

Lie Generators for Computing Steerable Functions

Patrick C. Teo

Yacov Hel-Or

Department of Computer Science NASA Ames Research Center

Stanford University

Moffet Field

Stanford, CA 94305

CA 94035-1000

Abstract

We present a computational, group-theoretic approach to steerable functions. The approach is group-theoretic in that the treatment involves continuous transformation groups for which elementary Lie group theory may be applied. The approach is computational in that the theory is constructive and leads directly to a procedural implementation. For functions that are steerable with n basis functions under a k -parameter group, the procedure is efficient in that at most $nk + 1$ iterations of the procedure are needed to compute all the basis functions. Furthermore, the procedure is guaranteed to return the minimum number of basis functions. If the function is not steerable, a numerical implementation of the procedure could be used to compute basis functions that approximately steer the function over a range of transformation parameters. Examples of both applications are described.

Keywords: *Steerable Filters, Filter Design, Low-level Vision, Feature detection, Lie Groups.*

1 Introduction

Steerable functions have been widely used in image processing [Freeman and Adelson 1991, Simoncelli et. al. 1992, Simoncelli and Farid 1995, Perona 1995], computer vision [Granlund and Knutsson 1990, Fleet 1990, Weng 1993], and recently, even in computer graphics [Gotsman 1994, Nimeroff et. al. 1994]. Simply put, a function is steerable under some transformation when all transformed versions of the function can be expressed as a linear combination of a fixed, finite set of basis functions. The weights of the linear combination depend solely on the transform parameters.

The importance of steerable functions stems from the property of superposition of linear systems. Hence, any linear operation applied to a transformed version of a steerable function can be expressed as a linear combination of the operation applied separately to the basis functions. The main advantage of this property is that the linear operations can be applied to the basis function once and off-line. In image processing, steerable functions have been used as filter kernels. Because convolution is a linear operation, the filter output of a transformed version of the filter kernel can be obtained by linearly combining the filter outputs of its associated basis filters.

Freeman and Adelson presented functions steerable with respect to rotation using derivatives of a Gaussian as the basis set [Freeman and Adelson 1991]. An extension of this technique to translation and scaling was shown by Simoncelli *et al.* [Simoncelli et. al. 1992]. Indeed, steerable filters are the most popular application of steerable functions; thus, the examples in this paper will pertain mainly to steerable filters.

Many of the commonly encountered families of transformations on images form continuous groups. Examples of these continuous groups include: image translation, rotation and

scaling. For these families of transformations, Lie group-theoretic tools have been used to analyze the property of steerability and to categorize the classes of functions which are steerable under different transformation groups. In spite of this, given an arbitrary function, there are several ways of computing a set of basis functions to steer this function. For example, Perona [Perona 1995] suggests sampling the space of transform parameters by constructing replicas of the common kernel, each of which is transformed according to a particular choice of parameters. The n -largest principal components of this set are then used to steer the given filter. Alternatively, Simoncelli and Farid [Simoncelli and Farid 1995], Michaelis and Sommer [Michaelis and Sommer 1995] and Hel-Or and Teo [Hel-Or and Teo 1996b] first approximate the function to be steered by a set of functions known to be steerable, and then steers the approximated function by analytically steering the set of steerable functions.

In this paper, we present a computational, group-theoretic approach to steerable functions. The approach is computational in that the theory is constructive and leads directly to a procedural implementation that computes a set of basis functions that steer an arbitrary function. This approach does not require the identification of the steerable function space in advance, and it is computationally feasible even for multi-parameter transformation groups; If the function is steerable with n basis functions under a k -parameter group, then the procedure is guaranteed to terminate in at most $nk + 1$ iterations.

2 Background on Lie Groups

Lie groups are often encountered as families of transformations acting on a function [Cohen 1911].

In this paper, we consider, primarily, the families of transformation groups acting on real-

Group	Generator
$g_{t_x}(\tau) f = f(x - \tau, y)$	$L_{t_x} = -\frac{\partial}{\partial x}$
$g_{s_x}(\tau) f = f(e^{-\tau} x, y)$	$L_{s_x} = -x \frac{\partial}{\partial x}$
$g_r(\tau) f = f(x \cos \tau - y \sin \tau, x \sin \tau + y \cos \tau)$	$L_r = x \frac{\partial}{\partial y} - y \frac{\partial}{\partial x} = -\frac{\partial}{\partial \theta}$
$g_s(\tau) f = f(e^{-\tau} x, e^{-\tau} y)$	$L_s = -x \frac{\partial}{\partial x} - y \frac{\partial}{\partial y} = -r \frac{\partial}{\partial r}$

Table 1: Several examples of one parameter groups and their generators. The transformation groups are (from top down) x-translation, x-scaling, rotation, and uniform scaling. In the rotation and uniform scaling examples, (r, θ) are the polar coordinates of the image.

valued, two-dimensional functions. We assume that these functions are non-zero only within a bounded region and denote them by $f(x, y) : \mathbf{R}^2 \mapsto \mathbf{R}$. We describe each family of transformations by *operators* $\{g(\boldsymbol{\tau})\}$ where $\boldsymbol{\tau} = (\tau_1, \dots, \tau_k) \in \mathbf{R}^k$ are parameters of the transformation. For example, consider the family of one-dimensional translations of a function in the x -direction:

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

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

A family of transformations $\{g(\boldsymbol{\tau})\}$ parameterized by $\boldsymbol{\tau}$ 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.

Lie groups are rich in structure and many properties of the group can be discerned by studying the properties of infinitesimal actions of the group. The infinitesimal actions of a k -parameter group are a set of differential operators $\{L_i \mid i = 1 \dots k\}$, called the *generators* of the group, corresponding to derivatives of the transformation at the identity with respect

to each parameter τ_i in turn; i.e.,

$$\left. \frac{d\hat{f}}{d\tau_i} \right|_{\boldsymbol{\tau}=\mathbf{0}} = L_i \hat{f} \quad \text{where} \quad L_i = \left(\frac{\partial x}{\partial \tau_i} \frac{\partial}{\partial x} + \frac{\partial y}{\partial \tau_i} \frac{\partial}{\partial y} \right) \Big|_{\boldsymbol{\tau}=\mathbf{0}}$$

The k generators provide a basis for the k -dimensional tangent space $\mathcal{G} = \{\tau_1 L_1 + \dots + \tau_k L_k | \boldsymbol{\tau} \in \mathbf{R}^k\}$. There is a correspondence between a k -parameter Lie group and its k -dimensional tangent space in the form of the exponential map:

$$g(\boldsymbol{\tau}) f(x, y) = e^{\tau_1 L_1} \dots e^{\tau_k L_k} f(x, y). \quad (1)$$

The notation $e^{\tau_i L_i}$ represents the series expansion $e^{\tau_i L_i} = I + \tau_i L_i + \frac{1}{2!} \tau_i^2 L_i^2 + \dots$, which is an infinite sum of differential operators [Cohen 1911]. The exponential map generates a group similar to the original group up to a change of parameterization. Examples of common one-parameter groups and their generators are given in Table 1. The generators of multi-parameter groups can be written in terms of the generators of one-parameter groups. More details can be found in [Hel-Or and Teo 1996a].

3 Generator Chains

In this section, we describe our approach to steerable functions in the context of one-parameter transformation groups; the treatment of multi-parameter transformation groups is deferred to the next section. We begin by formalizing the notion of a steerable function with the following definition.

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

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

where $\boldsymbol{\alpha}^T = (\alpha_1, \dots, \alpha_n)$ and $\Phi(x, y) = (\phi_1, \dots, \phi_n)^T$.

The functions α_i are known as the *steering functions* of f associated with the basis $\{\phi_i\}$ and depend solely on the transformation parameters. Without loss of generality, we assume that n is the minimum number of basis functions required and these basis functions are linearly independent.

The choice 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. Furthermore, if n is the minimum number of basis functions required to steer a function f , then any steerable basis of f will require only n basis functions as well. If a particular steerable basis contains $m > n$ basis functions, then $m - n$ of them are necessarily linearly dependent on the rest.

Theorem 1 (minimality of basis functions) : Let $\Phi = (\phi_1, \dots, \phi_n)^T$ 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 $\Omega = (\omega_1, \dots, \omega_m)^T$ of f with respect to G has exactly n linearly independent functions.

Proof 1:

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

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

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

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

Since \mathbf{B} is invertible it is possible to express Ω as a linear combination of Φ :

$$\Omega = \mathbf{B}^{-1}\mathbf{A}\Phi.$$

Now, if Ω 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 Ω are linearly independent. \square

In the following, we describe a method of constructing a basis for a given steerable function f under a transformation G . From Theorem 1, 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 Section 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 $L^2f = L(Lf)$. 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 2 (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.,*

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

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:

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

Using the exponential map given in Equation 1, the series formed by summing all the functions in the sequence is exactly the same as transforming f by an element $g(\tau) \in G$:

$$g(\tau) f = \exp(\tau L) f = \sum_{i=0}^{\infty} \frac{\tau^i}{i!} (L^i f). \quad (2)$$

Thus, the set of functions $\mathcal{C}(L) f$ provides a basis with which f can be steered. From Theorem 1 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 2 (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 denotes the 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 2 : Let the $(m + 1)^{\text{th}}$ 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

```

/* 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  $\cup$  {next_f}
    next_f =  $L$  next_f;
}
return( basis_set );

```

Figure 1: Procedure for computing the basis functions to steer an arbitrary function f under a one-parameter group.

induction where $j = m + 1$ is the base case. Let $L^m f = \sum_{i=0}^{m-1} a_i L^i f$. Then,

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

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 2 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, it follows from Theorem 1 that m must equal n . \square

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 . If the function f is steerable with n basis functions, then the procedure will terminate after $n + 1$ iterations. Figure 1 describes the procedure in pseudo-code. The

procedure is applied to the following two examples.

Example 1 : 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 2 : 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

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

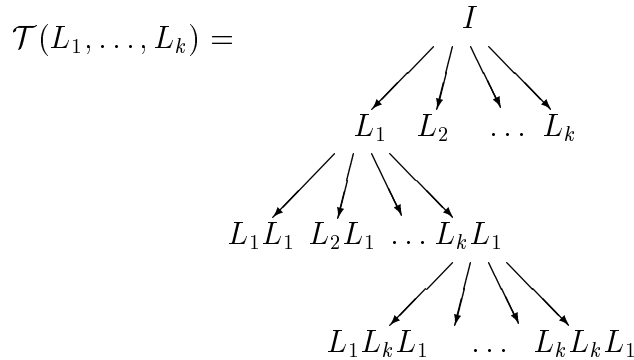
and $L^5 f = 16 \sin 2x + 2 \sin x = -4L^1 f - 5L^3 f$. Therefore, f is steerable under G with the five basis functions, $\{L^0 f, \dots, L^4 f\}$. The particular choice of basis functions is not unique; in this example, the function f is also steerable with the following five basis functions, $\{1,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

4 Generator Trees

In this section, we consider multi-parameter transformation groups. Let the set of differential operators, $\{L_1, \dots, L_k\}$, 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 3 (Generator Tree) : A generator tree $\mathcal{T}(L_1, \dots, L_k)$ is a k -ary tree of differential operators corresponding to repeated applications of the generators L_1, \dots, 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, \dots, L_k repeatedly for a total of l times.

For example,



Similar to generator chains, applying $\mathcal{T}(L_1, \dots, L_k)$ 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 1, transforming f by an element $g(\tau_1, \dots, \tau_k) \in G$ can be calculated by a linear combination of functions in the tree

$\mathcal{T}(L_1, \dots, L_k) f$:

$$g(\tau_1, \dots, \tau_k) f = e^{\tau_1 L_1} \dots e^{\tau_k L_k} f = \left(\prod_{i=1}^k \sum_{l=0}^{\infty} \frac{\tau_i^l}{l!} L_i^l \right) f \quad (3)$$

Thus, the set of functions obtained by applying $\mathcal{T}(L_1, \dots, L_k)$ 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 $\mathcal{T}(L_1, \dots, L_k) f$ with the same root. This property generalizes the minimality property of generator chains associated with one-parameter groups.

Theorem 3 (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, \dots, L_k denote the generators of G . The application of the generator tree $\mathcal{T}(L_1, \dots, L_k)$ to f results in a k -ary tree of functions such that there exists a subtree $\mathcal{T}'(L_1, \dots, L_k) f$ (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 3 : Let $\mathcal{T}'(L_1, \dots, L_k) f$ be a subtree of $\mathcal{T}(L_1, \dots, L_k) f$ (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 2. The induction, in this case, is on subtrees. Also, from Theorem 1 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. \square

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 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 $\{L_1, \dots, L_k\}$ 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. The proof of this claim is given in Appendix A. The procedure is applied to the following two examples.

Example 3 : 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:

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

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 4 : 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 $L_{\tau_2} = -\frac{\partial}{\partial y}$:

$$\mathcal{T}(L_{\tau_1}, L_{\tau_2}) f = \begin{array}{c} \sin x \sin y \\ \swarrow \quad \searrow \\ -\cos x \sin y \quad -\sin x \cos y \\ \searrow \\ \cos x \cos y \end{array} .$$

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.

5 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 for steering polynomials. The second application is a numerical implementation of the same procedure for approximately steering any sampled function.

5.1 Steering Polynomials

The procedure described in Section 4 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 [Hel-Or and Teo 1996b], the authors show 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 determined. This is done by representing each polynomial in terms of the basis of monomials $\{1, x, y, x^2, xy, y^2, \dots\}$. Specifically, let the matrix \mathbf{B} be an $m \times n$ matrix of coefficients and \mathbf{m} be the $n \times 1$ vector of monomials such that $\mathbf{B}\mathbf{m}$ 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 \mathbf{b} of coefficients. Since the monomials are linearly independent, the new polynomial is linearly dependent on the basis set if and only if \mathbf{b} is in the row space of \mathbf{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.

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')
```

$$x^3 + 3x^2y + 3xy^2 + y^3,$$

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

2. In this example, basis functions to steer the polynomial under the group of translations in the x and y directions 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')
```

$$\begin{aligned}
& -15x^3 - 39x^2y + 39xy^2 + 15y^3, \\
& 3x^3 - 15x^2y - 15xy^2 + 3y^3, \\
& 3x^3 + 3x^2y - 3xy^2 - 3y^3, \\
& x^3 + 3x^2y + 3xy^2 + y^3, \\
& -3x^2 - 6xy - 3y^2, \\
& -6x^2 + 6y^2, \\
& 6x + 6y, \\
& 6x - 6y, \\
& 24xy, \\
& -6.
\end{aligned}$$

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 generators of the group).

5.2 Numerical Simulations

A numerical version of the suggested 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. I.e. if ϕ_k is the k^{th} basis function that was generated and orthogonalized, then the residual of the $(k + 1)^{th}$ basis function is: $res(\phi_{k+1}) = \|(I - \Phi_k \Phi_k^T)\phi_{k+1}\|^2$, where Φ_k is a matrix having the k basis functions in a column order: $\Phi_k = [\phi_1, \phi_2, \dots, \phi_k]$. 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

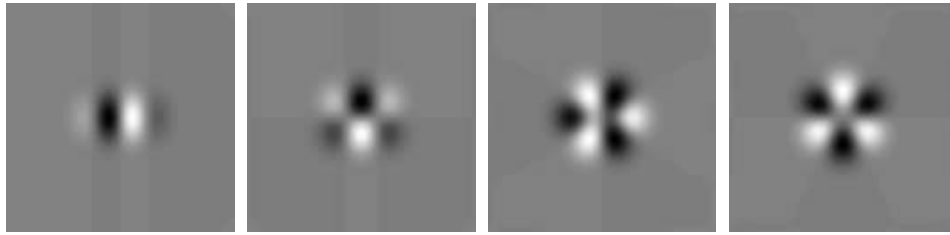


Figure 2: Basis functions that steer $(12x - 8x^3) \exp[-(x^2 + y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure.

see.

Figure 2 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 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 3 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 4 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 (i.e. $res(\phi)/\|\phi\|^2$) 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 5 shows the first 4 basis functions out of 22 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 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

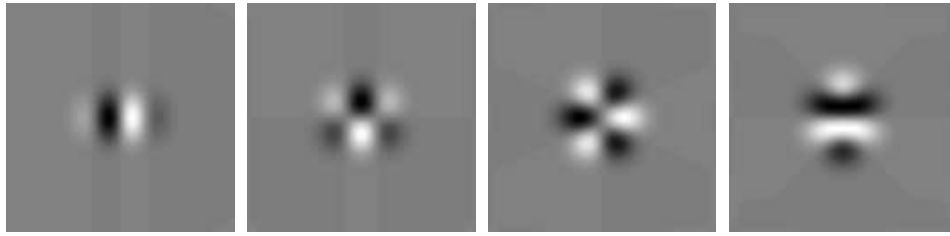


Figure 3: Orthogonal basis functions that steer $(12x - 8x^3) \exp[-(x^2 + y^2)]$, the third derivative of a Gaussian, under rotation. The leftmost image is the third derivative of a Gaussian that was used as input to the procedure.

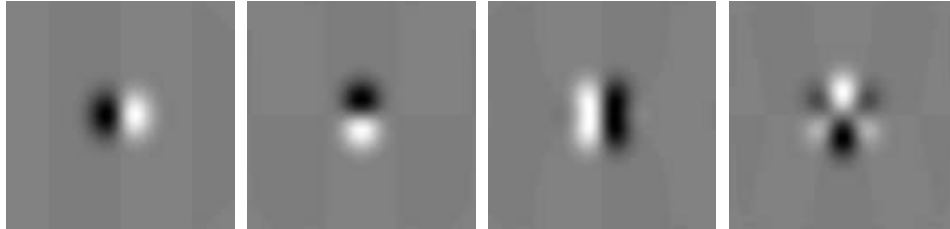


Figure 4: Basis functions that steer $\sin(x) \exp[-(x^2 + y^2)]$ under rotation. The leftmost image is the function that was used as input to the procedure.

squared error of the residual was 10%. Figure 6 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 6 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.

6 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

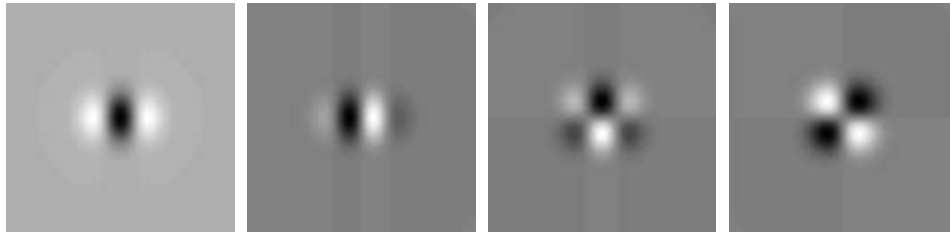


Figure 5: The first four basis functions that steer $(4x^2 - 2) \exp[-(x^2 + y^2)]$ under x, y - translation and rotation. The leftmost image is the function that was used as input to the procedure.

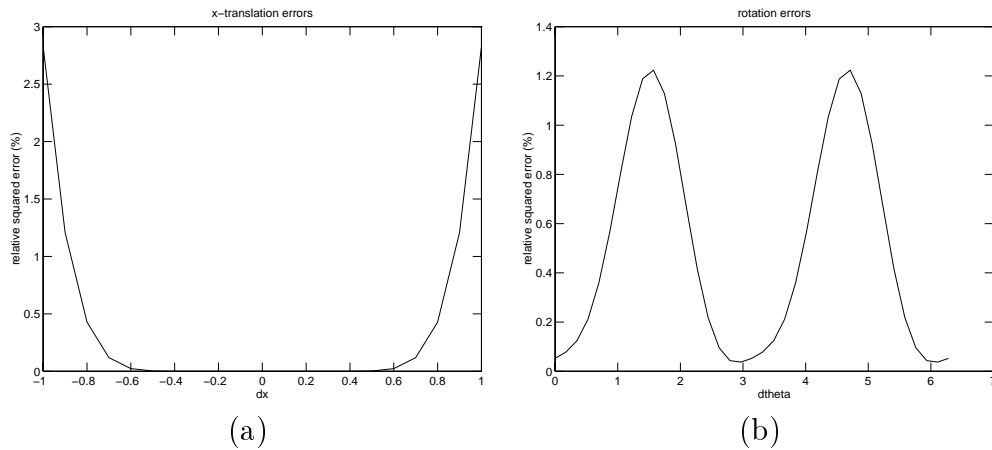


Figure 6: Graph (a): Relative squared errors of the steered approximations to the actual functions over a range of x -translations. The graph describing the errors with respect to the y -translations is virtually identical. Graph (b): Relative squared errors of the steered approximations to the actual 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.

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 2 of Section 3 and Equation 3 of Section 4 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. Furthermore, these higher order

derivatives can also be computed by repeated applications of the generators.

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. It can be verified from the numerical implementation of the procedure in Section 5. However, for some applications, where only a limited range of steering is required, the above approximation may be acceptable. For example, Manmatha [Manmatha 1994] 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, minimize the approximation error about the particular transform parameter (i.e. when we need the best accuracy at the identity transformation). Instead, if the criterion is to minimize the average error over a range of transform parameters, then fewer basis functions are required. In [Teo and Hel-Or], the authors propose an efficient method of computing the basis functions that minimize this approximation error.

In summary, we have presented a computational, group-theoretic approach to computing the basis functions of steerable functions. If the function is steerable with n basis functions under a k -parameter group, the procedure is efficient in that at most $nk + 1$ iterations of the procedure are needed to compute all the basis functions. Furthermore, the procedure is guaranteed to return the minimum number of basis functions. If the function is not steerable, a numerical implementation of the procedure could be used to compute basis functions that

approximately steer the function over a range of parameters.

References

- [Cohen 1911] 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.
- [Freeman and Adelson 1991] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.
- [Fleet 1990] D. Fleet. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.
- [Granlund and Knutsson 1995] G. Granlund and H. Knutsson. *Signal processing for computer vision*. Kluwer Academic Publishers, Boston, 1995.
- [Gotsman 1994] G. Gotsman. Constant-time filtering by singular value decomposition. *Computer Graphics Forum*, 13(2):153–163, 1994.
- [Hel-Or and Teo 1996a] Y. Hel-Or and P. Teo. Canonical decomposition of steerable functions. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 809–816, San Francisco, CA, 1996.
- [Hel-Or and Teo 1996b] Y. Hel-Or and P. Teo. A common framework for steerability, motion estimation and invariant feature detection. Technical Report STAN-CS-TN-96-28, Stanford University, 1996.

- [Manmatha 1994] R. Manmatha. A framework for recovering affine transforms using points, lines or image brightnesses. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 141–146, Seattle, WA, 1994.
- [Michaelis and Sommer 1995] M. Michaelis and G. Sommer. A Lie group-approach to steerable filters. *Pattern Recognition Letters*, 16(11):1165–1174, November, 1995.
- [Nimeroff et. al. 1994] J. Nimeroff and E. Simoncelli and J. Dorsey. Efficient re-rendering of naturally illuminated environments. In *5th Eurographics Workshop on Rendering*, 1994.
- [Perona 1995] Deformable kernels for early vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):488–499, 1995.
- [Simoncelli and Farid 1995] E. Simoncelli and H. Farid. Steerable wedge filters. In *Proc. Int. Conf. on Computer Vision*, pages 189–194, Boston, MA, 1995.
- [Simoncelli et. al. 1992] E. Simoncelli and W. Freeman and E. Adelson and D. Heeger. Shiftable multiscale transforms. *IEEE Trans. Information Theory*, 38(2):587–607, 1992.
- [Teo and Hel-Or] P. Teo and Y. Hel-Or. Design of multi-parameter steerable functions using cascade basis reduction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, to be published.
- [Weng 1993] J. Weng. Image matching using the windowed Fourier phase. *Int. J. Computer Vision*, 11(3):211–236, 1993.

Appendix:

A Number of Iterations

Claim 1 :

The procedure suggested in Section 4 tests the linear dependence of a function on the basis set at most $nk + 1$ times for a function that is steerable with n basis functions under a k -parameter group.

Before proving this claim, we proof the following useful lemma.

Lemma 1 : A k -ary tree with n nodes (internal nodes as well as leaves) has exactly $n(k - 1) + 1$ immediate children.

Proof of Lemma 1: A k -ary tree with one node has k immediate children. Each addition of a new node increases the number of immediate children by $k - 1$. Thus, adding $n - 1$ nodes results in a total number of $(n - 1)(k - 1) + k = n(k - 1) + 1$ immediate children in the tree.

Proof of Claim 1 : 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 $n + n(k - 1) + 1 = nk + 1$. □