Iterative Estimation of Rotation and Translation using the Quaternion

Mark D. Wheeler Katsushi Ikeuchi

December 10, 1995

CMU-CS-95-215

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we consider some practical issues when using the quaternion to represent rotation in conjunction with gradient- or Jacobian-based search algorithms. Iterative estimation techniques often incorporate gradient or Jacobian information to simultaneously solve for the parameters with respect to many non-linear constraints. We derive a simple form for the Jacobian of the rotation matrix with respect to its quaternion parameters and then use this form to predict some practical numerical problems of using gradient and Jacobian information for iterative search. These problems can be eliminated by some straightforward steps which we describe.

This research has been supported in part by the Advanced Research Projects Agency under the Department of the Army, Army Research Office grant number DAAH04-94-G-0006, and in part by the Office of Naval Research grant number N00014-93-1-1220. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Department of the Army, the Department of the Navy or the United State Government.

Keywords: computer vision, pose estimation, rotation and translation, quaternions, Jacobian, gradient, conditioning

1. Introduction

The need to estimate rotation and translation parameters, commonly referred to as pose estimation, is ubiquitous in the field of computer vision. Solving for rotation by itself is difficult and is exacerbated when coupled with translations. In some situations, a closed form solution for the optimal rotation and translation is available. In many problems, non-linearities and constraints often make it impossible to find a closed form solution. For these situations, iterative solutions are often required. These iterative solutions often use gradient or Jacobian information to make the search efficient.

In this paper, we consider some practical issues when estimating rotation and translation using gradient- and Jacobian-based search algorithms. Quaternions have proved a valuable representation for estimating and manipulating rotations [FH86, Hor87]; their use for such problems is motivated in Section 2. Quaternions and their relevant properties are described in Section 3. In Section 4, we derive a particularly simple form of the Jacobian of rotation with respect to its quaternion parameters. This form is fundamental to rotation, some reasons for which are discussed in Section 5. The form of the Jacobian allows us to predict some numerical problems when using the gradient and Jacobian of rotation in practice. These problems and their solutions are described in Section 6.

2. Why Quaternions?

A common problem in computer vision is solving for rigid body motions or poses consisting of a rotation and translation in 3D space. For example[1], given a set of points $\boldsymbol{x}_i$ and correspondences $\boldsymbol{p}_i$, it is often of interest to compute the 3x3 rotation matrix R and 3-vector translation $\boldsymbol{t}$ such that

$$R\boldsymbol{x}_i + \boldsymbol{t} = \boldsymbol{p}_i. \tag{1}$$

Although this system of equations is essentially linear, a number of problems arise when formulating solutions that account for the non-linear constraints on the components of R. The constraints arise from using nine values of rotation matrix R to represent three independent variables of 3D rotation. The rotation matrix is constrained to be orthogonal which is satisfied when $R^T R = \boldsymbol{I}$ (i.e., the rows and columns are orthonormal). Also, the rotation must not be a reflection; this is satisfied when the determinant is 1 (i.e., $|R| = 1$).

A number of techniques have been developed to deal with this added complexity. One of the most convenient is the quaternions representation. We will describe quaternions in some detail in what follows, but first we provide the reader a list of some of the advantages and mathematical niceties of the quaternion representation of rotation.

- Maintaining the constraints (orthogonal with unit determinant) of rotation is made simple with quaternions by standard vector normalization.

[1] We will use the convention of showing vector variables in bold and matrices as capital letters.

- Quaternions can be composed/multiplied in a straightforward manner to accumulate the effects of composed rotations.
- The inverse of a quaternion (specifying the inverse rotation) is obtained by simply negating 3 components of the quaternion vector.
- The rotation between two rotations can be computed by multiplying one quaternion with the inverse of the other.
- One can easily transform a quaternion into an axis-and-angle representation. Using this and the previous item, one can compute a rotational distance metric between two rotations—the angle of rotation between them.
- Quaternions can be easily transformed to a 3x3 rotation matrix for efficient computation when rotating vectors.
- It has been shown by Faugeras and Hebert [FH86] and Horn[Hor87] that with the quaternion representation, the rotation can be solved for in closed form when correspondences between three-dimensional point sets are available.

We would like to add to this list that the quaternion representation of rotation has some advantageous differential properties (to be described later). These differential properties combined with the properties of quaternions described above make quaternions particularly well suited to requirements of iterative gradient- or Jacobian-based search for rotation and translation. In this paper, we derive a simple form for the gradient and Jacobian of rotation with respect to quaternions. The form of the Jacobian leads to a straightforward analysis of the problem presented by scale when solving for rotation and translation, and leads to simple steps to ensure scale invariant performance of search algorithms.

3. Quaternions

In this section, we will define the quaternion and its essential properties for representing and algebraicly manipulating rotations. For further details on quaternions the reader is referred to [Ham69, FH86, Hor87, McC90]. The quaternion $\boldsymbol{q}$ is a four vector $[u,\, v,\, w,\;\, s]^T$ which is often considered as a three-vector $\boldsymbol{u} = [u,\, v,\, w]^T$ and a scalar s. We will often refer to $\boldsymbol{q}$ as $[\boldsymbol{u},\, s]^T$ for notational simplicity. The dot product and vector norm for quaternions is defined as usual

$$\boldsymbol{q}_1 \cdot \boldsymbol{q}_2 = \boldsymbol{u}_1 \cdot \boldsymbol{u}_2 + s_1\, s_2$$
$$|\boldsymbol{q}| = (\boldsymbol{q} \cdot \boldsymbol{q})^{-\frac{1}{2}}.$$

Multiplication is defined over quaternions as

$$\boldsymbol{q}_1\, \boldsymbol{q}_2 = [[s_1\, \boldsymbol{u}_2 + s_2\, \boldsymbol{u}_1 + \boldsymbol{u}_1 \times \boldsymbol{u}_2]\,,\;\; s_1\, s_2 - \boldsymbol{u}_1 \cdot \boldsymbol{u}_2]^T\,. \qquad (2)$$

The complex conjugate of a quaternion is defined by negating the vector component and is denoted $\bar{\boldsymbol{q}} = [-\boldsymbol{u},\ s]^T$. The complex conjugate of a unit quaternion, $|\boldsymbol{q}| = 1$, is the inverse of the quaternion with respect to multiplication, i.e.,

$$\boldsymbol{q}\bar{\boldsymbol{q}} = \boldsymbol{q}_I$$

where $\boldsymbol{q}_I = [0,\ 0,\ 0,\ 1]^T$ (we will refer to this often). From Equation 2, one can see that $\boldsymbol{q}\boldsymbol{q}_I = \boldsymbol{q}_I\boldsymbol{q} = \boldsymbol{q}$ which is why we refer to $\boldsymbol{q}_I$ as the identity quaternion.

A unit quaternion $\boldsymbol{q}$ can be used to perform a rigid rotation of a vector $\boldsymbol{x} = [x,\ y,\ z]^T$ by two quaternion multiplications

$$\boldsymbol{x}' = \boldsymbol{q} \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \bar{\boldsymbol{q}},$$

where the scalar component of $\boldsymbol{x}$ is simply set to zero. Observe that quaternion multiplication is not commutative; this is consistent with the fact that general three-dimensional rotations do not commute; however, quaternion multiplication is associative and distributive.

Working from this definition of quaternion rotation, one can derive a formula for the corresponding orthogonal (Euclidean) 3x3 rotation matrix from a unit quaternion

$$R_u(\boldsymbol{q}) = \begin{bmatrix} s^2 + u^2 - v^2 - w^2 & 2\,(u\,v - s\,w) & 2\,(u\,w + s\,v) \\ 2\,(u\,v + s\,w) & s^2 - u^2 + v^2 - w^2 & 2\,(v\,w - s\,u) \\ 2\,(u\,w - s\,v) & 2\,(v\,w + s\,u) & s^2 - u^2 - v^2 + w^2 \end{bmatrix}.$$

We use the subscript u in R_u to denote that this is the rotation matrix when given a unit quaternion. Given an arbitrary quaternion, R_u would no longer be unitary but rather a scaled rotation matrix. The reader can verify that for the identity quaternion (defined above) $R(\boldsymbol{q}_I) = \boldsymbol{I}$, the 3x3 identity matrix.

Finally, we define the relationship between quaternions and the axis-and-angle representation. A unit quaternion $\boldsymbol{q}$ can be straightforwardly interpreted to specify a rotation of angle θ around the unit vector $\hat{\boldsymbol{\omega}}$ using the relations

$$\boldsymbol{u} = \sin\frac{\theta}{2}\,\hat{\boldsymbol{\omega}}$$

$$s = cos\frac{\theta}{2}.$$

This relationship can be derived (see [Hor87]) from Rodrigues' formula for axis-and-angle rotation

$$\boldsymbol{x}' = \cos\theta\,\boldsymbol{x} + \sin\theta\,(\hat{\boldsymbol{\omega}} \times \boldsymbol{x}) + (1 - \cos\theta)(\hat{\boldsymbol{\omega}} \cdot \boldsymbol{x})\hat{\boldsymbol{\omega}} \tag{3}$$

where $\boldsymbol{x}$ is the vector being rotated.

The next section will examine some differential properties of quaternions with respect to rotation.

4. The Jacobian of Rotation With Respect to Quaternions

In this section, we explore a special case which greatly simplifies the form of the Jacobian of rotation with respect to quaternions. This form is later used to predict numerical problems that will occur for iterative search methods utilizing Jacobians and gradients with respect to quaternion rotation parameters.

The rigid transformation of Equation 1 is now a function of $\boldsymbol{q}$ instead of the nine elements of R

$$\boldsymbol{x}' = R(\boldsymbol{q})\boldsymbol{x}_i + \boldsymbol{t}, \tag{4}$$

and the derivatives with respect to $\boldsymbol{q}$ (which is all that we are interested in at the moment) are only a function of the rotation term.

We begin by eliminating the restriction to unit quaternions in our analysis. We can enforce the constraint that the rotation matrix is orthogonal without requiring $\boldsymbol{q}$ to be a unit vector by dividing the matrix by the squared length of the quaternion

$$R(\boldsymbol{q}) = \frac{1}{\boldsymbol{q}\cdot\boldsymbol{q}} R_u(\boldsymbol{q}). \tag{5}$$

This constraint is necessary in general to ensure the Jacobian and gradient accurately reflect the differential properties of a change in the quaternion parameters. As one might guess, the addition of this constraint greatly complicates the form of the Jacobian; however, as we will see shortly, this constraint actually slightly simplifies the Jacobian when evaluated at the identity quaternion, $\boldsymbol{q}_I$.

If we are computing the Jacobian of

$$\boldsymbol{x}' = f(\boldsymbol{q}, \boldsymbol{x}) = R(\boldsymbol{q})\boldsymbol{x}$$

with respect to $\boldsymbol{q}$, things are greatly simplified if we know $\boldsymbol{q} = \boldsymbol{q}_I$. Setting things up to make sure we evaluate all derivatives at $\boldsymbol{q}_I$ is quite painless. Say that quaternion specifying the current rotation of the data is $\boldsymbol{q}_c$. We can easily change coordinate systems so that our current quaternion is $\boldsymbol{q}_I$ by simply premultiplying all of the model points by $R(\boldsymbol{q}_c)$. That is, replace $\boldsymbol{x}$ with $\boldsymbol{x}_c = R(\boldsymbol{q}_c)\boldsymbol{x}$.

By premultiplying the data with the current rotation, we will now solve for a rotation that composes with our current rotation position instead of attempting to solve for the corrections of our current rotation parameters. The premultiplication should not increase the number of computations for most applications since $R(\boldsymbol{q}_c)\boldsymbol{x}$ must be computed anyway to compute error terms. After estimating this rotation, we can easily compute the quaternion of the complete rotation by quaternion multiplication (i.e., $\boldsymbol{q}' = \boldsymbol{q}\,\boldsymbol{q}_c$).

Fundamentally, we have not changed the problem since

$$f(\boldsymbol{q}_I, \boldsymbol{x}_c) = f(\boldsymbol{q}_c, \boldsymbol{x}),$$

and we can still represent the same set of rotations. However, the form for the Jacobian is much simpler when evaluated at $\boldsymbol{q}_I$ as we will see.

We now derive the Jacobian matrix $\frac{\delta}{\delta \boldsymbol{q}}(f(\boldsymbol{q}, \boldsymbol{x}))$ at $\boldsymbol{q}_I$

$$\left.\frac{\delta f}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} = \left.\frac{\delta R}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \boldsymbol{x}.$$

Using $z(\boldsymbol{q}) = \frac{1}{\boldsymbol{q} \cdot \boldsymbol{q}}$ and Equations 5, this can be broken up as follows

$$\begin{aligned} \left.\frac{\delta R}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} &= z(\boldsymbol{q}_I) \left.\frac{\delta R_u}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} + \left.\frac{\delta z}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} R_u(\boldsymbol{q}_I) & (6) \\ &= \left.\frac{\delta R_u}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} + \left.\frac{\delta z}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \boldsymbol{I}, & (7) \end{aligned}$$

using $R(\boldsymbol{q}_I) = \boldsymbol{I}$ and $z(\boldsymbol{q}_I) = 1$.

The first term can be computed by realizing that only the components in $R_u(\boldsymbol{q})$ with a factor of s in it will have a value in the derivative evaluated at $\boldsymbol{q}_I$

$$\begin{aligned} \left.\frac{\delta R_u}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} &= \left[\quad \left.\frac{\delta R_u}{\delta u}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \quad \left.\frac{\delta R_u}{\delta v}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \quad \left.\frac{\delta R_u}{\delta w}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \quad \left.\frac{\delta R_u}{\delta s}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \quad \right] \\ &= \left[\quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \right]. \end{aligned}$$

The second term is easily found to be

$$\left.\frac{\delta z}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \boldsymbol{I} = -2z(\boldsymbol{q}_I)\, \boldsymbol{q}_I\, \boldsymbol{I} = -2\, \boldsymbol{q}_I\, \boldsymbol{I} = [0,\ 0,\ 0,\ -2]\ \boldsymbol{I}.$$

Summing up the terms from Equation 7, we see that the term $\left.\frac{\delta R}{\delta s}\right|_{\boldsymbol{q}=\boldsymbol{q}_I}$ disappears (which is not true at other points in general). The normalization factor $z(\boldsymbol{q})$ effectively cancels out any gain in f from increasing or decreasing the s component at $\boldsymbol{q} = \boldsymbol{q}_I$; thus, the gradient of normalized rotation accurately reflects the effect of changes in the parameters. Since the Jacobian evaluated at $\boldsymbol{q}_I$ is nonzero only in the u, v and w components, we only have three rotation parameters to estimate.

We can now return to the original gradient equation and multiply through to get

$$\begin{aligned} \left.\frac{\delta f}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} &= \left.\frac{\delta R}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \boldsymbol{x} \\ &= \begin{bmatrix} 0 & 2z & -2y & 0 \\ -2z & 0 & 2x & 0 \\ 2y & -2x & 0 & 0 \end{bmatrix} \end{aligned}$$

where $\boldsymbol{x} = [x,\, y,\, z]^T$. Ignoring the last column of the Jacobian, the result will be familiar to most readers as -2 times the skew-symmetric matrix of $\boldsymbol{x}$. For example, the cross product of two vectors can be expressed as a matrix-vector multiplication

$$\boldsymbol{x} \times \boldsymbol{a} = C(\boldsymbol{x})\boldsymbol{a} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \boldsymbol{a},$$

where we refer to $C(\boldsymbol{x})$ as the skew-symmetric matrix of $\boldsymbol{x}$. Notice that, by the skew-symmetry of $C(\boldsymbol{x})$, $C^T = -C$. Thus, we have a very simple form for the Jacobian of our function at $\boldsymbol{q}_I$

$$\left.\frac{\delta f}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} = \left.\frac{\delta R}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} \boldsymbol{x} = 2C(\boldsymbol{x})^T. \tag{8}$$

The above derivation enables us to trivially and efficiently compute the gradient/Jacobian of any rotation with respect to quaternion parameters. The computation of these entities at quaternions other than $\boldsymbol{q}_I$ requires many more operations and involves a term for the s component as well.

Starting the search at $\boldsymbol{q}_I$ has some other advantages for gradient-based searches of rotation. Consider performing a line minimization in the (negative) gradient direction. Remember the gradient direction with respect to $\boldsymbol{q}$ will have no s component and will be of the form

$$\boldsymbol{dq} = -\left.\frac{\delta f}{\delta \boldsymbol{q}}\right|_{\boldsymbol{q}=\boldsymbol{q}_I} = [du,\, dv,\, dw,\, 0]^T.$$

and each step of the each successive rotation in this direction will be of the form

$$\boldsymbol{q}' = \boldsymbol{q}_I + \lambda \boldsymbol{dq} = [\lambda \boldsymbol{du},\, 1]^T \tag{9}$$

where $\boldsymbol{du} = [du,\, dv,\, dw]$. Remember that $\boldsymbol{q}'$ does not need to be normalized (we did that in Equation 5). This means that the rotation axis of our corresponding line search is constant while the angle

$$\theta = 2\cos^{-1}\left(\frac{1}{\lambda^2 + 1}\right) \tag{10}$$

increases with λ (assume $|\boldsymbol{dq}| = 1$). If the gradient is evaluated at $\boldsymbol{q} \neq \boldsymbol{q}_I$, we get a gradient with $s \neq 0$ and our steps in the search are now of the form

$$\boldsymbol{q}(\lambda) = \boldsymbol{q}_c + \lambda \boldsymbol{dq} = [\boldsymbol{u}_c + \lambda \boldsymbol{du},\;\; s_c + \lambda ds]^T. \tag{11}$$

It is interesting to note that searching across an arbitrary line in the 4D quaternion space is equivalent to searching along θ while rotating around a fixed rotation axis. Details of this relationship are presented in the Appendix.

Also note that θ is not a linear function of λ Equations 10 and 18. It is very close to a linear relationship for small angles and reasonably so for $\theta \leq 90°$. Beyond this linear region the search may run into trouble. The gradient search can compensate for this by linearly increasing the value of θ directly, but this should rarely be necessary.

As we have shown above, the quaternions possess some advantageous differential properties which make it amenable to gradient- and Jacobian-based iterative search. Since they are easily composable, we can maintain a single rotation estimate while simplifying the analysis by premultiplying our data with the current rotation estimate. The Jacobian can be computed very efficiently, and line searches in quaternion space effectively equate to linearly increasing the rotation angle (for all practical purposes) about a fixed rotation axis.

5. Rotation and the Cross Product

We showed that the Jacobian of rotation with respect to a quaternion has a particularly simple form (a cross product) when evaluated at the identity quaternion. In fact, the cross product is fundamental to rigid rotation. The differential equation [MLS94] describing a point rotating around an axis $\boldsymbol{\omega}$ at a constant angular speed is

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\omega} \times \boldsymbol{x}(t). \tag{12}$$

We can easily replace t for θ in this equation since the rotation rate is assumed to be a constant angle per unit time. Thus, we arrive at the derivative of the position with respect to rotation angle as a cross product of the axis of rotation with the point vector.

In our derivation, Equations 9 and 10 show that the search along the line $\boldsymbol{q}(\lambda)$ is equivalent to increasing the rotation angle θ around axis $\boldsymbol{du}$. Thus, we would expect that the Jacobian of rotation with respect to the axis parameters would be the transpose of the skew-symmetric matrix of $\boldsymbol{x}$ or $C(\boldsymbol{x})^T$. A related fact is that the tangent operator [McC90] of any rotation matrix is a skew-symmetric matrix (a vector cross product) where the corresponding vector is the axis of rotation. Multiplying a point by the tangent operator of a rotation matrix will give the point's tangent direction of motion under that rotation.

We would expect the same relationship for other rotation representations. The infinitesimal rotation approximation [Gol80, Hor87, KH94] and Rodrigues' formula are in fact derived from Equation 12 [MLS94]. From Rodrigues' formula, the relationship between the Jacobian of rotation and the cross product is readily apparent (note the term $\sin\theta\boldsymbol{\omega} \times \boldsymbol{x}$).

The same cross product relationship is obtained when using the x-y-z Euler angle representation [SI91],

$$R(\theta_x, \theta_y, \theta_z) = R_x(\theta_x)\ R_y(\theta_y)\ R_z(\theta_z)$$

where $R_x(\theta)$ is the rotation of angle θ about the x-axis. When evaluating the Jacobian for zero angles of rotation, the Jacobian reduces to the skew-symmetric matrix of the rotating point. However, accumulating the Euler angles is much more difficult and singularities in their representation must dealt with.

6. Scale Problems for Rotation and Translation Gradients and Jacobians

Any method that relies on gradients of Jacobians of a function with respect to rotation and translation will encounter one or more problems of scale. There are two scaling problems with respect to algorithms using gradients. The first problem is that gradient of rotation and translation parameters is dependent on the scale of the data. The second problem, closely related to the first, is that the scale of the data can cause numerical problems for techniques using the Jacobian matrix with respect to rotation and translation. The third problem is that gradient-based searches can be adversely affected by changes in the scaling of dissimilar variables in the search space as is the case with rotation and translation.

The fundamental problem is that rotation and translation are inherently incomparable. In controls theory, the special Euclidean group ($SE(3)$), the product space of rigid translations and rotations in three dimensions, is known to be non-metric [MLS94]. That is there is no intrinsic scaling of the six dimensional rotation and translation space that forms a strict metric space. This is a fundamental problem, but in practice, task specific knowledge can be used to generate effective solutions. We detail the problems and solutions below.

Because of the fundamental relationship between rotation and the cross product (described in Section 5), the findings of this section are not limited to the quaternion representation of rotation but are applicable for all representations of rotation.

6.1. Scale's Effect on Gradient Directions

A problem affecting both gradient- and Jacobian-based methods is that the amount of change due to rotation and translation is related to the scale of the data. The problem is that a unit of rotation results in a change in the function that is dependent on the scale of the vector being rotated, while a unit translation is independent of the data. Roughly, as the scale increases, the sensitivity of the rotation parameters increases at a rate that is much faster than that of the translation parameters. This is not a good situation for a gradient-based search. The implication is that if we have a given problem and simply scale the data, the algorithm which uses gradients will give two different solutions when rescaled.

From the derivation of the rotational Jacobian in Equation 8, one can quickly derive the gradient of the following example, a typical least squares error function,

$$f = (R(\boldsymbol{q}_c)\,\boldsymbol{x} + \boldsymbol{t} - \boldsymbol{p})^2.$$

Roughly, the rotational gradient is of the form

$$\begin{aligned}\frac{\delta f}{\delta \boldsymbol{q}} &= 4C(\boldsymbol{x}_c)^T(\boldsymbol{x}_c + \boldsymbol{t} - \boldsymbol{p}) \\ &= -4(\boldsymbol{x}_c) \times (\boldsymbol{t} - \boldsymbol{p})\end{aligned}$$

and the translational gradient is of the form

$$\frac{\delta f}{\delta \boldsymbol{t}} = 2(\boldsymbol{x}_c + \boldsymbol{t} - \boldsymbol{p}) \tag{13}$$

where $\boldsymbol{x}_c = R(\boldsymbol{q}_c)\,\boldsymbol{x}$. If the scale of our data, $\boldsymbol{x}$ and $\boldsymbol{p}$, (assume $\boldsymbol{t}$ is zero, or equivalently that it is already subtracted from $\boldsymbol{p}$), is roughly Σ, then

$$\left|\frac{\delta f}{\delta \boldsymbol{q}}\right| \approx \sin\theta\,|\boldsymbol{x}_c|\,|\boldsymbol{p}| \propto \Sigma^2$$

using the relation between the angle between two vectors and the magnitude of their cross product, while

$$\left|\frac{\delta f}{\delta \boldsymbol{t}}\right| \approx |\boldsymbol{x}_c - \boldsymbol{p}| \approx \frac{\sin\theta}{\sin(90-\frac{\theta}{2})}\,|\boldsymbol{x}_c| \propto \Sigma$$

using the law of sines and assuming that $\boldsymbol{x}_c$ and $\boldsymbol{p}$ approximately form an isosceles triangle. Thus, the scale of our data increases, the gradient direction shifts towards a pure rotation, and vice versa as the scale decreases.

When the gradient is a pure translation or pure rotation, gradient-based searches will have poor convergence characteristics. Imagine that the desired pose is a pure translation. The gradient with respect to rotation will dominate, and infinitesimal steps will have to be taken during each line minimization to ensure that progress is made[2]. What would be desirable for most applications is that the solution to a problem at a scale where rotation and translation have roughly the same influence would be the same (modulo scaling of $\boldsymbol{t}$) regardless if the data were rescaled or not. To effect this is not difficult, it simply involves normalizing the data to some canonical frame (say a unit cube). It is a good idea to include the normalization in any algorithm that will deal with objects of different sizes, otherwise algorithm performance may magically get worse when applying it to a new domain.

For gradient-based searches, normalization can be efficiently performed by dividing the rotational gradient by the squared scale and the translation gradient by the scale, and then rescaling the translation when the function is evaluated. Thus, the gradient can be used by the search algorithm (e.g., conjugate gradient search) in the normalized frame while the function evaluation and gradient evaluation work on the unnormalized data. The solution then remains the same at all scales.

6.2. Scale's Effect on the Condition of the Normal Equations

Hartley [Har95] recently showed that the scale of the data can cause numerical problems for Longuet-Higgens' 8-point algorithm for computing the fundamental matrix of two uncalibrated cameras. He showed that by simply normalizing the data before applying the

[2] A seemingly straightforward solution is to iteratively move in pure translations or pure rotations. This may be effective in some cases, but it suffers from the same problems of pure gradient-descent search. For example, if the error is pure translation, a step in rotation that reduces the error will later have to be undone to reach the desired result.

algorithm, the condition number of the solution matrix is greatly improved resulting in reduced error. The same logic applies here when using the Jacobian to perform Newton-Raphson search as in [Low91]. The Jacobian of Equation 4 with respect to $\boldsymbol{q}$ and $\boldsymbol{t}$ has the form

$$J = \left[C(\boldsymbol{x}_c)^T \;\; \boldsymbol{I} \right]$$

where $\boldsymbol{x}_c = R(\boldsymbol{q}_c)\,\boldsymbol{x}$. This gives the normal equations matrix

$$J^T J = \begin{bmatrix} C(\boldsymbol{x}_c)C(\boldsymbol{x}_c)^T & C(\boldsymbol{x}_c) \\ C(\boldsymbol{x}_c)^T & \boldsymbol{I} \end{bmatrix}.$$

As one can see, the upper left block of J will have a magnitude that is proportional to the square of the scale of the data while the lower right hand block remains constant. For example the respective blocks will be proportional to

$$J^T J \propto \begin{bmatrix} \Sigma^2 & \Sigma \\ \Sigma & 1 \end{bmatrix}.$$

Thus, the condition of the matrix will worsen (i.e., the matrix becomes nearly singular) when the scale of the data is not close to one. The result of which is inaccuracies in the resulting solution when inverting the matrix to solve the normal equations. The condition of the normal equations can be greatly improved by simply normalizing the data.

Hartley [Har95] also pointed out that the data should be centered as well so that large offsets in the data do not consume the numerical precision. This is true here as well. In practice only the rotated data, $\boldsymbol{x}$, needs to be centered.

6.3. Scaling of Dissimilar Parameters in Gradient-based Search

The third problem is a fundamental problem of gradient-based search methods [BS70]. The scaling behavior of dissimilar parameters (in our case, rotation and translation in rigid body motion) can wreak havoc on any gradient-based search algorithm. If the effective scales of the parameters are not roughly equal, the function being minimized tends to form a long trough which will make gradient-based/local-search methods inefficient. For example, consider a gradient-descent search [BS70, PFTV91] of a long narrow elliptical trough in two dimensions with the minimum at the center of the ellipse (see Figure 1 (a)). For most points, the gradient points in the direction of the narrow axis. The search will go back and forth down the trough taking many small steps. However, for the case of a perfect circular bowl (see Figure 1 (b)), the gradient always points directly to the minima which may be found in a small number of line searches. The difference between the two functions is simply scaling of the variables. A simple example of the effect that changing scales has on search efficiency is shown in Figure 1. In Figure 1 (a), the objective function is an elliptical bowl, and the gradient descent and conjugate gradient search steps are shown. In contrast, if the objective function variables are appropriately scaled (equivalent to a change in variable $z = x/c$) as in Figure 1 (b), the objective function becomes a perfect circular bowl and both gradient descent and conjugate gradient quickly reach the minimum.

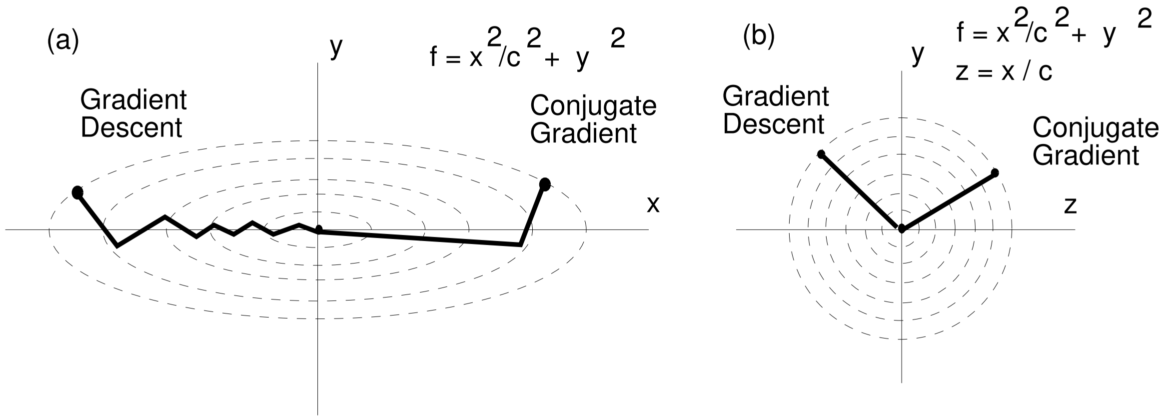

Figure 1: The number of line minimizations of gradient-based methods are dependent on the relative scale of the variables. (a) poorly scaled function f, (b) a simple change of variable, from x to z, improves search efficiency. The optimum value of x and y can be found in one line minimization by both methods.

For other gradient methods such as conjugate-gradient search [BS70, PFTV91] the scaling presents a practical problem. In theory, conjugate-gradient search avoids the problem of zig-zagging down long elliptical troughs, in practice it can become unmanageable to compute. This algorithm requires that the successive gradients are conjugate. This assumes that the minima along each line search are very accurate, especially across narrow sections of the trough where small steps produce large changes in the gradient direction. This is not in general possible without incurring a large number of function evaluations. When the trough is not so elongated, the accuracy requirement is relaxed since small steps do not drastically effect the gradient direction.

For our problem (rotation and translation estimation), the scaling issue is a little more complicated because the two parameters are not linearly related. The problem is that it's difficult to say how much rotation should accompany some amount of translation as you search. The specific data of the problem ultimately determines the shape of the function. However, the parameters can be scaled to change the shape so that it more closely resembles a bowl than a long trough.

For pure quadratic functions, a standard technique for rescaling the variables to form spherical objective functions is called preconditioning. Preconditioning involves computing an approximation of the inverse of the quadratic coefficient matrix of the objective function. The system is premultiplied with this approximation and if successful the condition number of the linear system can be improved (the objective is transformed to be more spherical than elongated).

Preconditioning and scaling the data will generally improve the performance of gradient-based searches. This problem is tolerable in the sense that for a given application, prior knowledge and experience can often be used to determine appropriate scales for the parameters. As we have seen in the previous section, scaling parameters by their relative magnitudes is not sufficient to ensure that the function has shape conducive for efficient gradient-based search. However, the two problems can be solved simultaneously if the data is scaled to a canonical frame such that the parameters effective scales are roughly equivalent in this frame. That is, we can choose one effective scaling of the data so that our function is "bowl shaped" with respect to the scaled data. Preconditioning can also be achieved by sensitivity analysis [Gle94] in which the parameters are scaled by their respective magnitude in the Jacobian. This is roughly equivalent to normalization of the data to a specific scale.

Another related issue is performing line search which usually requires some knowledge on the bounds of the search parameter λ. In object tracking for example [Low91, Gen92], this can be determined a priori from the task at hand by setting some limits on rotational and translational velocity. Knowing these limits, one can efficiently bracket the minima in a given direction of $\boldsymbol{dq}$ and $\boldsymbol{dt}$ and gradient-based search techniques (e.g., conjugate gradient [PFTV91]) can be effectively applied. The minima can be bracketed by choosing λ such that the limits of one of the parameters is reached. This should ensure that the minima can be efficiently found by one of many line minimization algorithms (see [PFTV91] for some examples).

7. Summary and Conclusions

We have described the use of quaternions for performing gradient- and Jacobian-based search of rotation and translation parameters. In this paper, we derive a simple form for the gradient and Jacobian of rotation with respect to quaternions. We used this form to predict numerical problems in gradient- and Jacobian-based searches resulting from the scale of the data.

Quaternions have a number of advantages when representing rotations as noted in Section 2. As we have shown, quaternions also has some advantageous differential properties which make it amenable to gradient- and Jacobian-based iterative search. Namely, they are easily composable, the Jacobian can be computed extremely quickly, and line searches in quaternion space effectively equate to linearly increasing the rotation angle (for all practical purposes) about a fixed rotation axis. One conceptual problem that many have with quaternions is of using 4 parameters to describe rotations which only have 3 degrees of freedom. However, if the data is premultiplied as suggested, the parameters being estimated essentially reduce to 3.

Three scale-related problems with Jacobian/gradient search of rotation and translation are discussed in detail. The first problem is that the scale of the data determines the direction of the gradient. Simply normalizing or transforming the data or gradient to a canonical reference frame can ensure that the algorithm performs properly independent of the scale of the data.

The second problem concerns the use of Jacobians of rotation in the normal equations. We showed that the condition of the normal equations is dependent on the scale of the data; hence, inverting the normal matrix will incur significant numerical errors if the scale of the data is not near unity. This finding is similar to the finding of Hartley [Har95] with respect to Longuet-Higgens' 8-point algorithm.

Finally, even when the data is scaled appropriately, the relative scale of the parameters can greatly affect the convergence characteristics and computational expense of gradient-based searches. Task specific knowledge and experience can be used to determine the appropriate scaling in most cases. The two problems can be solved simultaneously if the canonical reference frame is chosen such that the rotation and translation parameters are roughly of the same scale at unity.

The lesson to be learned is to make sure the data is appropriately scaled before applying search algorithms relying on gradients or Jacobians with respect to rotation and translation.

A Line Searches in Quaternion Space

Equation 5 of Section 4, while greatly complicating the algebraic form of the rotation matrix, allows us to search arbitrary lines[3] in four dimensional quaternion space. We are no longer restricted to the unit sphere. The line search corresponds to searching along an arc of the unit quaternion sphere (the quaternion is implicitly projected onto the unit sphere). It is interesting to note that this quaternion line search is equivalent to searching along θ while rotating around a fixed rotation axis.

For example, if we are searching along the quaternion space line of Equation 11 while increasing λ, we can express the current rotation as a composition of quaternions, i.e.,

$$\boldsymbol{q}(\lambda) = \boldsymbol{q}_c + \lambda \boldsymbol{dq} = \boldsymbol{q}'\boldsymbol{q}_c$$

where $\boldsymbol{q}'$ is the equivalent composing rotation to take $\boldsymbol{q}_c$ to $\boldsymbol{q}(\lambda)$. We can solve for

$$\boldsymbol{q}' = \boldsymbol{q}(\lambda)\bar{\boldsymbol{q}}_c$$

using the inverse property of the conjugate. Since we know the latter two vectors we can then multiply them out and algebraicly simplify them to

$$\begin{aligned}
\boldsymbol{q}' &= (\boldsymbol{q}_c + \lambda \boldsymbol{dq})\bar{\boldsymbol{q}}_c && (14)\\
&= (\boldsymbol{q}_c \bar{\boldsymbol{q}}_c + \lambda \boldsymbol{dq})\bar{\boldsymbol{q}}_c && (15)\\
&= (\boldsymbol{q}_I + \lambda \boldsymbol{dq})\bar{\boldsymbol{q}}_c && (16)\\
&= [\lambda\,[s_c\,\boldsymbol{du} - ds\,\boldsymbol{u}_c + \boldsymbol{u}_c \times \boldsymbol{du}]\,,\ \lambda \boldsymbol{q}_c \cdot \boldsymbol{dq} + 1] && (17)
\end{aligned}$$

assuming $|\boldsymbol{q}_c| = |\boldsymbol{dq}| = 1$. The step size λ does not influence the direction of the axis of rotation but only the angle of the rotation. The angle of rotation can be found to be

$$\theta = 2\cos^{-1}\left(\frac{\lambda\alpha + 1}{\lambda^2 + 2\lambda\alpha + 1}\right) \tag{18}$$

[3] Lines passing through the origin can potentially cause problems, however, since a search along such a line will not produce any change in the rotation angle, it is very unlikely that this singularity will be encountered in practice.

where $\alpha = \boldsymbol{q}_c \cdot \boldsymbol{dq}$ and again assuming $|\boldsymbol{q}_c| = |\boldsymbol{dq}| = 1$. Hence, an arbitrary line search through quaternion space defines a smooth rotation about a fixed axis where the composed rotation quaternion (current quaternion rotated by the search step) axis gradually shifts from $\boldsymbol{u}_c$ to $\boldsymbol{du}$. Equations 17 and 18 are general forms of the results in Equations 9 and 10. However, the simpler forms of Equations 9 and 10 are preferable for implementation and analysis.

Acknowledgments

Thanks to George Paul for helpful comments.

References

[BS70] Gordon S. Beveridge and Robert S. Schechter. *Optimization: Theory and Practice.* McGraw-Hill, 1970.

[FH86] O.D. Faugeras and Martial Hebert. The representation, recognition, and locating of 3-d objects. *Intl. J. Robot. Research*, 5(3):27–52, 1986.

[Gen92] Donald B. Gennery. Visual tracking of known three-dimensional objects. *Intl. J. Comput. Vision*, 7(3):243–270, 1992.

[Gle94] Michael Gleicher. *A differential approach to graphical interaction.* PhD thesis, Carnegie Mellon University, 1994.

[Gol80] Herbert Goldstein. *Classical Mechanics.* Addison-Wesley, 1980.

[Ham69] William R. Hamilton. *Elements of Quaternions.* New York Chelsea Pub. Co., 1969.

[Har95] Richard I. Hartley. In defense of the 8-point algorithm. In *Proc. ICCV*, pages 1064–1070, 1995.

[Hor87] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am.*, 4(4):629–642, 1987.

[KH94] Rakesh Kumar and Allen Hanson. Robust methods for estimating pose and a sensitivity analysis. *Computer Vision, Graphics and Image Processing: Image Understanding*, 60(3):313–42, 1994.

[Low91] David Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. Patt. Anal. Machine Intell.*, 13(5):441–450, 1991.

[McC90] J. Michael McCarthy. *An Introduction to Theoretical Kinematics.* The MIT Press, Cambridge, Massachusetts, 1990.

[MLS94] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC Press, 1994.

[PFTV91] William Press, Brian Flannery, Saul Teukolsky, and William Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1991.

[SI91] Takashi Suehiro and Katsushi Ikeuchi. Towards an assembly plan from observation : fine localization based on face contact constraints. Technical Report CMU-CS-91-168, Carnegie Mellon University, 1991.