ROTATIONAL SYMMETRY: THE LIE GROUP SO(3) AND ITS REPRESENTATIONS

Reiner Lenz
Dept. EE., Linköping University,
S-58183 Linköping, Sweden

Kazuhiro Homma
Mechanical Engineering Laboratory,
Tsukuba-City, Ibaraki 305, Japan

ABSTRACT
The paper describes iterative algorithms to normalize coefficient vectors computed by expanding functions on the unit sphere into a series of surface harmonics. Typical applications of the normalization procedure are the matching of different three-dimensional images, orientation estimations in low-level image processing or robotics. The method uses general methods from the theory of Lie-groups and Lie-algebras to linearize the non-linear original problem and can therefore also be adapted to applications involving groups different from the group of three-dimensional rotations. The performance of the algorithm is illustrated with a few experiments involving random coefficient vectors.

1. GROUP THEORETICAL MATCHING PROBLEMS

In many image processing and pattern recognition applications it is important to solve the following matching problem: Given are two functions $o_1, o_2$ and a set of transformations $\{R_i : i \in I\}$ we want to find the index $k \in I$ such that $R_k o_1$ is as similar to $o_2$ as possible. The case where the $o_i$ are images of the same object produced under different conditions is an obvious example of this problem. Another application is encountered in low-level image processing where the function $o_1$ may represent a prototype pattern (like an edge in a standard position) and $o_2$ is the actual pattern that we want to analyse. In this case we may want to compensate for the possible difference in orientation (i.e. compute the best rotation $R_k$) and then compute the structural similarity between the two patterns.

In the most general case with arbitrary transformations $R_i$ there is only one solution to the problem: to compute $R_{k_1}$ and $R_{k_2}$ for all $i$. Therefore it is necessary to make some assumptions about the transformations $R_i$. Here we assume that the transformations $R_i$ form a Lie-group depending on a finite number of parameters. The basic ideas described later on can be used for all $n$-parameter Lie-groups but for clarity of exposition we will restrict us to the simplest, non-trivial example: the group $SO(3)$ of three-dimensional rotations. In this case the problem is to find out if the two patterns $o_1$ and $o_2$ are similar up to a change in three-dimensional orientation. We will also restrict the object functions $o_k$ to be linear combinations of surface harmonics of a fixed degree. This choice is motivated by the property of the surface harmonics that every function on the unit sphere can be developed in a series of surface harmonics and that the set of linear combinations of surface harmonics of a fixed degree is closed under the action of 3-D rotations.

Even this restricted case has a number of interesting applications. The surface harmonics of degree one can be used as 3-D edge-detection filters. Second and third order harmonics can be applied for detection of other structures such as 3-D planes, lines and corners. The problem of recovering 3-D orientation from surface harmonics was previously considered in [1, 2, 3, 4, 5, 6].

An investigation of the general problem of recovering the 3-D orientation parameters from second order filters can be found in [4, 5]. There a procedure is described which simplifies the incoming filter vector iteratively. The iteration procedure is based on the Euler-angle description and it is limited to second order derivatives. In [6] the expansion of a function in surface harmonics is used for 3-D image registration. A similar direct approach is described in [7]. All these approaches are limited to first and second order surface harmonics.

Below we will use the matrix exponential description to linearise the problem and the method developed below can be easily generalized to higher order surface harmonics and other transformation groups.

2. MATHEMATICAL BACKGROUND

We summarize first some basic facts about surface harmonics and Lie theory necessary to give an intuitive understanding of the basic ideas. For more information about the vast field of Lie-theory and group representations the reader may consult the numerous textbooks on the subject [3, 8, 9, 10].

We recall that a group is a set of elements together with a group operation. If the group multiplication and the operation of inversion are differentiable mappings then the group is called a Lie-group. If a subset of a group together with the multiplication inherited from the original group forms a group then it is called a subgroup. A subgroup of a Lie group which depends on only one parameter is called a one-parameter subgroup. As an example we can take the group $SO(3)$ of all three-dimensional rotations. The elements are $3 \times 3$ matrices which depend on three parameters. One way to parameterize this group is known as the Euler-angle parametrisation:
Theorem 1 Each rotation $R$ is the product of three rotations $R_{\alpha}(\phi)R_{\beta}(\theta)R_{\gamma}(\psi)$ where $R_{\alpha}(\alpha)$ is a rotation around the $x$-axis with angle $\alpha$ and $R_{\beta}(\beta)$ is a rotation around the $z$-axis with angle $\beta$.

It is also well-known that a three-dimensional rotation can be described by its rotation axis and its rotation angle. The description of the axis needs two parameters (the coordinates of a point on the sphere) and the third parameter is the rotation angle. If we keep the rotation axis fixed then we get a subgroup of $SO(3)$ consisting of all rotations around this axis. This is a typical one-parameter group.

For a one-parameter group with elements $R(t)$ such that $R(t_1 + t_2) = R(t_1)R(t_2)$ and $R(0)$ the identity element we define the derivative:

$$X = \lim_{t \to 0} \frac{(R(t) - R(0))}{t}$$

The space of all matrices obtained in this way forms a vector space with dimension equal to the number of parameters of the original group. In the case of $SO(3)$ this is a three-dimensional vector space. The operation of differentiation can be reversed by the exponential map defined as:

$$e^{itX} = \sum_{n=0}^{\infty} \frac{t^n}{n!} X^n$$

Besides the usual vector space operations of addition and scalar multiplication these vector spaces have another multiplication operation, the bracket, defined as $[X, Y] = XY - YX$. Vector spaces with the bracket multiplication are called Lie-algebras. For each Lie-group we can construct its Lie-algebra and for each element in the Lie-algebra we can construct an element in the Lie-group. The correspondence between Lie-groups and Lie-algebras is however not unique: different Lie-groups can generate the same Lie-algebra. Locally around $t = 0$ the exponential mapping is invertible. For the rotation group this construction leads to the exponential description of rotations:

Theorem 2 Define the matrices

$$J_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad J_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

$$J_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Then there are numbers $u_1$, $u_2$ and $u_3$ such that the rotation matrix has the form:

$$R = e^{(u_1 J_1 + u_2 J_2 + u_3 J_3)}.$$

The elements $J_1$, $J_2$ and $J_3$ form a basis of $so(3)$, the Lie-algebra of $SO(3)$. The direction of $u = (u_1, u_2, u_3)$ gives the rotation axis and the length of $u$ is the rotation angle.

Next we introduce the concept of representations of groups and algebras. An $m$-dimensional representation $T$ of a group $G$ is a map from the group to the space of $m \times m$ matrices which preserves the group operation, i.e.:

$$T : R \mapsto T(R), \quad T(R_1R_2) = T(R_1)T(R_2)$$

If $R(t)$ is a one-parameter group and $T$ an representation of dimension $m$ then $T(R(t))$ is a one parameter group of matrices in the representation space. We can therefore compute the derivative $\dot{T} = \lim_{t \to 0} T(R(t)) - T(R(0))/t$. This mapping preserves the Lie-algebra operations: $\dot{T}(X, Y) = [\dot{T}(X), \dot{T}(Y)]$ and we will therefore call it a representation of the Lie-algebra. Again we can locally connect the representations of the algebra and the group by the exponential mapping.

Consider now the matching problem from the introduction with the functions $\alpha_0$ on the unit sphere and 3-D rotations $R$. The surface harmonics $Y_m^l$ form a complete orthonormal system for functions defined on the unit sphere and therefore there are coefficients $c_{l,m}$ such that each of the functions $\alpha_0$ has an expansion:

$$\alpha = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_{l,m} Y_m^l.$$

For a fixed value of $l$ we collect the coefficients in the $(2l+1)$-dimensional vector $c_l$ and the surface harmonics in $Y_l$. The expansion of $\alpha$ is written as:

$$\alpha = \sum_{l=0}^{\infty} c_l Y_l.$$

For $x$ on the unit sphere and $Rx$ its image under the rotation $R$ we define the rotated function $o^R(x)$ by $o^R(x) = \alpha(R^{-1}x)$. Expanding $\alpha$ and $o^R$ gives:

$$o^R = \sum_{l=0}^{\infty} c_l(R) Y_l = \sum_{l=0}^{\infty} T_l(R) c_l Y_l$$

where $T_l(R)$ are $(2l+1)$-dimensional matrices which satisfy the transformation equation: $T_l(R_1R_2) = T_l(R_1)T_l(R_2)$. They define thus a representation of $SO(3)$ in the space of surface harmonics of degree $l$.

We can now formulate the matching problem in terms of representation theory as follows:

Given are two functions $\alpha_1$ and $\alpha_2$ which are linear combinations of surface harmonics of degree $l$. They are thus completely characterized by two coefficient vectors $c_1^{(l)}$ and $c_2^{(l)}$. For two such vectors we want to find the rotation $R$ such that the distance between $T_l(R)c_1^{(l)}$ and $c_2^{(l)}$ is minimized.

Since the form of the representation matrices $T_l(R)$ is rather complicated this is a difficult non-linear problem. For $l = 1$ the solution can be easily found. For the case $l = 2$ it can be shown that the solution can (in principle) be solved by a closed expression of the vector elements. But this direct method requires the solution of a polynomial equation of degree three which is probably not very useful in practice. For larger values of $l$ it is impossible to find closed
form solutions. It is therefore necessary to find iterative techniques to solve these equations. Such methods (based on Euler angles) where for the case \( l = 2 \) developed in [4] and direct methods were investigated in [6, 7]. Here we show how Lie-theory can be used to linearize the problem and to find fast iterative algorithms.

From the basic theory we know that for each representation of a Lie group we can compute a representation of its Lie algebra by differentiation. We compute thus the representation matrices belonging to the three one-parameter subgroups corresponding to the rotations around the three coordinate axis. This gives us three matrices \( D_1, D_2 \), and \( D_3 \) (see equation 9 with \( d_k = \sqrt{l(l+1) - k(k-1)}, \ k = l, l-1, \ldots, -l+1 \) which correspond to the matrices \( J_k \) introduced in (3).

\[
D_1 = \frac{1}{2i} \begin{pmatrix}
0 & d_l & 0 & d_{l-1} & \cdots & d_{-l+1} \\
-d_l & 0 & 0 & \cdots & 0 & d_{-l+1} \\
0 & -d_{l-1} & 0 & \cdots & 0 & d_{-l+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & -d_{-l+1} & 0 \\
0 & 0 & \cdots & 0 & 0 & -d_{-l+1}
\end{pmatrix}
\]

\[
D_2 = \frac{1}{2} \begin{pmatrix}
0 & -d_l & 0 & d_{l-1} & \cdots & d_{-l+1} \\
d_l & 0 & 0 & \cdots & 0 & d_{-l+1} \\
0 & d_{l-1} & 0 & \cdots & 0 & d_{-l+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & d_{-l+1} & 0 \\
0 & 0 & \cdots & 0 & 0 & d_{-l+1}
\end{pmatrix}
\]

\[
D_3 = \frac{1}{i} \begin{pmatrix}
l & 0 & 0 & \cdots & 0 & 0 \\
0 & l-1 & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & -l+1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & -l
\end{pmatrix}
\]

An element of the representation of the Lie-algebra is thus given by a sum: \( D = u_1 D_1 + u_2 D_2 + u_3 D_3 \) and each such element can be exponentialized to a matrix in the Lie-group \( e^D = e^{u_1 D_1 + u_2 D_2 + u_3 D_3} \). Using this exponential form of the representation matrices gives the following approximation of the transformation properties of the coefficient vectors:

\[
T_l(R)c_l = T_l(e^{u_1 D_1 + u_2 D_2 + u_3 D_3})c_l \\
= e^{i(\xi_1 D_1 + \xi_2 D_2 + \xi_3 D_3)}c_l \\
= \approx (u_1 D_1 + u_2 D_2 + u_3 D_3)c_l
\]  

(10)

The non-linear expression \( T_l(R)c_l \) is replaced by the simpler linear combination \( u(D_1 + D_2 + D_3)c_l \). The general idea behind the following algorithms is to find a solution vector \( u = (u_1, \ u_2, \ u_3) \) by using the linear approximation. Then these values are inserted in the exponential and a new approximation is obtained.

As an example how to use this procedure we consider now the low-level image processing problem of normalizing the coefficient vectors \( c_l \) introduced in equations (6), (7). The relevant properties of \( c_2 \) are summarized in the following theorem (see [7]):

**Theorem 3** For each real function \( \sigma = c_2 Y_2 \) there is a rotation \( R \) such that the coefficient vector of the transformed function \( \sigma^R \) is \( (\xi_2, \ 0, \ \xi_0, \ 0, \ \xi_2) \) with \( \xi_0 \geq 0 \) and \( \xi_2 \geq 0 \). Furthermore if \( c_2 = 0 \) and \( c_0 < 0.5 \) then \( c_2 \geq 0.5 \) can be achieved by the rotations \( R_e(\pi/2) \) or \( R_e(\pi/2)R_x(\pi/2) \). \( \xi_2 \geq 0 \) can be achieved by a z-axis rotation.

This theorem shows that the main problem is to find a rotation which transforms the coefficient vector so that for the new vector we have \( c_2 = Im(c_2) = 0 \). For an arbitrary vector \( c_l \) we will therefore try to find a rotation \( R \) such that the transformed vector has entries \( c_l = Im(c_l) = 0 \). In the following we will write \( c_l = \xi_e + i\eta_e \) and only use real functions \( a \) for which \( c_l^a = (-1)^a c_l \).

Using equations (10) the transformed vector \( T_l(R)c_l \) is in a linear approximation given by \( u_1 D_1 + u_2 D_2 + u_3 D_3 c_l \). Using the definition of the \( D_3 \) we get the new values of the parameters as linear functions of the unknowns \( u_3 \).

For the second order case \( c_2 \) we find the equation

\[
\tilde{W} = W + A \cdot u
\]

(11)

where \( u = (u_1, \ u_2, \ u_3) \), and \( W = (\xi_1, \ \eta_1, \ \eta_2) \), \( \tilde{W} \) are the vectors of the optimization variables before and after the linear updating and \( A \) is the matrix:

\[
A = \begin{pmatrix}
-2\eta_2 & -2\xi_2 & 2\eta_1 & -2\eta_1 & 0 \\
-2\xi_2 & -2\eta_2 & 2\xi_0 & -2\xi_0 & 0 \\
2\eta_1 & 2\xi_0 & -\eta_1 & 0 & \eta_1 \\
2\eta_1 & 2\xi_0 & -\eta_1 & 0 & \eta_1 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

(12)

For the third order coefficient vectors the matrix \( A \) is given by

\[
A = \begin{pmatrix}
-\sqrt{3}\xi_2 & -\sqrt{3}\xi_2 & 0 & 0 & 0 \\
\sqrt{3}\xi_2 & 0 & 0 & 0 & 0 \\
0 & \sqrt{3}\xi_2 & -\sqrt{3}\xi_2 & 0 & 0 \\
0 & 0 & \sqrt{3}\xi_2 & -\sqrt{3}\xi_2 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

(13)

### 3. IMPLEMENTATION AND EXPERIMENTS

An implementation of this basic algorithm has to take into account two main problems:

- The basic step is to solve the system of linear equations \( W + A \cdot u = 0 \) but if the matrix \( A \) is singular or ill-conditioned there is no unique solution.

- Since we only compute a linear approximation, described by the parameter vector \( u \), this may lead to solutions which are worse than the original solution.

In our first implementation we did not attempt to solve these problems in the most efficient way but we only used some standard techniques to circumvent them. Faster solutions are certainly possible.

If the matrix \( A \) is nearly singular then we compute the QR-decomposition of \( A \) to search for a good solution. The problem that the original solution is better than the solution provided by the linear approximation is common in all optimization algorithms. We use a line-search in which the length of the original vector \( u \) computed from the linear approximation is gradually reduced to find better solutions.

For second order harmonics we use the post processing steps described in theorem 3 to bring the final result of the iteration process into the standard from described in the theorem.

205
In the first experiments we generated 500 vectors with the help of Euler angles. We first generated the vector \( o = (\xi_2, 0, \xi_0, 0, \xi_2) \) by selecting a random number \( \xi_0 \) in the range \([0.5..1]\). Then we computed \( \xi_2 \geq 0 \) such that the vector \( o \) had unit length. Three random numbers are used as Euler angles for the rotation \( R \) and the input vector \( c_0 = T_1(R)o \) is generated. We then applied the algorithm described above to recover the original vector \( o \) from this input vector.

In this experiment we stopped after at most seven iterations of the basic algorithm. We used seven steps to find a good solution when the matrix \( A \) was ill-conditioned and also at most seven steps to find a good step length in the gradient direction if the linear approximation failed. The algorithm was terminated when the length of the parameter vector \( (\xi_1, \eta_1, \eta_2) \) was less than 0.001. The threshold for the determinant of the matrix \( A \) (which decided when the QR-decomposition was used) was 0.1. In Figure 1 the sorted distance of the computed output vectors to the original prototype vectors \( o \) is shown.

![Figure 1: Distance to prototype vectors](image)

In the next experiment we generated first 500 seven-dimensional unit vectors \( c_0 \) with (uniformly distributed) random entries. These were then processed by the algorithm with the following parameter setting: determinantal threshold 0.001, final threshold 0.001, at most seven loops per input vector and seven loops for searching along the gradient and in the singularity case. The sorted lengths of the vectors \( (\xi_1, \eta_1, \eta_2) \) is shown in Figure 2.

![Figure 2: Length of parameter vector](image)

A summary of the convergence speed of the algorithm is given in Table 3, it shows the number of input vectors for which the norm of the parameter vector was below the threshold after a given number of iterations.

<table>
<thead>
<tr>
<th>Loop</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd order</td>
<td>0</td>
<td>7</td>
<td>198</td>
<td>413</td>
<td>478</td>
<td>497</td>
</tr>
<tr>
<td>3rd order</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>174</td>
<td>313</td>
<td>404</td>
</tr>
</tbody>
</table>

Table 1: Stopping times

Without going into details we want to make a few final remarks:

- In the experiments we only reduced the incoming vectors to a simple form by applying a class of transformations.
- In many cases it is also of interest to know which rotation brought the input vector into its final form.
- This rotation matrix can be easily computed since the algebras spanned by the matrices \( D_k \) and \( J_k \) are the same from an algebraical point.

From an implementation point of view an expansive step in the algorithm is the computation of the matrix exponential \( e^{\xi_1 D_1 + \eta_2 D_2 + \eta_2 D_3} \). This infinite sum of matrix products is usually computed by diagonalizing the matrix \( u_1 D_1 + u_2 D_2 + u_3 D_3 \). In this special case the computations can however be simplified by using well-known properties of the matrices \( D_k \) and \( J_k \).

4. REFERENCES