changes between frames. As such, various researchers have promoted the use of such filters in optical flow estimation algorithms [Fle90, Wen93].[2] The various basis

[2]While the phase of a complex band-pass filter is relatively insensitive to illumination changes, the use of complex phase in estimating motion requires some care when the magnitude of the filter response is small. This is because the estimate of the phase of the filter response becomes unreliable.

filters described above have been chosen usually for their computational efficiency or their relative insensitivity to noise. In [ETHO97], the authors describe a method of designing a least-squares optimal set of basis filters for measuring optical flow that takes into consideration the desired size of the filters, the expected image spectrum, and the distribution of optical flow vectors.

Steerable filters have also been used in generalizations of the basic optical flow problem. One common direction involves extending the translation motion model to an affine motion model. In this case, the number of parameters being estimated is six instead of two. Nevertheless, local steerability of the motion estimation filter is still required with respect to the affine motion model; the weights in the steering equation are now a function of the six motion parameters, and typically, more basis functions are necessary. In [XS95a] and [LHHC94], the authors show that hypergeometric filters and Hermite polynomial filters, respectively, can be used to estimate the affine motion parameters as well. In [MO93, Man94], the authors show how affine transformation parameters can be estimated using derivative of Gaussian filters with possibly elliptical kernels.

The ability to estimate local image distortion is useful to numerous other different applications in computer vision. For example, Malik and Rosenholtz [MR93, MR94] describe a method to compute the orientation and curvature of a textured surface from estimates of the affine transformation of the local power spectrum. Using a similar technique, the authors in [SC94b, SC94a] describe a technique to compute shape from estimates of the affine transformation of the local feature density. In [TSG94], Dijkstra *et al.* present a method to estimate local shape parameters directly from optical flow measurements. The stereo problem can also be considered as a one-dimensional analogue of the optical flow estimation problem. If disparities are kept small, one-dimensional optical flow techniques could be used. This is usually accomplished by computing the disparity in a coarse-to-fine manner as demonstrated in [FJ91, XS94]. The authors in [XS94] also used the same filters to estimate shape from focus.

A thorough analysis and solution of the problems involved in estimating motion from local phase information is presented in [Fle90].

1.1.3 Invariant Pattern Recognition

Many invariant pattern recognition systems begin by first computing several integrals over the pattern using different weighting functions; the result of these integrations are then combined such that they are invariant to some transformation. These invariant quantities could then be used in a classifier system to recognize the input pattern from a database of stored patterns. The most common of these integral transforms is the Fourier transform. It is well known that the power spectrum of the Fourier transform of any pattern is invariant to translations of that pattern. More precisely, the magnitude of each independent complex exponential in the transform is translation invariant. This can be explained in a variety of ways. In the context of steerable filters, consider the complex exponential that could be implemented using the filters $\sin(x)$ and $\cos(x)$, which are steerable with respect to each other. Any translation of the pattern is equivalent to an opposite translation of the filters; however, any translation of $\sin(x + \triangle x) \equiv \cos(\triangle x)\sin(x) + \sin(\triangle x)\cos(x)$, and a similar identity holds for $\cos(x)$. Therefore, if the outputs of integrating the pattern with the original filters were s and c, the corresponding outputs with a translated pattern would be $\cos(\triangle x)\, s + \sin(\triangle x)\, c$ and $-\sin(\triangle x)\, s + \cos(\triangle x)\, c$ respectively. Thus, the sum of the square of the outputs of the two filters is translation invariant; that is, independent of $\triangle x$.

The Fourier-Mellin transform can be viewed as a reparameterization of the Fourier transform such that rotation and scaling correspond to x- and y-translation. As a result, the magnitude of each complex coefficient of the Fourier-Mellin transform is independent of rotation and scale. A similar explanation in terms of steerable filters could also be made in this case; however, the filters and the steering functions would be different. In [FC88, RSZ91, SRZ92], the authors showed that it is possible to generalize this Fourier transform technique to other groups of transformations provided each of the transformations commute. Instead of forming invariants from the coefficients of single complex coefficients, it is also possible to form invariants using pairs of complex coefficients. This was demonstrated in [Sim96] where the author constructed such invariants over rotation.

Another important class of invariants found in pattern recognition are the algebraic or moment invariants. These are invariants that are formed by taking algebraic combinations of image moments, which are integrals of the image with polynomials. The most well-known of these invariants are the seven translation, scale and rotation invariants proposed by Hu [Hu62]. Using a similar technique, Reiss derived the corresponding affine moment invariants [Rei91]. In the context of steerable filters, computing image moments corresponds to integrating with steerable polynomial filters; in fact, it is easy to show that the polynomials are steerable with respect to affine transformations, and the set of moments used to construct the invariants form a complete set of basis functions.

Apart from these rather specific classes of invariants, the method of normalization is a general method of constructing invariants that is independent of the choice of linear filters. A pattern recognition system using this technique first derives features from the input pattern using a set of linear filters or some other choice of feature extraction mechanism. These features are then used to solve for the transformation that would put the input pattern into a canonical pose. This transformation is next applied to the input pattern; template matching of the input pattern in its canonical position is subsequently used to determine if the input pattern is present in the database of canonical templates. This technique can be used to derive a wide class of invariants easily [RSV96]. A theoretical account of the method can be found in [Ama68, Ama78]. The authors in [RSV96] provide a modern account of the technique along with several examples.

There are numerous other methods to construct invariants for pattern recognition, many differ by the choice of linear filters that are used to integrate with the input pattern. A good review of these invariants can be found in [Rei93, Woo96].

1.1.4 Computer Graphics

Steerable filters have recently been applied to computer graphics. One application is in the area of texture mapping. Texture mapping is a method of describing the surface property (e.g. color) of a model using an image. During rendering, the surface property of the model at a point is determined by sampling its texture at the point's corresponding texture coordinates. To avoid aliasing, point sampling should not be used; instead, an area-weighted average should be computed such that the shape and size of the filter kernel is dependent on the viewing and model geometry (and thus shift-variant). However, computing these weighted-averages during rendering is computationally expensive, especially if the size of the filter kernel is large; as a result, several solutions have been proposed to precompute these averages [Wil83, GH86] (see [Hec86] for a review). The sampling kernel can be implemented using a steerable function such that view-dependent transformations of it are expressible using a linear combination of a fixed set of basis kernels. These basis kernels are applied to the image texture prior to rendering, and during rendering, a linear combination of the outputs of the basis kernels is computed. A method applying this technique to a Gaussian sampling kernel is proposed in [Got94].

In a very different application, steerable functions have been used to efficiently re-render a synthetic scene subject to illumination changes [NSD94, DKNY95, DAG95, TSH97]. With this method, the scene is re-rendered after a change in illumination by linearly combining a set of basis images. The validity of this approach rests on a fundamental property of graphical rendering: linearity with respect to light source intensities. If the intensity distribution of the light source is modeled using a steerable function (i.e. a *steerable light source*), then transformations of that distribution (e.g. changes in orientation of a radial distribution) can be expressed as a linear combination of a set of basis intensity distributions. Since an image is linearly related to the intensity distribution of its illuminant, re-rendering the image with the same view-point but under a change of illumination (corresponding to a steerable transformation of the original intensity distribution) amounts to a linear combination of the basis images, i.e., the images rendered using the basis intensity distributions. This example demonstrates the wide applicability of steerable functions. Steerable filters and steerable light sources share one common characteristic: linearity of their respective operators (convolution and rendering respectively) allows steerability to be applied to the outputs of these operators. In fact, this extension is not limited to linear operators; algebraic operators also permit steerability of their outputs.

Finally, steerable filters have also been used in texture synthesis. In [HB95], a pyramid of steerable filters was used to analyze a sample of texture, and then to synthesize that texture in a destination image.

1.2 Survey of Existing Research

In this section, we review most of the existing results on constructing steerable functions. The body of literature can be divided into two categories: the numerical approach and the analytical approach. The numerical approach treats the problem as one of numerically computing the optimal set of basis and steering functions for a given function and family of transformations; optimality is measured using some function norm, typically the L^2 norm. Although this method is applicable to almost any choice of transformations, it tends to be computationally infeasible for transformations that are described by a moderate number of parameters. The analytical approach studies the problem by first identifying basis functions that are analytically steerable; for example, sinusoids under translation. The function to be steered is then approximated with a linear combination of these basis functions, and steered by steering the basis functions. As a result, the basis and steering functions can be written in analytic form.

1.2.1 The Numerical Approach

The numerical approach was pioneered by Perona [Per92, Per95]. The primary application of steerable functions for this author was its use in filters designed to detect local image structures like intensity edges and junctions such that the detection was independent of some transformation; e.g. rotation and scale. Analytically, the set of all transformed replicas of the function to be steered is treated as a compact linear operator for which a suitable discrete spectral representation was to be computed; the author showed that under certain conditions, the operator posseses a discrete spectrum. This discrete spectral representation provides a set of optimal basis and steering functions that could be used to steer the given function. In the case of steering a function over rotation, the author showed that the basis and steering functions can be derived analytically when the function to be steered is described by a Fourier series.

Nevertheless, in practice, the method is almost always implemented numerically by computing the singular value decomposition of a particular matrix $\boldsymbol{F}$ such that $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ where $\boldsymbol{S}$ is a diagonal matrix of positive singular values, $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$. This matrix corresponds to a sampled representation of the compact linear operator described above; sampling is carried out over both the spatial coordinates (i.e. the steered function's coordinates) and the transform parameters. Each column of $\boldsymbol{F}$ contains a spatially sampled representation of a particular transformed version of the function to be steered; thus, the columns of $\boldsymbol{F}$ enumerate all transformed replicas of the steered function. Likewise, the columns of $\boldsymbol{U}$ contain the orthogonal basis functions in sampled form, while each column of $\boldsymbol{V}$ contains a sampled version of the steering function associated with each basis function. The diagonal entries in $\boldsymbol{S}$ are typically arranged in decreasing order of magnitude such that $s_{ii} \geq s_{jj}$ for $i < j$. Setting the smaller singular values in $\boldsymbol{S}$ to zero effectively selects the first k basis and steering functions. These k basis and steering functions correspond to an optimal least-squares set.

There are several variants of this approach. In [SP94], the authors proposed a method of constructing an optimal set of basis functions that are also spatially separable. Spatially separable filters are computationally efficient since convolution with a multi-dimensional separable filter could be implemented as a sequence of one-dimensional convolutions. The basis functions and their steering functions were computed using iterative minimization. The authors in [GBG+94] demonstrated that an approximate steering equation can be computed for a fixed set of basis functions. Namely, the basis functions were from an efficient implementation of several orientation sensitive band-pass filters that were applied to each spatial frequency band of a Laplacian pyramid. Thus, the authors were able to steer the orientation sensitive band-pass filters over rotation. The obvious requirement here is that the filters be complete (or overcomplete). Finally, in [MP95], the basis filters to steer a given function over a large range of scales (e.g. two octaves) is computed such that the basis filters can be implemented recursively in a pyramidal framework; that is, the set of basis filters is divided into subsets such that the filters in each subset are applied to an increasingly low passed and decimated version of the original image. Since decimation is involved, the authors also minimize the amount of spatial aliasing in the system by optimizing over all (necessary) translates of the steered function. The entire optimization problem is solved iteratively.

The main advantage of this approach is that it is applicable to almost any transformation in practice. The method is the easier of the two approaches to understand and for many problems, its implementation simply involves computing the singular value decomposition of a matrix. Its main disadvantage is the computational inefficiency or infeasibility of using the technique to construct accurate basis functions over transformations involving more than two parameters. In cases where the transformations are also groups, this method does not explicitly take advantage of the many useful theoretical results of transformation groups. Finally, the basis and steering functions are also typically computed in sampled form.

1.2.2 The Analytical Approach

The analytical approach precedes the numerical approach and was popularized by Freeman and Adelson [FA91]. The authors were primarily concerned with steering a filter over orientation. They provided an analytical form for functions that were capable of being steered over orientation; this form essentially requires that the polar representation of the steered function have a finite angular decomposition in terms of complex exponentials. They also showed that polynomials windowed by radially symmetric functions were steerable over orientation. Because the basis functions were derived analytically, the steering functions were also obtained in analytic form.

The results of Freeman and Adelson were extended by Simoncelli *et al.* in [SFAH92] where the authors investigated the possibility of analytically steering a function over translation, rotation, or scaling. They also explored the issue of simultaneously steering a function over several domains. They concluded that it was not possible to jointly steer over translation and scaling unless one of the domains, for example translation, was fully sampled (i.e. not subsampled). In the case of steering a function over scale, the authors described a method for steering purely symmetric or anti-symmetric functions. The method considered scaling as translation in a suitably warped spatial domain. The ideal spatial warping is logarithmic; however, to cope with the singularity at the origin, the authors used a modified logarithmic warp that intersected with the origin and was linear about it. This was possible because the steerability property is independent of the particular choice of spatial warping. However, this meant that, at fine scales, the functions were not scaled copies of one another. Finally, to consider scaling only over a small interval, the authors assumed that the function to be steered has a radially periodic extension.

A framework based on tensor invariance to derive the functional form of functions steerable under n-D rotation is presented in [Bei94]. The author observed that in order for a function to be steerable under rotation, it has to be invariant under simultaneous rotation of *both* its basis and steering functions; that is, rotating the basis function and rotating the steering function by the same amount should leave the steered function unchanged. This is an observation based on the fact that the origin of the rotation parameter is arbitrary, or more accurately, dependent on the orientation of the basis functions. It follows then that the steered function must be expressible as a function of such invariants. The set of all such invariants was derived by the author using Cartesian tensor calculus.

Actually, analytical steerability has been investigated by several other authors under the notion of filter adaptivity ([And92, Hag92]; see [GK95] for a review). In particular, Andersson [And92] proposed a robust method of estimating local orientation in 2D and 3D using a particular set of orientation and phase adaptive filters. Haglund [Hag92] described a phase-based representation of images using orientation adaptive filters; he also demonstrated the use of orientation adaptive filters in several applications, including image enhancement, optical flow estimation and disparity estimation from stereo pairs.

Much of the earlier analytical work on steerable filters examined the construction of suitable basis filters under particular transformations like translation, rotation, or scaling. These transformations were special because under well-known reparameterizations, they could be treated as translations in the new domain for which Fourier analysis provided a well-understood theoretical framework. In [Mic95a, Mic95b], the authors applied the mathematical theory of Lie transformation groups to show that a function is steerable under a group of transformations (i.e. a family of transformations that also possesses algebraic group properties) only if its basis functions form an invariant subspace with respect to the infinitesimal generator of the group, which is the differential operator obtained by computing the derivative of the transform at the identity. In particular, the complex exponentials form an invariant subspace under the infinitesimal generator of translation and thus, provide a basis for steering functions under translation.

The authors point out that any one-parameter transformation group can be reparameterized to resemble the group of translations for which the complex exponentials (in the new domain) are appropriate as basis functions. The case of steering over scale is treated in greater detail where the authors proposed an alternative method to the one suggested by Simoncelli *et al.* [SFAH92]. Unlike the latter, the new method does not involve modifying the logarithmic warp near the origin. Instead, the authors argue that the projection of the steered function onto the complex exponentials in logarithmic space is not singular due to the presence of a non-identity measure that cancels the singularity; furthermore, the singularity at zero of the complex exponentials in logarithmic space can be dealt with by defining the value at zero to be zero for all the basis functions except one, which can be chosen arbitrarily.

The main advantage of the analytical approach is that the basis and steering functions are derived in analytic form. The availability of the steering function in analytic form allows steering to be carried out analytically; this is necessary for certain applications like motion estimation and the computation of invariants as will be described in Chapter 6. Another advantage of this approach is that its computational complexity is often independent of the number of parameters required to describe the transformation. In the case of group transformations, Lie group theory provides the most appropriate mathematical tool to understanding steerability. The obvious disadvantage is the restriction of the analysis to group transformations. Fortunately, this restriction is not too severe for applications in image processing and computer vision as many commonly used transformations are group transformations.

1.3 Contributions of this Thesis

In this thesis, we establish a framework based on Lie theory that facilitates the analysis and promotes the understanding of steerable functions. We argue that Lie theory is the appropriate mathematical tool for discussing steerable functions under smooth transformation groups. This position is supported by the fact that all existing analytical approaches to steerability can be consistently explained using Lie theory (because they involve smooth transformation groups).

The framework is useful as it describes the nonlinear[3] steerability property in terms of first order, linear differential operators. In doing so, it provides a general technique for constructing steerable functions under any transformation group. Furthermore, the linearity of the differential operators implies that the wealth of concepts and techniques from linear algebra can be applied to analyzing steerable functions. In particular, the Jordan decomposition of square matrices is used to derive a complete classification of all functions steerable under any one-parameter or any multi-parameter Abelian transformation group.

In addition to classifying the functions steerable under different transformation groups, the framework is also used to design two very different algorithms for constructing basis functions to steer a given function. The first algorithm is a computational, symbolic method of deriving the basis functions of an arbitrary analytically steerable function. It could be implemented in any symbolic package (like Maple or Mathematica) and used to automatically derive the set of basis functions for a

[3]Steerability is nonlinear because the group transformation that is applied to the coordinates of a function could be arbitrarily nonlinear (e.g. projective transformation). Steerability is, however, linear in terms of the function being transformed because transforming the sum of two functions is equivalent to the sum of two independently transformed functions.

given steerable function. The second algorithm addresses two problems: approximate steerability and local steerability. In practice, functions that need to be steered might not be analytically steerable; that is, steerable with a finite number of basis functions. Thus, the problem becomes one of finding the best set of k basis functions. Although Lie transformation groups are common in many applications, one can often assume that only a small amount of transformation (e.g. a small amount of translation or scaling) is ever applied or encountered. Therefore, it would be unnecessary to design functions steerable over the entire family of transformations. The second method addresses these two practical concerns by combining the Lie group-theoretic approach of the framework and the numerical singular value decomposition technique. The result is an algorithm that computes the best set of k basis functions to steer a given function under a subset of the group of transformations. Unlike the vanilla SVD approach, this technique is more computationally efficient for multi-parameter transformations.

Finally, we demonstrate that steerable functions are useful in a variety of applications. In particular, we describe the use of steerable functions in five applications: (1) approximating an infinite number of interacting mechanisms in a model of early human visual processing (continuum approximation), (2) the design of optimal steerable filters for gradient-based motion estimation, (3) efficient linear re-rendering of synthetic scenes under changes in illumination, (4) the construction of invariants from steerable filters, and (5) the application of steerable functions to discrete sets of points and lines.

The rest of the thesis is organized into five chapters. In Chapter 2, we provide an introduction to the Lie theory of transformation groups and introduce the definition of a steerable function that will be used in the rest of this work. Chapter 3 elucidates the mathematical framework surrounding steerable functions and proposes a canonical decomposition of all functions steerable under any one-parameter or multiparameter Abelian transformation groups. Chapter 4 presents a symbolic algorithm for deriving the basis functions of any steerable function. Chapter 5 describes an efficient numerical method for computing the optimal k basis functions to steer any given function under a subset of the group of transformations. In Chapter 6, the use of steerable functions in five different applications is presented. Finally, the thesis is concluded in Chapter 7.

1.4 Lie Theory in Related Areas

Before concluding this chapter, we describe the application of Lie theory in several areas related to image processing and computer vision. Lie theory is concerned with the study of topological groups that can be given an analytical structure such that the group composition and inversion are analytic. One important class of such groups in computer vision and image processing is the class of smooth spatial (coordinate) transformation groups; for example, translation, rotation, scaling, and shearing. The theory provides a mathematical framework within which to study such groups and their action (for example, the coordinate transformation of functions). Issues such as the equivalence of translation and scaling (up to a change of parameterization) are easily resolved within the framework. The nonlinear effect of the coordinate transformations on functions is more easily explained using the theory.

Although Lie theory has only recently been introduced to the study of steerable functions, it has been applied to several other areas in computer vision and image processing. S. Amari was one of the earliest to recognize the relevance of Lie theory to invariant feature detection [Ama68, Ama78, Ama87]. R. Lenz in numerous publications [Len89b, Len89a, Len90a, Len91, Len94, LH97] and in his book on the application of group theoretical methods in image processing [Len90b] provide a comprehensive introduction to the theory with examples of applications to feature extraction, motion analysis, and other applications. The use of Lie theory in motion analysis had also been investigated by R. Eagleson [Eag92a, Eag92b]. In a recent work by K. Nordberg [Nor94], the author proposed a representation, using Lie theory, for signals that are transformed by operator groups.

Lie theory has also been used in constructing nonlinear curve and surface evolution equations that are invariant under the group of special affine transformations [Sap93]. This is accomplished by choosing an affine-invariant parameterization of the curve (the affine arc length) and describing the evolution of the curve using only differential affine-invariant quantities like the affine-invariant normal vector and the affine-invariant curvature. Similar methods were used by the authors in [OST96] to construct an affine-invariant edge detector and an affine-invariant active contour. A good collection of early works on the subject can be found in [Rom94]. [Gug63] provides a good explanation of differential invariants using Lie theory. In a different application, T. Moons *et al.* use Lie theory to construct semi-differential invariants, which are invariants made up of a combination of points and derivatives [MPGO95]. Other approaches to constructing invariants are described in [Me92].

Finally, similarities between the Lie theory of transformation groups and aspects of the human visual system have also been suggested by W. C. Hoffman and others [Hof66a, Hof66b, Hof70, Dod83, PJ89].

Chapter 2

Concepts and Mathematical Preliminaries

In this chapter, we describe the properties of Lie groups, both as transformation groups and as matrix groups. We then introduce the Lie algebra associated with a Lie group. This Lie algebra is related to the original group via an exponential map that is bijective under certain conditions. This close association between Lie groups and Lie algebras allows properties of a nonlinear Lie group to be identified with properties of its linear Lie algebra. Finally, we provide a definition of steerability in conjunction with Lie transformation groups. We also show that polynomials of steerable functions are steerable. The mathematical treatment of Lie groups and Lie algebras in this chapter is rudimentary; only the necessary concepts are presented. For a more detailed exposition of Lie theory, please refer to the numerous books on the subject [Coh11, BK89, CSM95, Her66, SW73, Kir76, Kna86, Tal68, Olv95].

2.1 Lie Groups

Lie groups are often encountered as transformation groups, that is, as families of transformations acting on a signal. Common examples in image processing and computer vision include: translations, scalings, rotations, affine, and projective transformations of images, lines or points. In this work, we consider, primarily, the families of transformation groups acting on real-valued, two-dimensional images. We assume that these images are non-zero only within a bounded region and denote them by $s(x, y) : \mathbf{R}^2 \mapsto \mathbf{R}$. We describe each family of transformations by *operators* $\{g(\tau_1, \ldots, \tau_k)\}$ where $\tau_i \in \mathbf{R}$ are parameters of the transformation. The restriction of the range of s to $\mathbf{R}$ is arbitrary; most of the properties to be presented will also apply to $\mathbf{C}$ (complex-valued images).

For example, consider the family of one-dimensional translations of an image in the x-direction:

$$\hat{s}(\hat{x}, \hat{y}) = g_{t_x}(\tau)\ s(x, y) = s(x - \tau, y)$$

where τ denotes the amount of translation. In words, the operator $g_{t_x}(\tau)$ acts on the original image $s(x, y)$ to yield a new translated image $\hat{s}(\hat{x}, \hat{y}) = s(x - \tau, y)$.

A family of transformations $\{g(\tau_1, \ldots, \tau_k)\}$ parameterized by $\tau_1, \ldots, \tau_k$ over some predefined range is a Lie group if: (1) it satisfies the group conditions of closure under composition, associativity, inverse and the existence of an identity, and (2) the maps for inverse and composition are smooth (infinitely differentiable). Thus, the family of translations forms a Lie group: First, every translation operator $g_{t_x}(\tau)$ has an inverse, namely, $g_{t_x}(\iota(\tau))$ where $\iota(\tau) = -\tau$. Since $\iota(0) = 0$, $g_{t_x}(0)$ is the identity operator.[1] Second, composition of two operators can be described by a third operator which also belongs to the same family, i.e. $g_{t_x}(\tau_a)g_{t_x}(\tau_b) = g_{t_x}(\rho(\tau_a, \tau_b))$ where $\rho(\tau_a, \tau_b) = \tau_a + \tau_b$. In addition, composition is associative, that is to say, $g_{t_x}(\tau_a)(g_{t_x}(\tau_b)g_{t_x}(\tau_c)) = (g_{t_x}(\tau_a)g_{t_x}(\tau_b))\ g_{t_x}(\tau_c)$; as such, $\rho(\tau_a, \rho(\tau_b, \tau_c)) = \rho(\rho(\tau_a, \tau_b), \tau_c)$. Finally, both the inverse map, $\iota(\tau)$, and the composition map, $\rho(\tau_a, \tau_b)$ are smooth. In abstract mathematical terms, a Lie group is a smooth manifold that has a smooth map satisfying the

[1]We will typically parameterize the group such that $\boldsymbol{\tau} = \mathbf{0}$ corresponds to the identity operator; this is always possible since every element of the group has an inverse.

group properties described above; in the previous example, the manifold corresponds to the parameter space $\boldsymbol{\tau}$ and the smooth map is the composition map ρ. The dimension of the parameter space of a Lie transformation group may be different from the dimension of the image space upon which it acts. Here, the family of translations in the x-direction forms a one-parameter Lie group ($\tau \in \mathbf{R}$) while the space upon which it acts is two-dimensional ($(x, y) \in \mathbf{R}^2$).

Another familiar family of transformations that is also a Lie group is the group of rotations in the plane $g_r(\tau)$ such that $\hat{s}(\hat{x}, \hat{y}) = g_r(\tau)\ s(x, y) = s(x \cos\tau - y \sin\tau, x \sin\tau + y \cos\tau)$. It is straightforward to check that the necessary conditions, verified in the previous example, are also satisfied here. The family of transformations defined by $g_{t_x,t_y}(\tau_1, \tau_2)\ s(x, y) = s(x - \tau_1, y - \tau_2)$ is a two-parameter Lie group of x- and y-translations in the plane. Alternatively, the family of rotation and x-translation, $g_{r,t_x}(\tau_1, \tau_2)\ s(x, y) = s(x \cos\tau_1 - y \sin\tau_1 + \tau_2, x \sin\tau_1 + y \cos\tau_1)$, is another two-parameter Lie transformation group.

A subgroup of a group is a subset of the original group such that, together with the composition map, is a group over the subset alone. This subset therefore has to include the identity element. For example, in the group of x- and y-translations, the set of all x-translations with a fixed y-translation is a one-parameter subgroup since any composition of transformations from the subset yields a transformation also in that subset. Likewise, the set of rotations about the origin, is a subgroup of the group of rotations and x-translations. However, the set of rotations together with a fixed, non-zero x-translation is not a subgroup. It can be shown that any multi-parameter group can be decomposed into a collection of one-parameter subgroups where each of the one-parameter subgroup is formed by fixing all but one of the parameters to the value corresponding to the identity element. This is a useful way of understanding multi-parameter groups.

A group is called Abelian (or commutative) if all pairs of elements from the group commute; that is, $g(\boldsymbol{\tau_a})g(\boldsymbol{\tau_b}) = g(\boldsymbol{\tau_b})g(\boldsymbol{\tau_a})$ for all $\boldsymbol{\tau_a}, \boldsymbol{\tau_b}$, or equivalently, $\rho(\boldsymbol{\tau_a}, \boldsymbol{\tau_b}) = \rho(\boldsymbol{\tau_b}, \boldsymbol{\tau_a})$. It can be shown that all one-parameter groups (or subgroups) are Abelian; examples of these include the group of one-dimensional translation or the group of rotation. Two subgroups are said to commute if all pairs of elements from each subgroup commute; that is, $g_a(\boldsymbol{\tau_a})g_b(\boldsymbol{\tau_b}) = g_b(\boldsymbol{\tau_b})g_a(\boldsymbol{\tau_a})$ for all $\boldsymbol{\tau_a}, \boldsymbol{\tau_b}$, where g_a, g_b are two different subgroups. As a result, a multi-parameter group is Abelian if and only if it is made up of one-parameter subgroups that commute. The group of x- and y-translations is Abelian, for instance, since translating first by x- and then by y- is the same the other way around. Conversely, the group of rotations and x-translations is not commutative since changing the order of appying the two one-parameter transformations produces different results.

Lie groups are not restricted to transformation groups. In fact, one of the most popular examples of Lie groups are matrix groups, the most common of which is the group of invertible $n \times n$ complex or real matrices known as the general linear group, $GL(n, \mathbf{C})$ or $GL(n, \mathbf{R})$, respectively. These matrices form a group with matrix multiplication as the composition map and the identity matrix as the identity element. One common subgroup of the general linear group is the orthogonal group $O(n, \mathbf{C})$ or $O(n, \mathbf{R})$, which consists of orthogonal matrices where orthogonality is defined with respect to some appropriate inner-product. Matrix multiplication preserves the orthogonality property of the matrices; thus, matrices belonging to $O(n, \mathbf{C})$ or $O(n, \mathbf{R})$ form a subgroup of the respective general linear group. A closely related subgroup is the special orthogonal group $SO(n, \mathbf{C})$ or $SO(n, \mathbf{R})$, which is made up of orthogonal matrices with determinants equal to one. These subgroups are popular because while the group $O(3, \mathbf{R})$ is related to the group of 3D rotations and reflections about the origin, the group $SO(3, \mathbf{R})$ is related to the group of pure 3D rotations. Besides matrix groups, another common group is the additive group in either $\mathbf{R}^n$ or S^n.[2] The elements of the group are elements in $\mathbf{R}^n$ or S^n while the composition map is addition; in the case of S^n, addition is performed modulo 2π. When n equals one, we get the additive group over the reals $\mathbf{R}$ and the additive group over the circle S.

Similarities between two groups can be made precise through the use of a mapping between elements of the two group known as a *homomorphism*. A homomorphism is a mapping from the one group to another that commutes with composition of the two groups. That is, a homomorphism is a map $\Theta : G \rightarrow H$ such that $\Theta(g_1 g_2) = \Theta(g_1)\Theta(g_2)$ where G, H are two groups $g_1, g_2 \in G$, and $\Theta(g_1), \Theta(g_2) \in H$.

[2] S^n is defined as $S \times \ldots \times S$ where S is a topological space equivalent to a unit circle.

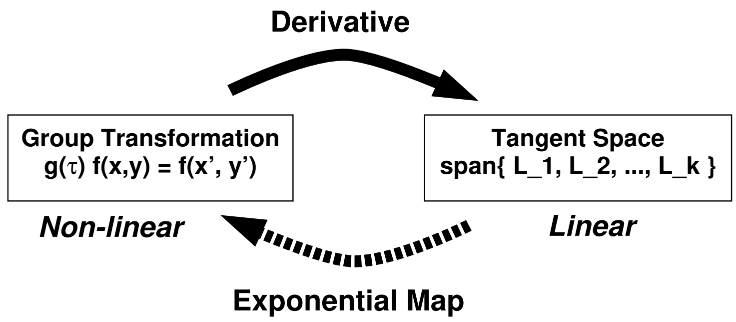


Figure 1: The relationship between the Lie group and its tangent space (Lie algebra) is due to the exponential map, which maps an element in the tangent space onto an element of the group in the neighborhood of the identity.

A homomorphism that is bijective is known as an *isomorphism*. Groups that can be related via an isomorphism are said to be *isomorphic* to each other, and have identical algebraic properties. For example, n-parameter Abelian groups are isomorphic only to other n-parameter Abelian groups. The Abelian property is preserved across isomorphisms. Another example of a pair of isomorphic groups is the group of rotations in 3D with the group of matrices $SO(3, \mathbf{R})$.

2.2 Lie Algebras

Lie groups are rich in structure and many properties of the group can be discerned by studying its corresponding Lie algebra. The close relationship between Lie groups and Lie algebras allows properties of a group to be associated one-to-one, in many cases, with properties of the algebra.

2.2.1 Tangent Space

The Lie algebra associated with a group is defined on a vector space known as the *tangent space* of the group that is spanned by a set of *infinitesimal generators* of the group. Each infinitesimal generator corresponds to the total derivative of the group action with respect to one of its parameters, evaluated at the identity. Conversely, the tangent space generates a group via the exponential map that is similar to the original group. The generated group and the original group are equivalent when the original group is simply connected. This fundamental relationship between Lie groups and Lie algebras is depicted in Figure 1, and forms the basis of the connection between Lie groups and Lie algebras.

One-parameter Transformation Groups. For a one-parameter transformation group parameterized by τ and acting on an image $s(x, y)$, the infinitesimal transformation of the group about the identity ($\tau = 0$) is defined using Leibnitz's chain rule:

$$\frac{d}{d\tau}\left(g(\tau)\; s\right)\Big|_{\tau=0} = \frac{d\hat{s}}{d\tau}\Big|_{\tau=0} = \left(\frac{\partial x}{\partial\tau}\frac{\partial}{\partial x} + \frac{\partial y}{\partial\tau}\frac{\partial}{\partial y} + \frac{\partial}{\partial\tau}\right)\Big|_{\tau=0} \hat{s}.$$

The differential operator on the right hand side of the equation is the infinitesimal generator of the transformation and is denoted by L, i.e.

$$L = \left(\frac{\partial x}{\partial\tau}\frac{\partial}{\partial x} + \frac{\partial y}{\partial\tau}\frac{\partial}{\partial y} + \frac{\partial}{\partial\tau}\right)\Big|_{\tau=0} \tag{1}$$

The partial derivative with respect to τ is zero when the group does not affect the signal in ways other than through a coordinate transformation.

The set of elements $\mathcal{G} = \{\tau L \mid \tau \in \mathbf{R}\}$ forms the one-dimensional tangent space of the group where L can be thought of as a one-dimensional basis vector. There is a strong connection between the tangent space and the Lie group from which it was derived. Namely, each element $g(\tau)$ of the group can be generated by an element in the tangent space, $\tau L \in \mathcal{G}$, via the exponential map:[3]

$$g(\tau)\; s(x, y) = e^{\tau L}\; s(x, y) \tag{2}$$

where τ is the parameter of the group. The notation $e^{\tau L}$ represents the series expansion $e^{\tau L} = I + \tau L + \frac{1}{2!}\tau^2 L^2 + \cdots$, which is an infinite sum of differential operators [Coh11]. This is a rather surprising result since the operator $g(\tau)$ can transform the image in highly nonlinear ways while $\mathcal{G}$ is simply a linear vector space.

[3]To be precise, this is only true for group elements sufficiently close to the identity element so that their Taylor expansions converge, and for elements within the connected component containing the identity. We will typically consider simply connected transformation groups for which the exponential map is bijective.

Group	Transformation	Generator
x-translation	$g_{t_x}(\tau)\, f(x,y) = f(x-\tau, y)$	$L_{t_x} = -\frac{\partial}{\partial x}$
x-scaling	$g_{s_x}(\tau)\, f(x,y) = f(e^{-\tau}x, y)$	$L_{s_x} = -x\frac{\partial}{\partial x}$
x-projective	$g_{p_x}(\tau)\, f(x,y) = f(x/(1+\tau x), y)$	$L_{p_x} = -x^2\frac{\partial}{\partial x}$
y-translation	$g_{t_y}(\tau)\, f(x,y) = f(x, y-\tau)$	$L_{t_y} = -\frac{\partial}{\partial y}$
y-scaling	$g_{s_y}(\tau)\, f(x,y) = f(x, e^{-\tau}y)$	$L_{s_y} = -y\frac{\partial}{\partial y}$
y-projective	$g_{p_y}(\tau)\, f(x,y) = f(x, y/(1+\tau y))$	$L_{p_y} = -y^2\frac{\partial}{\partial y}$
Rotation	$g_r(\tau)\, f(x,y) = f(x\cos\tau - y\sin\tau, x\sin\tau + y\cos\tau)$	$L_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$
Uniform scaling	$g_s(\tau)\, f(x,y) = f(e^{-\tau}x, e^{-\tau}y)$	$L_s = -x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y}$

Table 1: Several examples of one-parameter transformation groups and their infinitesimal generators.

Recall the group of translations in the x-direction presented earlier. The derivative of the transformation about the identity is

$$\left.\frac{d\hat{s}}{d\tau}\right|_{\tau=0} = -\frac{\partial}{\partial x}\,\hat{s}$$

and hence its generator is $L_{t_x} = -\frac{\partial}{\partial x}$. Using the exponential map suggested in Equation 2, we find that

$$\begin{aligned} g_{t_x}(\tau)\ s &= e^{\tau L_{t_x}}\ s \\ &= (1 - \tau\tfrac{\partial}{\partial x} + \tfrac{1}{2!}\tau^2\tfrac{\partial^2}{\partial x^2} + \cdots)\ s \\ &= s - \tau\tfrac{\partial s}{\partial x} + \tfrac{1}{2!}\tau^2\tfrac{\partial^2 s}{\partial x^2} + \cdots \end{aligned}$$

which is exactly the Taylor expansion of $s(x-\tau, y)$ about $\tau = 0$. Further examples of one-parameter transformation groups and their generators are given in Table 1.

The infinitesimal generators of the general linear group $GL(n, \mathbf{R})$ or $GL(n, \mathbf{C})$ of matrices is defined in a similar way. Each infinitesimal generator, in this case, is an arbitrary (possibly singular) $n \times n$ matrix. For example, the one-parameter subgroup of 2×2 rotation matrices:

$$\boldsymbol{A}(\tau) = \begin{pmatrix} \cos(\tau) & -\sin(\tau) \\ \sin(\tau) & \cos(\tau) \end{pmatrix}$$

has the following matrix as its infinitesimal generator:

$$\boldsymbol{B} = \frac{d}{d\tau}\boldsymbol{A}(\tau)\bigg|_{\tau=0} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Applying the exponential map to the matrix $\boldsymbol{B}$ (multiplied by the scalar τ) yields the matrix $\boldsymbol{A}(\tau)$. In this case, exponentiating the infinitesimal generator, which is a matrix, involves computing a power series in terms of the matrix:

$$e^{\tau \boldsymbol{B}} = \boldsymbol{I} + \tau\boldsymbol{B} + \frac{1}{2!}\tau^2\boldsymbol{B}^2 + \ldots = \boldsymbol{A}(\tau),$$

where $\boldsymbol{B}^i$ refers to the result of multiplying the matrix $\boldsymbol{B}$ by itself i times.

Multi-parameter Transformation Groups. The situation with multi-parameter Lie groups is analogous. The infinitesimal generators of a multi-parameter group are the differential operators $\{L_i \mid i = 1 \ldots k\}$ corresponding to derivatives of the transformation at the identity with respect to each parameter τ_i in turn, i.e.

$$\frac{d\hat{s}}{d\tau_i}\bigg|_{\boldsymbol{\tau}=\mathbf{0}} = L_i\ \hat{s}$$

where

$$L_i = \left(\frac{\partial x}{\partial \tau_i}\frac{\partial}{\partial x} + \frac{\partial y}{\partial \tau_i}\frac{\partial}{\partial y} + \frac{\partial}{\partial \tau_i}\right)\bigg|_{\boldsymbol{\tau}=\mathbf{0}}$$

The k generators provide a basis for the k-dimensional tangent space $\mathcal{G} = \{\tau_1 L_1 + \cdots + \tau_k L_k | \tau_1, \ldots, \tau_k \in \mathbf{R}\}$.[4] As before, there is a correspondence between a k-parameter Lie group and its k-dimensional tangent space in the form of the exponential map:

$$g(\tau_1, \ldots, \tau_k)\ s(x,y) = \exp\{\sum_{i=1}^{k} \tau_i L_i\}\ s(x,y). \tag{3}$$

Although the exponential map provides a correspondence between every operator in the Lie group and every element in its tangent space (barring the considerations raised earlier for one-parameter groups), the parameterization of the group generated by the exponential map may be different from that of the original group. Hence,

[4] Loosely speaking, the linear independence of the k generators is assured if the k-parameter group from which it was derived cannot be replaced by another with fewer parameters [Coh11].

the exponential map generates a group similar to the original group up to a change of parameterization. For example, consider the two parameterizations of the two-parameter affine group acting solely on the x coordinate:

$$\begin{aligned} g_1(\tau_1, \tau_2)\ s(x, y) &= s(e^{\tau_1} x - \tau_2, y), \\ g_2(\tau_1, \tau_2)\ s(x, y) &= s(e^{\tau_1}(x - \tau_2), y). \end{aligned}$$

Both yield the same generators:

$$L^1_{\tau_1} = L^2_{\tau_1} = x\frac{\partial}{\partial x}, \qquad L^1_{\tau_2} = L^2_{\tau_2} = -\frac{\partial}{\partial x}.$$

Hence, the exponential map will generate the same group for both. In fact, using Equation 3, we obtain the group parameterized as follows:

$$g'(\tau_1, \tau_2)\ s(x, y) = s(e^{\tau_1} x - \frac{\tau_2}{\tau_1}(e^{\tau_1} - 1)).$$

This is not a problem as we are often interested in the group of transformations and not the particular parameterization of it. Furthermore, we can easily reparameterized the generated group using the original parameterization.

As exponentiating long sums of differential operators can become rather cumbersome, another more useful map, proposed by Lie himself, is the following:

$$g(\tau_1, \ldots, \tau_k)\ s(x, y) = (\prod_{i=1}^{k} e^{\tau_i L_i})\ s(x, y) = e^{\tau_1 L_1} \cdots e^{\tau_k L_k}\ s(x, y). \tag{4}$$

The ordering of the individual exponential maps is arbitrary but that is not to say that different orderings give rise to the same parameterization of the group. Actually, the choice of ordering determines the parameterization of the group generated by the composition of exponential maps. This is easily demonstrated with the previous example:

$$\begin{aligned} e^{\tau_2 L_2} e^{\tau_1 L_1}\ s(x, y) &= s(e^{\tau_1} x - \tau_2, y), \\ e^{\tau_1 L_1} e^{\tau_2 L_2}\ s(x, y) &= s(e^{\tau_1}(x - \tau_2), y). \end{aligned}$$

With multi-parameter groups, if we vary a single parameter τ_i and keep the others fixed, we get a one-parameter group of transformations $\{g_i(\tau_i)\}$ that is a subgroup of the original k-parameter group. Hence, by varying each of the k different parameters separately, we can construct k different one-parameter subgroups. When two one-parameter subgroups commute, exponentiating their respective generators can be done in either order, i.e. $e^{\tau_i L_i} e^{\tau_j L_j} = e^{\tau_j L_j} e^{\tau_i L_i} = e^{\tau_i L_i + \tau_j L_j}$. This is not true of non-commuting subgroups. The two-parameter group of the previous example is not Abelian. Hence, as demonstrated earlier, $e^{\tau_1 L_1} e^{\tau_2 L_2} \neq e^{\tau_2 L_2} e^{\tau_1 L_1} \neq e^{\tau_1 L_1 + \tau_2 L_2}$. On the other hand, for the one-parameter transformation groups listed in Table 1, the pairs, $\{g_{t_x}, g_{t_y}\}, \{g_{t_x}, g_{s_y}\}, \{g_{t_y}, g_{s_x}\}, \{g_{s_x}, g_{s_y}\}, \{g_r, g_s\}$, etc., are commutative.

The infinitesimal generators of multi-parameter subgroups of the general linear group $GL(n, \mathbf{R})$ or $GL(n, \mathbf{C})$ are general $n \times n$ matrices corresponding to the partial derivatives of the group, evaluated at the identity. For example, the two-parameter subgroup of 2×2 matrices:

$$\boldsymbol{A}(\tau_1, \tau_2) = \begin{pmatrix} e^{\tau_1} & \tau_2 \\ 0 & 1 \end{pmatrix}$$

have the following matrices as its infinitesimal generators:

$$\boldsymbol{B}_{\boldsymbol{\tau_1}} = \frac{d}{d\tau_1} \boldsymbol{A}(\tau_1, \tau_2)\Big|_{\boldsymbol{\tau}=\mathbf{0}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

$$\boldsymbol{B}_{\boldsymbol{\tau_2}} = \frac{d}{d\tau_2} \boldsymbol{A}(\tau_1, \tau_2)\Big|_{\boldsymbol{\tau}=\mathbf{0}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Linear combinations of the above infinitesimal generators (weighted by τ_1, τ_2 respectively) can be exponentiated to yield a group of matrices. As discussed above, whether the original matrix group is obtained depends on the choice of exponentiation method. The particular method of exponentiating that derives the original matrix subgroup $\boldsymbol{A}(\tau_1, \tau_2)$ is:

$$e^{\tau_2 \boldsymbol{B}_{\boldsymbol{\tau_2}}} e^{\tau_1 \boldsymbol{B}_{\boldsymbol{\tau_1}}} = \begin{pmatrix} 1 & \tau_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} e^{\tau_1} & 0 \\ 0 & 1 \end{pmatrix} = \boldsymbol{A}(\tau_1, \tau_2).$$

2.2.2 Vector Fields

The elements of the tangent spaces of transformation groups can also be viewed as vector fields where the spatial partial derivatives provide a local coordinate frame. For

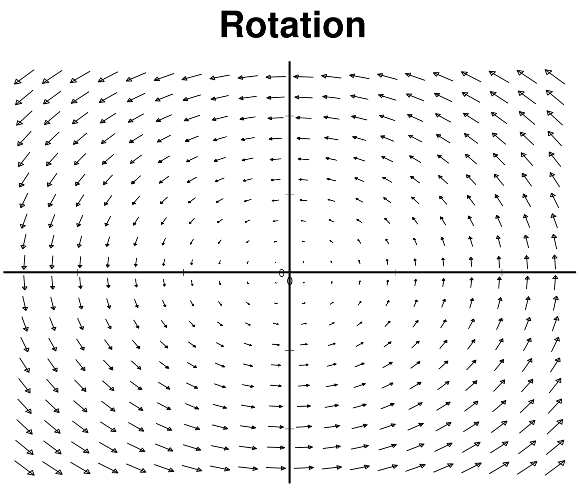


Figure 2: The vector field representing the infinitesimal generator of the one-parameter group of rotations in the plane, the infinitesimal generator being $L_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$.

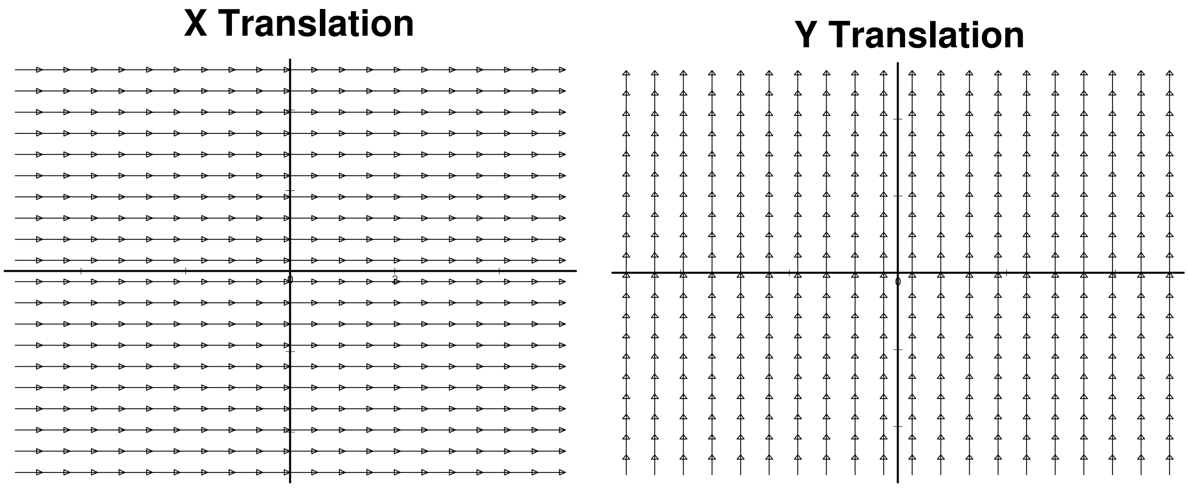


Figure 3: The vector fields associated with the infinitesimal generators of the two-parameter group of translations in the plane, the infinitesimal generators being $L_{t_x} = \frac{\partial}{\partial x}$ and $L_{t_y} = \frac{\partial}{\partial y}$.

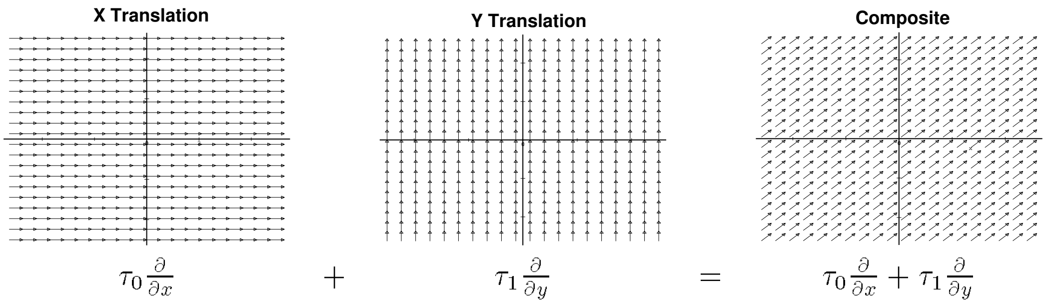


Figure 4: The vector field associated with the element $(\tau_0 \frac{\partial}{\partial x} + \tau_1 \frac{\partial}{\partial y})$ of the tangent space of the two-parameter group of translations in the plane. The vector field is shown as a linear combination of the vector fields associated with the infinitesimal generators.

example, the infinitesimal generator of the one-parameter group of rotations in the plane, $L_r = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}$, can be regarded as a vector field over x, y. Figure 2 plots the vector field such that the vectors correspond to $(-y, x)$. Likewise, the infinitesimal generators of a multi-parameter group each has its respective vector field. The vector fields for the two-parameter group of translations in the plane is shown in Figure 3. Any element of the tangent space can also be represented by a vector field via linear combinations of the vector fields associated with the infinitesimal generators that span the tangent space. This is depicted in Figure 4 for the group of x- and y-translations.

2.2.3 Lie Bracket

The tangent space is made into an algebra by the introduction of a multiplication known as the *Lie bracket*. The simplest way of defining the Lie bracket is as the commutator of pairs of elements of the tangent space. That is, the Lie bracket denoted by $[l_a, l_b]$ where $l_a, l_b \in \mathcal{G}$ is defined as $[l_a, l_b] = l_a l_b - l_b l_a$. This is not the most general definition of the Lie bracket but it suffices for the cases of transformation groups and matrix groups. In the case of transformation groups, $l \in \mathcal{G}$ is a linear combination of the infinitesimal generators $\{L_1, \cdots, L_k\}$ and is thus a differential operator. Composition of two operators $l_a l_b$ is defined to be the composition of differential operators. As for matrix groups, the elements of the tangent space corresponds to a linear combination of the $n \times n$ matrices $\{\boldsymbol{B_1}, \cdots, \boldsymbol{B_n}\}$ and composition is simply matrix multiplication.

Consider the group of x-translations and x-scalings whose infinitesimal generators are $L_{t_x} = -\frac{\partial}{\partial x}$ and $L_{s_x} = -x\frac{\partial}{\partial x}$. The Lie bracket of the two generators is

$$\begin{aligned}
[L_{t_x}, L_{s_x}] &= L_{t_x}L_{s_x} - L_{s_x}L_{t_x} \\
&= \left(-\frac{\partial}{\partial x}\right)\left(-x\frac{\partial}{\partial x}\right) - \left(-x\frac{\partial}{\partial x}\right)\left(-\frac{\partial}{\partial x}\right) \\
&= \frac{\partial}{\partial x}\left(x\frac{\partial}{\partial x}\right) - x\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\right) \\
&= \left(x\frac{\partial^2}{\partial x^2} + \frac{\partial}{\partial x}\right) - \left(x\frac{\partial^2}{\partial x^2}\right) \\
&= \frac{\partial}{\partial x} = -L_{t_x}.
\end{aligned}$$

Likewise, the generators of a matrix representation of the group are $\boldsymbol{B_{t_x}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ and $\boldsymbol{B_{s_x}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. The Lie bracket of these two matrix generators are

$$\begin{aligned}
[\boldsymbol{B_{t_x}}, \boldsymbol{B_{s_x}}] &= \boldsymbol{B_{t_x}}\boldsymbol{B_{s_x}} - \boldsymbol{B_{s_x}}\boldsymbol{B_{t_x}} \\
&= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} = -\boldsymbol{B_{t_x}}.
\end{aligned}$$

It can be shown that the Lie bracket between any two elements from the tangent space is an element also in the tangent space. Therefore, the Lie bracket is a bilinear map from the tangent space onto itself. Furthermore, it is easy to see that the Lie bracket is anti-symmetric such that $[l_a, l_b] = -[l_b, l_a]$ and the Lie bracket of two identical elements is zero. However, the Lie bracket is not associative; i.e. $[l_a, [l_b, l_c]] \neq [[l_a, l_b], l_c]$. Instead, it satisfies a different property known as the Jacobi identity: $[l_a, [l_b, l_c]] + [l_b, [l_c, l_a]] + [l_c, [l_a, l_b]] = 0$.

Since the Lie bracket is bilinear, it can be characterized by its results over pairs of the infinitesimal generators that span the tangent space. Consider, for example, the tangent space spanned by two generators L_1, L_2. Let $aL_1 + bL_2$ and $cL_1 + dL_2$ be two elements of the tangent space. Thus, their Lie bracket is $[aL_1 + bL_2, cL_1 + dL_2] = ac\,[L_1, L_1] + ad\,[L_1, L_2] + bc\,[L_2, L_1] + bd\,[L_2, L_2]$ which simplifies to $(ad - bc)\,[L_1, L_2]$ because $[L_1, L_1] = [L_2, L_2] = 0$ and $[L_2, L_1] = -[L_1, L_2]$.

The Lie bracket is useful in that properties of the Lie bracket with respect to a given Lie algebra can be used to deduce properties of the generated group; conversely, properties of a Lie group induces properties in its Lie algebra. For example, it can be shown that a multi-parameter Lie group is Abelian if and only if the Lie bracket is zero for any pair of elements from its tangent space. Correspondingly, this implies that $[L_i, L_j] = 0$ for any pair of infinitesimal generators. As a result, one immediately sees that all one-parameter groups are commutative since $[L, L] = 0$. Another example is the fact that a subgroup of a Lie group defines a specific subalgebra within the original Lie algebra. A *subalgebra* $\mathcal{H}$ is a subspace of the tangent space of the Lie algebra $\mathcal{G}$ that is invariant with respect to the Lie bracket; this means that $[L_i, L_j] \in \mathcal{H}$ for all $L_i, L_j \in \mathcal{H} \subseteq \mathcal{G}$. This is trivially true for all one-parameter subgroups since $[L_i, L_i] = 0$.

In the same way, homomorphisms between Lie groups relate groups with similar properties, homomorphisms between Lie algebra relate the algebras with similar properties. A homomorphism $\Theta : \mathcal{G} \to \mathcal{H}$ is a linear map from the Lie algebra $\mathcal{G}$ to the Lie algebra $\mathcal{H}$ such that the Lie bracket is preserved; i.e., $\Theta([l_1, l_2]) = [\Theta(l_1), \Theta(l_2)]$ where $l_1, l_2 \in \mathcal{G}$. Note that the Lie bracket on the left hand side of the equation is the Lie bracket defined on $\mathcal{G}$ while the Lie bracket on the right hand side of the equation refers to the Lie bracket defined on $\mathcal{H}$. An isomorphism between two Lie algebras is a homomorphism that is also bijective. Since homomorphisms of Lie algebras are linear mappings, this implies that isomorphisms only occur between Lie algebras of the same dimension. Homomorphic (isomorphic) Lie groups have Lie algebras that are homomorphic (isomorphic); conversely, homomorphic Lie algebras generate Lie groups that are homomorphic. Therefore, Lie algebras that are isomorphic with each other are essentially equivalent and generate groups that are equivalent, up to a reparameterization. Homomorphisms between Lie algebras are generally easier to study since they are linear maps while homomorphisms between Lie groups are typically nonlinear.

2.3 Steerable Functions

In this section, we present the definition of steerability that will be used in the rest of this work. We discuss the natural extension of steerability of a single function to steerability of an entire function space, which we describe as being equivariant. Finally, we show how equivariant function spaces can be combined to construct larger function spaces that are also equivariant.

2.3.1 Definition

The original definition of steerability was proposed by Freeman and Adelson [FA91] for the special case of rotation. Simoncelli *et al.* [SFAH92] extended this definition to include translation and scaling, and coined the term "joint-shiftability." Perona [Per95] used the term "deformable" to refer to functions steerable under arbitrary compact transformations, not necessarily having the group properties. We retain the term steerability in our definition but generalize its definition to encompass any transformation group.

Definition 1 (Steerability) *: A function $f(x,y) : \mathbf{R}^2 \mapsto \mathbf{C}$ is* **steerable** *under a k-parameter Lie transformation group G if any transformation $g(\boldsymbol{\tau}) \in G$ of f can be written as a linear combination of a fixed, finite set of* basis functions *$\{\phi_i(x,y)\}$:*

$$g(\boldsymbol{\tau})\ f(x,y) = \sum_{i=1}^{n} \alpha_i(\boldsymbol{\tau})\ \phi_i(x,y) = \boldsymbol{\alpha}^T(\boldsymbol{\tau})\Phi(x,y)$$

The vector $\boldsymbol{\tau}$ parameterizes the family of transformations in the group G. Vector $\boldsymbol{\alpha}$ is a vector of the functions α_i such that $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Likewise, vector Φ contains the functions ϕ_i such that $\Phi = (\phi_1, \ldots, \phi_n)$. The functions α_i are known as the *steering functions* of f associated with the basis $\{\phi_i\}$ and depend solely on the transform parameters. Without loss of generality, we assume that n is the minimum number of basis functions required and these basis functions are linearly independent. Clearly, the set of basis functions required to steer a given function is not unique; any (non-singular) linear transformation of the set of basis functions could also be used. For example, the first x-derivative of a 2D Gaussian G'_x is steerable under the one-parameter group of rotations with the basis set $\{G'_x, G'_y\}$ such that $g(\theta)G'_x = \cos(\theta)G'_x + \sin(\theta)G'_y$. Rotation is represented by the operator $g(\theta)$ where θ is the angle of rotation. The functions $\cos(\theta), \sin(\theta)$ are the steering functions while the functions G'_x, G'_y are the basis functions.

If a function f is steerable with a set of basis functions Φ, then each of the basis functions ϕ_i are themselves steerable with the same basis functions. This is true since each basis function can be rewritten as a linear combination of transformed replicas of f (chosen to be linearly independent):

$$\Phi = \begin{bmatrix} \boldsymbol{\alpha}^T(\boldsymbol{\tau}^1) \\ \vdots \\ \boldsymbol{\alpha}^T(\boldsymbol{\tau}^k) \end{bmatrix}^{-1} \begin{pmatrix} g(\boldsymbol{\tau}^1)f \\ \vdots \\ g(\boldsymbol{\tau}^k)f \end{pmatrix}.$$

Thus, transforming a basis function is equivalent to linearly combining the set of transformed replicas of f, which are themselves steerable.

Since steerability of the given function f implies steerability of its basis functions ϕ_i as well, it is more natural to express steerability in terms of a function space, i.e. in terms of the space spanned by the basis functions $\{\phi_i\}$.

Definition 2 (Equivariant Function Space) *:*
An n-dimensional function space $\mathcal{F}$ =span$\{\phi_1, \ldots, \phi_n\}$ is **equivariant** *under a k-parameter Lie transformation group G if every ϕ_i is steerable with respect to the basis $\{\phi_1, \ldots, \phi_n\}$, i.e., there is an $n \times n$ matrix function $\boldsymbol{A}(\boldsymbol{\tau})$, called the* **interpolation matrix**, *such that:*

$$g(\boldsymbol{\tau})\Phi(x,y) = \boldsymbol{A}(\boldsymbol{\tau})\Phi(x,y) \quad \textit{for all} \quad g(\boldsymbol{\tau}) \in G$$

This equation is called the **interpolation equation**.

Following the previous example of the first x-derivative of a 2D Gaussian, the two-dimensional function space spanned by $\Phi = \left(G'_x, G'_y\right)^T$ is equivariant with the interpolation matrix $\boldsymbol{A}(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$.

The term "equivariance" was originally proposed by Wilson and Knutsson [WK88]. From the definition, it follows that an equivariant function space is a function space that is invariant under the associated transformation group. More generally, any function $f \in \mathcal{F}$, such that $f = \sum c_i \phi_i = \mathbf{c}^T \Phi$ is steerable by steering the basis of $\mathcal{F}$:

$$g(\boldsymbol{\tau}) f = g(\boldsymbol{\tau}) \boldsymbol{c}^T \Phi = \boldsymbol{c}^T \boldsymbol{A}(\boldsymbol{\tau}) \Phi.$$

As a result, any function f is steerable under a k-parameter transformation group if and only if it belongs to some function space that is equivariant under the same transformation group.

For example, consider the function space $\mathcal{F}_\theta = \operatorname{span}\{\cos\theta, \sin\theta\}$ under the one-parameter group of rotations: $g_r(\tau) f(\theta) = f(\theta - \tau)$. It is easy to verify the following two identities:

$$\begin{aligned} \cos(\theta - \tau) &= \cos\tau\cos\theta + \sin\tau\sin\theta, \\ \sin(\theta - \tau) &= -\sin\tau\cos\theta + \cos\tau\sin\theta. \end{aligned}$$

Thus, rotated versions of any basis function in $\mathcal{F}_\theta$ can always be expressed as linear combinations of the basis functions. Hence, any $f \in \mathcal{F}_\theta$ is steerable under the rotation group.

The interpolation matrix is an $n \times n$ matrix whose entries are functions of the group parameters $\boldsymbol{\tau}$. In fact, the set of all matrices $\{\boldsymbol{A}(\boldsymbol{\tau})\}$ is a subgroup of $GL(n, \mathbf{R})$. This is because $\boldsymbol{A}(\boldsymbol{\tau_{12}}) = \boldsymbol{A}(\boldsymbol{\tau_1})\boldsymbol{A}(\boldsymbol{\tau_2})$ where $\boldsymbol{A}(\boldsymbol{\tau_{12}}), \boldsymbol{A}(\boldsymbol{\tau_1}), \boldsymbol{A}(\boldsymbol{\tau_2})$ are the interpolation matrices corresponding to the transformation $g(\tau_{12}), g(\tau_1), g(\tau_2)$ respectively, and $g(\tau_{12}) = g(\tau_1)g(\tau_2)$. Thus, the subgroup of matrices $\{\boldsymbol{A}(\boldsymbol{\tau})\}$ is a homomorphism of the group of transformations denoted by $\{g(\boldsymbol{\tau})\}$. This is exploited in the next chapter where the infinitesimal generator of the transformation group is related to the infinitesimal generator of the interpolation matrix group.

2.3.2 Construction of Equivariant Function Spaces

Equivariant function spaces under the same transformation group can be combined to construct larger functions spaces that are also equivariant under the same transformation group. One approach computes the new equivariant function space as the *direct sum* of two equivariant function spaces. If Φ_1 and Φ_2 contain the basis functions of two distinct equivariant function spaces, then the function space spanned by all the basis functions together is also equivariant. For example, if $\Phi_1 = (\sin(x), \cos(x))^T$ and $\Phi_2 = (\sin(2x), \cos(2x))^T$, then the space spanned by $\Phi_1 \oplus \Phi_2 = (\sin(x), \cos(x), \sin(2x), \cos(2x))$ is also equivariant. Since the vector sum of two equivariant function spaces is equivariant, this implies that the sum of two steerable functions is also steerable.

A second approach computes the new equivariant function space as the *tensor product* of two equivariant function spaces. The transformation group is applied to the tensor product space by applying the transformation to each component of the tensor simultaneously. The basis functions of the tensor product space are equivariant. The multi-linear tensor product space can be converted into a linear space using the set of basis functions formed by taking the Kronecker product of the original basis functions from the two equivariant function spaces spaces (i.e. the pairwise products of functions from Φ_1 and Φ_2). This linearized space is also equivariant. Using the same example as above, the linearized tensor product space spanned by $\Phi_1 \otimes \Phi_2 = (\sin(x)\sin(2x), \sin(x)\cos(2x), \cos(x)\sin(2x), \cos(x)\cos(2x))$ is equivariant. Since the tensor product of two equivariant function spaces is equivariant, this implies that the product of two steerable functions is also steerable.

Using these two approaches, any polynomial of equivariant function spaces (or of a single equivariant function space) is equivariant where addition and multiplication is vector sum and tensor product respectively. Similarly, any polynomial of steerable functions is also steerable.

Chapter 3

Canonical Decomposition

Designing basis functions that can steer an arbitrary function under a given transformation group is the main problem concerning steerable functions. Once the basis functions are chosen, its steering functions can be computed numerically or even analytically as will be evident later in this chapter. From the last chapter, is is clear that the problem of designing suitable basis functions is equivalent to identifying an appropriate equivariant function space, namely, the lowest-dimensional one that contains the orbit (i.e. all transformed replicas) of the steered function. For example, when steering the first derivative of a Gaussian under rotation, the lowest-dimensional equivariant function space is the two-dimensional space spanned by the first derivatives of the Gaussian in two linearly independent directions. Note that this minimal equivariant function space may not be finite dimensional for an arbitrary function; the definition of a steerable function defines precisely those functions that have finite-dimensional equivariant function spaces as being steerable.

In this chapter, a general mathematical recipe for constructing equivariant function spaces under any Lie transformation group is presented. The method involves solving a system of linear, first order partial differential equations, the partial derivatives being taken over the spatial variables. In the case of one-parameter transformation groups or multi-parameter Abelian transformation groups, we describe a canonical decomposition of all equivariant function spaces. This decomposition provides a

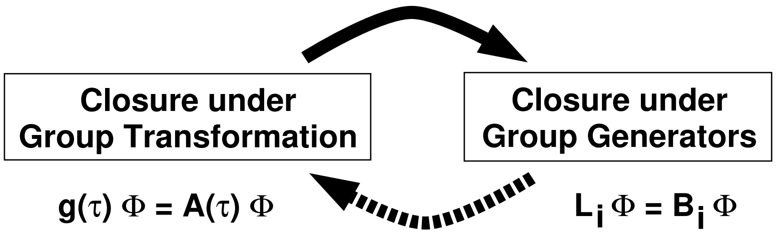


Figure 5: Closure of the equivariant function space with respect to the group transformation is equivalent to closure of the function space with respect to the infinitesimal generators of the group.

canonical functional form for all equivariant function spaces under these transformation groups. For the general case of non-Abelian transformation groups, a catalog of equivariant function spaces for several common subgroups of the affine transformation group is provided.

The material presented in this chapter can be found in [HOT98]; an early version of it is described in [HOT96].

3.1 Construction of Equivariant Function Spaces

Equivariant function spaces are defined as function spaces that are closed under some transformation group. Since a coordinate transformation of an arbitrary function is a nonlinear operation, the closure condition is nonlinear. Fortunately, because we are dealing with Lie transformation groups, the closure of a function space under elements of the transformation group $g(\boldsymbol{\tau})$ can be reformulated, more simply, in terms of the group generators $\{L_1, \ldots, L_k\}$. Specifically, closure of a function space under a transformation group is equivalent to closure of the function space under the action of the infinitesimal generators of the group. This relationship is depicted in Figure 5. This approach is an extension of the seminal work of S. Amari [Ama68, Ama78] who originally proposed it in the context of invariant feature detection in pattern recognition.

Theorem 1 (Interpolation Equation) *:*

The function space $\mathcal{F} = span\{\phi_1, \cdots, \phi_n\}$ is equivariant under the transformation group G if and only if $\mathcal{F}$ is closed under the action of each generator L_i of G. That is, $g(\boldsymbol{\tau})\Phi = \boldsymbol{A}(\boldsymbol{\tau})\Phi$ if and only if there is a set of $n \times n$ matrices $\{\boldsymbol{B_1}, \ldots, \boldsymbol{B_k}\}$ such that:

$$L_i \ \Phi = \boldsymbol{B_i} \ \Phi \quad \textit{for all} \quad i = 1, \ldots, k$$

In particular, the interpolation matrix can be written as follows:

$$\boldsymbol{A}(\boldsymbol{\tau}) = e^{\tau_k \boldsymbol{B_k}} \cdots e^{\tau_1 \boldsymbol{B_1}} \ .$$

The matrices $\boldsymbol{B_i}$ are the infinitesimal generator of the matrix group $\boldsymbol{A}(\boldsymbol{\tau})$.

Proof 1 : Let $\hat{\Phi}(x, y) = g(\boldsymbol{\tau})\Phi(x, y)$, the transformed vector of basis functions. Since $g(\boldsymbol{\tau})$ is a Lie group, it follows from the exponential map in Equation 4 that

$$\begin{aligned}
\hat{\Phi}(\boldsymbol{\tau}) &= e^{\tau_1 L_1} \cdots e^{\tau_k L_k} \ \Phi \\
&= (I + \tau_1 L_1 + \cdots) \cdots (I + \tau_k L_k + \cdots) \ \Phi \\
&= (I + \tau_1 L_1 + \cdots) \cdots (I + \tau_k \boldsymbol{B_k} + \cdots) \ \Phi \\
&= (I + \tau_1 L_1 + \cdots) \cdots (I + \tau_{k-1} L_{k-1} + \cdots) \ e^{\tau_k \boldsymbol{B_k}} \ \Phi \\
&= (I + \tau_1 L_1 + \cdots) \cdots \ e^{\tau_k \boldsymbol{B_k}} \ (I + \tau_{k-1} L_{k-1} + \cdots) \ \Phi \\
&= (I + \tau_1 L_1 + \cdots) \cdots \ e^{\tau_k \boldsymbol{B_k}} \ e^{\tau_{k-1} \boldsymbol{B_{k-1}}} \ \Phi \\
&\vdots \\
&= e^{\tau_k \boldsymbol{B_k}} \cdots e^{\tau_1 \boldsymbol{B_1}} \ \Phi,
\end{aligned}$$

in which the substitution $(L_i)^m \Phi = (\boldsymbol{B_i})^m \Phi$ is used repeatedly. It can easily be verified that $L_i \ \Phi = \boldsymbol{B_i} \ \Phi$ implies $(L_i)^m \Phi = (\boldsymbol{B_i})^m \Phi$ via the linearity of the differential operator L_i. The order in which the generators L_i are applied is arbitrary. However, as pointed out in Section 2.2.1, the order will determine the parameterization of the generated group. Conversely, if $\hat{\Phi} = e^{\tau_k \boldsymbol{B_k}} \cdots e^{\tau_1 \boldsymbol{B_1}} \ \Phi$, taking derivatives with respect to τ_i (about $\boldsymbol{\tau} = \boldsymbol{0}$) on both sides of the equation yields the system of equations $L_i \ \Phi = \boldsymbol{B_i} \ \Phi$. □

Theorem 1 provides a recipe for verifying whether a space spanned by a set of functions $\{\phi_i\}$ is equivariant, and if it is, derives the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$.

1. Derive the generators $L_1, \ldots, L_k$ of the given transformation group $g(\boldsymbol{\tau})$.
2. Verify for each generator L_i that
$$L_i \, \Phi = \boldsymbol{B_i} \, \Phi$$
where $\boldsymbol{B_i}$ is some $n \times n$ matrix.
3. If so, the function space $\mathcal{F} = \text{span}\{\phi_1, \cdots, \phi_k\}$ is equivariant and the interpolation equation is simply
$$g(\boldsymbol{\tau})\Phi = \boldsymbol{A}(\boldsymbol{\tau}) \, \Phi$$
where the interpolation matrix is
$$\boldsymbol{A}(\boldsymbol{\tau}) = e^{\tau_k \boldsymbol{B_k}} \ldots e^{\tau_1 \boldsymbol{B_1}}.$$

Figure 6: Recipe for verifying that the function space of span(Φ) is equivariant. If so, the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$ is also derived.

Figure 6 summarizes the procedure. Unfortunately, the construction of all possible n-dimensional equivariant function spaces is not as methodical in general, and has to be done by inspection. For one-parameter and multi-parameter Abelian transformation groups, however, the construction is straightforward and will be treated extensively in the next section.

3.2 Function Spaces for One-Parameter Groups

In this section, we attend to the construction of all possible equivariant function spaces with respect to any one-parameter transformation group. First, we provide examples of several such equivariant function spaces. After that, we show that any one-parameter group can be re-parameterized to appear as a group of translations in the new parameterization. Finally, we propose a canonical decomposition of all the function spaces equivariant under the translation group (and correspondingly under any one-parameter group that has been appropriately re-parameterized).

3.2.1 The Translation Group

Consider the group of one-dimensional translations in the x-direction: $\hat{f}(\hat{x}, \hat{y}) = g_{t_x}(\tau)\ f(x, y) = f(x + \tau, y)$ whose generator $L_{t_x} = \frac{\partial}{\partial x}$. An n-dimensional function space Φ is equivariant with respect to $g_{t_x}(\tau)$ if $L_{t_x}\Phi = \frac{\partial}{\partial x}\Phi = \boldsymbol{B}\Phi$ for a given $n \times n$ matrix $\boldsymbol{B}$. The general solution to this ordinary differential equation is

$$\Phi(x, y) = e^{\boldsymbol{B}x}\ \Phi(0) \tag{5}$$

where $\Phi(0)$ is the value of Φ at $x = 0$. Actually, the product of $\Phi(x, y)$ with any function solely in y leaves it equivariant; thus, without loss of generality, we refer to $\Phi(x, y)$ only as $\Phi(x)$. Since $\Phi(0)$ can be arbitrary chosen, any element in the column space of $e^{\boldsymbol{B}x}$ is a possible solution. We will denote this by $\Phi(x) \in \mathcal{R}(e^{\boldsymbol{B}x})$ where $\mathcal{R}$ refers to the column space of the matrix $e^{\boldsymbol{B}x}$. Regardless of the choice of $\Phi(0)$, the interpolation equation is the same, i.e. $\hat{\Phi} = e^{\boldsymbol{B}\tau}\Phi$.

In the following examples, we present different choices for the matrix $\boldsymbol{B}$ and derive the corresponding equivariant function spaces. We show that several commonly used steerable functions are the result of particular choices of the matrix $\boldsymbol{B}$.

Example 1 : Consider the simplest case where $\boldsymbol{B}$ is a 1×1 matrix, i.e. $\boldsymbol{B} = [\lambda]$ where λ is a scalar value (which may be complex). From Equation 5, the space of equivariant functions is: $\Phi(x) = ae^{\lambda x}$, where a is some scalar value (the value at $\Phi(0)$), while the interpolation equation is $\hat{\Phi} = e^{\tau\lambda}\Phi$. Proving equivariance is straightforward since $\hat{\Phi} = ae^{\lambda(x+\tau)} = e^{\lambda\tau}ae^{\lambda x} = e^{\lambda\tau}\Phi$. When λ is purely imaginary, the functions are complex exponentials. In phase-based motion estimation, the parameter τ is regarded as the difference in phase. Fleet and Jepson [FJ91] proposed an accurate method of measuring disparity by estimating the difference in phase between two (windowed) complex exponentials.

Example 2 : Now, let

$$\boldsymbol{B} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

In this case, the solution to Equation 5 implies that

$$\Phi(x) \in \mathcal{R}(e^{\boldsymbol{B}x}) = \mathcal{R}\left[\begin{pmatrix} e^{\lambda_1 x} & 0 \\ 0 & e^{\lambda_2 x} \end{pmatrix}\right]$$

and the interpolation equation is

$$\hat{\Phi} = e^{\tau \boldsymbol{B}}\ \Phi = \begin{pmatrix} e^{\lambda_1 \tau} & 0 \\ 0 & e^{\lambda_2 \tau} \end{pmatrix} \Phi.$$

Simoncelli *et al.* [SFAH92] proposed a criterion for shiftability in position that decomposes the filter into a set of complex exponentials (using Fourier decomposition). In this example, it would correspond to having $\boldsymbol{B}$ being a diagonal matrix with distinct and purely imaginary λ's.

Example 3 : Let

$$\boldsymbol{B} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

In this case, the equivariant functions and the interpolation equation are

$$\Phi(x) \in \mathcal{R}(e^{\boldsymbol{B}x}) = \mathcal{R}\left[\begin{pmatrix} 1 & x & \frac{1}{2!}x^2 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix}\right] \qquad \text{and} \qquad \hat{\Phi} = e^{\tau \boldsymbol{B}}\ \Phi = \begin{pmatrix} 1 & \tau & \frac{1}{2!}\tau^2 \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{pmatrix} \Phi$$

This example produces the kernels of moment filters which are used in many applications involving invariant feature detection [Hu62] and motion estimation [XS95b].

3.2.2 The Rotation Group

Another commonly encountered one-parameter transformation group is the group of rotations in the plane:

$$g_r(\tau)\ f(x,y) = f(x\cos\tau + y\sin\tau, -x\sin\tau + y\cos\tau)$$

where τ represents the amount of rotation. The generator of the rotation group is: $L_r = y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y}$. It is easy to see that if we represent the function $f(x, y)$ in polar coordinates (r, θ), then rotation becomes similar to translation: $g_r(\tau)f(r, \theta) = f(r, \theta + \tau)$. In these coordinates, the generator is $L_r = \frac{\partial}{\partial \theta}$. Therefore, as before, an n-dimensional vector of functions $\Phi(r, \theta)$ is equivariant with respect to $g_r(\tau)$ if it satisfies the equation

$$L_r\Phi \doteq \frac{\partial \Phi}{\partial \theta} = \boldsymbol{B}\Phi$$

where $\boldsymbol{B}$ is an $n \times n$ matrix. The general solution to the above equation is simply

$$\Phi(\theta) = e^{\boldsymbol{B}\theta}\Phi(0)$$

where $\Phi(0)$ is the value of $\Phi(\theta)$ at $\theta = 0$. Since $\Phi(0)$ is arbitrarily chosen, $\Phi(\theta) \in \mathcal{R}(e^{\boldsymbol{B}\theta})$.

Example 4 : In this example, we show that a vector of functions is equivariant with respect to rotation and derive its interpolation matrix. Let $\Phi(x, y)$ be a 2D-vector containing the spatial derivatives of a Gaussian $G = \exp(-(x^2+y^2)/2) = \exp(-r^2/2)$ in the x- and y- directions:

$$\Phi(x, y) = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} G = \begin{pmatrix} -x \\ -y \end{pmatrix} G = \begin{pmatrix} -r\cos\theta \\ -r\sin\theta \end{pmatrix} G.$$

Applying the generator $L_r = \frac{\partial}{\partial \theta}$ to Φ, we obtain

$$L_r\ \Phi = \begin{pmatrix} r\sin\theta \\ -r\cos\theta \end{pmatrix} G = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\ \Phi = \boldsymbol{B}\ \Phi.$$

Thus, the elements of $\Phi(x, y)$ span an equivariant function space whose interpolation function is

$$\hat{\Phi} = e^{\tau\boldsymbol{B}}\ \Phi = \begin{pmatrix} \cos\tau & -\sin\tau \\ \sin\tau & \cos\tau \end{pmatrix} \Phi.$$

This is an example of the steerable filters suggested in Freeman and Adelson [FA91].

3.2.3 Canonical Coordinates of One-Parameter Transformation Groups

The construction of equivariant function spaces depends on the existence of a solution to the system of partial differential equations $L\Phi = \boldsymbol{B}\Phi$. It was shown that for translations and planar rotations, solutions exist for any given matrix $\boldsymbol{B}$. In this section, we show that solutions exist for *any* one-parameter transformation group. The simplest way to show this is via a re-parameterization of the current coordinates into some canonical coordinates where solutions are known to exist. For any one-parameter transformation group $g(\tau)$, there exists a change of coordinates such that the group resembles a translation in the new parameterization [Coh11]. Hence, given a function $f(x, y)$, one can determine a change of coordinates $f(\eta(x, y), \xi(x, y))$ such that

$$g(\tau)\, f(\eta, \xi) = f(\eta + \tau, \xi).$$

Segman *et al.* [SRZ92] used this re-parameterization to construct invariant kernels for pattern recognition. Ferraro and Caelli [FC94] used this method in a similar context and suggested its relevance to biological vision.

Since the group operation is the same as one-dimensional translation, the equivariant condition with respect to the canonical coordinates is also the same:

$$L_{\eta,\xi}\, \Phi(\eta, \xi) = \frac{\partial}{\partial \eta} \Phi(\eta, \xi) = \boldsymbol{B}\, \Phi(\eta, \xi).$$

Therefore, its equivariant function spaces also resemble the equivariant function spaces for translation (up to a change of coordinates).

Example 5 : In Section 3.2.2, polar coordinates were used for the group of rotations in the plane. It is easy to show that polar coordinates are the canonical coordinates for this group. Recall the change of coordinates from Cartesian to polar:

$$\eta = \arctan(y/x) = \theta \quad ; \quad \xi = \sqrt{x^2 + y^2} = r.$$

Rotating a function $f(x, y)$ in Cartesian coordinates is the same as translating the function in polar coordinates: $g_r(\tau)\, f(\eta, \xi) = f(\eta + \tau, \xi)$ where $\tau \in [0, 2\pi)$.

Example 6 : Consider next the one-parameter group of scaling in the x direction, i.e. $g_{s_x}(\tau)\ f(x,y) = f(e^\tau x, y)$ where e^τ ensures that the scaling constant is always positive. The canonical coordinates of this transformation group are obtained by the coordinate changes:

$$\eta = \ln(x) \quad \text{and} \quad \xi = y.$$

In this case,

$$g_{s_x}(\tau)\ f(\eta,\xi) = f(\ln(e^\tau x),\xi) = f(\ln(x) + \ln(e^\tau),\xi) = f(\eta+\tau,\xi)$$

which is a translation in the new coordinate system. Suppose now that

$$\boldsymbol{B} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{where} \quad e^{\boldsymbol{B}\eta} = \begin{pmatrix} 1 & \eta & \frac{1}{2!}\eta^2 \\ 0 & 1 & \eta \\ 0 & 0 & 1 \end{pmatrix}.$$

Thus, the equivariant function space is spanned by the functions in $\Phi(\eta) \in \mathcal{R}(e^{\boldsymbol{B}\eta})$, like in Example 3 of Section 3.2.1. In this case, however, the function space is in η coordinates. After a change of coordinates, the function space in x coordinates is spanned by the functions in

$$\Phi(x) \in \mathcal{R}\left[\begin{pmatrix} 1 & \ln x & \frac{1}{2!}(\ln x)^2 \\ 0 & 1 & \ln x \\ 0 & 0 & 1 \end{pmatrix}\right].$$

3.2.4 Canonical Decomposition of One-Parameter Equivariant Spaces

For any one-parameter transformation group, the n-vector of equivariant functions Φ depends on the apriori choice of the $n \times n$ matrix $\boldsymbol{B}$. However, the same function space, $\text{span}(\Phi) \doteq \text{span}\{\phi_1, \cdots, \phi_n\}$, may be generated by different matrices $\boldsymbol{B}$. The following theorem provides an equivalence condition among the various matrices $\boldsymbol{B}$ that generate the same equivariant function space.

Theorem 2 *: Let Φ, Ψ be two n-vectors of equivariant functions (with respect to the same k-parameter transformation group) and $\boldsymbol{B_i}, \boldsymbol{B_i'}$ are such that $L_i\ \Phi = \boldsymbol{B_i}\ \Phi$ and*

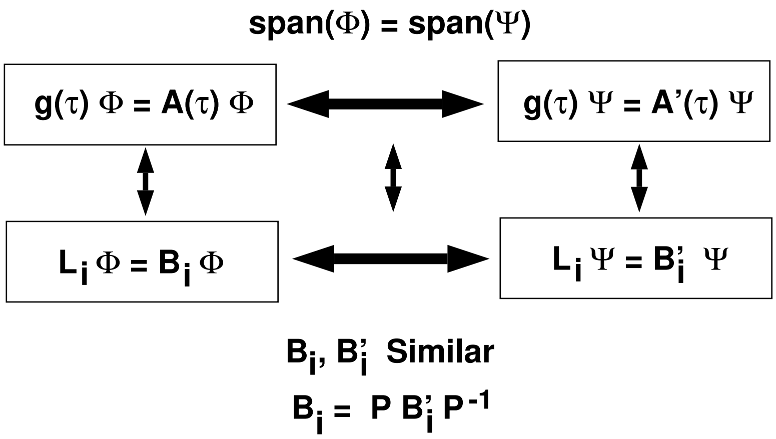


Figure 7: Two equivariant function spaces span the same function space if and only if their corresponding $\boldsymbol{B_i}$ matrices are similar.

$L_i\ \Psi = \boldsymbol{B_i}'\ \Psi$ where L_i and $\boldsymbol{B_i}, \boldsymbol{B}_i'$ are the infinitesimal generators of the transformation and interpolation matrix groups $\boldsymbol{A}(\boldsymbol{\tau}), \boldsymbol{A}'(\boldsymbol{\tau})$ respectively, then

$$\Phi = \boldsymbol{P}\ \Psi \qquad \text{iff} \qquad \boldsymbol{B_i} = \boldsymbol{P}\boldsymbol{B}_i'\boldsymbol{P}^{-1}$$

for all $1 \leq i \leq k$ and some non-singular $n \times n$ matrix $\boldsymbol{P}$.

Proof 2 : If $\Phi = \boldsymbol{P}\ \Psi$, then substituting into $L_i\ \Phi$, we get

$$L_i\Phi = L_i(\boldsymbol{P}\Psi) = \boldsymbol{P}\boldsymbol{B}_i'\Psi = (\boldsymbol{P}\boldsymbol{B}_i'\boldsymbol{P}^{-1})\boldsymbol{P}\Psi$$

and since $\boldsymbol{P}\ \Psi = \Phi$, it follows that $\boldsymbol{P}\boldsymbol{B}_i'\boldsymbol{P}^{-1} = \boldsymbol{B_i}$. Conversely, if $\boldsymbol{B_i} = \boldsymbol{P}\boldsymbol{B}_i'\boldsymbol{P}^{-1}$, then

$$\boldsymbol{B}_i'(\boldsymbol{P}^{-1}\Phi) = \boldsymbol{P}^{-1}\boldsymbol{B_i}\Phi = L_i(\boldsymbol{P}^{-1}\Phi);$$

thus, $L_i\ \Phi' = \boldsymbol{B_i}'\ \Phi'$ where $\Phi' = \boldsymbol{P}^{-1}\ \Phi$. Since Ψ is uniquely determined by the set $\boldsymbol{B_i}'$ (given $\Psi(\boldsymbol{0})$), $\Psi = \Phi' = \boldsymbol{P}^{-1}\ \Phi$. $\square$.

Thus, two equivariant function spaces under the same k-parameter transformation group span the same function space if and only if the infinitesimal generators $\boldsymbol{B_i}, \boldsymbol{B}_i'$ of their respective interpolation matrix groups are simulataneously similar for $1 \leq i \leq k$. This relationship is depicted in Figure 7. In particular, two vectors of functions, Φ and Ψ, which are equivariant with respect to the same one-parameter transformation group, span the same function space if and only if all their corresponding matrices $\boldsymbol{B}, \boldsymbol{B'}$ are *similar*. Hence, it suffices to examine all matrices $\boldsymbol{B}$ that are unique up to a similarity transformation. The *Jordan decomposition* is useful to this end since any two matrices that are similar share the same Jordan form [Str88].

With the Jordan decomposition, any $n \times n$ matrix $\boldsymbol{B}$ can be rewritten as $\boldsymbol{PJP^{-1}}$ such that $\boldsymbol{P}$ is a non-singular $n \times n$ matrix and $\boldsymbol{J}$ is a block-diagonal matrix of the form

$$\begin{bmatrix} \boldsymbol{J_1} & & \\ & \ddots & \\ & & \boldsymbol{J_k} \end{bmatrix}.$$

Each block $\boldsymbol{J_i}$ is a upper bidiagonal matrix with a single eigenvalue λ_i and one eigenvector:

$$\boldsymbol{J_i} = \begin{bmatrix} \lambda_i & 1 & & \\ & \cdot & \cdot & \\ & & \cdot & 1 \\ & & & \lambda_i \end{bmatrix}.$$

The matrix $\boldsymbol{J}$ is called the Jordan form of $\boldsymbol{B}$ and $\boldsymbol{J_i}$ are its Jordan blocks. A special case of the Jordan decomposition occurs when the matrix $\boldsymbol{B}$ is normal, i.e. $\boldsymbol{BB^H} = \boldsymbol{B^H B}$ where $\boldsymbol{B^H}$ is the complex conjugate of the transpose of $\boldsymbol{B}$. In this case, the Jordan decomposition yields a diagonal matrix $\boldsymbol{J}$; hence, each $\boldsymbol{J_i}$ is simply a 1×1 matrix containing the eigenvalue λ_i.

Let Φ_B, Φ_J be vectors of equivariant functions with respect to the translation group having corresponding matrices $\boldsymbol{B}, \boldsymbol{J}$ such that $\boldsymbol{J}$ is the Jordan form of $\boldsymbol{B}$, i.e. $\boldsymbol{B} = \boldsymbol{PJP^{-1}}$. From Theorem 2, then $\Phi_B = \boldsymbol{P}\ \Phi_J$. In other words, the function spaces spanned by Φ_B and Φ_J are identical. Furthermore,

$$\Phi_J(x) \in \mathcal{R}(e^{\boldsymbol{J}x}) = \mathcal{R}\begin{bmatrix} e^{\boldsymbol{J_1}x} & & \\ & \ddots & \\ & & e^{\boldsymbol{J_k}x} \end{bmatrix}.$$

Since $e^{\boldsymbol{J}x}$ is block diagonal, the function space spanned by Φ_J can be decoupled into a direct sum of function spaces spanned by each Jordan block:

$$\Phi_J(x) \in \mathcal{R}(e^{\boldsymbol{J}x}) = \mathcal{R}(e^{\boldsymbol{J_1}x}) \oplus \mathcal{R}(e^{\boldsymbol{J_2}x}) \oplus \cdots \oplus \mathcal{R}(e^{\boldsymbol{J_s}x}).$$

Moreover, each $\mathcal{R}(e^{\boldsymbol{J_i} x})$ is a solution to $L_{t_x}\Phi = \boldsymbol{J_i}\ \Phi$ and thus by itself equivariant under translation. Finally, from the identity [Str88],

$$e^{\boldsymbol{J_i} x} = \begin{bmatrix} e^{\lambda_i x} & x e^{\lambda_i x} & \frac{1}{2!} x^2 e^{\lambda_i x} & . \\ & e^{\lambda_i x} & x e^{\lambda_i x} & . \\ & & . & . \\ & & & e^{\lambda_i x} \end{bmatrix},$$

it follows that any equivariant function space spanned by $\Phi_J(x)$ can be represented by a direct sum of the equivariant function basis Φ_{J_i} of the form:

$$\Phi_{J_i} = (e^{\lambda_i x}, e^{\lambda_i x} x, e^{\lambda_i x} x^2, \cdots, e^{\lambda_i x} x^{n_i - 1})^T$$

where n_i is the dimension of the Jordan block $\boldsymbol{J_i}$ and λ_i is its eigenvalue. Note that if the matrix $\boldsymbol{B}$ is real, its eigenvalues appear in conjugate pairs; i.e., if one of the eigenvalues λ is complex, its conjugate $\bar{\lambda}$ is also an eigenvalue of $\boldsymbol{B}$. In this case, the equivariant spaces will appear in pairs:

$$\Phi_{J_i} \oplus \Phi_{\bar{J}_i} = (e^{\lambda_i x}, \cdots, e^{\lambda_i x} x^{n_i - 1})^T \oplus (e^{\bar{\lambda}_i x}, \cdots, e^{\bar{\lambda}_i x} x^{n_i - 1})^T.$$

When λ is zero, the equivariant space is spanned by the first n_i monomials (moments). Alternatively, when n_i is one and λ is purely imaginary, the space is spanned by the complex exponentials, which are also the Fourier basis functions. Since any one-parameter transformation group can be put into its canonical coordinates (where the group operation becomes a translation in these new coordinates), the decomposition of equivariant function spaces for translation applies directly to all other one-parameter transformation groups as well (after re-parameterization). Table 2 is a summary of several common one-parameter groups and their equivariant function spaces.

Example 7 : The following functions span an equivariant function space under the group of translations $g_{t_x}(\tau)$:

$$\Phi = \begin{pmatrix} \cos(kx) \\ \sin(kx) \end{pmatrix}$$

since

$$L_{t_x}\,\Phi = \boldsymbol{B}\,\Phi \qquad \text{where} \quad \boldsymbol{B} = \begin{pmatrix} 0 & -k \\ k & 0 \end{pmatrix}.$$

The interpolation equation associated with Φ is:

$$\hat{\Phi} = e^{\boldsymbol{B}\tau}\Phi \quad \text{where} \quad e^{\boldsymbol{B}\tau} = \begin{pmatrix} \cos(k\tau) & -\sin(k\tau) \\ \sin(k\tau) & \cos(k\tau) \end{pmatrix}$$

A different way to represent span(Φ) is by using functions generated by the Jordan form of $\boldsymbol{B}$:

$$\boldsymbol{J} = \boldsymbol{P}\boldsymbol{B}\boldsymbol{P}^{-1} = \begin{pmatrix} ik & 0 \\ 0 & -ik \end{pmatrix} \quad \text{where} \quad \boldsymbol{P} = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$$

By Theorem 2, it can verified that

$$\Phi_J = \boldsymbol{P}\Phi_B = \begin{pmatrix} e^{ikx} \\ e^{-ikx} \end{pmatrix}.$$

where Φ_J are two of the Fourier basis functions. The new interpolation equation in this case is:

$$\hat{\Phi} = e^{\boldsymbol{J}\tau}\Phi = \begin{pmatrix} e^{ik\tau} & 0 \\ 0 & e^{-ik\tau} \end{pmatrix}\Phi.$$

Example 8 : The following functions span an equivariant function space under $g_{t_x}(\tau)$:

$$\Phi = \left(\sin^3 x, \cos^3 x, 3\cos^2 x \sin x, 3\sin^2 x \cos x\right)^T$$

since

$$L_{t_x}\Phi = \boldsymbol{B}\Phi \qquad \text{where} \quad \boldsymbol{B} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 3 & 0 & -2 \\ -3 & 0 & 2 & 0 \end{pmatrix}$$

Transformation Group	Equivariant Function Space
$g_{t_x}(\tau)\ f(x,y) = f(x-\tau, y)$	$\{h(y)x^p e^{\alpha x}\}$
$g_{s_x}(\tau)\ f(x,y) = f(e^{-\tau}x, y)$	$\{h(y)x^{\alpha}(\ln x)^p\}$
$g_{p_x}(\tau)\ f(x,y) = f(x/(1+\tau x), y)$	$\{h(y)x^{-p}e^{\alpha/x}\}$
$g_{t_y}(\tau)\ f(x,y) = f(x, y-\tau)$	$\{h(x)y^p e^{\alpha y}\}$
$g_{s_y}(\tau)\ f(x,y) = f(x, e^{-\tau}y)$	$\{h(x)y^{\alpha}(\ln y)^p\}$
$g_{p_y}(\tau)\ f(x,y) = f(x, y/(1+\tau y))$	$\{h(x)y^{-p}e^{\alpha/y}\}$
$g_r(\tau)\ f(x,y) = f(x\cos\tau - y\sin\tau, x\sin\tau + y\cos\tau)$	$\{h(r)e^{ik\theta}\}$
$g_s(\tau)\ f(x,y) = f(e^{-\tau}x, e^{-\tau}y)$	$\{h(\theta)r^{\alpha}(\ln r)^p\}$

Table 2: Several examples of one-parameter transformation groups and their equivariant function spaces. The parameter k is any integer while the parameter α is any complex number. The function h is any arbitrary function. The variables r, θ refer to polar coordinates. The set notation describes a set of functions indexed by p for $0 \leq p \leq k$.

A different way to represent span(Φ) is with the basis functions determined by the Jordan form of $\boldsymbol{B}$:

$$\boldsymbol{J} = \boldsymbol{P}\boldsymbol{B}\boldsymbol{P}^{-1} = \begin{pmatrix} i & & & \\ & -i & & \\ & & 3i & \\ & & & -3i \end{pmatrix} \quad \text{where} \quad \boldsymbol{P} = \begin{pmatrix} 1 & -i & \frac{1}{3} & -\frac{1}{3}i \\ 1 & i & \frac{1}{3} & \frac{1}{3}i \\ 1 & -i & -1 & i \\ 1 & i & -1 & -i \end{pmatrix}.$$

and hence span(Φ_J) is determined by

$$\text{span}(\mathcal{R}(e^{\boldsymbol{J}x})) = \text{span}\left(e^{ix}, e^{-ix}, e^{3ix}, e^{-3ix}\right)^T.$$

The interpolation equation in this case is:

$$g_{t_x}(\tau)\Phi_J = e^{\boldsymbol{J}\tau}\Phi_J = \begin{pmatrix} e^{i\tau} & & & \\ & e^{-i\tau} & & \\ & & e^{3i\tau} & \\ & & & e^{-3i\tau} \end{pmatrix} \Phi_J.$$

3.3 Function Spaces for Multi-Parameter Groups

With one-parameter transformation groups (in their canonical coordinates), various equivariant function spaces can be constructed by choosing different $\boldsymbol{B}$ matrices; solutions to the system of partial differential equations $L\Phi = \boldsymbol{B}\Phi$ exist for arbitrary choices of $\boldsymbol{B}$. Unfortunately, there is no systematic way to construct general n-dimensional equivariant spaces for multi-parameter groups. Unlike one-parameter groups, arbitrary choices of $\boldsymbol{B_i}$ for multi-parameter groups will often not yield solvable systems of differential equations. For Abelian multi-parameter groups, i.e. groups made up of one-parameter subgroups that commute, however, a categorization of the equivariant spaces similar to that for one-parameter groups can be carried out. In the following, the categorization of equivariant spaces for Abelian multi-parameter groups is presented. After that, a technique for handling non-Abelian multi-parameter groups is suggested.

Abelian Multi-Parameter Groups When the multi-parameter transformation group is Abelian, there exists a re-parameterization of the group so that the group action is equivalent to independent translations in the new parameterization [Coh11, SRZ92, FC94]. Formally, for any two-parameter Abelian group, there exists a re-parameterization of the function $f(\eta(x,y),\xi(x,y))$ so that

$$g(\tau_1,\tau_2)\ f(\eta,\xi) = f(\eta+\tau_1,\xi+\tau_2).$$

Segman *et al.* in [SRZ92] describe a constructive way of determining this canonical re-parameterization. In the new parameterization, the equivariant space for the two-parameter group is simply the product of the equivariant spaces for each one-parameter translation group:

$$\mathrm{span}(\Phi(\eta,\xi)) = \mathrm{span}(\ \eta^p e^{\alpha\eta}\ ) \otimes \mathrm{span}(\ \xi^q e^{\beta\xi}\ )$$

for $0 \le p \le m$ and $0 \le q \le l$. Note that multi-parameter groups acting on a two-dimensional image with more than two parameters are necessarily not Abelian as there are only two independent translations in an image.

Example 9 : Consider the group of rotation and uniform scaling made up of the two one-parameter subgroups $g_r(\tau_1)$ and $g_s(\tau_2)$ from Table 2. The generators for these groups are $L_r = -x\frac{\partial}{\partial y} + y\frac{\partial}{\partial x}$ and $L_s = x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y}$ respectively. Recall that two one-parameter groups are Abelian if their generators commute; i.e., $[L_r, L_s] = L_r L_s - L_s L_r = 0$. It is easy to verify that this equality holds in our case. The re-parameterization that makes $g_r(\tau_1)$ and $g_s(\tau_2)$ act as translations on the image is:

$$\begin{aligned} \eta(x,y) &= \arctan(y/x) = \theta \\ \xi(x,y) &= \ln(\sqrt{x^2+y^2}) = \ln(r) \end{aligned}$$

Thus, the equivariant spaces for rotation and scaling are:

$$\text{span}(\ r^{\beta}(\ln r)^p\ ) \otimes span(\ e^{iq\theta}\ ) \quad \text{for} \quad 0 \leq p \leq m \quad \text{and} \quad q \in \mathbf{Z}\ .$$

The slight differences between the equivariant function spaces for rotation and scaling is due to the topology of the two transformation groups: rotation is topologically equivalent to a S while scaling is topologically equivalent to $\mathbf{R}$.

Non-Abelian Multi-Parameter Groups For multi-parameter transformation groups that are not Abelian, there are no re-parameterizations such that the group behaves like the group of independent translations in the new parameterization. One way to approach the problem is to start with the largest Abelian subgroup of the multi-parameter transformation group. The rest of the subgroups impose constraints on this initial equivariant function space by way of the differential equations: $L_i\Phi = \boldsymbol{B_i}\Phi$. Thus, the equivariant function space for the multi-parameter transformation group can be constructed by successively constraining the equivariant function space of the largest Abelian subgroup.

Example 10 : Consider the multi-parameter transformation group made up of translations in the x and y directions together with the group of rotations, i.e. g_{t_x}, g_{t_y} and g_r respectively. The largest Abelian subgroup is the two-parameter group of translations in the x and y directions. The equivariant function space for this group is: $\text{span}(\Phi) = \text{span}(x^p y^q e^{\alpha x + \beta y})$ for $0 \leq p \leq m$ and $0 \leq q \leq l$. The group of rotations yields the additional constraint that $L_r\Phi = \boldsymbol{B_r}\Phi$ where $L_r = -x\frac{\partial}{\partial y} + y\frac{\partial}{\partial x}$. By observation, we can rule out the exponentials $e^{\alpha x+\beta y}$ since applying L_r to each term introduces a monomial factor each time; repeated applications of the infinitesimal generator will not terminate since the monomials of different degrees are independent; thus $\alpha = \beta = 0$. Applying L_r to the monomial $x^p y^q$, however, raises the power in one variable and decreases the power in the other. Successive applications will result in one of the variables being reduced to zero. Hence, $\{x^p y^q\}$ is an equivariant function space under this group where $0 \leq p+q \leq m$ for some m. Note that this is not the only finite dimensional equivariant function space for this transformation group. Proceeding by first identifying the equivariant function space for rotation and then enforcing the constraints imposed by the infinitesimal generators of x and y translations L_{t_x} and L_{t_y} will produce the family of Bessel functions [Len90b].

Thus far, equivariant function spaces have been constructed in two steps: (1) matrices $\boldsymbol{B_i}$ are selected, (2) equivariant functions spaces are derived by solving the corresponding system of partial differential equations. Unfortunately, only for one-parameter transformation groups are we guaranteed to find a solution. For multi-parameter transformation groups, arbitrary choices of $\boldsymbol{B_i}$ will not always yield a solution. Alternatively, we could begin by selecting an interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$ and then derive its infinitesimal generators $\boldsymbol{B_i}$ and corresponding system of partial differential equations. The equivariant function space that is the solution to this system of partial differential equations can then be combined with itself via vector sums and tensor products to produce larger equivariant function spaces.

Example 11 : Consider the two-parameter group of translations and scalings in the x-direction:

$$g(\tau_1, \tau_2)\; s(x, y) = s(e^{-\tau_1}x - \tau_2, y).$$

We select the following interpolation matrix $\boldsymbol{A}$ for this group:

$$\boldsymbol{A}(\tau_1, \tau_2) = \begin{pmatrix} e^{-\tau_1} & -\tau_2 \\ 0 & 1 \end{pmatrix}.$$

Groups (# parameters)	Equivariant Function Space
x, y-translation (2)	$\{x^p y^q e^{\alpha x+\beta y}\}$ for $0 \leq p \leq m$ and $0 \leq q \leq l$.
x, y-scaling (2)	$\{x^\alpha y^\beta (\ln x)^p (\ln y)^q\}$ for $0 \leq p \leq m$ and $0 \leq q \leq l$.
Rotation Uniform-scaling (2)	$\{r^\alpha (\ln r)^p e^{ik\theta}\}$ for $0 \leq p \leq m$.
x, y-translation Rotation (3)	$\{x^p y^q\}$ for $0 \leq p+q \leq m$.
x, y-translation x, y-scaling (4)	$\{x^p y^q\}$ for $0 \leq p \leq m$ and $0 \leq q \leq l$.
x, y-translation x, y-scaling Rotation (5)	$\{x^p y^q\}$ for $0 \leq p+q \leq m$.

Table 3: Several examples of multi-parameter groups and their equivariant measuring spaces. The parameters $p, q, m, l, k \in \mathbf{Z}$ and $\alpha, \beta \in \mathbf{C}$.

Note that composition of two transformations $g(\tau_1^b, \tau_2^b)\, g(\tau_1^a, \tau_2^a)$ resembles the multiplication of the two corresponding interpolation matrices $\boldsymbol{A}(\tau_1^b, \tau_2^b)\, \boldsymbol{A}(\tau_1^a, \tau_2^a)$:

$$g(\tau_1^b, \tau_2^b)\, g(\tau_1^a, \tau_2^a) = g(\tau_1^a + \tau_1^b, e^{-\tau_1^b}\tau_2^a + \tau_2^b)$$

and

$$\boldsymbol{A}(\tau_1^b, \tau_2^b)\, \boldsymbol{A}(\tau_1^a, \tau_2^a) = \boldsymbol{A}(\tau_1^a + \tau_1^b, e^{-\tau_1^b}\tau_2^a + \tau_2^b).$$

This is not a mere coincidence; in fact, all interpolation matrices are related to their transformation groups in a similar way. More precisely, the group of interpolation matrices is homomorphic to the transformation group; likewise, their respective Lie algebras are also homomorphic to each other. From the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$ the infinitesimal generators can be derived:

$$\begin{aligned}
\boldsymbol{B_1} &\doteq \left.\frac{\partial}{\partial \tau_1}\boldsymbol{A}(\boldsymbol{\tau})\right|_{\boldsymbol{\tau}=\mathbf{0}} = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}, \\
\boldsymbol{B_2} &\doteq \left.\frac{\partial}{\partial \tau_2}\boldsymbol{A}(\boldsymbol{\tau})\right|_{\boldsymbol{\tau}=\mathbf{0}} = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}.
\end{aligned}$$

Subsequently, the vector of functions Φ are the solutions of the following system of partial differential equations:

$$\begin{aligned} L_1\,\Phi &= -x\tfrac{\partial}{\partial x}\,\Phi &= \boldsymbol{B_1}\,\Phi, \\ L_2\,\Phi &= -\tfrac{\partial}{\partial x}\,\Phi &= \boldsymbol{B_2}\,\Phi. \end{aligned}$$

The solution to these equations is $\Phi(x) = h(y)(x,1)^T$ where h is an arbitrary functions solely in y. Setting $h(y) \equiv 1$ for simplicity, the linearized tensor product space of $\oplus_{i=1}^{n}\Phi^i(x)$ where $\Phi^i(x) = \Phi(x) \otimes \cdots \otimes \Phi(x)$ multiplied i times is spanned by the functions $\{x^p\}$ for $0 \leq p \leq n$.

Table 3 is a summary of several common multi-parameter groups and their equivariant function spaces derived in similar ways.

3.4 Summary

We conclude with a summary of the results presented in this chapter.

1. Determining the basis functions for a steerable function under a given transformation group is equivalent to finding the lowest-dimensional equivariant function space containing the orbit of the steered function. Steerable functions are precisely those functions with finite dimensional equivariant spaces.

2. A function space is closed under a transformation group if and only if it is closed under each of the group's infinitesimal generators. This implies that the functions of an equivariant function space have to satisfy a system of linear, first order partial differential equations. Solving these differential equations results in a function space that is closed under the group transformation.

3. The steering function corresponding to a set of basis functions can be obtained analytically by exponentiating a series of matrices $\boldsymbol{B_i}$ that are the infinitesimal generators of the group of interpolation matrices $\boldsymbol{A}(\boldsymbol{\tau})$.

4. Two equivariant function spaces under the same group span the same function space if and only if the infinitesimal generators $\boldsymbol{B_i}, \boldsymbol{B_i'}$ of their group of interpolation matrices $\boldsymbol{A}(\boldsymbol{\tau}), \boldsymbol{A'}(\boldsymbol{\tau})$ are simultaneously similar $\boldsymbol{B_i} = \boldsymbol{P}\boldsymbol{B_i'}\boldsymbol{P^{-1}}$.

5. The function spaces equivariant under any Abelian multi-parameter group are made up of products of monomials and complex exponentials when expressed in canonical coordinates.

Chapter 4

Generator Trees

Once the equivariant function spaces for a given transformation group has been determined, selecting the minimal set of basis functions for a particular steerable function involves choosing the equivariant function space with the smallest dimension that contains the orbit of that function. This is often done by inspection. Typically, finding any equivariant function space that contains the orbit of the given function is relatively easy; one simply decomposes the function into a polynomial of simpler steerable functions for whom minimal equivariant function spaces have already been identified. However, determining the minimal equivariant function space is more complicated.

In this chapter, we investigate further the properties of the infinitesimal generators of a transformation group. We propose two graph-theoretic structures: in the case of one-parameter transformation groups, the structure is an ordered list or chain; in the case of multi-parameter transformation groups, the structure is a k-ary tree where k is equal to the number of parameters in the group. The nodes of both structures are differential operators constructed via repeated applications of the various infinitesimal generators of the group. Hence, the chain of differential operators is described as a *generator chain* while the tree of differential operators is described as a *generator tree*. These structures are derived from properties of the transformation group and are independent of the function that is to be steered.

We show that for one-parameter transformation groups, the minimal n-dimensional equivariant function space of any steerable function is spanned by the set of functions obtained by applying the first n differential operators in the generator chain to the steered function. Likewise, for k-parameter transformation groups, the minimal n-dimensional equivariant function space of any steerable function is spanned by the set of functions obtained by applying n differential operators to the steered function; these n differential operators correspond to all the nodes in a subtree of the k-ary generator tree such that the subtree and the generator tree share a common root. Applying the differential operators of the remaining nodes in the generator chain or tree to the steered function produces functions that are linearly dependent on the basis functions. These two results lead to a computationally efficient procedure for computing the minimal set of basis functions for any given steerable function. The computational complexity of the procedure is linear with respect to the actual minimal number of basis functions required, even in the case of multi-parameter groups.[1] We describe the results of a symbolic version of the procedure as well as a numerical implementation of it.

The material presented in this chapter can be found in [THO98c]; an early version of it is described in [THO97].

4.1 Generator Chains

In this section, we consider one-parameter transformation groups and introduce the generator chain associated with any one-parameter group. Before doing so, however, we recall that a function is steerable with a minimum of n basis functions provided there exists an n-dimensional equivariant function space containing the orbit of that function. The choice of basis functions for that equivariant function space is not unique; any (non-singular) linear transformation of a chosen set of basis functions could also be used. Furthermore, if the function can be steered with $m > n$ other basis functions, then $m-n$ of them are necessarily linearly dependent on the remaining n basis functions, and all choices of n basis functions span the same function space.

[1]The linear complexity refers to the number of nodes in either the generator chain or tree that needs to be visited. At each node, a test for linear dependence on the current basis set needs to be performed. Thus, a naive implementation would have quadratic complexity with respect to the number of inner-products necessary.

Thus, the minimal equivariant function space for any steerable function is unique. We formalize this notion in the following theorem.

Theorem 3 (Minimality of Basis Functions) *: Let $\Phi = (\phi_1, \cdots, \phi_n)^T$ be the minimum set of independent basis functions required to steer a function f under a Lie transformation group G. Then, any other steerable basis $\Psi = (\psi_1, \cdots, \psi_m)^T$ of f with respect to G has exactly n linearly independent functions.*

Proof 3:

Assume that m is the minimum number of linearly independent functions in Ψ to steer f. Therefore, it is possible to find m transformed replicas of f that are linearly independent (otherwise m is not minimal):

$$\begin{pmatrix} g(\boldsymbol{\tau}^1)f \\ \vdots \\ g(\boldsymbol{\tau}^m)f \end{pmatrix} = \begin{bmatrix} \boldsymbol{\beta}^T(\boldsymbol{\tau}^1) \\ \vdots \\ \boldsymbol{\beta}^T(\boldsymbol{\tau}^m) \end{bmatrix} \Psi \doteq \boldsymbol{B}\Psi$$

where $\boldsymbol{\beta}^T(\boldsymbol{\tau})$ is composed of the steering functions associated with Ψ and $\boldsymbol{\tau}^1, \ldots, \boldsymbol{\tau}^m$ are particular choices of steering parameters. Since the m transformed replicas are linearly independent, $\boldsymbol{B}$ is a non-singular matrix. These m transformed replicas of f can also be constructed using the n basis functions of Φ:

$$\begin{pmatrix} g(\boldsymbol{\tau}^1)f \\ \vdots \\ g(\boldsymbol{\tau}^m)f \end{pmatrix} = \begin{bmatrix} \boldsymbol{\alpha}^T(\boldsymbol{\tau}^1) \\ \vdots \\ \boldsymbol{\alpha}^T(\boldsymbol{\tau}^m) \end{bmatrix} \Phi \doteq \boldsymbol{A}\Phi = \boldsymbol{B}\Psi \ .$$

Since $\boldsymbol{B}$ is invertible it is possible to express Ψ as a linear combination of Φ:

$$\Psi = \boldsymbol{B}^{-1}\boldsymbol{A}\Phi.$$

Now, if Ψ includes $m > n$ functions, it is obvious from the above equation that $m - n$ of them are linearly dependent. This contradicts the minimality assumption of m. On the other hand, if $m < n$, then n is not minimal. Thus, it must be true that $m = n$ and all the n functions in Ψ are linearly independent. □

We now describe a method of constructing a basis for a given steerable function f under a transformation G. From Theorem 3, this basis can be related to any other steerable basis of f via a linear transformation. Associated with each one-parameter transformation group G is its generator L. As shown in Chapter 2, the generator L is a differential operator corresponding to an infinitesimal transformation about the identity. Applying L to a function f results in a new function Lf; likewise, applying L a second time to the previous result yields another function which we denote by $L^2 f = L(Lf)$. Alternatively, we could also regard L^2 (or L^j, $j \geq 0$) as a new differential operator that is applied to f. The set of all such differential operators is collected into a sequence in the following definition.

Definition 3 (Generator Chain) *: A* generator chain $\mathcal{C}(L)$ *is an ordered sequence of differential operators corresponding to repeated applications of the given generator* L*; i.e.,*

$$\mathcal{C}(L) = (I,\ L,\ L^2,\ L^3,\ \ldots)$$

where I *corresponds to zero applications of the generator.*

The result of applying $\mathcal{C}(L)$ to a function f is defined to be the ordered sequence of functions:

$$\mathcal{C}(L)\ f = (f,\ Lf,\ L^2 f,\ L^3 f,\ \ldots).$$

Using the exponential map given in Equation 4, the series formed by summing all the functions in the sequence is exactly the same as transforming f by an element $g(\tau) \in G$:

$$g(\tau)\ f = \exp(\tau L)\ f = \sum_{i=0}^{\infty} \frac{\tau^i}{i!} \left(L^i f\right). \tag{6}$$

Thus, the set of functions $\mathcal{C}(L)\ f$ provides a basis with which f can be steered. From Theorem 3 it follows that this basis is redundant if n functions are required to steer f; only n of the functions in $\mathcal{C}(L)\ f$ are linearly independent. It turns out that these n functions are necessarily the *first* n functions of the chain. The minimality of the generator chain is formalized in the following theorem.

Theorem 4 (Minimality of Generator Chain) *: Let f be a steerable function under a one-parameter Lie transformation group G such that transformations of f by any element $g \in G$ can be written as a linear combination of (no less than) n basis functions. Let L denote the infinitesimal generator of G. The application of the generator chain $\mathcal{C}(L)$ to f results in an ordered sequence of functions such that the elements $i > n$ of the sequence, corresponding to $L^{(i-1)}f$, are linearly dependent on the first n elements. Furthermore, the first n functions of the sequence are linearly independent.*

Proof 4 : Let the $(m+1)^{\text{st}}$ function in the sequence be the first linearly dependent function. That is, $L^m f$ can be written as a linear combination of the first m linearly independent functions. As a result, all subsequent functions in the sequence $L^j f$ where $j > m$ are necessarily linearly dependent on the first m functions as well. This can be proven by strong induction where $j = m+1$ is the base case. let $l^m f = \sum_{i=0}^{m-1} a_i l^i f$. then,

$$\begin{aligned} l^{m+1} f &= l(l^m f) = l(\textstyle\sum_{i=0}^{m-1} a_i l^i f) \\ &= \textstyle\sum_{i=0}^{m-1} a_i l^{i+1} f \\ &= a_{m-1} l^m f + \textstyle\sum_{i=0}^{m-2} a_i l^{i+1} f \end{aligned}$$

but since l^m is linearly dependent on the first m functions, $l^{m+1} f$ can also be expressed as a linear combination of these functions. the proof of the inductive case is similar. As a result, Equation 6 implies that transformations of f can be written as a linear combination of the first m functions in $\mathcal{C}(L)\ f$. Because f is steerable with n basis functions (by assumption), it follows from Theorem 3 that m must equal n. □

This theorem suggests the following procedure to compute a set of basis functions to steer an arbitrary function f. The generator L is applied to f repeatedly and each time, the linear dependence of the new function upon the previously computed functions is checked. If it is linearly dependent, then the set of all functions computed prior to this one is sufficient to steer f. It the function f is steerable with n basis functions, then the procedure will terminate after $n+1$ iterations. Figure 8 describes the procedure in pseudo-code. The procedure is applied to the following two examples.

```
/* Compute the basis functions needed to */
/* steer f under a one-parameter group. */
next_f = f;
basis_set = {};
while (next_f not linearly dependent on basis_set) {
      basis_set = basis_set ∪ {next_f}
      next_f = L next_f;
}
return( basis_set );
```

Figure 8: Procedure for computing the basis functions to steer an arbitrary function f under a one-parameter group.

Example 12 : Let $f(x,y) = -2x\ e^{-(x^2+y^2)}$ and G be the group of rotations in the plane. The generator of G is $L = x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}$, and

$$L^0 f = -2x\ e^{-(x^2+y^2)} = f \quad ; \quad L^1 f = 2y\ e^{-(x^2+y^2)},$$

and $L^2 f = 2x\ e^{-(x^2+y^2)} = -f$. Therefore, f is steerable under G with two basis functions; i.e., the derivative of a Gaussian in any direction can be expressed as a linear combination of two functions.

Example 13 : Let $f(x) = (\cos x + 1)^2$ and G be the group of x-translation: $g_r(\tau)\ f(x) = f(x-\tau)$. The generator of G is $L = -\frac{\partial}{\partial x}$, and

$$\begin{aligned}
L^0 f &= (\cos x + 1)^2, \\
L^1 f &= 2(\cos x + 1)\sin x = \sin 2x + 2\sin x, \\
L^2 f &= -2\cos 2x - 2\cos x, \\
L^3 f &= -4\sin 2x - 2\sin x, \\
L^4 f &= 8\cos 2x + 2\cos x,
\end{aligned}$$

and $L^5 f = 16\sin 2x + 2\sin x = -4L^1 f - 5L^3 f$. Therefore, f is steerable under G with the five basis functions, $\{L^0 f, \ldots, L^4 f\}$. The particular choice of basis functions is not unique; in this example, the function f is also steerable with the following five basis functions, $\{1, \sin x, \cos x, \sin 2x, \cos 2x\}$.

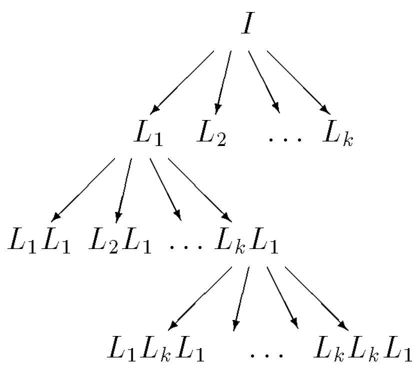
4.2 Generator Trees

In this section, we consider multi-parameter transformation groups. Let the set of differential operators, $\{L_1, \ldots, L_k\}$, be the k generators of the k-parameter transformation group G. In the context of multi-parameter transformation groups, the generator chain is no longer a chain since more than one generator may be applied; instead, we have a tree of differential operators. Nodes in the tree correspond to all possible compositions of the generators.

Definition 4 (Generator Tree) *: A* generator tree $\mathcal{T}(L_1, \ldots, L_k)$ *is a k-ary tree of differential operators corresponding to repeated applications of the generators $L_1, \ldots, L_k$. Each node of the tree has k children, which correspond to applying each of the L_k different generators. Level l of the tree contains k^l nodes, each of which represents the different permutations of applying $L_1, \ldots, L_k$ repeatedly for a total of l times.*

For example,

$\mathcal{T}(L_1, \ldots, L_k) =$



Similar to generator chains, applying $\mathcal{T}(L_1, \ldots, L_k)$ to a function f results in a k-ary tree where each node corresponds to the function obtained by applying the generators to f. Furthermore, using the exponential map given in Equation 4, transforming f by an element $g(\tau_1, \ldots, \tau_k) \in G$ can be calculated by a linear combination of functions in the tree $\mathcal{T}(L_1, \ldots, L_k)\ f$:

$$g(\tau_1, \ldots, \tau_k)\ f = e^{\tau_1 L_1} \cdots e^{\tau_k L_k}\ f = \left(\prod_{i=1}^{k} \sum_{l=0}^{\infty} \frac{\tau_i^l}{l!} L_i^l \right)\ f \tag{7}$$

Thus, the set of functions obtained by applying $\mathcal{T}(L_1, \ldots, L_k)$ to a function f provides a basis with which to steer f. Similar to the case with generator chains, this basis is redundant if only n functions are required to steer f. It turns out that the n functions needed to steer f necessarily form a subtree of $\mathcal{T}(L_1, \ldots, L_k)\ f$ with the same root. This property generalizes the minimality property of generator chains associated with one-parameter groups.

Theorem 5 (Minimality of Generator Tree:) *Let f be a steerable function under a k-parameter Lie transformation group G such that transformations of f by any element $g \in G$ can be written as a linear combination of (no less than) n basis functions. Let $L_1, \ldots, L_k$ denote the generators of G. The application of the generator tree $\mathcal{T}(L_1, \ldots, L_k)$ to f results in a k-ary tree of functions such that there exists a subtree $\mathcal{T}'(L_1, \ldots, L_k)\ f$ (with the same root) satisfying the following two conditions: (1) all functions within the subtree are linearly independent of one another, and (2) all functions in the original tree but not in the subtree are linearly dependent on functions within the subtree.*

Proof 5 : Let $\mathcal{T}'(L_1, \ldots, L_k)\ f$ be a subtree of $\mathcal{T}(L_1, \ldots, L_k)\ f$ (with the same root) such that: (1) all the functions within the subtree are linearly independent, and (2) all the functions that are immediate children of the subtree (as part of the original tree) are linearly dependent on the functions within the subtree. Then, all the descendents of the immediate children are also linearly dependent on the functions within the subtree. This can be proven in a way similar to that for generator chains in Theorem 4. The strong induction, in this case, is on subtrees. Also, from Theorem 3 it follows that the total number of linearly independent functions in the original tree is necessarily n since f is steerable. As a result of the former property, the size of the subtree must be n as well. □

Unlike the situation with generator chains, this minimal subtree is not unique. That is, there may be two subtrees of the same size (and with the same root as

```
/* Compute the basis functions needed to */
/* steer f under a k-parameter group. */
next_set = { f };
basis_set = { f };
while (next_set is not empty) {
   next_set' = {};
   for each (next_f ∈ next_set) {
      for each (L ∈ {L_1, ..., L_k}) {
         next_f' = L next_f;
         if (next_f' not linearly-dependent
                        on basis_set) {
            next_set' = next_set' ∪ {next_f'};
            basis_set = basis_set ∪ {next_f'};
         }
      }
   next_set = next_set';
   }
}
return( basis_set );
```

Figure 9: Procedure for computing the basis functions to steer an arbitrary function f under a k-parameter group. The nodes in the generator tree are tested in a breadth-first manner.

the original tree) that could be used to steer f. However, since f is steerable, the functions in these two trees necessarily span the same space.

This theorem also suggests a procedure for computing the basis functions needed to steer an arbitrary function f. Each of the generators $\{L_1, \ldots, L_k\}$ is applied to f repeatedly. Each new function is then checked to determine if it is linearly dependent on all the previously computed functions. If it is linearly dependent, then one need not further apply any generators to this function. If the function f is steerable with n basis functions under a k-parameter group, then the procedure will terminate after testing at most $nk+1$ functions.[2] Figure 9 describes the procedure in pseudo-code. The procedure is applied to the following two examples.

Example 14 : Let $f(x) = x^2$ and G be the group of one-dimensional scaling and translations: $g(\tau_1, \tau_2)\ f(x) = f(e^{-\tau_1}x - \tau_2)$. The generators of G are $L_{\tau_1} = -x\frac{\partial}{\partial x}$ and $L_{\tau_2} = -\frac{\partial}{\partial x}$. It can be seen that:

$$f = x^2 \quad ; \quad L_{\tau_1} f = -2x \quad ; \quad L^2_{\tau_1} f = 2$$

are the first three functions that span the entire generator tree $\mathcal{T}(L_{\tau_1}, L_{\tau_2})$. Thus, any other node in the generator tree is linearly dependent on these functions.

Example 15 : Let $f(x,y) = \sin x \sin y$ and G be the group of translation along the x and y dimensions: $g(\tau_1, \tau_2)\ f(x) = f(x - \tau_1, y - \tau_2)$. The generators of G are $L_{\tau_1} = -\frac{\partial}{\partial x}$ and $L_{\tau 2} = -\frac{\partial}{\partial y}$:

$$\mathcal{T}(L_{\tau_1}, L_{\tau_2})\ f \quad = \quad \begin{tikzcd} & \sin x \sin y \arrow[dl] \arrow[dr] & \\ -\cos x \sin y \arrow[dr] & & -\sin x \cos y \\ & \cos x \cos y & \end{tikzcd} .$$

Since translation in the x and y dimensions are commutative, their generators commute as well; i.e., $L_{\tau_1} L_{\tau_2} = L_{\tau_2} L_{\tau_1}$. Thus, the left child of the node with $-\sin x \cos y$ is automatically pruned since it will be the same as the right child of the node with $\cos x \cos y$. Therefore, f is steerable under G with four basis functions.

[2] A k-ary tree with one node has k immediate children. Each addition of a new node increases the number of immediate children by $k-1$; adding $n-1$ nodes results in a total number of $(n-1)(k-1)+k = n(k-1)+1$ immediate children in the tree. Thus, a k-ary tree with n nodes (internal nodes as well as leaves) has exactly $n(k-1)+1$ immediate children. The number of times the linear dependence test is invoked is equal to the sum of the number of basis functions required and the number of immediate children in the resultant k-ary generator tree. Therefore, the total number of times the test is applied is $n + n(k-1) + 1 = nk + 1$.

4.3 Simulations

In this section, we present two applications of the theory described in the previous section. The first application is an implementation of the procedure described in Section 4.2 for steering polynomials. The second application is a numerical implementation of the same procedure for approximately steering any sampled function.

4.3.1 Steering Polynomials

The procedure described in Section 4.2 was implemented in MATLAB to automatically determine the basis functions needed to steer an arbitrary two-dimensional polynomial under any subgroup of the two-dimensional affine transformation. In Chapter 3, we saw that such polynomials can be steered, with a finite number of basis functions, under any subgroup of the affine group. Thus, the procedure is guaranteed to terminate after a finite number of iterations.

In the procedure, the linear independence of a polynomial with respect to the current basis set needs to be determine. This is done by representing each polynomial in terms of the basis of monomials $\{1, x, y, x^2, xy, y^2, \ldots\}$. Specifically, let the matrix $\boldsymbol{B}$ be an $m \times n$ matrix of coefficients and $\boldsymbol{m}$ be the $n \times 1$ vector of monomials such that $\boldsymbol{Bm}$ yields an $m \times 1$ vector corresponding to the m basis polynomials. Similarly, expressing the new polynomial in the monomial basis results in a $1 \times n$ vector $\boldsymbol{b}$ of coefficients. Since the monomials are linearly independent, the new polynomial is linearly dependent on the basis set if and only if $\boldsymbol{b}$ is in the row space of $\boldsymbol{B}$. The generators for each one-parameter subgroup (e.g. x-translation, y-translation, etc.) are implemented as operations on the coefficients of the polynomial representation. This is possible since applying any generator to a polynomial always results in another polynomial.

The cubic polynomial $x^3 + 3x^2y + 3xy^2 + y^3$ (`mpoly2`) is used in the following examples. The basis functions needed to steer the function under different multi-parameter groups are computed automatically.

1. In this example, basis functions to steer the polynomial under the group of

uniform scaling and rotation are computed. The generators are $L_s = -x\frac{\partial}{\partial x} - y\frac{\partial}{\partial y}$ and $L_r = x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}$ respectively.

```
poly2_mat = steer_poly2(mypoly2, 'lscale', 'lrot')
```

$$\begin{aligned}
&x^3 + 3x^2y + 3xy^2 + y^3,\\
&3x^3 + 3x^2y - 3xy^2 - 3y^3,\\
&3x^3 - 15x^2y - 15xy^2 + 3y^3,\\
&-15x^3 - 39x^2y + 39xy^2 + 15y^3.
\end{aligned}$$

2. In this example, basis functions to steer the polynomial under the group of translations in the x and y directions are computed. The generators are $L_x = -\frac{\partial}{\partial x}$ and $L_y = -\frac{\partial}{\partial y}$ respectively.

```
poly2_mat = steer_poly2(mypoly2, 'ltransx', 'ltransy')
```

$$\begin{aligned}
&x^3 + 3x^2y + 3xy^2 + y^3,\\
&-3x^2 - 6xy - 3y^2,\\
&6x + 6y,\\
&-6.
\end{aligned}$$

3. In this example, basis functions to steer the polynomial under the group of translations in the x and y directions and rotation are computed. The generators are $L_x = -\frac{\partial}{\partial x}$, $L_y = -\frac{\partial}{\partial y}$, and $L_r = x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}$ respectively.

```
poly2_mat = steer_poly2(mypoly2, 'ltransx', 'ltransy', 'lrot')
```

$$\begin{aligned}
&-15x^3 - 39x^2y + 39xy^2 + 15y^3,\\
&3x^3 - 15x^2y - 15xy^2 + 3y^3,\\
&3x^3 + 3x^2y - 3xy^2 - 3y^3,\\
&x^3 + 3x^2y + 3xy^2 + y^3,\\
&-3x^2 - 6xy - 3y^2,\\
&-6x^2 + 6y^2,
\end{aligned}$$

$$6x + 6y,$$
$$6x - 6y,$$
$$24xy,$$
$$-6.$$

Clearly, the basis comprising of all the monomials in x, y up to powers of three, a total of ten, is sufficient to steer the cubic polynomial. However, as can be seen in the examples above, fewer than ten are actually needed in some situations. The procedure selectively retains only those necessary by removing those that are linearly dependent (with respect to the infinitesimal generators of the group).

4.3.2 Numerical Simulations

A numerical version of the procedure was also implemented. The program automatically computes a set of basis functions that can be used to steer a given two-dimensional function. The derivatives in the generators were approximated using numerical derivatives. The linear dependence of a function on the current set of basis functions is verified by projecting the function onto an orthogonalized version of the basis set and measuring the relative magnitude of the residual. The set of orthogonal basis functions can be efficiently computed by using the Gram-Schmidt technique iteratively.

Since the procedure is not guaranteed to terminate for arbitrary functions as an infinite number of basis functions might be required, the check for linear dependence was replaced by a numerical condition that the residual between the function and its projection is below some threshold. Furthermore, the maximum depth of the generator tree was also used as a termination criteria since higher-order numerical derivatives are less accurate. As a result, the steering of the given function with the basis set is only accurate to within some range of transform parameters as we shall see.

Figure 10 shows the four basis functions that could be used to steer (under rotation) the function $(12x - 8x^3)\exp[-(x^2 + y^2)]$, which is the third derivative of a

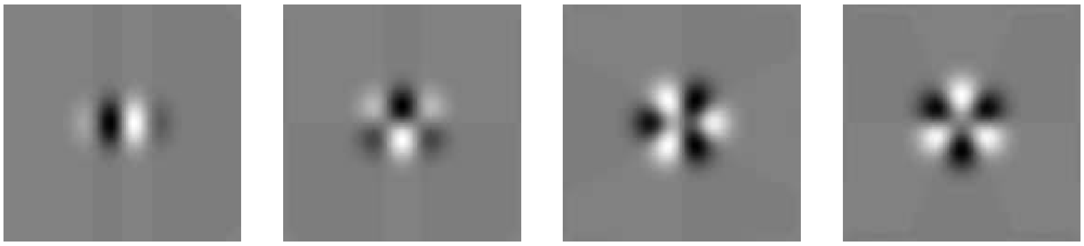


Figure 10: Basis functions that steer $(12x - 8x^3)\exp[-(x^2 + y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure.

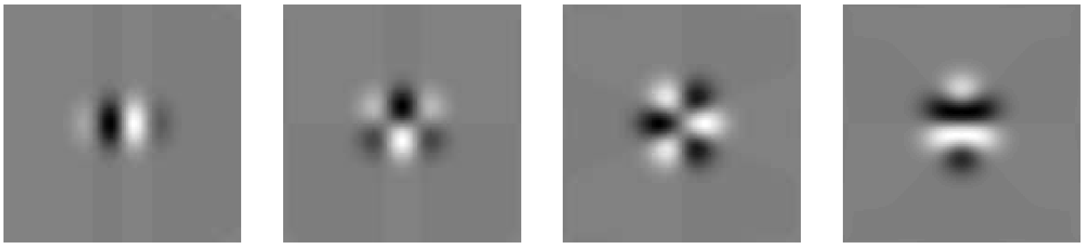


Figure 11: Orthogonal basis functions that steer $(12x - 8x^3)\exp[-(x^2 + y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure.

Gaussian. The leftmost image is the function that was used as input to the procedure; i.e., the function to be steered. The spatial extent of each image ranges from -5 to 5 units both horizontally and vertically. Figure 11 shows images of four orthogonal basis functions that could also be used to steer the function. These basis functions were computed by the Gram-Schmidt component of the procedure.

Unlike the third derivative of a Gaussian, the function $\sin(x)\exp[-(x^2 + y^2)]$ cannot be perfectly steered under rotation. Figure 12 shows images of the four basis functions returned by the procedure. The maximum tree-depth was set at 6 and the maximum relative squared error of the residual was 5%. The relative squared error of using these basis functions to steer the function under any rotation was always less than 0.1%. Figure 13 shows images of four orthogonal basis functions that could also be used to steer the function.

Figure 14 shows the 22 basis functions that were computed to steer the function $(4x^2 - 2)\exp[-(x^2 + y^2)]$ under any x, y translation and rotation. Again, the steering

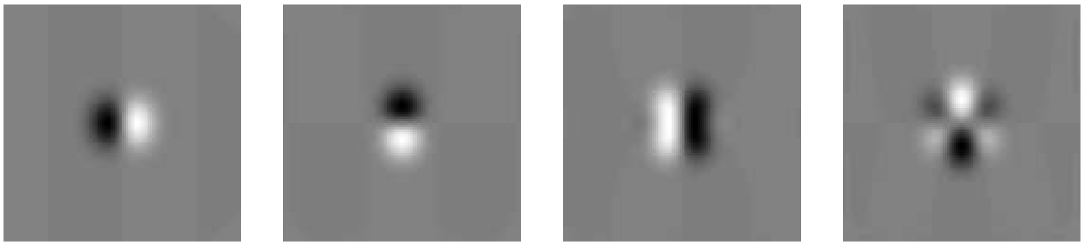


Figure 12: Basis functions that steer $\sin(x)\exp[-(x^2+y^2)]$ under rotation. The leftmost image is the function that was used as input to the procedure.

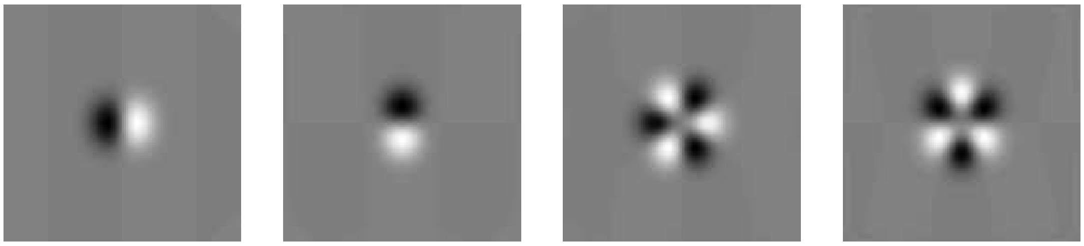


Figure 13: Orthogonal basis functions that steer $\sin(x)\exp[-(x^2+y^2)]$ under rotation. The leftmost image is the function that was used as input to the procedure.

is only approximate since the function cannot be steered with a finite number of basis functions. The maximum tree-depth of the procedure was set at 3 and the maximum relative squared error of the residual was 10%. Figure 15 shows the corresponding 22 orthogonal basis functions. Figure 16 (left) plots the relative squared errors of steering the function using this basis for a range of translations. The approximation is very good about the origin (zero translation) and worsens when the translations are large. Figure 16 (right) plots the relative squared errors of steering a translated version of the function over all rotation angles. The errors in steering the untranslated function are negligible since the second derivative of a Gaussian can be perfectly steered with three basis functions under orientation.

4.4 Discussion

The proposed method for computing basis functions to steer a given function essentially computes the Taylor expansion of the function with respect to the transform parameters. The expansion is evaluated at the origin of the transform parameters. Several simplifications arise because the transform is a Lie transformation group. The principal one being that the Taylor expansion can be written solely in terms of the first order derivatives, namely, the generators. Equation 6 of Section 4.1 and Equation 7 of Section 4.2 describe the Taylor expansion in terms of the generator(s) for one-parameter and multi-parameter groups respectively. Since higher order derivatives can be determined from these generators, properties involving the higher order derivatives can be proven. These properties are precisely those that were used to show the minimality of generator chains and generator trees.

As a result of this close connection with Taylor expansions, the errors incurred in approximately steering a function increases with the deviation of the transform parameters from the origin. This happens when the function to be steered cannot be steered by a finite number of basis functions. This can be seen from the numerical implementation of the procedure in Section 4.3. This may be acceptable for some applications where only a limited range of steering is required. For example, Manmatha [Man94] uses a similar approach to estimate the affine transformation of points, lines and image intensities. However, if the function needs to be steered over a larger range of parameters, then either more basis functions could be computed by increasing the maximum tree-depth or by applying the Taylor expansion about another location other than the origin. The basis functions computed by this method, in fact, minimizes the approximation error about the particular transform parameter. Instead, if the criterion is to minimize the average error over a range of transform parameters, then fewer basis functions are required. In the next chapter, an efficient method of computing the basis functions that minimizes this approximation error is described.

If the function to be steered is obtained from a space of functions that is steerable, then the function can be steered with a finite number of basis functions. Consequently, an analytic version of the procedure could be applied to determine the smallest basis set required. The example for polynomials is illustrated in Section 4.3. As shown in that section, while each monomial could be used as a basis function to steer the given polynomial, often fewer basis functions are sufficient because of various linear dependencies. A similar algorithm could also be implemented for sinusoids over translation/rotation or spherical harmonics over $3D$ rotation.

4.5 Summary

We conclude this chapter with a summary of the results presented.

1. Any function steerable with n basis functions under some transformation group can be associated with a unique n-dimensional equivariant function space.

2. For one-parameter transformation groups, any function steerable with n basis functions can be steered using the n functions obtained by applying to the steered function the differential operators of the first n nodes in the generator chain of the group.

3. For k-parameter transformation groups, any function steerable with n basis functions can be steered using the n functions obtained by applying to the steered function the differential operators corresponding to all the n nodes of some subtree sharing the same root as the generator tree of the group.

4. The procedure to compute the minimal n basis functions of a steerable function is linear in n.

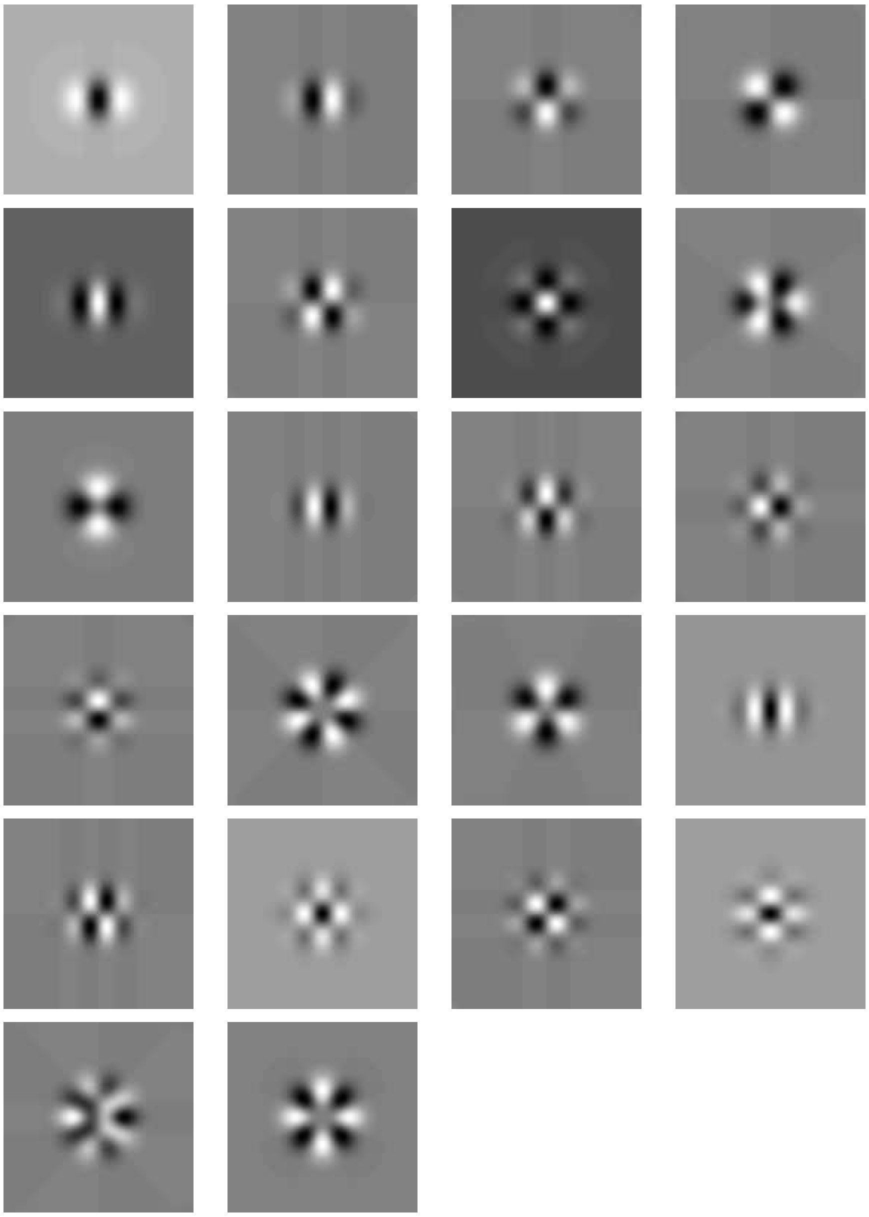


Figure 14: The 22 basis functions that steer $(4x^2 - 2)\exp[-(x^2 + y^2)]$ under $x, y-$ translation and rotation. The image at the top left is the function that was used as input to the procedure.

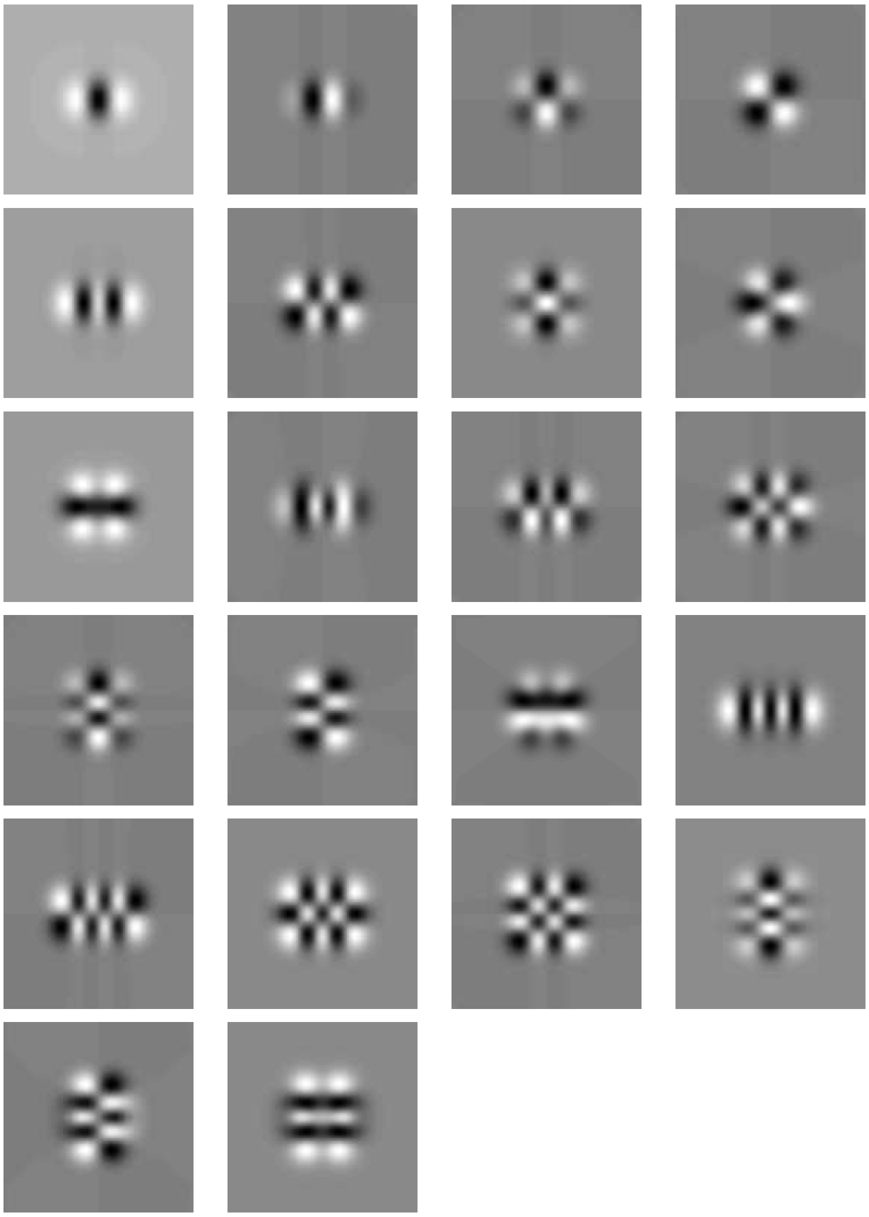
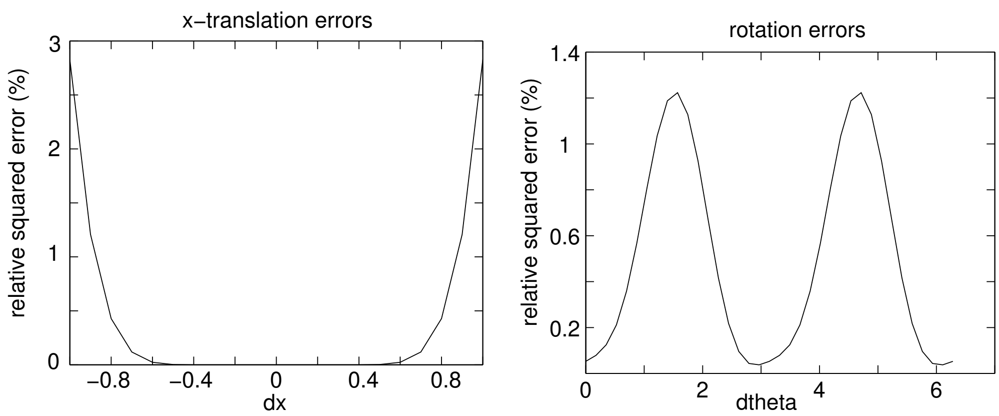


Figure 15: The 22 orthogonal basis functions that steer $(4x^2 - 2)\exp[-(x^2 + y^2)]$ under $x, y-$ translation and rotation. The image at the top left is the function that was used as input to the procedure.



(a) (b)

Figure 16: Graph (a): Relative squared errors of the steered approximations to the actual functions over a range of x-translations. The graph describing the errors with respect to the y-translations is virtually identical. Graph (b): Relative squared errors of the steered approximations to the actual functions over the entire range of rotation angles. The actual function has been translated by 0.5 units in both the x and y dimensions; i.e., $(4(x-0.5)^2-2)\exp[-((x-0.5)^2+(y-0.5)^2)]$. The percentage errors for an untranslated function are negligible.

Chapter 5

Cascade Basis Reduction

In earlier chapters, we investigated functions that were perfectly steerable; that is, functions that could be expressed (in analytic form) as linear combinations of a fixed set of basis functions when transformed under some transformation group. However, as was pointed out earlier, not all functions are perfectly steerable. In practice, the recipe for approximately steering such a non-steerable function is to first approximate it using an equivariant function space and then steer the approximation. Such a method is empirically effective; however, typically a moderate number of basis functions is required.

An alternative approach to designing basis functions that can be used to steer a given function is the singular value decomposition method proposed by Perona [Per95]. As described in Chapter 1, the technique computes the optimal (in a least-squares sense) set of basis functions to steer a given function under an arbitrary (compact) transformation. Unlike the Lie group-theoretic method involving equivariant function spaces, this method is not restricted to Lie transformation groups. This restriction, fortunately, is not too severe as many useful transformations are Lie transformation groups. Because it computes the least-squares optimal set of basis functions, this method typically requires fewer basis functions than the previous method when steering a function over a transformation group. The main drawback of the SVD method, however, is that its computational complexity increases exponentially with the number of transform parameters. Hence, using this method even for groups with a moderate number of parameters, like the four-parameter group of linear image transformations, is infeasible.

Besides commonly requiring more basis functions, the main shortcoming of the Lie group-theoretic approach is that the steerability property is enforced *globally*; that is, a function is designed to be steered by *any* transformation in the group. For non-compact groups (like translation and scaling), the basis functions spanning their equivariant function spaces have infinite support. If the function to be steered has compact-support, then a large number of basis functions are needed to approximate it accurately. In practice, it is reasonable to assume that only transformations over a limited range of parameters can be expected. The SVD method proposed by Perona can be used when the number of transform parameters is less than or equal to two; however, for larger number of parameters, the method is computationally intractable.

In this chapter, we present a new method of computing the optimal least-squares set of basis functions to steer a given function within a limited range of transform parameters that is computationally efficient for larger numbers of transform parameters. The efficiency of the method is demonstrated by the design of a set of basis functions to steer a Gabor function under the four-parameter linear transformation group. The method combines the Lie group-theoretic and the singular value decomposition approaches in such a way that their respective strengths complement each other. The hybrid method comprises two steps. First, the Lie group-theoretic approach is used to compute the basis functions to steer the given function locally, i.e., within a compact range of transform parameters. Since these basis functions (equivariant functions) are already known to be steerable under the given transformation group, the computational complexity of this step is independent of the number of transform parameters. In the second step, the singular value decomposition technique is used to determine the optimal least-squares set of basis functions and thereby reduce the current number of basis functions. The computational complexity of this second stage is shown to be only dependent on the number of basis functions used in the first stage. Since the original group-theoretic basis functions (and their steering functions) are available in analytic form, the least-squares optimal set of basis functions (and their steering functions) can also be derived in analytic form.

The material presented in this chapter can be found in [THO98a]; an early version of it is described in [THO98b].

5.1 Local Steerability

In this section, we introduce the concept of local steerability to allow functions to be steered under compact subsets of the family of transformations. We also show that a compactly-supported function can be steered locally with a set of equivariant basis functions by approximating it with these basis functions over an appropriate compact domain.

Definition 5 (Local Steerability) *: A function $f : \mathbf{R}^m \mapsto \mathbf{C}$ is* **locally steerable** *under a k-parameter Lie transformation group G if any transformation $T(g)$ of f by any element $g \in G' \subseteq G$ can be written as a linear combination of a fixed, finite set of* basis functions $f_i : \mathbf{R}^m \mapsto \mathbf{C}$*:*

$$T(g)\, f = \sum_{i=1}^{n} \alpha_i(g)\, f_i \tag{8}$$

We will also assume that the region over which $g \in G'$ is compact in some parameterization. Also, this subset G' need not be a subgroup of G. If G' were a subgroup of G, then the function f would simply be globally steerable under the new subgroup. As with the definition of global steerability, we will, in practice, also consider the case where the local steerability property holds only in approximation.

If a function f is locally steerable with a set of basis functions f_i, then arbitrary linear combinations of f_i (or even the basis functions themselves) are *not* necessarily locally steerable. Unlike the situation with global steerability, the function f is only steerable within a local range of parameter space; thus, each basis function f_i is only locally steerable within a different, possibly smaller, range of parameter space. Hence, the property of local steerability cannot be associated with function spaces but has to be discussed with respect to the particular function.

A compactly supported function is a function that is non-zero only over some compact region of its domain, and zero everywhere else. A non-compact transformation group refers to a group whose parameter space is non-compact. For example, the group of translations is non-compact since its parameter space is **R** while the group of rotations whose parameter space is S^1 is compact. For compactly-supported functions, there are no finite-dimensional function spaces that can be used to globally steer these functions under a non-compact transformation group. The simple example of steering a (single-period) raised cosine[1] under translation is illustrative of this point: in order to steer a raised cosine under all possible translations, an infinite number of raised cosines are needed.

Fortunately, if only local steerability is desired, then a finite number of functions might be sufficient to steer a compactly-supported function. The function to be steered is first approximated using an appropriate equivariant function space. This approximation is then steered by steering the basis functions spanning the space. Since only local steerability is desired, the domain over which the function is approximated need only be a subset of its actual domain; the size of this subset depends on the range of parameter space over which local steerability is expected.

Intuitively, we need to approximate the function over a large enough subset of its domain so that all transformed replicas of it will also be adequately approximated. For example, consider the problem of steering a one-dimensional raised cosine under translation. The raised cosine is compactly-supported over the interval $[-1, 1]$. The range of translations over which it is to be steered is $[-1, 1]$. Thus, the union of the support of all possible translated raised cosines is $[-2, 2]$. We refer to this interval as the *integration region* as this would be the (fixed) interval of integration for a corresponding steerable filter. Clearly, the original raised cosine needs to be well approximated over this interval $[-2, 2]$. Unfortunately, approximating it over this interval is not enough. When the raised cosine is translated to the left by -1, for example, the interval $[2, 3]$ (the right tail) of the original raised cosine's domain enters the integration interval. If the original raised cosine is poorly approximated in this region, then the interval $[1, 2]$ of this translated raised cosine will be poorly approximated as well. The same holds when the raised cosine is translated to the right by 1. Hence, the original raised cosine needs to be well approximated over the interval $[-3, 3]$. We refer to this interval as the *approximation region*. The integration region

[1]The definition of a raised cosine adopted in this paper is $(\cos(\pi x) + 1)/2$.

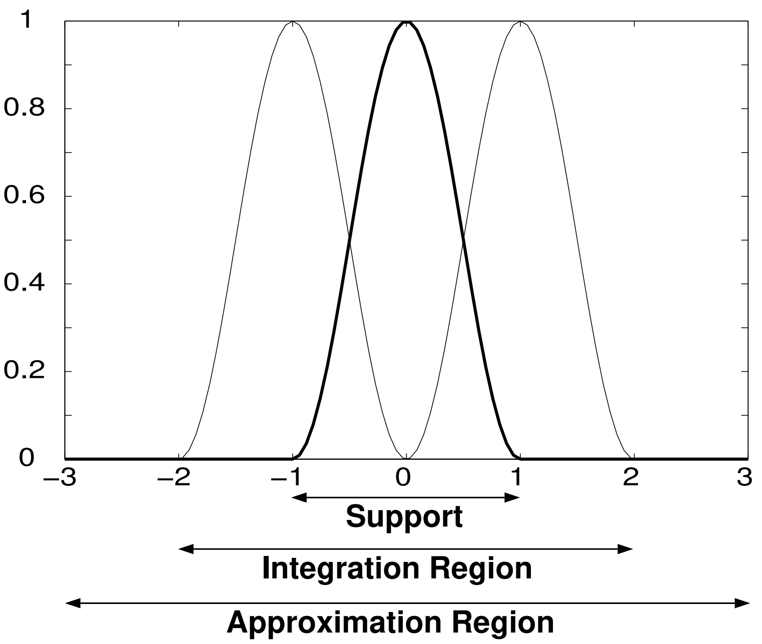


Figure 17: The support of the raised cosine is within the interval $[-1,1]$. If the function is to be steered in translation over the range $[-1,1]$, then the *integration region* corresponding to a steerable filter would be $[-2,2]$. The interval over which the raised cosine needs to be approximated, i.e. the *approximation region* is $[-3,3]$.

is a subset of the approximation region; the compact support of the original function is, in turn, a subset of the integration region. Figure 17 illustrates the approximation and integration regions for a one-dimensional raised cosine steered under translation.

The integration and approximation regions can be defined mathematically. Let R_f be the compact support of the steered function outside of which is zero identically. The integration region is therefore:

$$R_{\text{int}} = \bigcup_{g \in G'} T(g)\ R_f$$

where G' is the subset of G over which the function is to be locally steered. The application of the group operator g to the region R_f produces the corresponding region $T(g)\ R_f$ of the transformed function. The approximation region is defined in terms of the inverse of the group operator:

$$R_{\text{approx}} = \bigcup_{g \in G'} T(g^{-1})\ R_{\text{int}}$$

where $T(g^{-1})\,T(g) = I$, the identity map, for all $g \in G$.

5.2 Cascade Basis Reduction

The number of equivariant basis functions required to approximate the steerable function over the approximation region is not least-squares optimal. While these basis functions span the space of all transformations of the original function, not all possible linear combinations of these basis functions give rise to transformed replicas of the original function. In fact, very few will; that is, only those belonging to a k-dimensional manifold within that function space, where k is the number of parameters required to describe the transformation. In this section, we describe a method which finds an ordered set of functions such that the first n elements of this set span the optimal least-squares function space that best contains this manifold and thus can be used to steer f.

5.2.1 Singular Value Decomposition

Perona [Per95] showed that this problem could be solved numerically by computing the singular value decomposition (SVD) of a particular matrix $\boldsymbol{F}$ whose column vectors are transformed replicas of a discretely sampled version of the function f. Thus, each column in $\boldsymbol{F}$ corresponds to a specific sample of the parameter space over which the function is to be steered and each row in $\boldsymbol{F}$ corresponds to a specific sample of the function's domain. The SVD decomposes the matrix $\boldsymbol{F}$ into a product of three matrices:

$$\boldsymbol{F} = \begin{bmatrix} \vdots & & \vdots \\ T(g_1)f & \cdots & T(g_{s_p})f \\ \vdots & & \vdots \end{bmatrix} = \boldsymbol{U_F}\ \boldsymbol{S_F}\ \boldsymbol{V_F}^T = \boldsymbol{U_F}\ \boldsymbol{W_F}$$

where s_p indexes over samples of the parameter space, $\boldsymbol{U_F}^T\boldsymbol{U_F} = \boldsymbol{I}$, $\boldsymbol{V_F}^T\boldsymbol{V} = \boldsymbol{I}$, and $\boldsymbol{S_F}$ is a diagonal matrix of non-negative singular values, in decreasing order of magnitude. It can be shown that the first n columns of $\boldsymbol{U_F}$ represents the optimal least-squares set of basis functions (of size n) needed to steer f. The first n rows of the matrix $\boldsymbol{W_F}$ tabulate the weights of the linear combination needed to steer f.

The SVD of matrix $\boldsymbol{F}$ could also be computed by first computing the eigenvalues and eigenvectors of $\boldsymbol{F}^2 \doteq \boldsymbol{F}^T\boldsymbol{F}$. Denoting the eigenvalues and eigenvectors of $\boldsymbol{F}^2$ by $\boldsymbol{S_{F^2}}$ and $\boldsymbol{V_{F^2}}$ respectively, the SVD of $\boldsymbol{F}$ is: $\boldsymbol{S_F} = \boldsymbol{S_{F^2}}^{\frac{1}{2}}$, $\boldsymbol{V_F} = \boldsymbol{V_{F^2}}$, and $\boldsymbol{U_F} = \boldsymbol{F V_F S_F^{\#}}$ where $\boldsymbol{S_F^{\#}}$ is the pseudo-inverse of $\boldsymbol{S_F}$. Assuming that s_d and s_p samples of the domain and parameter space respectively are taken, if $s_p < s_d$, then it is computationally more efficient to compute the SVD of $\boldsymbol{F}$ in this manner as the size of $\boldsymbol{F}^T\boldsymbol{F}$ is smaller than the size of $\boldsymbol{F}$. Conversely, if $s_d < s_p$, then a similar method using $\boldsymbol{F}\boldsymbol{F}^T$ could be derived. Thus, the computational complexity of computing the SVD of $\boldsymbol{F}$ is upper-bounded by the smaller of the row and column dimensions of $\boldsymbol{F}$. For one or two-parameter groups, s_d often exceeds s_p and s_p is also manageably small. As a result, the SVD of $\boldsymbol{F}$ could be computed from the eigenvalues and eigenvectors of $\boldsymbol{F}^T\boldsymbol{F}$. Unfortunately, s_p increases exponentially with the number of parameters. For example, with a four parameter group and a discretization of only ten samples per dimension, the number of columns would be 10^4. Computing the eigenvalues and eigenvectors of a square matrix this size is computationally infeasible.

5.2.2 Basis Reduction

Alternatively, the matrix $\boldsymbol{F}$ could be written as a product of a $s_d \times m$ matrix $\boldsymbol{B}$ and an $m \times s_p$ matrix $\boldsymbol{H}$ such that columns of $\boldsymbol{B}$ are a set of m appropriately chosen, discretely sampled, basis functions (not necessarily orthogonal) and the columns of $\boldsymbol{H}$ contain the weights needed to reconstruct each column $T(g_i)f$ in $\boldsymbol{F}$. Typically, if appropriate basis functions are chosen, then $m < s_d$ and $m < s_p$. Thus, although the dimensionality of matrix $\boldsymbol{F}$ ($s_d \times s_p$) is quite large, its rank is only m which is much smaller than s_d and s_p. When the matrix $\boldsymbol{F}$ can be decomposed into the product of $\boldsymbol{B}$ and $\boldsymbol{H}$, the SVD of $\boldsymbol{F}$ can be computed economically by a sequence of two singular value decompositions, each of which involves computing the eigenvalues and eigenvectors of a square matrix whose size is equal to m. From the decomposition of

$\boldsymbol{F}$, we have

$$\begin{aligned}
\boldsymbol{F} &= \boldsymbol{B}\boldsymbol{H} \\
&\stackrel{(a)}{=} \left(\boldsymbol{U_B}\boldsymbol{S_B}\boldsymbol{V_B}^T\right)\boldsymbol{H} \\
&= \boldsymbol{U_B}\boldsymbol{H'} \\
&\stackrel{(b)}{=} \boldsymbol{U_B}\left(\boldsymbol{U_{H'}}\boldsymbol{S_{H'}}\boldsymbol{V_{H'}}^T\right) \\
&= \left(\boldsymbol{U_B}\boldsymbol{U_{H'}}\right)\boldsymbol{S_{H'}}\,\boldsymbol{V_{H'}}^T \\
&= \boldsymbol{U_F}\,\boldsymbol{S_F}\,\boldsymbol{V_F}^T.
\end{aligned}$$

Thus, the SVD of $\boldsymbol{F}$ is such that $\boldsymbol{U_F} = \boldsymbol{U_B}\boldsymbol{U_{H'}}$, $\boldsymbol{S_F} = \boldsymbol{S_{H'}}$ and $\boldsymbol{V_F} = \boldsymbol{V_{H'}}$. Two singular value decompositions need to be computed: one at (a) involving $\boldsymbol{B}$ and a second at (b) involving $\boldsymbol{H'}$. These decompositions could be obtained by computing the eigenvalues and eigenvectors of $\boldsymbol{B}^T\boldsymbol{B}$ and $\boldsymbol{H}\boldsymbol{H}^T$ respectively. Each of these matrix products are square matrices of size m. If the basis functions are orthonormal, then $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}$. Thus, $\boldsymbol{U_F} = \boldsymbol{B}\boldsymbol{U_H}$, $\boldsymbol{S_F} = \boldsymbol{S_H}$ and $\boldsymbol{V_F} = \boldsymbol{V_H}$. That is, only the SVD of $\boldsymbol{H}$ needs to be computed. Alternatively, if the steering functions are orthonormal, then $\boldsymbol{H}\boldsymbol{H}^T = \boldsymbol{I}$ and only the SVD of $\boldsymbol{B}$ needs to be computed.

5.2.3 Basis Reduction using Equivariant Function Spaces

In the previous section, we saw that the optimal least-squares set of n basis functions to steer a function f under any k-parameter transformation group could be efficiently computed if an appropriate set of basis functions $\boldsymbol{B}$ were available. These basis functions have to be chosen so that they span the column space of $\boldsymbol{F}$; i.e., these basis functions must be sufficient to locally steer the function f within the local parameter space of the k-parameter group.

In Section 5.1, we saw how equivariant functions could be used to steer a function f under a limited range of transformations. Essentially, the function f is approximated with linear combinations of the globally steerable equivariant functions $\boldsymbol{B}_{\mathrm{glob}}$ (within some appropriate domain of approximation). Steering the function f then amounts to steering the equivariant functions:

$$T(g)\,f \approx \boldsymbol{B}_{\mathrm{glob}}\boldsymbol{A}(g)\,\boldsymbol{c} \tag{9}$$

where $\boldsymbol{c}$ is a vector of weights that approximate f with $\boldsymbol{B}_{\text{glob}}$, i.e. $f \approx \boldsymbol{B}_{\text{glob}}\boldsymbol{c}$. The matrix $\boldsymbol{A}(g)$ is the matrix of steering functions used to steer each equivariant function. Thus, these equivariant functions are suitable candidates for the basis functions of $\boldsymbol{B}$ such that $\boldsymbol{B} = \boldsymbol{B}_{\text{glob}}$ and $\boldsymbol{H} = [(\boldsymbol{A}(g_1)\,\boldsymbol{c})\cdots(\boldsymbol{A}(g_s)\,\boldsymbol{c})]$. The columns of $\boldsymbol{H}$ correspond to a discrete sampling of a local range of the parameter space. Likewise, the rows of $\boldsymbol{B}$ correspond to a discrete sampling of the domain of the globally steerable basis functions. The SVD of $\boldsymbol{B}$ and $\boldsymbol{H}$ (and thus of $\boldsymbol{F}$) are then computed from the eigenvalues and eigenvectors of $\boldsymbol{B}^T\boldsymbol{B}$ and $\boldsymbol{H}\boldsymbol{H}^T$ respectively.

5.2.4 Analytic Form of Basis and Steering Functions

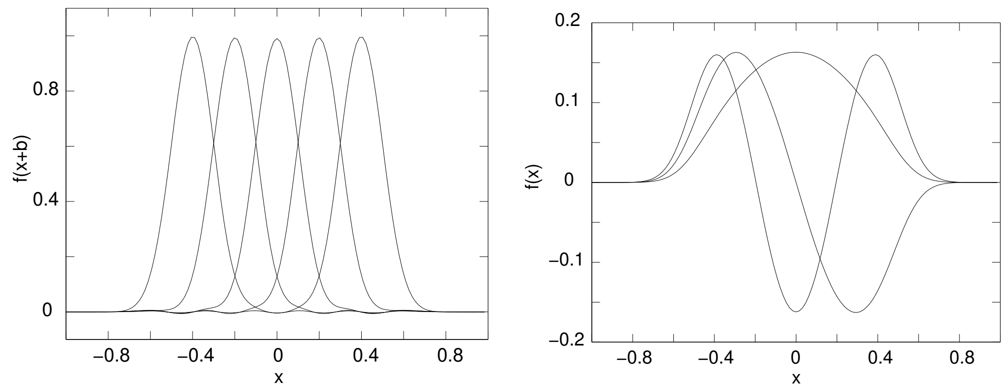
Since the globally steerable basis functions and their corresponding steering functions are in analytic form, the new basis and steering functions computed from the SVD of $\boldsymbol{F}$ can also be described in analytic form. To obtain an analytic description of the new basis functions, we simply write them in terms of the globally steerable functions in the columns of $\boldsymbol{B}$. Observe that $\boldsymbol{B} = \boldsymbol{U_B}\boldsymbol{S_B}\boldsymbol{V_B}^T$ and $\boldsymbol{U_F} = \boldsymbol{U_B}\boldsymbol{U_{H'}}$. Thus, $\boldsymbol{U_F} = \boldsymbol{B}(\boldsymbol{V_B}\boldsymbol{S_B^{\#}}\boldsymbol{U_{H'}})$. However, each column of $\boldsymbol{B}$ is simply a sampled version of a basis function. Therefore, the vector of the new basis functions (described analytically) is:

$$\boldsymbol{u_F}(x,y) = (\boldsymbol{V_B}\boldsymbol{S_B^{\#}}\boldsymbol{U_{H'}})^T\;\boldsymbol{b}(x,y) \tag{10}$$

where $\boldsymbol{b}(x,y)$ is the vector of original basis functions (described analytically). Likewise, to obtain an analytic description of the new steering functions $\boldsymbol{W_F} = \boldsymbol{S_F}\boldsymbol{V_F}^T$, we simply write them in terms of the original steering functions in the columns of $\boldsymbol{H}$. Since $\boldsymbol{H'} = \boldsymbol{S_B}\boldsymbol{V_B}^T\boldsymbol{H}$, $\boldsymbol{H'} = \boldsymbol{U_{H'}}\boldsymbol{S_{H'}}\boldsymbol{V_{H'}}^T$, and $\boldsymbol{S_F}\boldsymbol{V_F}^T = \boldsymbol{S_{H'}}\boldsymbol{V_{H'}}^T$, we have $\boldsymbol{W_F} = \boldsymbol{S_F}\boldsymbol{V_F}^T = (\boldsymbol{U_{H'}}^T\boldsymbol{S_B}\boldsymbol{V_B}^T)\boldsymbol{H}$. Again, each column of the matrix $\boldsymbol{H}$ is simply a sampled version of the steering function. Therefore, the vector of new steering functions (described analytically) is:

$$\boldsymbol{w_F}(g) = (\boldsymbol{U_{H'}}^T\boldsymbol{S_B}\boldsymbol{V_B}^T)\;\boldsymbol{h}(g) \tag{11}$$

where $\boldsymbol{h}(g)$ is the vector of original steering functions (described analytically); i.e., $\boldsymbol{h}(g) = \boldsymbol{A}(g)\;\boldsymbol{c}$. Denoting $\boldsymbol{\Lambda} = \boldsymbol{U_{H'}}\boldsymbol{S_B}\boldsymbol{V_B}^T$, we can write the overall analytic steering



(a) (b)

Figure 18: (a) Reconstructions of translated replicas of the original function using 10 basis functions. (b) Basis functions corresponding to the three largest singular values computed using the cascade basis reduction method.

equation as

$$T(g)\ f(x,y) = (\boldsymbol{b}(x,y)^T \boldsymbol{\Lambda}^{\#})\ (\boldsymbol{\Lambda}\ \boldsymbol{A}(g)\ \mathbf{c}). \tag{12}$$

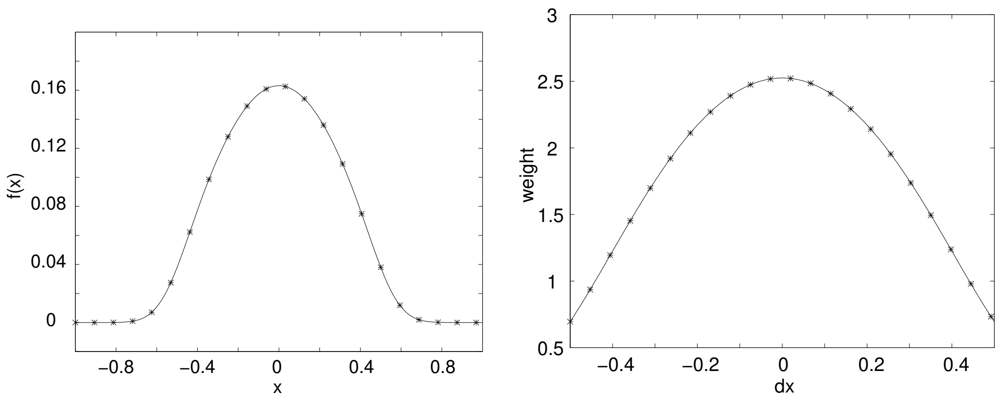
This equation is essentially the same as Equation 9. To compute the optimal least squares set of n basis functions, only the first n columns of $\boldsymbol{U_{H'}}$ in $\boldsymbol{\Lambda}$ (and correspondingly, in $\boldsymbol{\Lambda}^{\#}$) are retained; the rest are set to zero.

5.3 Results

5.3.1 Comparison with Conventional SVD

In this experiment, a one-dimensional Gaussian function $(\exp(-((x+\triangle x)/\sigma)^2/2), \sigma = 0.1)$ was steered in translation over the parameter range $-0.5 \leq \triangle x \leq 0.5$. The domain of the function was discretized using 128 evenly-spaced samples from $[-1, 1]$. The parameter range was also discretized using 128 evenly-spaced samples. Thus, using the conventional SVD method, the singular value decomposition of a 128×128 matrix was computed.

For the cascade basis reduction method, the sinusoids (and co-sinusoids) with integer frequencies over the domain $[-1, 1]$ were used as the equivariant functions (see Table 2 of Chapter 2). A total of 21 were required to approximate the Gaussian over



(a) (b)

Figure 19: Graph (a) plots the analytic form of the basis function with the largest singular value. The asterisks represent the corresponding discretely sampled basis function computed using the conventional SVD method. Graph (b) plots the analytic form of the steering function for the basis function with the largest singular value. The asterisks represent the corresponding discretely sampled steering function computed using the conventional SVD method.

this interval (one DC component, and 10 pairs of sinusoids and co-sinusoids of increasing integral frequencies). The SVD of the matrix $\boldsymbol{F}$ was then computed via two consecutive SVD's, each of which involve only a 21×21 matrix. In both methods, we selected the optimal 10 basis functions and used them to steer the Gaussian. The basis functions derived using both methods are virtually identical. Figure 18 (a) shows translated replicas of the steered function constructed using the 10 basis functions. Figure 18 (b) shows the first three basis functions obtained using the cascade basis reduction method. Figure 19 (a) plots the analytically derived first basis function. The asterisks denote the numerically computed basis function obtained using the conventional SVD technique. The analytically derived basis function obtained using the cascade basis reduction method interpolate the numerically computed sample points very well. Figure 19 (b) plots the analytically derived steering function corresponding to the first basis function.

5.3.2 Steering a Gabor Function under General Linear Transformation

In this experiment, a two-dimensional odd-phase Gabor function ($\sin(x/\sigma_x)\exp(-((x/\sigma_x)^2+(y/\sigma_y)^2)/2)$, $\sigma_x = \sigma_y = 0.2$) was steered over a range of linear transformations (combinations of rotations, independent scalings along each axis, and skew-transformations). The domain was sampled uniformly over $[-1,1]\times[-1,1]$ with 64×64 samples. The linear transformation was parameterized in a unique way: $\boldsymbol{A} = \boldsymbol{R}(\theta_2)\ \boldsymbol{S_x}(s_x)\ \boldsymbol{S_y}(s_y)\ \boldsymbol{R}(\theta_1)$ where $\boldsymbol{R}(\theta_1), \boldsymbol{R}(\theta_2)$ are rotation matrices and $\boldsymbol{S_x}, \boldsymbol{S_y}$ represent pure scaling in the x- and y- directions respectively. Thus, we are disallowing reflections. The validity of this parameterization can be understood in terms of the singular value decomposition of $\boldsymbol{A}$. The range of parameter space over which the Gabor function was steered was: $\theta_1, \theta_2 \in [0, 2\pi)$ and $s_x, s_y \in [1, 5/3]$. The Legendre polynomials over the interval $[-1,1]\times[-1,1]$ were used as the equivariant basis functions to approximate the Gabor function (see Chapter 3 for a catalog of equivariant function spaces for different multi-parameter transformations). A total of 231 Legendre polynomials were used. This set included all products of one-dimensional Legendre polynomials whose total degree was less than or equal to 20; i.e., $\bigcup_{0\leq d\leq 20} P^d_{x,y}$ where $P^d_{x,y} \doteq \{P_x^{d_x} P_y^{d_y} | d_x + d_y = d, d_x \geq 0, d_y \geq 0\}$.

The results of using the cascade basis reduction method to compute the basis functions are shown in Figure 21. The singular values of the SVD decrease rather rapidly such that a total of 11 basis functions were found to be sufficient to steer the odd-phase Gabor function (see Figures 20 (a) and (b)). These 11 basis functions accounts for 99.9% of the total squared norm in the approximation. Figure 21 (a) shows the first ten of these eleven basis functions. Figure 21 (b) shows replicas of the Gabor function steered to various linear transformations. The mean squared error involved in steering the Gabor as compared to the actually transforming the Gabor itself over the range of transformed parameters was 0.17% (expressed with respect to the squared norm of the transformed Gabor in each case). The maximum absolute error with respect to the maximum absolute value of the transformed Gabor in each case was 4.82%. A total of 22,500 samples of the parameter space were used in this

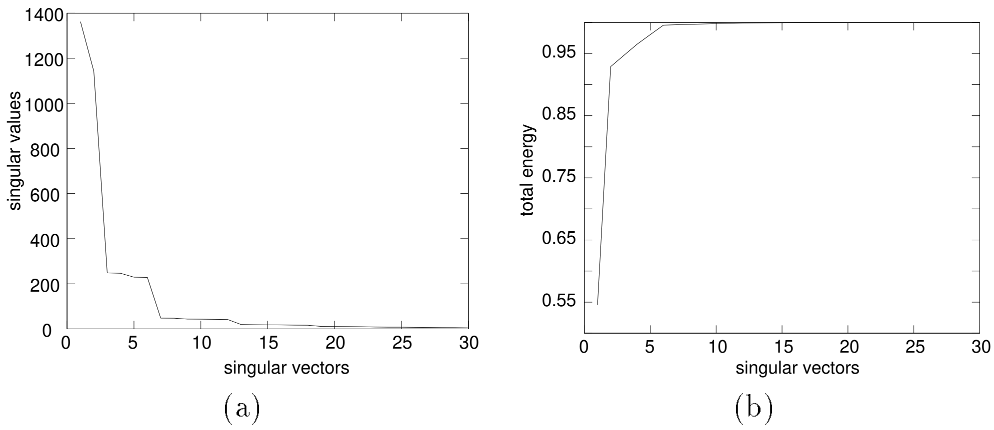


Figure 20: Graph (a) plots the magnitude of the singular values for each singular vector. Each singular vector corresponds to a single basis function. A total of 231 singular vectors were present but only the largest 30 of them are plotted. Graph (b) plots the cummulative sum of the squared magnitudes of the singular values. The squared magnitudes of the singular values have been normalized so that their sum equals one.

experiment. Since the domain was sampled with $64 \times 64 = 4096$ samples, applying the conventional method would have required computing the SVD of a 4096×4096 matrix! The cascade basis reduction method, however, required the calculation of the SVD of two 231×231 matrices.

Figure 22 shows the results of using the cascade basis reduction method to design a set of basis functions for an even-phased Gabor function, $(\cos(x/\sigma_x)\exp(-((x/\sigma_x)^2 + (y/\sigma_y)^2)/2),\ \sigma_x = \sigma_y = 0.2)$. Figure 22 (a) shows the eight basis functions that account for 99.9% of the total squared error. Figure 22 (b) shows replicas of the Gabor function steered to various linear transformations.

5.4 Summary

We conclude this chapter with a summary of its highlights.

1. A definition of local steerability was proposed. With global steerability, the basis functions are themselves steerable. However, the basis functions of a

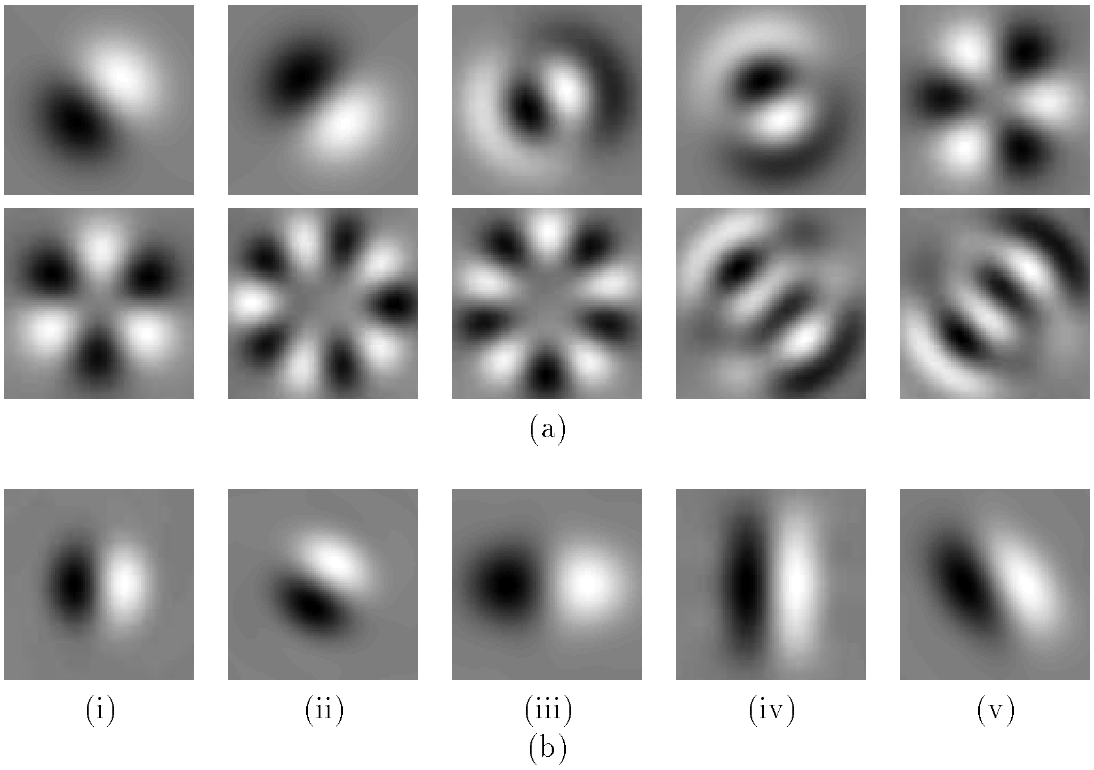
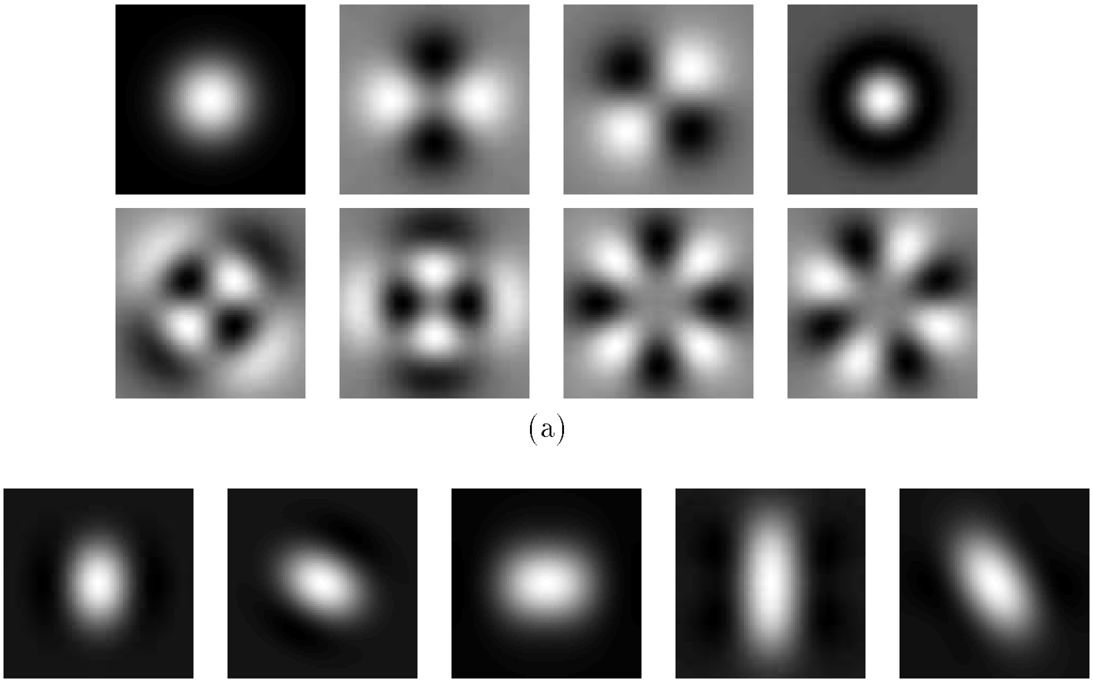


Figure 21: (a) Ten out of the eleven basis functions (99.9% total squared norm) computed to steer the odd-phase Gabor under any local linear transformation. The basis functions are arranged in descending order of the magnitudes of their singular values from left to right and from top to bottom. (b) Image (i) shows a reconstruction of the original function. Image (ii) shows a reconstruction of the function rotated by 60 degrees. Image (iii) shows a reconstruction of the function scaled along the x-axis. Image (iv) shows a reconstruction of the function scaled along the y-axis. Image (v) shows a reconstruction of the function skewed along the x-axis and uniformly scaled. All of these functions were reconstructed using the 11 basis functions.



(a)

(i) (ii) (iii) (iv) (v)

(b)

Figure 22: (a) Eight basis functions (99.9% total squared norm) computed to steer the even-phase Gabor under any local linear transformation. The basis functions are arranged in descending order of the magnitudes of their singular values from left to right and from top to bottom. (b) Images (i) through (v) show reconstructions of the original function under various linear transformations. See the caption of Figure 21 for further description.

locally steerable function are not locally steerable within the same range of transform parameters.

2. The integration and approximation regions of a locally steered function were defined. The integration region is the region of support within which locally steering the function with a set of basis functions is valid. The approximation region is the enlarged region of support over which the function to be steered needs to be approximated.

3. A cascade basis reduction method was proposed where the locally steered function is first approximated using an appropriate equivariant function space and then a series of two singular value decompositions are carried out to compute the optimal least-squares set of basis functions to steer the given function. The two singular value decompositions are performed on $n \times n$ matrices where n is the number of basis functions spanning the equivariant function space.

4. The basis and steering functions of the locally steered function were derived in analytic form.

5. Two sets of basis functions to steer an odd-phased and an even-phased Gabor function over a limited range of the four-parameter group of linear transformations were designed.

Chapter 6

Applications

In this chapter, we describe five applications of steerable function. In the first application, steerable filters are used to model hypothetical orientation-sensitive mechanisms in a model of human spatial pattern detection. Because the outputs of a steerable filter can be synthesized arbitrarily, the model is able to compute the collective response of an *infinite* number of these hypothetical orientation-sensitive units. In the second application, two techniques for designing filters for gradient-based motion estimation are described. These techniques are rooted in the observation that such motion estimation filters have to be approximately steerable. The third application shows the use of steerable functions to represent the emitted radiance distribution of a computer graphics model of a light source. This means that the radiance distribution of the light source is steerable. Due to the linearity of the rendering operation, re-rendering of scenes under steered illumination changes can thus be efficiently computed as linear combinations of a collection of basis images. The fourth application details the construction of invariants within equivariant function spaces and describes the relevance of equivariant function spaces to invariant feature detection. Finally, the fifth application points out that the use of steerable functions is not limited to the continuous case. It describes the use of steerable functions over sets of points and lines.

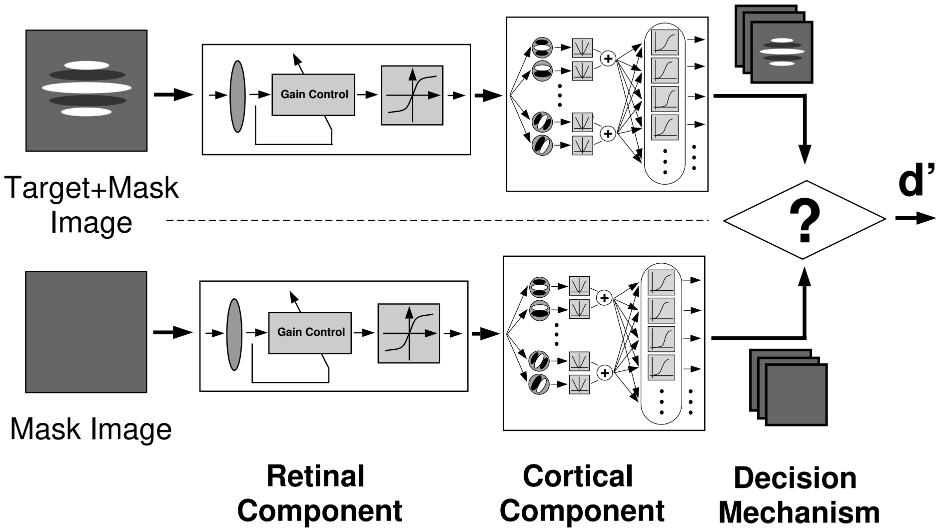


Figure 23: Overview of the human spatial pattern detection model. The model consists of three major components: a retinal component, a cortical component and a decision mechanism. The model simulates a discrimination task; the output of the model is a number representing the discriminability between the two input images.

6.1 Continuum Approximation in Modeling Vision

In this section, we describe the use of steerable functions in a mechanistic model of human spatial pattern detection. The part of the model that uses steerable functions will be presented in detail; the rest of the model will only be briefly mentioned. Details of the model can be found in [TH94b, TH94a, TH95].

The model takes as input two spatial patterns and outputs a single number representing the predicted visual discriminability between the two input images. In a psychophysical experiment, these two patterns would be the patterns presented to a human subject whose task then is to determine if they are visibly different; the goal of the model is to predict the human observer's performance. In a more applied setting like image compression, the two images would correspond to the original image and a lossy-compressed replica; the model would then be used to determine if the compression errors are visible, and possibly tune the compression algorithm for the particular image.

The model consists of three major components: a retinal component, a cortical

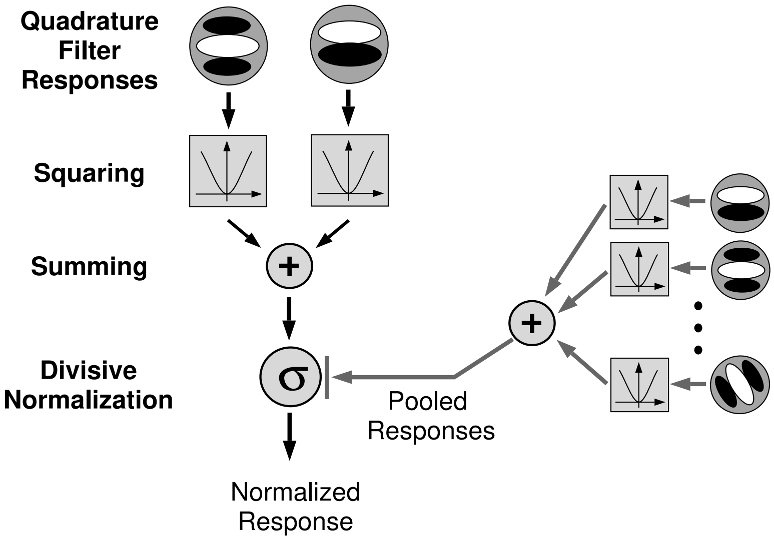


Figure 24: Schematic of contrast normalization in the cortical component of the model. Contrast normalization involves: (1) computing energy responses as the sum of the squared responses of a quadrature pair of linear responses, and (2) divisive normalization, that is, dividing each energy response by a sum of other energy responses.

component, and a decision mechanism (Figure 23). Each of the patterns is processed through the retinal and cortical components of the model. The retinal component controls the model's sensitivity to changes in the average light level. The cortical component controls the model's sensitivity to spatial patterns of different spatial frequencies and orientations. The final component of the model, the decision mechanism, then compares the outputs of the cortical component for the two patterns. In particular, the decision mechanism computes d', a measure of discriminability commonly used in signal detection theory. Under certain assumptions, d' is related to the probability of correctly detecting when the two patterns are different. For example, a $d' = 1$ would imply that the detection mechanism is correct 84% of the time.

Steerable filters are used in the cortical component of the model to model the receptor fields of hypothetical cortical mechanisms involved with early visual processing. Within any local region of the visual field, there is a large number of such mechanisms sensitive to different spatial frequency, phase, and orientation. The steerable filters used in the model are thus also localized in spatial frequency, phase, and orientation. These steerable filters are designed to be steerable over orientation; their phase and spatial frequency selectivity are achieved through the use of a pair of multi-resolution pyramids that are in quadrature-phase [SF95b]. The multiplicity of cortical mechanisms tuned to different orientations is modeled by steering the outputs of the steerable filters to different orientations.

The linear outputs of the steerable filters are normalized by a method known as *contrast normalization* [AG91, Hee92b, Hee92a, Hee93, FB94], which is depicted in Figure 24. A simplified version of contrast normalization involves squaring the output of the linear filter, and dividing the result by the sum of squared outputs of all filters tuned to different orientations together with a semi-saturation constant. The resulting contrast normalized output is a value bounded between zero and some constant. Mathematically,

$$\bar{O}(\theta_i) = k \, \frac{(O(\theta_i))^2}{\frac{1}{N}\sum_{j=1}^{N}(O(\theta_j))^2 + \sigma^2} \tag{13}$$

where j indexes over the orientation tunings of all mechanisms, and $O(\theta_i) = \langle f_{\theta_i}, I \rangle$ is the output of the steerable filter f tuned to orientation θ_i. Because f is steerable, it follows by linearity that its output is also steerable; that is, $O(\theta_i)$ can be written as a linear combination of the outputs of a fixed set of M basis filters: $O(\theta_i) = \sum_{j=1}^{M} \alpha_j(\theta_i) O(\theta_j)$.

The discriminability value computed by the model is a function of the discriminability of individual mechanisms. In particular, the overall discriminability between all orientation sensitive mechanisms is computed from the differences of their normalized outputs:

$$d' = \frac{1}{N}\sum_{i=1}^{N}(\bar{O}_1(\theta_i) - \bar{O}_2(\theta_i))^2 \tag{14}$$

where $\bar{O}_1(\theta_i), \bar{O}_2(\theta_i)$ represent the normalized outputs of a steerable filter applied to the two input patterns respectively; the filter is steered to a variety of orientations θ_i.

If the number of different orientation-selective mechanisms N is small, Equation 14 can be computed efficiently. Since each $\bar{O}(\theta_i)$ represents the output of a single cortical unit and it is likely that there are a large number of these cortical units, N is expected to be extremely large. As N grows larger and larger, the sum in the equation can be considered as computing a Riemann sum, which converges to the Reimann integral of the expression within the sum. That is,

$$d' \rightarrow \int (\bar{O}_1(\theta) - \bar{O}_2(\theta))^2 d\theta \tag{15}$$

as $N \rightarrow \infty$. Here, we assume that the distribution of orientation-selective mechanisms is uniform. In general, a non-uniform distribution could be used, in which case the integral becomes a weighted integral. This technique of approximating a (convergent) discrete sum of a large number of terms with its Riemann integral is known in the neural computation community as a *continuum approximation* [HKP91].

The continuum approximation of Equation 14 can be computed in closed form because the normalized outputs $\bar{O}_1(\theta), \bar{O}_2(\theta)$ are steerable. In particular,

$$\bar{O}(\theta) \approx k \, \frac{(O(\theta))^2}{\int (O(\phi))^2 d\phi + \sigma^2} \tag{16}$$

where the denominator, after applying the continuum approximation, is independent of θ and can be computed in terms of the outputs of the basis filters $O(\theta_j)$ directly as $\int (\sum_{j=1}^{M} \alpha_j(\phi) O(\theta_j))^2 d\theta + \sigma^2$. Thus, Equation 15 can be expressed in terms of $O(\theta)$:

$$\begin{aligned} d' &\approx \int (\kappa_1 (O_1(\theta))^2 - \kappa_2 (O_2(\theta))^2)^2 d\theta \\ &= \int (\kappa_1 (\textstyle\sum_{j=1}^{M} \alpha_j(\theta) O_1(\theta_j))^2 - \kappa_2 (\textstyle\sum_{j=1}^{M} \alpha_j(\theta) O_2(\theta_j))^2)^2 d\theta \end{aligned} \tag{17}$$

where $\kappa_i = k/(\int (O_i(\phi))^2 d\phi + \sigma^2)$ and $O_i(\theta_j)$ are the outputs of the basis filters. The steering functions $\alpha_j(\theta)$ are expressible in analytic form and their integrals can be derived in closed form. As a result, the discriminability d can be computed directly from the outputs of the basis filters $O_i(\theta_j)$. The actual computations in the model are more complicated than what has been presented, involving quadrature pairs of steerable filters and multiple contrast normalization units. By assuming an infinite number of orientation-selective mechanisms, the model is rotation invariant since the discriminability value is computed independent of θ. The model is also approximately translation invariant because of the use of quadrature pairs.

Nevertheless, by assuming an infinite number of orientation-selective mechanisms, the model is not only translation invariant, but also rotation invariant.

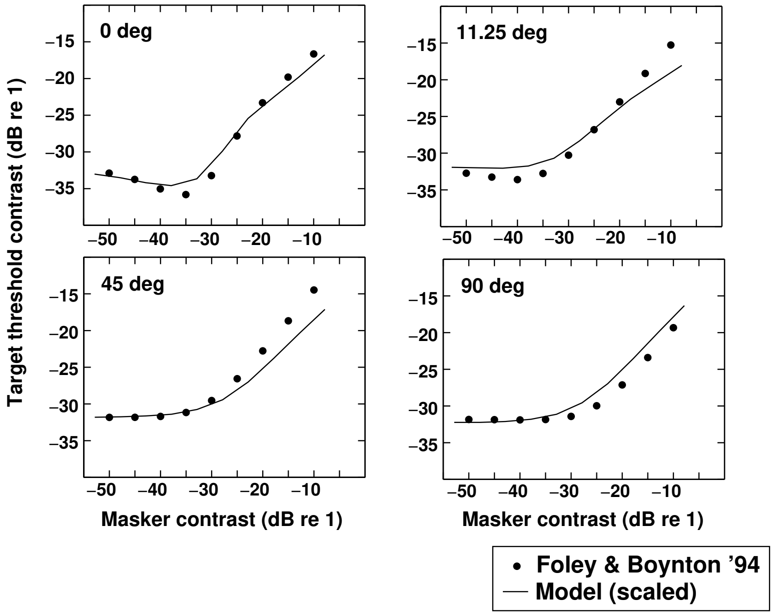


Figure 25: Summary of model's fits to contrast masking data with maskers of different orientations. Empirical data from Foley and Boynton [Fol94, FB94] is represented by filled circles while the model's predictions are described by the solid lines.

Figure 25 plots the model's predictions to the results of a set of contrast masking experiments. The filled circles represent experimental measurements obtained by Foley and Boynton [Fol94, FB94]; the solid lines represent the model's predictions. During the experiment, the human subject was presented two spatial patterns, one followed by the other. One of the patterns was a full-field sinusoidal grating; the other was a small (oriented) Gabor pattern superimposed on a similar full-field sinusoidal grating. The order of presentation was randomized and the contrast (or equivalently, the amplitude) of the Gabor patch was varied. The object of the experiment was to determine the detection threshold (minimum contrast) of the Gabor patch for it to be discernable. Each graph in the figure plots the detection thresholds of the Gabor patch for sinusoidal gratings of different contrasts. The different graphs in the figure correspond to different relative orientations between the Gabor pattern and the sinusoidal grating.

Figure 26 demonstrates the predictions of the model on a real image. The original image of Einstein was distorted in two ways, one of which produced more visible

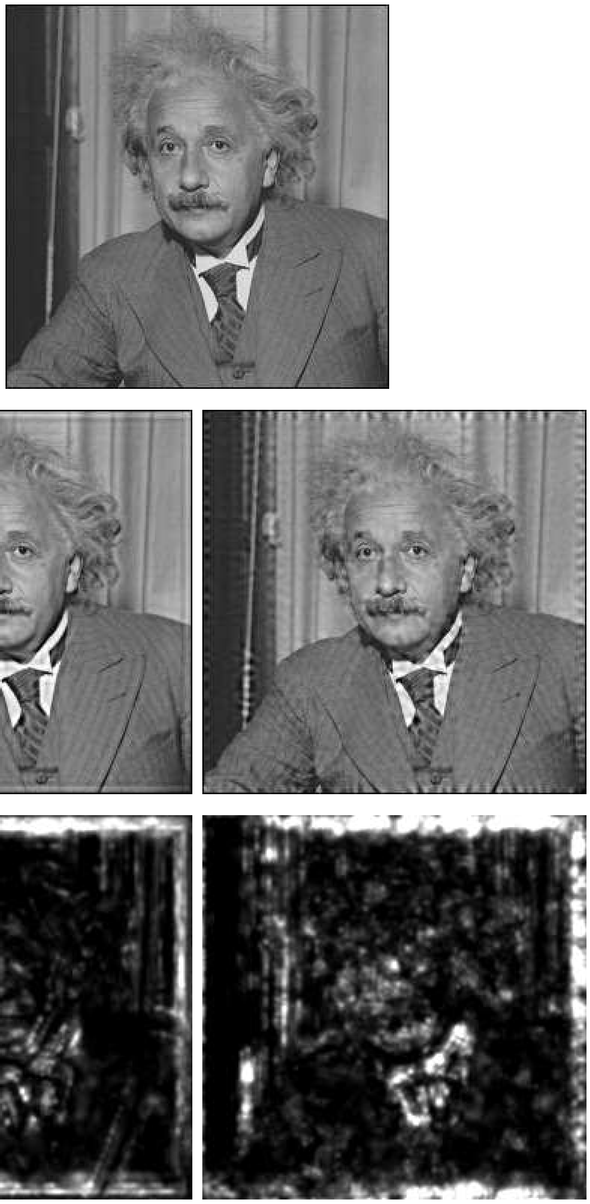


Figure 26: (Top) Original Einstein image. (Middle-left) Image was distorted so as to minimize perceptual distortion (RMSE = 9.01, peak-SNR = 29.04 dB). (Middle-right) Image was distorted so as to maximize perceptual distortion (RMSE = 8.50, peak-SNR = 29.54 dB). While the left image looks generally less distorted than the right image, it has a larger root mean squared error. (Bottom-left) Perceptual distortion measured from the minimally distorted image. Darker regions correspond to areas of lower perceptual distortion while brighter regions indicate areas of greater perceptual distortion. (Bottom-right) Perceptual distortion measured from the maximally distorted image.

errors than the other. The root mean squared error (RMSE) between the more visibly distorted image and the original image is *less* than the RMSE between the less visibly distorted image and the original. This demonstrates that RMSE is a poor measure of perceptual distortion. The model's predictions are shown below each of the distorted images in Figure 26. Brighter regions in the model's output correspond to areas that have more visible distortions; darker regions correspond to areas with less noticeable distortions. The model accurately predicts regions of the image where the errors are more noticeable, for example, the region around the tie or the vertical structure in the background at the bottom-left of the image. In the image where the artifacts are less visible, the orientation of the artifacts are similar to the orientation of the signal (the local image content); thus, the artifacts are masked more effectively by the signal. In the more visibly distorted image, on the other hand, the orientation of the artifacts is very different from the orientation of the signal; thus, masking is less effective and the artifacts are more visible. The visibility of these artifacts is well predicted by the model because of its contrast normalization component which accounts for contrast masking.

6.2 Optimal Filters for Motion Estimation

Gradient based motion estimation techniques compute the local optical flow from the outputs of a set of filters. The brightness constancy assumption between two images, $I_2(x) = I_1(x + \tau)$, provides the constraint equation between the filter outputs, which is given below for the one-dimensional case:

$$\langle h, I_2 \rangle = \langle h, I_1 \rangle + \tau \langle g, I_1 \rangle + O(\tau^2) \tag{18}$$

where h is typically a pre-filter that smoothes the images and g is a filter that measures the first derivative of the image. The parameter τ measures the amount of translation between the two images I_1 and I_2. The above equation was obtained by computing the Taylor expansion of I_1 with respect to τ keeping only terms up to first order. Simoncelli [Sim94] and others have advised that the filters h and g should be well-matched; that is, the filter g should be the first derivative of the filter h. For digital

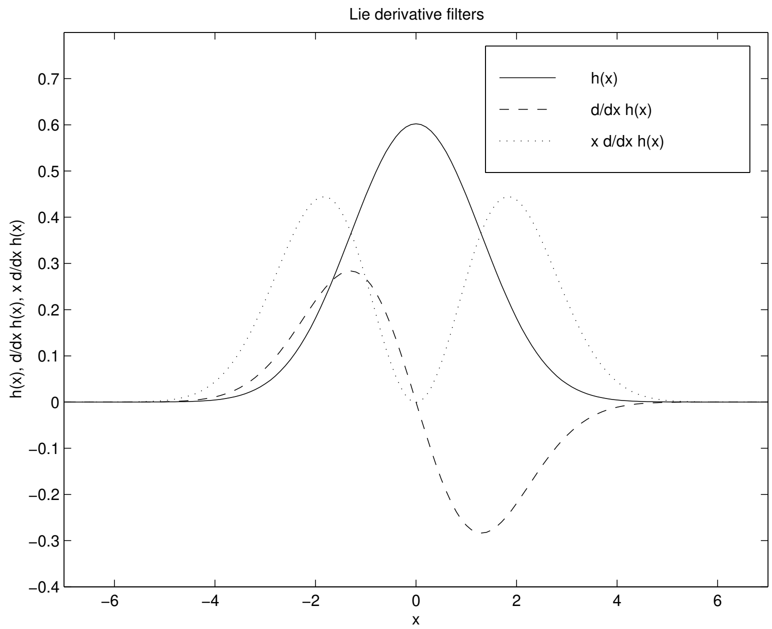


Figure 27: Continuous profiles $h(x), g_1(x), g_2(x)$ of the optimal set of 5-tap filters such that $g_1(x) = \frac{\partial}{\partial x} h(x)$ and $g_2(x) = x \frac{\partial}{\partial x} h(x)$. The continuous profiles are created by interpolating the digital filters using a narrow Gaussian interpolant. The optimal digital filters are $\boldsymbol{h}$ = (0.0167, 0.1847, 0.3738, 0.1847, 0.0167)^T, $\boldsymbol{g_1}$ = (0.0602, 0.2675, 0.0000, -0.2675, -0.0602)^T, and $\boldsymbol{g_2}$ = (0.2959, 0.3366, -0.4893, 0.3366, 0.2959)^T.

filters, this requirement is stated with respect to some choice of interpolant.

Since $I_2(x) = I_1(x+\tau)$, we can eliminate references to I_2 and rewrite Equation 18:

$$\langle h(x-\tau), I_1(x)\rangle = \langle h(x), I_1(x)\rangle - \tau \langle(-g(x)), I_1(x)\rangle + O(\tau^2) \tag{19}$$

where the relationship $\langle h(x), I_1(x+\tau)\rangle = \langle h(x-\tau), I_1(x)\rangle$ was used to apply the translation to the filter h instead of the image.[1] The double negation in the equation was used so that the negative translation of h by τ on the left hand side resulted in a negative sign in front of τ on the right hand side. With Equation 18 in this form, it is clear that the function h is an approximately steerable function with basis functions h and $-g$ and steering functions 1 (the constant function) and τ.

[1]In the infinitesimal case, where the temporal difference between the two frames tends to zero, the equation becomes the more familiar gradient constraint for motion. Specifically, $I_t - \tau I_x = 0$ where the temporal derivative of I, $I_t \approx \langle h(x-\tau), I(x)\rangle - \langle h(x), I(x)\rangle$ and the spatial derivative of I, $I_x \approx \langle g(x), I(x)\rangle$.

Thus, the suggestion that the function g be the first derivative of h can be understood in terms of steerable functions, namely that $g = L\ h$ where $L = \frac{\partial}{\partial x}$ is the infinitesimal generator for translation. This is a more general requirement since it is applicable to any transformation group. For example, if instead of translation, scaling is to be measured, then $L = x\frac{\partial}{\partial x}$ and g should be equal to $x\frac{\partial}{\partial x}\ h$. In the case of multi-parameter groups, the requirement is generalized to involve all the filters. For example, if both translation and scaling are to be measured, then three filters h, g_1, g_2 are needed with the requirement that $g_1 = L_1\ h$ and $g_2 = L_2\ h$ such that $L_1 = \frac{\partial}{\partial x}$ is the infinitesimal generator for translation and $L_2 = x\frac{\partial}{\partial x}$ is the infinitesimal generator for scaling. Figure 27 plots the continuous profiles of the optimal set of 5-tap Lie derivative filters satisfying this relationship (so called because the infinitesimal generators of transformation groups are also known as Lie derivatives). These filters minimize the squared error of the two constraints ($g_1 = L_1\ h$ and $g_2 = L_2\ h$) over all triples of 5-tap filters (for a given choice of interpolant); the minimization is solved by computing the eigenvector corresponding to the smallest eigenvalue of a particular positive-definite matrix in a scheme similar to the one proposed in [Sim94] for ordinary derivative filters.

Equation 19 could be generalized by relaxing the requirement that the steerable filter be one of the basis filters; that is,

$$\langle m(x+\tau), I_1(x)\rangle = \langle h(x), I_1(x)\rangle + \tau\ \langle g(x), I_1(x)\rangle + O(\tau^2) \tag{20}$$

where m, h, g are arbitrary filters satisfying the above equation. Allowing m and h be different filters implies that the pre-smoothing applied to the two images could likewise be different. We can also express the equality constraint as an error functional:

$$E(m, h, g) = \int_{\tau=-D}^{D} \langle m(x+\tau) - h(x) - \tau g(x), I_1(x)\rangle^2\, d\tau. \tag{21}$$

Thus, the filters m, h, g minimizing this error functional represent, in a certain sense, the optimal first-order motion estimation filters over the range of translations, $\tau = -D \ldots D$.[2]

[2]First-order refers to the first order Taylor expansion in the equation.

In practice, several modifications to this error functional is required. Since $m = h = g = 0$ satisfies this equation exactly, additional constraints on m, h, g are required to produce non-trivial results (e.g., $< m, m >= 1$). In the energy functional, the filters are specified as continuous functions; in practice, the optimal set of digital filters satisfying this equation is sought instead. As a result, some suitable interpolant is typically assumed. Given this interpolant, it can be shown that the optimal set of digital filters, minimizing this modified energy functional, can be computed as the eigenvector corresponding to the smallest eigenvalue of a particular positive-definite matrix.

Unlike in the minimization problem to compute Lie derivative filters, the image I_1 is included in the minimization. This allows one to design optimal filters over a collection of images or more practically, over images bearing certain properties, for example, a decaying power spectrum with a certain exponential decay constant. Alternatively, we can remove the dependence over the choice of images by computing the inner-product with respect to the first term: $||m(x + \tau) - h(x) - \tau g(x)||$.

Two important features distinguish this method from other methods of designing optimal filters. First, by allowing the filter m to differ from the filter h (that is, the pre-smoothing filters applied to the two images may be different), the estimated motion when there is no motion could be small but non-zero. However, this relaxation results in a reduction in error for the larger non-zero translations such that the filters are optimal in a least-squares sense. Second, and more importantly, the distribution of translation could be explicitly specified. Although Equation 21 integrates over translations from $\tau = -D$ to $\tau = D$, any arbitrary distribution of motion could be used. Thus, the motion estimation filters could be designed to be optimal with respect to a particular class of motion. Further details of this technique are elaborated in Elad *et al.* [ETHO97] where comparisons with other filters are described and evaluation of the optimal filters performance in estimating optical flow are compared with those of popular filters.

6.3 Steerable Light Sources

In computer graphics, images of a synthetic scene rendered from a common viewpoint are linear with respect to the radiance distributions of the light sources. In other words, given two images of a scene rendered under two different light sources, an image of the scene illuminated by a linear combination of the two *basis light sources* is simply a linear combination of the two *basis images*. Thus, a complex scene can be efficiently re-rendered under a change of illumination if the radiance distribution of the new illuminant can be expressed as a linear combination of the radiance distributions of the basis light sources. This simple approach is general as it depends only on the linearity of the rendering operation; it is independent of scene complexity (geometry and surface reflectances) and rendering complexity (shadows and complex inter-reflections). This approach was introduced by Nimeroff *et al.* [NSD94] to re-render outdoor scenes under illumination changes; it has subsequently been used to efficiently re-render interior scenes [DKNY95] and also applied to theatrical lighting design [DAG95].

The linearity of the rendering operation with respect to the scene's illuminant suggests the use of steerable functions to describe the radiance distribution of the illuminant. In this section, we describe several extensions to the results of [NSD94]. We elucidate a design methodology for: (1) directional spot lights whose directions of foci and angular radiance distributions can be continuously varied, (2) area light sources whose positions and spatial radiance distributions can be continuously varied, and (3) light sources that are a combination of the first two types (i.e., directional area lights whose directions of foci and positions can all be changed). In addition, we describe a general method for reducing the number of basis images that is applicable to any of the above cases. Details of the method can be found in [TSH97].

Any light source can be fully described by the spatial and angular distribution of its emitted radiance. We write this function as $L(\boldsymbol{x}, \boldsymbol{\omega})$ where $\boldsymbol{x}$ specifies the position and $\boldsymbol{\omega}$ specifies the angular direction. For example, an isotropic point light source centered at location $\boldsymbol{x_0}$ would be described by the distribution $L_{\text{point}}(\boldsymbol{x}, \boldsymbol{\omega}) = \boldsymbol{\delta}(\boldsymbol{x} - \boldsymbol{x_0})$ where $\boldsymbol{\delta}(\boldsymbol{x}') = 1$ when $\boldsymbol{x}' = \mathbf{0}$ and zero otherwise. A parameterized light source is a family of light source distributions denoted by $\{L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p})\}$. A parameterized light source is said to be steerable in its parameters $\boldsymbol{p}$ if its radiance distribution can be written as a linear combination of a finite set of basis lights, where the weights involved in the linear combination are functions solely of the parameter vector $\boldsymbol{p}$. Mathematically, $\{L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p})\}$ is steerable if

$$L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p}) = \sum_{i=i}^{N} \alpha_i(\boldsymbol{p})\ L_i(\boldsymbol{x}, \boldsymbol{\omega}) \tag{22}$$

where α_i are the steering functions and L_i are the radiance distributions of the basis lights. In particular, we will restrict ourselves to the case where the family of light source distributions are generated by a Lie transformation group, in which case $L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p})$ is a steerable function.

Let $\mathcal{R}$ denote the linear rendering operator of a given model scene from a fixed viewpoint. This operator takes as input the radiance distribution of the light source L and produces an image $I = \mathcal{R}(L(\boldsymbol{x}, \boldsymbol{\omega}))$. Combining this notation with Equation 22 gives an expression describing the re-rendering process:

$$\begin{aligned} \mathcal{R}(L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p})) &= \mathcal{R}\left(\textstyle\sum_{i=1}^{N} \alpha_i(\boldsymbol{p})\ L_i(\boldsymbol{x}, \boldsymbol{\omega})\right) \\ &= \textstyle\sum_{i=1}^{N} \alpha_i(\boldsymbol{p})\ \mathcal{R}(L_i(\boldsymbol{x}, \boldsymbol{\omega})) \\ &= \textstyle\sum_{i=1}^{N} \alpha_i(\boldsymbol{p})\ I_i \end{aligned} \tag{23}$$

where $L_i(\boldsymbol{x}, \boldsymbol{\omega})$ are the basis lights and I_i are the basis images. That is, an image of the model scene with the new light source $L(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p})$ can be re-rendered by linearly combining the basis images.

A directional spot light is a point light source whose emitted angular radiance distribution is anisotropic. Typically, spot lights are rotationally symmetric about their directions of foci; thus, their radiance distributions can be described as: $L_{\mathrm{spot}}(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p}) = \boldsymbol{\delta}(\boldsymbol{x} - \boldsymbol{x_0})\ f(\boldsymbol{\omega} \cdot \boldsymbol{p})$ where $\boldsymbol{p}$ is a unit vector parameterizing the direction of focus of the spot light, and $\boldsymbol{x_0}$ denotes the origin of the spot light. In particular, we use $f(\boldsymbol{\omega} \cdot \boldsymbol{p}) = (1 + \boldsymbol{\omega} \cdot \boldsymbol{p})^N$ in our examples where $(N+1)^2$ basis lights (and correspondingly, basis images) are required. $f(\boldsymbol{\omega} \cdot \boldsymbol{p})$ can be expanded as follows:

$$f(\boldsymbol{\omega} \cdot \boldsymbol{p}) \quad = \quad (1 + \boldsymbol{\omega} \cdot \boldsymbol{p})^N$$

$$\begin{aligned} &= (1 + p_x\omega_x + p_y\omega_y + p_z\omega_z)^N \\ &= \sum_{n=0}^{N} \sum_{\substack{i+j+k=n, \\ i,j,k\geq 0.}} \alpha^n_{i,j,k}(\boldsymbol{p}) \left(\omega_x^i \omega_y^j \omega_z^k\right) \end{aligned}$$

where the weighting coefficients are:

$$\alpha^n_{i,j,k}(\boldsymbol{p}) = \frac{N!}{(N-n)!(i)!(j)!(k)!} p_x^i p_y^j p_z^k,$$

and the functions $\omega_x^i \omega_y^j \omega_z^k$ are the basis functions. The above expansions seems to indicate that the total number of basis light sources is $\sum_{n=0}^{N} \sum_{i=0}^{n} \sum_{j=0}^{n-i} 1 = (N+1)(N+2)(N+3)/6$. Fortunately, the actual number of basis lights required is less because the basis light source distributions $\omega_x^i \omega_y^j \omega_z^k$ are not linearly independent. The linear dependence is evident when one considers $\omega_x, \omega_y, \omega_z$ as components of a unit vector, i.e., $\omega_x^2 + \omega_y^2 + \omega_z^2 \equiv 1$. The actual number of basis functions required is only $(N+1)^2$, the number of spherical harmonics up to degree N.

Instead of the monomial basis light sources $\omega_x^i \omega_y^j \omega_z^k$, a sampled basis set comprising the desired spot light aimed in different directions was used. In particular, unit vectors $\boldsymbol{p_i}$, for $1 \leq i \leq (N+1)^2$, distributed on the sphere were randomly selected and used to construct the corresponding basis lights: $L(\boldsymbol{x_0}, \boldsymbol{\omega}; \boldsymbol{p_i}) = (1 + \boldsymbol{\omega} \cdot \boldsymbol{p_i})^N$. Since each new (directional spot) basis light can be expressed as a linear combination of the monomial basis lights, the new sampled basis set can be used in place of the monomial basis set.

Figure 28 shows images of a model scene illuminated by spot lights of degree $N = 5$. Observe that in Figure 28 (a), the reflection of the white wall in the sphere is brighter then the reflection of the red wall. This is because the white wall is being illuminated by the spot light; thus, linear re-rendering captures all ray interactions.

The radiance distribution of an area light source is a four-dimensional function: two dimensions specify the angular distribution and two dimensions specify the spatial distribution. For the purpose of steering over position, we assume that the function is separable in its angular and spatial dimensions: $L_{\text{area}}(\boldsymbol{x}, \boldsymbol{\omega}; \boldsymbol{p}) = f_x(\boldsymbol{x} - \boldsymbol{p})\ f_\omega(\boldsymbol{\omega})$ where $\boldsymbol{p}$ is now a two-dimensional vector parameterizing the position of the light

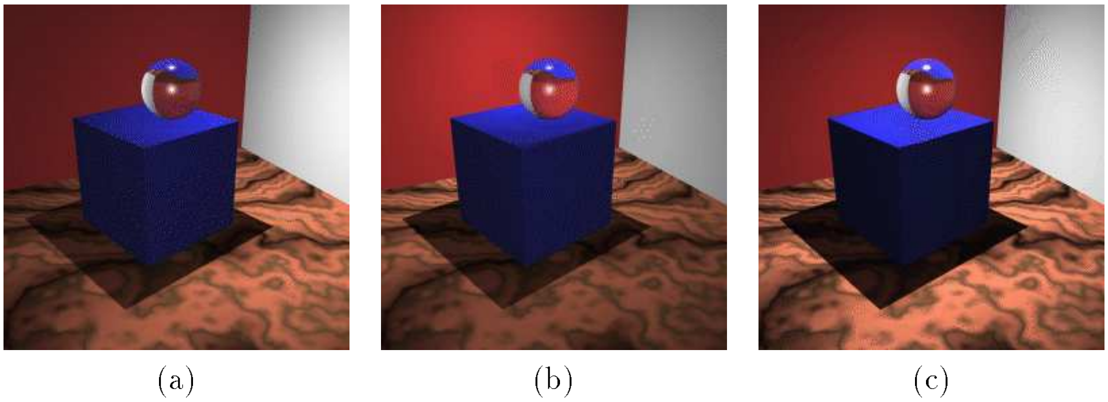


Figure 28: Images of a scene illuminated by a directional spot light. The angular radiance distribution of the light source is a fifth degree polynomial. Each of the images were re-rendered by linearly combining a set of 12 basis images. The left and middle images show the scene re-rendered with the spot light pointed in different directions. The right image shows the scene re-rendered with two spot lights in the same position, but pointing in different directions.

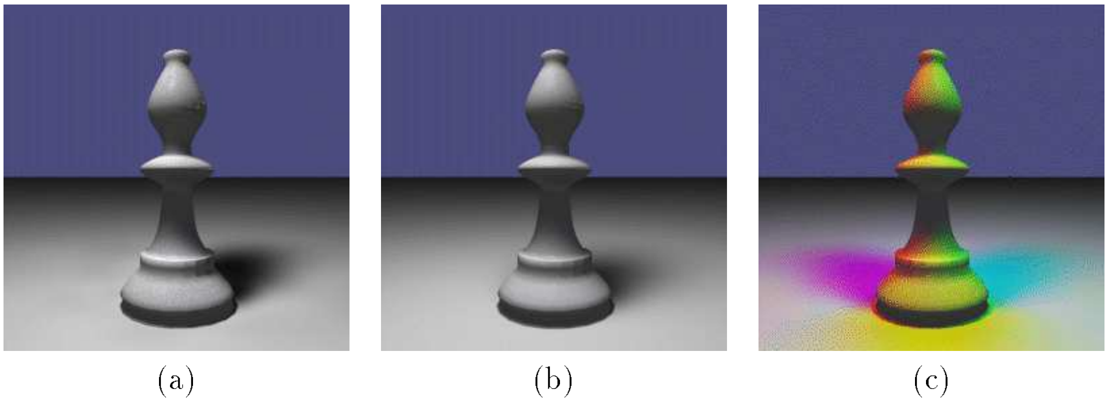


Figure 29: Images of a chess piece illuminated by an area light source. The left image shows the scene re-rendered with the area light source positioned to the front and left of the object. The middle image is a re-rendering of the scene with a broader area light source. The right image shows the object illuminated by three primary colored lights. A total of 20 basis images were used to re-render all three images.

source. For example, when the area light source is defined over a plane, $\boldsymbol{x}$ is the two-dimensional coordinates on the plane, and $\boldsymbol{p}$ parameterizes the origin of the coordinate system. Since the function f_ω is not involved in the steering, it can be arbitrarily complex. Figure 29 shows a chess piece illuminated by an area light source with a raised-cosine spatial radiance distribution (in each dimension). A two-dimensional raised-cosine can be approximated by its Fourier decomposition, i.e., by a linear combination of sinusoids and cosinusoids of different frequencies. Since sinusoids and cosinusoids are steerable over translation, the raised-cosine approximation can also be steered by steering its Fourier basis functions.

Figure 30 shows an example of steering both in position and direction. The light source was separable in its angular and spatial dimension; the spatial distribution was made up of raised cosines, and the angular distribution was composed of polynomials of degree $N = 3$. The basis functions of this light source are the pairwise products of the basis functions of the spatial and the angular basis functions. Likewise, the weighting functions are the pairwise products of the spatial and the angular weighting functions. Notice that the shadow is unchanged when the direction of the light is steered (left and middle images), but the shadow changes when the light's position is shifted (right image).

The time required to re-render an image is proportional to the number of basis images. Principal component analysis can be used to compute a reduced set of basis images best approximating the original set. Specifically, the first k principal components are the best (in a least-squares sense) k basis images approximating the original basis set. Due to the number of pixels in each basis image, it is infeasible to compute the principal components using techniques like the singular value decomposition directly. Details of how to compute the principal components of the original set of basis images are described in [TSH97]. All the images in Figures 28, 29, and 30 were computed with reduced basis sets (that accounted for 90% of the total variance). The reduction in the number of basis images is significant. The images in Figure 28 were rendered with 12 principal component basis images instead of the 36 original basis images. The images in Figure 29 were re-rendered with 20 principal components instead of the 81 original basis images. For Figure 30, there were 400 original basis images, but only 50 principal components were used.

The appeal of the linear re-rendering approach lies in its efficiency: only linear combinations of the basis images are required, independent of scene and initial rendering complexity. Unfortunately, re-rendering scenes illuminated by light sources with narrow angular or spatial distributions requires a large number of basis images. Fortunately, existing ray-tracing rendering techniques are particularly efficient for such narrow light sources (and conversely, inefficient when the radiance distribution of the light sources are broad). This observation suggests a hybrid scheme: the original light source distribution is decomposed into the sum of two components: a smooth, steerable component and a narrow, compactly supported second component. Since the first component is smooth, only a small number of basis images are required to re-render the scene using the linear re-rendering method. The contribution of the second component can be efficiently rendered using ray-tracing rendering techniques for each new light source position. The two resulting images are then pixelwise summed together to produce the final image. This hybrid approach would be useful for re-rendering scenes illuminated by skylight since the distribution of skylight comprises the sum of a widespread slowly varying component and a strong, narrow component (the sun). Since the scene illuminated by the narrow component of the light source has to be rendered using conventional techniques for each new light source position, this method cannot be used for interactive lighting design without adequate hardware acceleration. Nevertheless, its efficiency is likely to be useful in off-line renderings of animations involving illumination changes.

6.4 Invariants from Equivariant Function Spaces

Image invariants are quantities computed from the image that are invariant under certain transformations. For example, the magnitude of the local image gradient is a rotation invariant. The magnitude of the local image gradient measures the presence of an intensity edge in the image and is invariant to rotation of that edge; that is, the magnitude is independent of the orientation of the intensity edge. Image invariants are useful for feature detection, recognition, and matching because they represent the properties of an image that are intrinsic and not artifacts of some transformation. A face recognition algorithm, for instance, should rely on features that are insensitive to moderate changes in pose and illumination; otherwise, the algorithm will, incorrectly, distinguish two instances of the same face with different pose as being different.

Such image invariants can be computed directly from an image via an integral transform like the Fourier transform (for translational invariance), the Fourier-Mellin transform (for rotation and scale invariance), or a generalized version of these transforms (for an arbitrary Abelian two-parameter transformation group) [FC88, RSZ91, SRZ92]. Alternatively, special invariants like the moment invariants can be constructed using nonlinear functions of the image moments [Hu62, Rei91]; these invariants are typically determined by inspection, with the exception of [RSV96] where the authors present a method of deriving them using the normalization method. As pointed out in Section 1.1.3 of Chapter 1, the kernels of the integral transforms and the polynomials used to compute image moments are steerable functions under their respective transformation groups. Sets of these functions used to compute the various invariants span specific equivariant function spaces. In this section, we present several methods for constructing invariants from arbitrary equivariant function spaces. We demonstrate that since these equivariant function spaces are finite-dimensional, techniques for computing invariants over points can be applied.

We consider an n-dimensional function space equivariant under a given transformation group to be an n-dimensional *equivariant measurement space*. A vector of n basis functions ϕ_i of the function space can be viewed as a vector of n measurement kernels; these kernels, applied as inner-products with an image such that $f_i = \langle \phi_i, I \rangle$, measure certain features in the image. For example, if ϕ_i correspond to the first partial derivatives of a Gaussian, the features f_i measure the local (regularized) partial derivatives of the image. Since the function space is equivariant, the features of a transformed image can be computed in terms of the original vector of features. Mathematically, we write

$$\hat{\boldsymbol{f}} = \boldsymbol{A}(\boldsymbol{\tau})\, \boldsymbol{f} \tag{24}$$

where $\boldsymbol{f}$ is an n-dimensional vector $(f_1, \ldots, f_n)^T$ representing the original feature set and $\hat{\boldsymbol{f}}$ denote the feature vector of the transformed image. The vector $\boldsymbol{\tau}$ describes the k-parameter transformation group and $\boldsymbol{A}(\boldsymbol{\tau})$ is the $n \times n$ interpolation matrix of the measurement kernels. Again, if $\boldsymbol{f} = (f_x, f_y)$ measure the first partial derivatives of the image (at the origin) and the image undergoes rotation, the transformed feature set $\hat{\boldsymbol{f}} = (\hat{f}_x, \hat{f}_y)$ is related to the original by a 2×2 rotation matrix $\boldsymbol{A}(\tau) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$. Equation 24 can easily be derived from the interpolation equation of the measurement kernels. Thus, the vector space of features is also closed with respect to the same group of transformations; that is, the features corresponding to the transformed image can be computed from the features derived from the original image. We refer to the feature space as an *equivariant feature space.*

Each point $\boldsymbol{f}$ in an n-dimensional equivariant feature space represents a vector of n measurements, which are the inner-products of the n measurement kernels with a particular image. Transforming the image under a k-parameter group traces out a k-dimensional manifold from the original point, and is known as the orbit of $\boldsymbol{f}$. This manifold is described parametrically by the interpolation equation, and is constructed by applying the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$ to the original point $\boldsymbol{f}$ for all values of $\boldsymbol{\tau}$. In the case of one-parameter groups, the orbits are one-dimensional space curves in an n-dimensional space.

With the equivariant feature space consisting of the first order partial derivatives of the image in the x- and y-directions, the original 2-vector of measurements traces out a circular orbit as the image being measured is rotated. That is, rotating the image causes the feature vector $\boldsymbol{f}$ to rotate in feature space as well. This is true since the transformed feature vector $\hat{\boldsymbol{f}}$ is computed by pre-multiplying the original feature vector $\boldsymbol{f}$ by a 2×2 rotation matrix $\boldsymbol{A}(\tau)$. Thus, the orbit or the set of feature vectors measured from all rotated versions of a single image is a circle in feature space. Figure 31 plots these orbits, which have been computed by setting $h(f_1, f_2) = f_1^2 + f_2^2$ to different values. Each circle represents the orbit corresponding to the feature vectors of a possibly different family of rotated images. Two rotated versions of the same image will have feature vectors residing on the same circle; on the other hand, two feature vectors on the same circle may not correspond to rotated

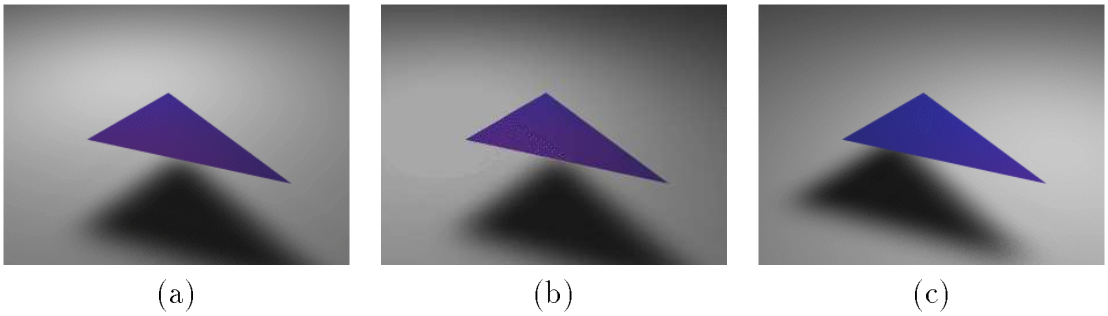


Figure 30: Images of a single polygon illuminated by an area light source that has an anisotropic angular distribution. The left image shows a re-rendering with the light source pointing downwards, and positioned to the rear and left of the object. The middle image shows a re-rendering with the light source in the same position as before but pointing in a different direction. The right image is a re-rendering with the light source centered at a different position. A total of 50 basis images were used to re-render all three images. These basis images were computed using the singular value decomposition; the actual number of basis images required was 400.

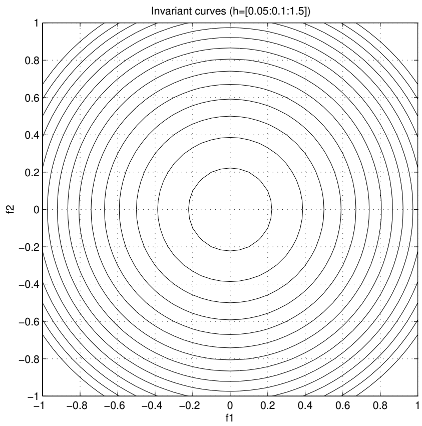


Figure 31: Orbits of a two-dimensional function space equivariant under a one-parameter transformation group. Each curve (circle) is an orbit representing some fixed value of the invariant $h(f_1, f_2) = f_1^2 + f_2^2$.

versions of the same image. More importantly, however, two feature vectors not on the same circle, thus having different values for the invariant h, are necessarily not computed from rotated versions of the same image. Thus, the discriminative power of this invariant is one-dimensional.

In general, an n-dimensional equivariant feature space under a k-parameter transformation group possesses k-dimensional orbits and admits $n-k$ invariants. Thus, the dimension of the equivariant feature space n must be strictly larger than the number of parameters describing the transformation group for the feature space to possess any invariants. Figure 32 plots the orbits of a different three-dimensional feature space equivariant under a one-parameter group; thus, its orbits are one-dimensional and it contains a two-dimensional family of invariants.

Since two feature vectors derived from transformed versions of the same image must reside on the same k-dimensional orbit manifold, functions of features that are constant over each orbit are invariant under the transformation. Formally, an invariant is a function h such that $h(\hat{\boldsymbol{f}}) = h(\boldsymbol{f}) = \text{const}$ for any two vectors $\hat{\boldsymbol{f}}$ and $\boldsymbol{f}$ such that $\hat{\boldsymbol{f}}$ is the feature vector measured from the transformed image. In general, deriving functions that are invariant over arbitrary families of manifolds is difficult. However, the orbit manifolds in an equivariant feature space are far from arbitrary. This is because they are described by the matrix $\boldsymbol{A}(\boldsymbol{\tau})$, and the matrix $\boldsymbol{A}(\boldsymbol{\tau})$ is a k-parameter subgroup of the general linear group of invertible matrices. As a result, we can employ a theorem from Lie theory that states that a function is invariant under a transformation group if and only if applying any of the infinitesimal generators of the group to it results in zero identically [OST96]. In our case, this implies that a function $h(\hat{\boldsymbol{f}}) = h(e^{\tau_k B_k} \cdots e^{\tau_1 B_1} \boldsymbol{f})$ is invariant if and only if

$$L_i \; h(\hat{\boldsymbol{f}}) = \boldsymbol{B}_i \hat{\boldsymbol{f}} \cdot \nabla h = \boldsymbol{0} \tag{25}$$

for $1 \leq i \leq k$ and $\nabla h = (\frac{\partial h}{\partial f_1}, \cdots, \frac{\partial h}{\partial f_n})^T$. L_i is an infinitesimal generator of the transformation group, and the matrix $\boldsymbol{B}_i$ is the corresponding infinitesimal generator of the interpolation matrix $\boldsymbol{A}(\boldsymbol{\tau})$. Solving Equation 25 amounts to solving a system of homogeneous, first order partial differential equations. A good review of different techniques to solve such equations can be found in [MPGO95].

The next two examples illustrate how invariants can be derived for different equivariant feature spaces using Equation 25. First, the infinitesimal generator L and the matrix B from the corresponding equivariant measurement space are identified. Second, the partial differential equation with L and B substituted is solved for h, which is invariant to the given transformation.

Example 16 : Let $\Phi(x) = (\cos kx, \sin kx)^T$ be a measurement space that is equivariant under translation such that $L\ \Phi = \frac{\partial}{\partial x}\ \Phi = B\ \Phi$ where

$$B = \begin{pmatrix} 0 & -k \\ k & 0 \end{pmatrix}.$$

The same matrix B also holds for our previous example of first order partial derivatives. By Equation 25, a function $h(\hat{f}_1, \hat{f}_2)$ is invariant under translation if and only if

$$B\ \hat{\mathbf{f}} \cdot \nabla h = -k\hat{f}_2 \frac{\partial h}{\partial \hat{f}_1} + k\hat{f}_1 \frac{\partial h}{\partial \hat{f}_2} = 0.$$

Solving the above equation, we get $h(\hat{f}_1, \hat{f}_2) = h^\star(\hat{f}_1^2 + \hat{f}_2^2)$ for any function $h^\star$, in particular for the identity function as plotted in Figure 31. Hence, we have verified that the sum of squares of the inner-product of any pair of the Fourier basis functions with a signal is invariant under translation, and likewise that the magnitude of the gradient is invariant under rotation.

Example 17 : Consider the measurement space $\Phi(x) = (\frac{x^2}{2!}, x, 1)^T$ that is equivariant under translation such that $L\ \Phi = \frac{\partial}{\partial x}\ \Phi = B\ \Phi$ where

$$B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \hat{\boldsymbol{f}} = \begin{pmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \end{pmatrix} = \begin{pmatrix} 1 & \tau & \frac{\tau^2}{2!} \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{f} = \boldsymbol{A}(\tau)\ \boldsymbol{f}.$$

By Equation 25, a function $h(\hat{f}_1, \hat{f}_2, \hat{f}_3)$ is invariant under translation if and only if

$$B\ \hat{\boldsymbol{f}} \cdot \nabla h = \hat{f}_2 \frac{\partial h}{\partial \hat{f}_1} + \hat{f}_3 \frac{\partial h}{\partial \hat{f}_2} = 0.$$

Since the orbits are one dimensional manifolds in a three dimensional space, there exists two independent solutions to this equation. These are the functions: $h_1(\hat{\boldsymbol{f}}) = \hat{f}_3$ and $h_2(\hat{\boldsymbol{f}}) = \hat{f}_1 - \frac{1}{2}\frac{\hat{f}_2^{\,2}}{\hat{f}_3}$. Actually, any function $h^\star(h_1, h_2)$ is invariant with respect to translation. It is straightforward to verify that h_2 is an invariant:

$$\begin{aligned}
\hat{f}_1 - \frac{1}{2}\frac{\hat{f}_2^{\,2}}{\hat{f}_3} &= f_1 + \tau f_2 + \frac{\tau^2}{2!} f_3 - \frac{1}{2}\frac{(f_2+\tau f_3)^2}{f_3} \\
&= f_1 + \tau f_2 + \frac{\tau^2}{2!} f_3 - \frac{1}{2}\left(\frac{f_2^2}{f_3} + \tau^2 f_3 + 2\tau f_2\right) \\
&= f_1 - \frac{1}{2}\frac{f_2^2}{f_3}.
\end{aligned}$$

The two invariants h_1, h_2 are plotted in Figure 32.

Another approach to constructing invariants in an equivariant feature space is by deriving implicit representations of the feature orbits. The interpolation equation $\hat{\boldsymbol{f}} = \boldsymbol{A}(\boldsymbol{\tau})\ \boldsymbol{f}$ can be seen as a *parametric* description of the orbit manifold. An *implicit* representation of this manifold is a description of the manifold that is independent of the parameters $\boldsymbol{\tau}$ and is represented by a set of independent functions $\{h_i(\hat{\boldsymbol{f}}) = 0\}, 1 \leq i \leq n-k$ whose algebraic variety (the function space spanned by functions that are polynomial combinations of h_k) coincides with the manifold itself. In particular, $h_i(\hat{\boldsymbol{f}}) = h_i(\boldsymbol{f})$ for each function in the set and for any function in the variety. Thus, functions in the variety are invariant under any transformation in the group. Furthermore, any function $h^\star(h_1(\hat{\boldsymbol{f}}), \ldots, h_2(\hat{\boldsymbol{f}}))$ is constant over the orbit manifold and therefore also invariant.

Constructing implicit representations from parametric representations is the well-studied subject of elimination theory from which one can find many techniques (see [WG95a, WG95b] for a short review). One particular general technique for implicitizing parametric descriptions consisting of multivariate polynomials is the method of Groebner bases [CLO92]. This method is a generalization of Gaussian elimination of a linear system of equations. Gaussian elimination can be viewed as a systematic process of eliminating the free variables in the system of equations; when all but one of the variables have been eliminated, this trivial equation is solved and use to solve the other equations (back-substitution). The method of Groebner bases is a technique of elimination generalized to work with multivariate polynomials. Using this technique for implicitization amounts to eliminating the parameters $\boldsymbol{\tau}$ from the interpolation equation; the resulting equations will only depend on $\hat{\boldsymbol{f}}$ and $\boldsymbol{f}$. The use of Groebner bases to generate invariants in computer vision was also recently suggested in [WS95].

Numerous techniques for computing invariants from features like points and derivatives have been proposed by researchers in the past [Me92]. Many of these techniques can be readily applied to constructing invariants on equivariant feature spaces since these spaces are finite-dimensional. The method used in the first two examples above is similar to the one used by Moons *et al.* [MPGO95] to construct invariants on points and derivatives. Another simple method of constructing polynomial invariants was recently proposed by Keren [Ker94] in which instead of seeking to determine all possible invariants, the author describes a procedure for symbolically deriving polynomial invariants of a given polynomial order. The method can also be employed in this context to derive polynomial invariants over feature vectors $\hat{\boldsymbol{f}}$ (or even over prolongations, i.e. multiple feature vectors).

Invariant Feature Detection. Invariant feature or pattern detectors are used to identify specific patterns like edges and corners in an image independent of some family of transformations. For example, detecting an image template of a corner in a larger image, independent of the orientation of the template (i.e. rotation-independent). Within this framework, invariant feature detectors are computed in a sequence of stages:

1. A set of equivariant measurement functions is chosen such that they can represent faithfully the image template via linear combinations. These equivariant measurement functions are then orthogonalized. The coefficients of the image template is then computed by projecting the image template onto each of these orthogonalized measurement functions.

2. A set of possible invariant functions based on the outputs of the orthogonalized measurement functions are derived using the techniques discussed in this section. These functions yield an invariant feature descriptor when computed using the projection coefficients of the image template.

3. Each overlapping region of the larger image, the size of the image template, is projected onto the equivariant measurement functions in turn to compute its projection coefficients. These projection coefficients are then used to compute the invariant feature descriptor of that region of the image.

4. The invariant feature descriptor of each region in the image is compared to the invariant feature descriptor of the image template to ascertain the presence of the image template at that location.

Generating invariant pattern detectors in this manner has the advantage that one first constructs equivariant measurement spaces pthat are rich enough to fully characterize a given pattern. This is done prior to the construction of the invariants. Since the dimension of the feature space is finite and often relatively small, we can then easily compute all the invariants associated with the given equivariant feature space.

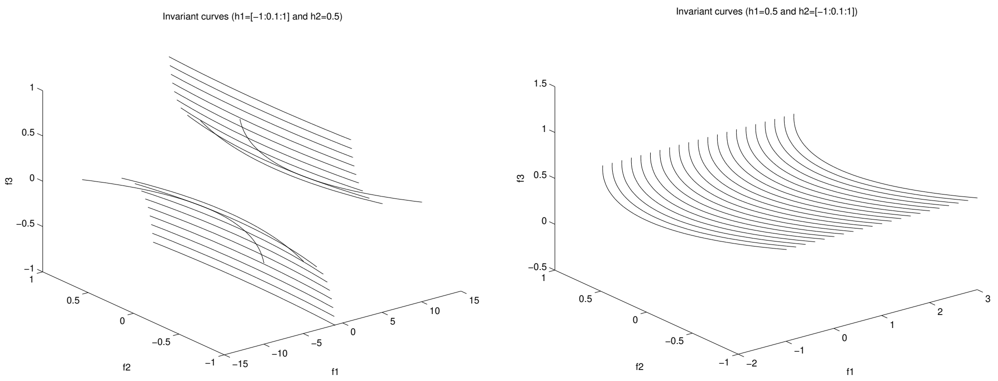
6.5 Equivariance of Points and Lines

Throughout this thesis, steerable functions have been discussed with reference to continuous functions. In particular, steerable filters have been applied to images, which are mathematically represented as continuous functions. The mathematical framework and its application to motion estimation and invariant feature detection can be applied to points and lines in a straightforward manner by representing the points and lines by a sum of delta functions. For example, a set of N points $\{(x_i, y_i)\}$ can be written as an image: $s(x,y) = \sum_{i=1}^{N} \delta(x - x_i, y - y_i)$. As a result, taking the inner-product of a set of basis filters with this image yields

$$\langle \Phi(x,y), s(x,y) \rangle = \int\int \Phi(x,y) \sum_{i=1}^{N} \delta(x - x_i, y - y_i) dx dy = \sum_{i=1}^{N} \Phi(x_i, y_i). \qquad (26)$$

The last equality follows from the definition of the delta function. In a similar way, steerable filters can also be applied to lines (curves); however, proper attention has to be given when defining the integral over curves. A unified way to understand this treatment of points and lines is as distributions; therefore, the steerability of points and lines is a special case of the steerability of distributions.

Since equivariant feature spaces can be constructed over a set of points or lines, all the methods described in Section 6.4 on constructing invariants in equivariant feature spaces can also be used to derive invariants over the set of points or lines. The important difference between this method of constructing point (or line) invariants is that correspondence is not required; that is, the invariants derived using this method are independent of the ordering of the set of points or lines. This is clearly desirable as correspondence is often difficult to obtain. We note, however, that this is not the most general way of constructing point invariants that do not require correspondence. A more general technique involves constructing an equivariant function space over the finite group of permutations (permutations of the points or lines). Preliminary investigation of this idea can be found in [LM94] where the authors describe point invariants based on the cross-ratio that are permutation-invariant.



(a) (b)

Figure 32: Orbits of a three-dimensional function space equivariant under a one-parameter transformation group. The space of invariants is two-dimensional. Each one-dimensional space curve is an orbit representing some pair of fixed values of the invariants $h_1(f_1, f_2, f_3) = f_3$ and $h_2(f_1, f_2, f_3) = f1f3 - \frac{1}{2}f_2^2$. Each curve in (a) is specified by $h_1 \in [-1, 1]$ and $h_2 = 0.5$. Each curve in (b) is specified by $h1 = 0.5$ and $h2 \in [-1, 1]$.

Chapter 7

Conclusions

Steerable functions were first introduced to the image processing, computer vision, and computer graphics communities in the form of steerable filters, a class of linear filters whose kernels are composed of steerable functions. The obvious advantages that steerable filters provide in image analysis over traditional filters are computational efficiency and numerical accuracy, the latter because steerability is fundamentally an analytic property. In this thesis, it was shown that the steerability property is more general and far more useful than was described in its original application. This thesis generalized the steerability property to be a property associated, not with specific classes of transformations, but with any smooth transformation group. Smooth transformation groups are common in image processing and computer vision as they include the groups of 2D and 3D translation, rotation, affine, and projective transformations. This generalization allowed the powerful techniques of Lie group theory to be employed in analyzing the steerability property, and led to a formulation of a mathematical framework in which theoretical questions about steerable functions with respect to any smooth transformation group could be examined abstractly. This framework also resulted in a classification of all one-parameter and multi-parameter steerable functions, and a recipe for constructing and verifying steerable functions.

The design of a suitable set of basis functions given any arbitrary steerable function is one of the main problems concerning steerable functions. To this end, two very different algorithms were developed within the framework. The first algorithm is a symbolic method that can be implemented in any symbolic package. Typically, the basis functions of a steerable function are derived by inspection; this algorithm derives the minimal set of basis functions automatically given an arbitrary steerable function. The second algorithm addresses two practical considerations: approximate steerability and local steerability. In practice, functions that need to be steered might not be steerable with a finite number of basis functions. Moreover, it is often the case that only a small subset of transformations within the group of transformations needs to be considered. In response to these two concerns, the second algorithm computes the optimal set of k basis functions to steer an arbitrary function under a subset of the group of transformations.

As alluded to earlier, the usefulness of the steerability property extends beyond steerable filters. By distinguishing, beyond the application, the property that makes steerable filters so attractive, this thesis identified numerous applications in image processing, computer vision, and computer graphics that either explicitly or implicitly takes advantage of the property of steerability. In particular, it presented five new applications that could benefit from the use of steerable functions: (1) continuum approximation in vision modeling (a method of approximating an infinite number of interacting mechanisms in a model), (2) the design of optimal steerable filters for gradient-based motion estimation, (3) efficient linear re-rendering of synthetic scenes under changes in illumination, (4) the construction of invariants from steerable filters, and (5) the application of steerable functions to discrete sets of points and lines.

The wide applicability of the property of steerability implies that the mathematical framework proposed in this thesis provides a common language with which to discuss the problems in these different fields. In particular, the problems involved with designing steerable filters, designing motion estimation filters, and constructing invariant feature detectors all have at their core the notion of steerability. These problems have traditionally been pursued separately; their joint exploration, within the context of steerability, opens up interesting possibilities. For example, the design of accurate motion estimation filters typically involves the design of accurate derivative filters, without regard to the fact that the intended use of these filters is motion estimation and not derivative estimation. In contrast, the design of optimal steerable filters has been the focus of numerous bodies of research (and is also dealt with in this thesis). Recent research, presented partially in this thesis, has begun to employ the techniques used to compute optimal steerable filters, to compute optimal filters for motion estimation.

The close connection between constructing invariant feature detectors and designing accurate motion estimation filters also suggests the possibility of designing motion estimation filters that are simultaneously invariant to other transformations. One useful application of this is the design of optical flow (translational) filters that are also invariant to rotation and/or scaling. The techniques for designing least-squares optimal steerable functions could also be used to design invariant motion estimation filters such that their invariance and motion estimation accuracy are least-squares optimal within a given desired operating range.

In the construction of invariant feature detectors, one deals with the issues of completeness and invariance of the underlying representation. Trivial representations that are completely invariant can easily be constructed but provide little discriminative power; i.e., all the features to be differentiated are indistinguishable within the representation. For example, a filter with a kernel that is identically zero is perfectly invariant but is completely useless. The connection between steerable filters and invariant features suggests a strategy whereby a suitable suite of steerable filters are first designed such that they provide an adequate representation for the different features, following which invariants based on the outputs of these steerable filters are derived. These invariants do not have to hold over the entire group of transformations but can be local since the steerability property of the filters could be local. Unlike differential invariants which are also local about some operating point, these invariants are local within some operating range, with respect to some optimality criterion like the average least-squares error.

Two important and possibly promising areas of research is the application of steerability to sets of points and lines, and the issue of robustness. Although the property of steerability has been discussed in this thesis with regards to continuous functions, it is also applicable to discrete sets of points and lines. This implies a possible association between steerability and traditional problems of recognition and pose estimation from sets of point and line features. This association is important as it could suggest novel algorithms in both areas, for example, the possibility of least-squares optimal local algorithms in recognition and the use of alignment or geometric-hashing techniques in developing motion estimation filters and invariant features. The issue of robustness is of practical importance particularly in the area of motion estimation and invariant feature detection. While the sensitivity of the outputs of steerable filters with respect to the transformation group for which they were designed can be accounted, their sensitivity in unexpected situations is unclear. For example, it is not known the amount of error that would be introduced in the output of a motion estimation steerable filter in the presence of multiple motions (at a depth discontinuity, for example). Similarly, the robustness of an invariant feature detector undergoing some other transformation than it was designed for is unclear.

In conclusion, the mathematical framework presented in this thesis provides both an analytical framework with which to understand the property of steerability as well as a common framework to discuss problems in different areas that are connected by their common assumption of steerability. With regards to the former, it is demonstrated that Lie group theory is the appropriate mathematical tool for understanding the properties of functions that are steerable under smooth transformation groups. This position is supported by the fact that all existing analytical approaches to steerability can be consistently explained within the framework. A common framework with which to investigate problems from different areas, namely steerable filter design, motion estimation and invariant feature detection, facilitates the transfer of results between these different areas more readily and opens up numerous possibilities for the integration of the various techniques.

Bibliography

[AB85] E. Adelson and J. Bergen. Spatiotemporal energy models for the perception of motion. *J. Optical Society of America A*, 2(2):284–299, February 1985.

[AG91] D G Albrecht and W S Geisler. Motion sensitivity and the contrast-response funtion of simple cells in the visual cortex. *Visual Neuroscience*, 7:531–546, 1991.

[Ama68] S. Amari. Invariant structures of signal and feature spaces in pattern recognition problems. *RAAG Memoirs*, 4:19–32, 1968.

[Ama78] S. Amari. Feature spaces which admit and detect invariant signal transformations. In *Int'l Conf. Pattern Recognition*, pages 452–456, 1978.

[Ama87] S. Amari. A theory on the determination of 3D motion and 3D structure from features. *Spatial Vision*, 2(2):151–168, 1987.

[And92] M. Andersson. Controllable multidimensional filters and models in low level computer vision. Technical Report Dissertations No. 282, Department of Electrical Engineering, Linköping University, 1992.

[Bei94] W. Beil. Steerable filters and invariance theory. *Pattern Recognition Letters*, 15(5):453–460, 1994.

[BK89] J. Belinfante and B. Kolman. *A Survey of Lie Groups and Lie Algebras with Applications and Computational Methods*. SIAM, Philadelphia, 1989.

[CLO92] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, Berlin; New York, 1992.

[Coh11] A. Cohen. *An Introduction to the Lie Theory of One-Parameter Groups; With Applications to the Solution of Differential Equations*. D. C. Heath & Co., Boston; New York, 1911.

[CSM95] R. Carter, G. Segal, and I. Macdonald. *Lectures on Lie Groups and Lie Algebras*. Cambridge University Press, Great Britain, 1995.

[DAG95] J. Dorsey, J. Arvo, and D. Greenberg. Interactive design of complex time-dependent lighting. *IEEE Computer Graphics and Applications*, 15(2):26–36, 1995.

[DKNY95] Y. Dobashi, K. Kaneda, H. Nakatani, and H. Yamashita. A quick rendering method using basis functions for interactive lighting design. In *Eurographics*, pages 229–240, 1995.

[Dod83] P. C. Dodwell. The Lie transformation group model of visual perception. *Perception and Psychophysics*, 34:1–16, 1983.

[Eag92a] R. Eagleson. Group-theoretic approach to motion analysis for 3D robotic tracking. In A. Sood and H. Wechsler, editors, *Active Perception and Robot Vision*, pages 373–394. Springer-Verlag, Berlin; New York, 1992.

[Eag92b] R. Eagleson. Measurement of the 2D affine Lie group parameters for visual motion analysis. *Spatial Vision*, 6(3):183–198, 1992.

[ETHO97] M. Elad, P. Teo, and Y. Hel-Or. Optimal filters for gradient-based motion estimation. Technical report, Hewlett Packard Laboratories, 1997.

[FA91] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[FB94] J. Foley and G. Boynton. A new model of human luminance pattern vision mechanisms: Analysis of the effects of pattern orientation, spatial phase, and temporal frequency. In *Computational Vision Based on Neurobiology, SPIE Proc. vol. 2054*, 1994.

[FC88] M. Ferraro and T. Caelli. Relationship between integral transform invariances and Lie group theory. *J. Optical Society of America A*, 5:738–742, 1988.

[FC94] M. Ferraro and T. Caelli. Lie transformation groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8(1):33–44, 1994.

[FJ91] D. Fleet and A. Jepson. Phase-based disparity measurement. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):198–210, 1991.

[Fle90] D. Fleet. Computation of component image velocity from local phase information. *Int'l J. Computer Vision*, 5(1):77–104, 1990.

[Fol94] J. Foley. Human luminance pattern-vision mechanisms: masking experiments require a new model. *J. Optical Society of America A*, 11:1710–1719, 1994.

[GBG+94] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 222–228, 1994.

[GH86] N. Green and P. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, 1986.

[GK95] G. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, Boston, 1995.

[Got94] C. Gotsman. Constant-time filtering by singular value decomposition. *Computer Graphics Forum*, 13(2):153–163, 1994.

[Gug63] H. W. Guggenheimer. *Differential Geometry.* McGraw-Hill Book Company, New York, 1963.

[Hag92] L. Haglund. Adaptive multidimensional filtering. Technical Report Dissertations No. 284, Department of Electrical Engineering, Linköping University, 1992.

[HB95] D. Heeger and J. Bergen. Pyramid-based texture analysis/synthesis. *Computer Graphics (Siggraph)*, pages 229–238, 1995.

[Hec86] P. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[Hee87] D. Heeger. Optical flow using spatiotemporal filters. *Int'l J. Computer Vision*, 1(4):279–302, 1987.

[Hee92a] D. J. Heeger. Half-squaring in responses of cat simple cells. *Visual Neuroscience*, 9:427–443, 1992.

[Hee92b] D. J. Heeger. Normalization of cell responses in cat striate cortex. *Visual Neuroscience*, 9:181–198, 1992.

[Hee93] D J Heeger. Modeling simple cell direction selectivity with normalized, half-squared, linear operators. *Journal of Neurophysiology*, 70:1885–1898, 1993.

[Her66] R. Hermann. *Lie Groups for Physicists.* W. A. Benjamin, Inc., New York, 1966.

[HKP91] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation: Lecture Notes Vol. 1.* Addison-Wesley Publishing Company, Redwood City, CA, 1991.

[Hof66a] W. C. Hoffman. The Lie algebra of visual perception. *J. Mathematical Psychology*, 3:65–98, 1966.

[Hof66b] W. C. Hoffman. The neuron as a Lie group germ and a Lie product. *Quarterly. J. Applied Mathematics*, 25:423–441, 1966.

[Hof70] W. C. Hoffman. Higher visual perception as prolongations of the basic Lie transformation group. *Mathematical Biosciences*, 6:437–471, 1970.

[HOT96] Y. Hel-Or and P. Teo. Canonical decomposition of steerable functions. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 809–816, 1996.

[HOT98] Y. Hel-Or and P. Teo. Canonical decomposition of steerable functions. *J. Mathematical Imaging and Vision*, 1998. To appear.

[HS93] D. Heeger and E. Simoncelli. Model of visual motion sensing. In L. Harris and M. Jenkin, editors, *Spatial Vision in Humans and Robots*, pages 367–392. Cambridge University Press, 1993.

[Hu62] M. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Information Theory*, February:179–187, 1962.

[Ker94] D. Keren. Using symbolic computation to find algebraic invariants. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(11):1143–1149, 1994.

[Kir76] A. Kirillov. *Elements of the Theory of Representations*. Springer-Verlag, New York, 1976.

[Kna86] A. Knapp. *Representation Theory of Semisimple Groups*. Princeton University Press, Princeton, New Jersey, 1986.

[Len89a] R. Lenz. Describing and recognizing group-invariant pattern classes with group-sampling. *Pattern Recognition Letters*, 9:169–173, 1989.

[Len89b] R. Lenz. Group theoretical model of feature extraction. *J. Optical Society of America A*, 6(6):827–834, 1989.

[Len90a] R. Lenz. Group invariant pattern recognition. *Pattern Recognition*, 23(1/2):199–217, 1990.

[Len90b] R. Lenz. *Group Theoretical Methods in Image Processing*. Springer-Verlag, Berlin; New York, 1990.

[Len91] R. Lenz. On probabilistic invariance. *Neural Networks*, 4:627–641, 1991.

[Len94] R. Lenz. Group theoretical transforms in image processing. *Current Topics in Pattern Recognition Research*, 1:83–106, 1994.

[LH97] R. Lenz and K. Homma. Constructing iterative matching algorithms with the use of Lie theory: three-dimensional orientation example. *J. Optical Society of America A*, 14(8):1734–1741, 1997.

[LHHC94] H. Liu, T. Hong, M. Herman, and R. Chellappa. A generalized motion model for estimating optical flow using 3D Hermite polynomials. In *Int'l Conf. Pattern Recognition*, pages 9–13, Jerusalem, Israel, 1994.

[LM94] R. Lenz and P. Meer. Point configuration invariants under simultaneous projective and permutation transformations. *Pattern Recognition*, 27(11):1523–1532, 1994.

[Man94] R. Manmatha. A framework for recovering affine transforms using points, lines or image brightnesses. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 141–146, Seattle, WA, 1994.

[Me92] J. Mundy and A. Zisserman (ed.). *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Massachusetts, 1992.

[Mic95a] M. Michaelis. A Lie group approach to steerable filters. *Pattern Recognition Letters*, 16:1165–1174, 1995.

[Mic95b] M. Michaelis. Low level image processing using steerable filters. Technical Report PhD Dissertation, Fakultät für Informatik, University of Kiel, Germany, 1995.

[MO93] R. Manmatha and J. Oliensis. Extracting affine deformations from image patches I: finding scale and rotation. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 754–755, New York, NY, 1993.

[MP95] R. Manduchi and P. Perona. Pyramidal implementation of deformable kernels. In *Int'l Conf. Image Processing*, pages 378–381, 1995.

[MPGO95] T. Moons, E. Pauwels, L. Van Gool, and A. Oosterlinck. Foundations of semi-differential invariants. *Int'l J. Computer Vision*, 14:25–47, 1995.

[MR93] J. Malik and R. Rosenholtz. A differential method for computing local shape-from-texture for planar and curved surfaces. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 267–273, 1993.

[MR94] J. Malik and R. Rosenholtz. Recovering surface curvature and orientation from texture distortion: a least squares algorithm and sensitivity analysis. In *European Conf. Computer Vision*, pages 353–364, 1994.

[Nor94] K. Nordberg. Signal representation and processing using operator groups. Technical Report Dissertations No. 366, Department of Electrical Engineering, Linköping University, 1994.

[NSD94] J. Nimeroff, E. Simoncelli, and J. Dorsey. Efficient re-rendering of naturally illuminated environments. In *5th Eurographics Workshop on Rendering*, pages 373–388, 1994.

[Olv95] P. Olver. *Equivalence, Invariants, and Symmetry.* Cambridge University Press, Cambridge, 1995.

[OST96] P. Olver, G. Sapiro, and A. Tannenbaum. Affine invariant detection: edges, active contours, and segments. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 520–525, 1996.

[Per92] P. Perona. Steerable-scalable kernels for edge detection and junction analysis. *Image and Vision Computing*, 10(10):663–672, 1992.

[Per95] P. Perona. Deformable kernels for early vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):488–499, 1995.

[PJ89] T. V. Papathomas and B. Julesz. Lie differential operators in animals and machine vision. In J. C. Simon, editor, *From Pixels to Features*, pages 115–126. Elsevier Science Publishers B. V., North Holland, 1989.

[Rei91] T. Reiss. The revised fundamental theorem of moment invariants. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:830–834, 1991.

[Rei93] T. Reiss. *Recognizing Planar Objects Using Invariant Image Features.* Springer-Verlag, Berlin, 1993.

[Rom94] B. Romeny. *Geometry Driven Diffusion in Computer Vision.* Kluwer Academic Press, Boston, 1994.

[RSV96] I. Rothe, H. Süsse, and K Voss. The method of normalization to determine invariants. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(4):366–375, 1996.

[RSZ91] J. Rubinstein, J. Segman, and Y. Zeevi. Recognition of distorted patterns by invariance kernels. *Pattern Recognition*, 24(10):959–967, 1991.

[Sap93] G. Sapiro. Affine invariant scale-space. *Int'l J. Computer Vision*, 11(1):25–44, 1993.

[Sap96] G. Sapiro. Vector (self) snakes: a geometric framework for color, texture, and multiscale image segmentation. In *Int'l Conf. Image Processing*, pages 817–820, vol 1, 1996.

[SC94a] J. Sato and R. Cipolla. Extracting the affine transformation from texture moments. Technical Report CUED/F-INFENG/TR 167, Department of Engineering, University of Cambridge, 1994.

[SC94b] J. Sato and R. Cipolla. Image registration using multi-scale texture moments. Technical Report CUED/F-INFENG/TR 177, Department of Engineering, University of Cambridge, 1994.

[SF95a] E. Simoncelli and H. Farid. Steerable wedge filters. In *Int'l Conf. Computer Vision*, pages 189–194, Boston, MA, 1995.

[SF95b] E. Simoncelli and W. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *Int'l Conf. Image Processing*, pages 444–447, vol 3, 1995.

[SFAH92] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multiscale transforms. *IEEE Trans. Information Theory*, 38(2):587–607, 1992.

[SH98] E. Simoncelli and D. Heeger. A velocity population coding model of MT neurons. *Vision Research*, 1998. In press.

[Sim94] E. Simoncelli. Design of multi-dimensional derivative filters. In *Int'l Conf. Image Processing*, pages 790–794, vol 1, 1994.

[Sim96] E. Simoncelli. A rotation-invariant pattern signature. In *Int'l Conf. Image Processing*, pages 185–188, vol 3, 1996.

[SP94] D. Shy and P. Perona. X-Y separable pyramid steerable scalable kernels. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 237–244, Seattle, WA, 1994.

[SRZ92] J. Segman, J. Rubinstein, and Y. Zeevi. The canonical coordinate method for pattern deformation: theoretical and computational considerations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(12):1171–1183, 1992.

[Str88] G. Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, Inc., San Deigo, CA, 1988.

[SW73] A. Sagle and R. Walde. *Introduction to Lie Groups and Lie Algebras.* Academic Press, New York, 1973.

[Tal68] J. Talman. *Special functions; a group theoretic approach.* W. A. Benjamin, Inc., New York, NY, 1968.

[TH94a] P. Teo and D. Heeger. Perceptual image distortion. In *Int'l Conf. Image Processing*, pages 982–986, vol 2, 1994.

[TH94b] P. Teo and D. Heeger. Perceptual image distortion. In *Human Vision, Visual Processing, and Digital Display V, SPIE Proc. vol. 2179*, pages 127–141, 1994.

[TH95] P. Teo and D. Heeger. A general, mechanistic model of spatial pattern detection. In *Association for Research in Vision and Ophthalmology*, 1995.

[THO97] P. Teo and Y. Hel-Or. A computational approach to steerable functions. In *Int'l Conf. Computer Vision and Pattern Recognition*, pages 313–318, 1997.

[THO98a] P. Teo and Y. Hel-Or. Design of multi-parameter steerable functions using cascade basis reduction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1998. To appear.

[THO98b] P. Teo and Y. Hel-Or. Design of multi-parameter steerable functions using cascade basis reduction. In *Int'l Conf. Computer Vision*, 1998. To appear.

[THO98c] P. Teo and Y. Hel-Or. Lie generators for computing steerable functions. *Pattern Recognition*, 1998. To appear.

[TSG94] T. M. H. Tijkstra, P. R. Snoeren, and C. C. A. M. Gielen. Extraction of three-dimensional shape from optical flow: a geometric approach. *J. Optical Society of America A*, 11(8):2184–2196, 1994.

[TSH97] P. Teo, E. Simoncelli, and D. Heeger. Efficient linear re-rendering for interactive lighting design. Technical Report STAN-CS-TN-97-60, Stanford University, 1997.

[WA85] A. Watson and A. Ahumada. Model of human visual-motion sensing. *J. Optical Society of America A*, 2(2):322–342, February 1985.

[Wen93] J. Weng. Image matching using the windowed Fourier phase. *Int'l J. Computer Vision*, 11(3):211–236, 1993.

[WG95a] C. Wee and R. Goldman. Elimination and resultants; part 1: elimination and bivariate resultants. *IEEE Computer Graphics and Applications*, 15(1):69–77, 1995.

[WG95b] C. Wee and R. Goldman. Elimination and resultants; part 2: multivariate resultants. *IEEE Computer Graphics and Applications*, 15(2):60–69, 1995.

[Wil83] L. Williams. Pyramidal parametrics. *Computer Graphics (Siggraph)*, 17(3):1–11, 1983.

[WK88] R. Wilson and H. Knutsson. Uncertainty and inference in the visual system. *IEEE Trans. Systems, Man, and Cybernetics*, 18(2):305–312, 1988.

[Woo96] J. Wood. Invariant pattern recognition: a review. *Pattern Recognition*, 29(1):1–17, 1996.

[WS95] M. Werman and A. Shashua. The study of 3D-from-2D using elimination. In *Int'l Conf. Computer Vision*, pages 473–479, 1995.

[XS94] Y. Xiong and S. Shafer. Moment and hypergeometric filters for high precision computation of focus, stereo and optical flow. Technical Report CMU-RI-TR-94-28, Carnegie Mellon University, 1994.

[XS95a] Y. Xiong and S. Shafer. Hypergeometric filters for optical flow and affine matching. In *Int'l Conf. Computer Vision*, pages 771–776, Boston, MA, 1995.

[XS95b] Y. Xiong and S. Shafer. Moment filters for high precision computation of focus and stereo. In *Proc. Int. Conf. on Intelligent Robots and Systems*, pages 108–113, 1995.