Lie Groups in Computer Vision

Hasnain Vohra

30th April 2017

1 Introduction

1.1 Where are Lie Groups used in computer vision?

Lie Groups are used to represent 2D and 3D transformations in computer vision [2, 1]. Most computer vision problems involve minimizing a cost function over a set of variables. A very common example is minimizing the squared distance between a transformation of a set of points and a corresponding set of points. This 2D or 3D transformation is usually one of SO(2), SE(2), Sim(2), SO(3), SE(3) or Sim(3).

Lie Groups are introduced in computer vision to have a coherent framework for dealing with differential spaces of rotation matrices.
A 2D rotation about an angle θ is represented using the rotation matrix

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{1}$$

Even though a 2D rotation is represented with four parameters, it really has just one underlying parameter (θ). If a cost function involves a rotation, how do we differentiate the rotation matrix with respect to θ?

Similarly, a 3D rotation (θ_1, θ_2, θ_3) about the three dimensions (x,y,z) is represented as

$$R(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_1 & -sin\theta 1 \\ 0 & sin\theta_1 & cos\theta_1 \end{bmatrix}; R(\theta_2) = \begin{bmatrix} cos\theta_2 & 0 & sin\theta_2 \\ 0 & 1 & 0 \\ -sin\theta_2 & 0 & cos\theta_2 \end{bmatrix}; R(\theta_3) = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 \\ sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

1.2 What is a Lie Group?

A group is a set of elements with an operation that combines any two group elements to form a third element.

For example, the set of integers ($-\infty$...-1,0,1,...∞) form a group with the operation being addition. Multiplication is also a property of the group of integers.

Another example of a group is the group of all 3x3 matrices, with the operations being addition and multiplication again.

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} + \begin{bmatrix} y_1 & y_2 & y_3 \\ y_4 & y_5 & y_6 \\ y_7 & y_8 & y_9 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \tag{3}$$

A Lie group is speical type of group that is also a smooth differentiable manifold. Multiplication and inversion operations map a group element to another group element, i.e. if you multiply an element of the group with another group element, the result will also lie in the group.

2 Lie Groups and Lie Algebras

Every Lie group has an associated Lie algebra, which is really the tangent space of the Lie group at the identity element. In other words, the Lie algebra is a vector space obtained when a Lie group is differentiated around the identity element.

The Lie algebra space is useful for representing the Jacobian (first order derivative) of a transformation, because it has the same degrees of freedom as the degrees of freedom of the transformation.

The vector space containing a Lie algebra can be represented with a linear combination of the basis vectors (generators). Every Lie algebra can be transformed to its corresponding Lie group using the exponential map and the adjoint property. This group element now can be used for matrix computations, mainly updating a transformation matrix.

3 The Exponential Map

The exponential map maps an element from the Lie algebra space to the corresponding Lie group.
Lie algebra space so(2) has a single generator

$$G = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

The Lie algebra element for so(2) parametrized by θ is

$$\theta_X = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \tag{5}$$

Then, the Lie group element is given by the exponentiaion of the Lie algebra element. The Lie algebra element is exponentiated using Taylor series expansion

$$exp(\theta_x) = e^{\begin{bmatrix}0 & -\theta \\ \theta & 0\end{bmatrix}} \tag{6}$$

$$= I + \theta_X + \frac{1}{2!}\theta_X^2 + \frac{1}{3!}\theta_X^3 + \frac{1}{4!}\theta_X^4 + ... \tag{7}$$

$$= I + \begin{bmatrix}0 & -\theta \\ \theta & 0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\theta^2 & 0 \\ 0 & \theta^2\end{bmatrix} + ... \tag{8}$$

$$= \begin{bmatrix}1 + \frac{\theta^2}{2!} + ... & -\theta + \frac{\theta^3}{3!} + ... \\ \theta - \frac{\theta^3}{3!} + ... & 1 + \frac{\theta^2}{2!} + ...\end{bmatrix} \tag{9}$$

$$= \begin{bmatrix}cos\theta & -sin\theta \\ sin\theta & cos\theta\end{bmatrix} \tag{10}$$

4 The Lie Bracket and the Adjoint

4.1 Lie bracket

The generators of the se(2) Lie algebra are

$$G_1 = \begin{bmatrix}0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} G_2 = \begin{bmatrix}0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} G_3 = \begin{bmatrix}0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0\end{bmatrix} d \tag{11}$$

What would the exponentiation of the Lie algebra elements look like?
Consider the generators for rotation and translation in X direction

$$\delta_1(\theta, t_x) = \theta \begin{bmatrix}0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} + t_x \begin{bmatrix}0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} \tag{12}$$

Exponentiating the above equation

$$exp(\delta_1) = e^{\begin{bmatrix}0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & t_x \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}} \tag{13}$$

$$= e^{\begin{bmatrix}0 & -\theta & 0 \\ \theta & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}} * e^{\begin{bmatrix}0 & 0 & t_x \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}} \tag{14}$$

$$= \begin{bmatrix}cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}1 & 0 & t_x \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix} \tag{15}$$

Now, this is not the same as exponentiating δ_2, where

$$\delta_2 = t_x \begin{bmatrix}0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} + \theta \begin{bmatrix}0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} \tag{16}$$

because,

$$\begin{bmatrix}cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}1 & 0 & t_x \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix} \neq \begin{bmatrix}1 & 0 & t_x \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix} \tag{17}$$

This is because translation and rotation do not commute. If two group elements commute, then the Lie bracket of corresponding Lie algbera elements is a zero element, where the Lie bracket is defined as

$$[A, B] = AB - BA \tag{18}$$

Translation in X direction commutes with translation in Y direction and translation in both directions commutes with isotropic scaling. But none of these commute with rotation.

When two group elements do not commute, the composition of their algebra elements is given by the Baker- Cambell-Hausdorff (BCH) formula

$$\begin{aligned} Z(X,Y) = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}([X,[X,Y]] + [Y,[Y,X]]) \\ - \frac{1}{24}[Y,X[X,Y]] \\ - \frac{1}{720}([Y,[Y,[Y,[Y,X]]]] + [X,[X,[X,[X,Y]]]]) + ... \end{aligned} \tag{19}$$

Now, most problems in the field can be addressed with a rigid transform, one from the family of Sim(3). A closed form solution is available for all the transforms in the family of Sim(3).

But, if higher order transforms are required (i.e. affine or higher), there are no closed form solutions to exponentiating their Lie algebra elements. In that case, the BCH formula can be used to approximate the exponentation of their Lie algebra.

4.2 Adjoint

In the previous cases, we have examined cases of the exponentiation of the Lie algebra space at the identity element. But what if the tangent space is not identity? That is, what if a non-commutative tangent vector has been applied? For example, the element at identity is first translated and then rotated.

This is the case where we need the adjoint operator. The adjoint operator transforms a tangent vector from the tangent space around one element to the tangent space around another.

Suppose a group element B is transformed using the group element transform A. The resulting group element is now AB. We would like to get the Lie algebra element for group B at identity (b). In other words, we need to trnasform the tangent vector AB' to the tangent space at identity.
The adjoint operator is thus given by,

$$Adj_A(b) = AbA^{-1} \tag{20}$$

Consider, the exponentiation of the adjoint applied to the tangent vector, i.e. $e^{AbA^{-1}}$

$$e^{AbA^{-1}} = I + AbA^{-1} + \frac{1}{2!}(AbA^{-1})^2 + \frac{1}{3!}(AbA^{-1})^3 + ... \tag{21}$$
$$= I + AbA^{-1} + \frac{1}{2!}Ab^2A^{-1} + \frac{1}{3!}Ab^3A^{-1} + ... \tag{22}$$
$$= AA^{-1} + AbA^{-1} + \frac{1}{2!}Ab^2A^{-1} + \frac{1}{3!}Ab^3A^{-1} + ... \tag{23}$$
$$= A(1 + b + \frac{1}{2}b^2 + \frac{1}{3}b^3 + ...)A^{-1} \tag{24}$$
$$= Ae^bA^{-1} \tag{25}$$
$$\tag{26}$$

Thus,

$$e^{AbA^{-1}}A = Ae^bA^{-1}A \tag{27}$$
$$= Ae^b \tag{28}$$

The above equation implies that an adjoint action on the group is equivalent to the group of adjoint action on the tangent vector at identity.
The adjoint is very useful in computing the Jacobians in minimization problems.

5 SO(2)

The Lie algebra space so(2) has a single generator

$$G_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{29}$$

An element of SO(2) is represented as a linear combination of the generators

$$\omega \in R \tag{30}$$
$$\omega G \in so(2) \tag{31}$$
$$e^{\omega G} \in SO(2) \tag{32}$$

There exists a closed form solution to $e^{\omega G}$, which is simply

$$\begin{bmatrix} cos\omega & -sin\omega \\ sin\omega & cos\omega \end{bmatrix} \tag{33}$$

6 Sim(2)

The Lie algebra space se(2) has four generators. One for rotation, two for translation and one for isotropic scaling

$$G_1 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} G_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} G_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} G_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{34}$$

The mapping from sim(2) to Sim(2) has a closed form solution.

7 Sim(3)

The Lie algebra space sim(3) has seven generators. Three for rotation, Three for translation and one for isotropic scaling

$$
\begin{aligned}
G_1 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
G_4 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
G_7 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{35}
$$

The mapping from sim(3) to Sim(3) also has a closed form solution.

8 Affine

The Lie algebra space aff(3) has twelve generators. Seven from sim(3) and three for hyperbolic scaling and two for anisotropic scaling.

$$
\begin{aligned}
G_8 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_9 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_{10} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
G_{11} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} G_{11} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{36}
$$

There is no closed form solution for the exponential map from aff(3) to Aff(3).

Given a 12 dimensional tangent vector ω, $\sum \omega_i G_i = \delta$ gives the algebra matrix. The group element can be obtained by exponentiating this matrix using numerical exponentiation e^{δ}. Updating a group element is (U= Te^{δ}). The updated Lie algebra element δ_{new} is obtained by taking a numerical logarithm of the updated group element (U).

On the other hand, the BCH formula can also be used to update a transform. Updating a group element U using a 12-d affine tangent vector ω (or the Algebra matrix δ), the update equation is U = BCH(T, e^{δ}). The updated algebra element δ_{new} is obtained by taking the logarithm of the updated group matrix (U).

9 Jacobians

Consider a cost function that minimizes the distance between a set of n fixed world points (x_w) and a set of points (x) projected using the SO(2) transform R.

$$y = \sum_{i=1}^{n} Rx_i - x_{wi} \tag{37}$$

The overall cost function is the sum of the cost function assocated with edge point correspondence. For now, consider the cost function of just a single correspondence.

$$y = Rx - x_w \tag{38}$$

Differentiation with respect to x is

$$\frac{\partial y}{\partial x} = R \tag{39}$$

Differentiation with respect to a Lie group is performed by left multiplying the group element by the exponentiation of the generator Lie algebra at identity and differentiating the expression around origin.

To understand this more intuitively, draw a comparison with differentation w.r.t. scalars. Differentiation can be treated as finding the slope of a tangent to curve. For, an infinitesimal change in x, the derivative of y w.r.t. x is change in y divided by change in x

$$\dot{y}x = \frac{\Delta y}{\Delta x} \tag{40}$$

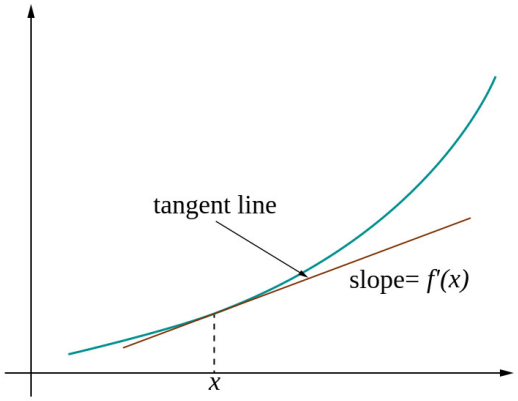

Figure 1: Tangent to a curve. Source: Wikipedia

For a Lie group, what is the equivalent of addition of an infinitesimally small number?
The equivalent of addition is multiplicaiton in Lie groups. The equivalent of small is the exponentiation of inifinitesimally small in Lie algebra.

$$\frac{\partial y}{\partial R} = \frac{\partial (e^{\omega} R)}{\partial \omega_{|\omega=0}} \cdot x \tag{41}$$

$$= \frac{\partial e^{\omega}}{\partial \omega_{|\omega=0}} (R \cdot x) \tag{42}$$

$$= \frac{\partial e^{\omega}}{\partial \omega_{|\omega=0}} (R \cdot x) \tag{43}$$

$$= (G_1|G_2|G_3)(R \cdot x) \tag{44}$$

Differentiaton of a product of Lie groups is a little more involved. Product of two Lie groups transforms the element on the right of the product term to a non-identity space.

The differentiation w.r.t. the term on the left-most side of the product will be as shown above. The differentiation w.r.t. any group element on the right side of another element needs to be transformed back to the origin using the adjoint.

Consider this cost function,

$$y = R_1 \cdot R_0 \tag{45}$$

Then,

$$\frac{\partial y}{\partial R_0} = \frac{\partial (R_1 e^{\omega} R_0)}{\partial \omega_{|\omega=0}} \tag{46}$$

Recall from the definition of adjoint,

$$\frac{\partial y}{\partial R_0} = \frac{\partial (e^{R_1 \omega R_1^{-1}} R_1 \cdot R_0)}{\partial \omega_{|\omega=0}} \tag{47}$$

$$= R_1 \omega_{|\omega=0} R_1^{-1} (e^{R_1 \omega R_1^{-1}} R_1 \cdot R_0) \tag{48}$$

Since all the above quantities are known, the derivative can be calculated w.r.t. all parameters of R_0.

10 Lessons

Care should be taken when dealing with Lie groups of rotation, translation and scaling. Even though translation and scaling components are commonly represented as vectors and scalars respectively, they should be treated as Lie groups. Translation and scaling group elements should never be added, because addition is not a group operation for Lie groups (multiplication is!). Their Lie algebras can however be added or subtracted and that is the correct way of dealing with them.

For a fish-eye or an RPC camera model, two cameras can be related by a transform higher than projective. But a simple pinhole camera model is used in a majority of applications and if the intrinsic parameters of cameras are known beforehand, they are related by a rigid (SE(3)) transform. However, two pinhole cameras are related by an affine transform when there is a change in the intrinsic camera parameters. When and how can a camera's intrinsic parameters change? Think.

11 Acknowledgement

I would like to thank Joseph Mundy and William Stephenson for helping me understand Lie Groups.

References

[1] Ethan Eade. Lie groups for 2d and 3d transformations. http://ethaneade.com/lie.pdf.

[2] Ethan Eade. Lie groups for computer vision. http://ethaneade.com/lie_groups.pdf.