Assignment 3: Motion Estimation using Lie theory

Ali Babaei
Robotics and Computer vision Lab
Ryerson University
Toronto, Canada
ali.babaei@ryerson.ca

***Abstract*—This is a report for last assignment of the course. We simulated 2D and 3D pose estimation problem using formulation based on Lie theory .**

***Index Terms*—pose estimation, Lie Theory, Lie algebra**

I. Introduction

The complex-step derivative approximation is a numerical differentiation technique that can achieve analytical accuracy, to machine precision, with a single function evaluation [1].

II. The complex-step derivative approximation

Generally speaking, algorithmic differentiation can be time consuming to implement and finite-differencing is prone to subtractive cancellation errors, thus limiting precision [1]. The complex-step derivative approximation is a numerical method for computing first derivatives that does not suffer from subtractive cancellation errors [2] . This method has gained popularity due to its ability to realize machine-precision accuracy of derivative computations, and doing so without tuning the step size, since it can be reduced to an arbitrarily small value. The complex-step derivative also requires only one complex function evaluation, which is beneficial compared to central-differencing when the function is expensive to evaluate [2].

III. Why Lie Theory?

We can represent the position of the entire robot by specifying the positions of each joint and the end effector. Euler Angles via roll, pitch and yaw are a popular way to represent the position of a 3-d object but they have 2 major issues:

- Singularities in representation: Let's say the pen is facing 90 degrees upwards, then a sensor at the top of it can't tell the difference between yaw and roll. Motors can burn themselves out, joints can snap and are likely to get you hurt.
- Coordinate dependent: Let's say we have 10 joints, which one of them should be the reference frame. How do you transform the coordinates of joint 9 relative to joint 2? Each new kind of robot would require deriving complex trigonometric expressions and applying them.

Quaternions help alleviate the singularity issue but it still requires a coordinate transform for each joint which is why we'll be looking at an alternate formalism inspired by Lie Groups called Screws and Twist. [3]

A. Rotations form a Group

Rotations have particularly interesting properties, more specifically Rotations form an algebraic group which is just a fancy way to say that rotations can be inverted and that the composition of a bunch of rotations is still a rotation.

A group G is a group if for any two elements of the group a, b the following properties hold:
Closure: $\forall$ a, b $\in$ G $\rightarrow$ a $\circ$ b $\in$ G

Associativity: $\forall$ a, b, c $\in$ G $\rightarrow$ (a $\circ$ b) $\circ$ c = a $\circ$ (b $\circ$ c)

Identity: $\exists$ e $\in$ G s.t $\forall$ a $\in$ G $\rightarrow$ a $\circ$ e = e $\circ$ a = a

Inverse: $\forall$ a $\in$ G, $\exists$ b s.t a $\circ$ b = e

In other words, they mean:

1) Closure: You can compose elements of the group
2) Associativity: Parentheses don't matter
3) Identity: There's a null operator that does nothing
4) Inverse: The action of any element can be reversed

2-d rotations have an additional property called Commutativity a $\circ$ b = b $\circ$ a which means that the order in which you apply 2 different rotations doesn't matter.

B. Lie Group

A Lie Group is a smoothly differentiable Group which means it's an algebraic group in which there are no singularities i.e: holes or points at infinity. This means that no single small change in a joint configuration will cause a large change in the end effector position which means smooth and stable motions.

The group of rotation matrices is called the Special Orthogonal Group SO(n) and it's a Lie Group.

In 3-d SO(3) is equivalent to all 3 × 3 matrices orthogonal matrices with determinant = 1. A matrix is orthogonal if :

$$A^T = A^{-1}$$

so(3) consists of all 3 × 3 skew-symmetric real matrices which is just a fancy way of saying:

$$A^T = -A$$

Now the interesting part is that:

$$A \in so(3) \rightarrow e^A \in SO(3)$$

If SO(3) represents a rotation then so(3) represents a tiny step towards that rotation or more specifically a differentiation. SO(3) is a Lie Group and so(3) is a Lie algebra.

It's important to remember that rotation matrices have distinct applications:

1) Displacement: multiplication by a rotation matrix represents a displacement
2) Representation of position: a rotation matrix by itself represents a position
3) Change of reference frame: a base reference frame can be multiplied by a rotation matrix to get a new reference frame

C. Rigid Body Motion

Lie Groups only let us represent rotations but motions are a combination of translations and rotations so we need to introduce the Special Euclidean Group SE(3) and it's associated Algebra se(3).

To fully describe a rigid body in space (e.g: joint, link) we need a total of 6 numbers, 3 numbers for the x, y, z position and 3 numbers for the rotation yaw, roll, pitch.

So given some starting position p we can represent a rigid body motion to a new position p' via a translation t and a rotation matrix R.

$$p' = pR + t$$

Rigid body motions are also known as the special Euclidean Group SE(3) and can be represented as a matrix where $R \in SO(3)$ and $p \in \mathbb{R}^3$.

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

Transformation matrices form a group, so they can be composed and inverted.

IV. Implementation

We used *kalmanif* library which is a Kalman filter on Lie groups library for state-estimation targeted at robotics applications [4]. It is developed based on *manif* which is a Lie theory library [5].

At the moment, it implements:

- Extended Kalman Filter (EKF)
- Square Root Extended Kalman Filter (SEKF)
- Invariant Extended Kalman Filter (IEKF)
- Unscented Kalman Filter on manifolds (UKFM)

We used the library to simulate a pose estimation problem in 2D and 3D: a Kalman filter for landmark-based localization. They are based on a common setup, explained as follows.

We consider a robot in the plane surrounded by a small number of punctual landmarks or beacons. The robot receives control actions in the form of axial and angular velocities, and is able to measure the location of the beacons with respect to its own reference frame.

The robot pose X is in SE(2) and the beacon positions in R^2. The control signal u is a twist in se(2) comprising longitudinal velocity v and angular velocity w, with no lateral velocity component, integrated over the sampling time dt. The control is corrupted by additive Gaussian noise, with covariance Q. At the arrival of a control u, the robot pose is updated. Landmark measurements are of the range and bearing type, though they are put in Cartesian form for simplicity. Their noise n is zero mean Gaussian, and is specified with a covariances matrix R. We also consider the beacons situated at known positions. We define the pose to estimate as X in SE(2). The estimation error dx and its covariance P are expressed in the tangent space at X. The result of simulation in 2D are shown in figure 1, as well as error plots in figure 2. We also used Invariant

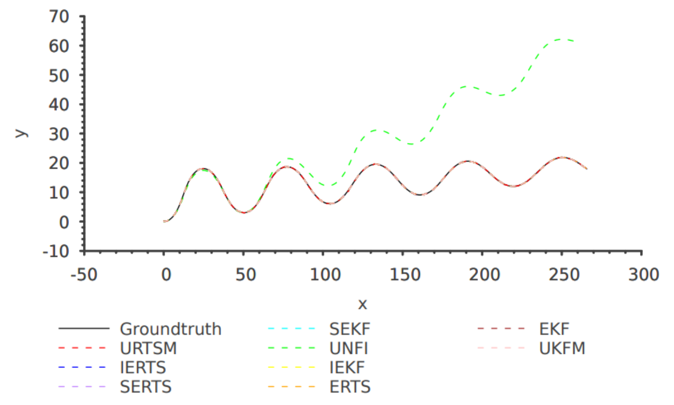

Fig. 1. Estimated pose and Ground Truth

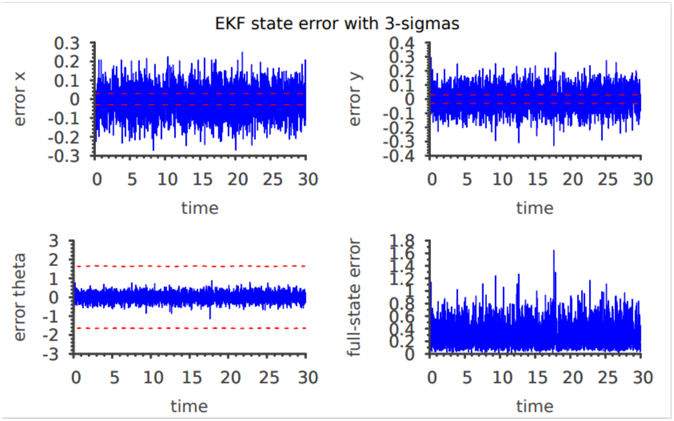

Fig. 2. EKF estate error with 3-sigmas

Extended Kalman Filter (IEKF) in the simulation. the error results are shown in figure 3. To extend the simulation in 3D, we only need to follow the same procedure using elements in SE(3) and corresponding algebra. The setting of the simulation is identical to the 2D simulation. the results of estimated trajectory is shown in figure 4.

The error diagram for estimated pose in x, y and z axis using EKF and IEKF are also shown in figure 5 and figure 6, respectively.

V. Conclusion

The Lie groups and algebra can be applied in a couple of different areas. The first is in the case of camera motion estimation and the establishment of image geometry from said motion. If we have a free motion camera in a surgical setting

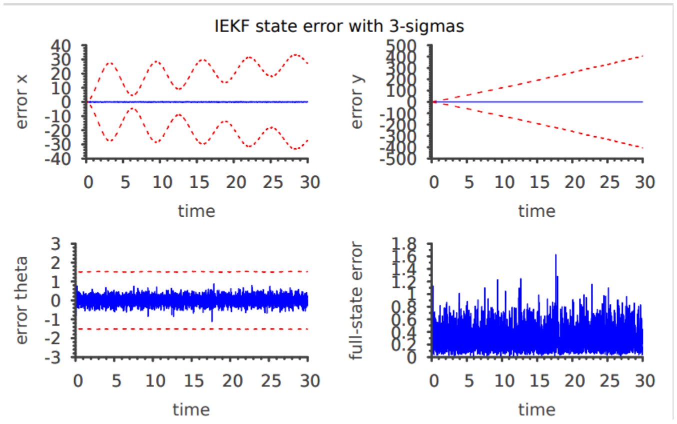

Fig. 3. IEKF estate error with 3-sigmas

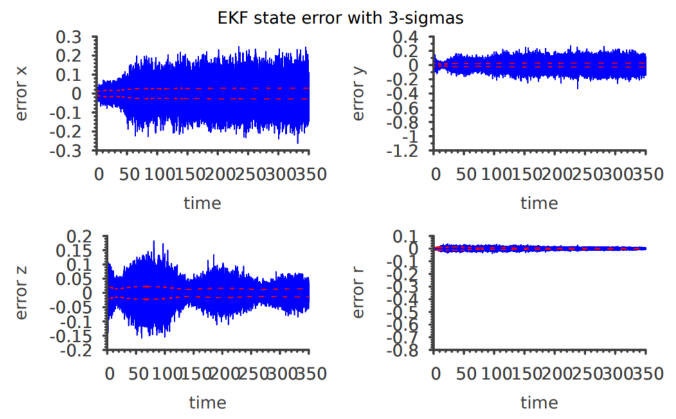

Fig. 5. EKF estate error with 3-sigmas

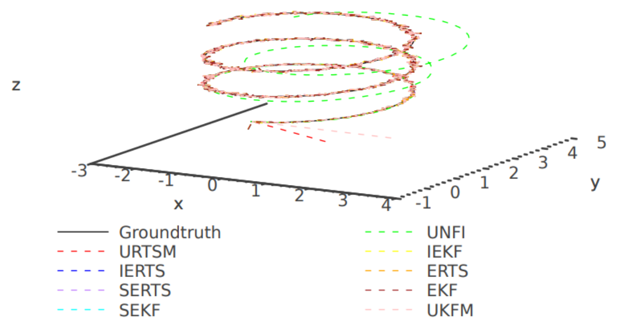

Fig. 4. Estimated pose and Ground Truth

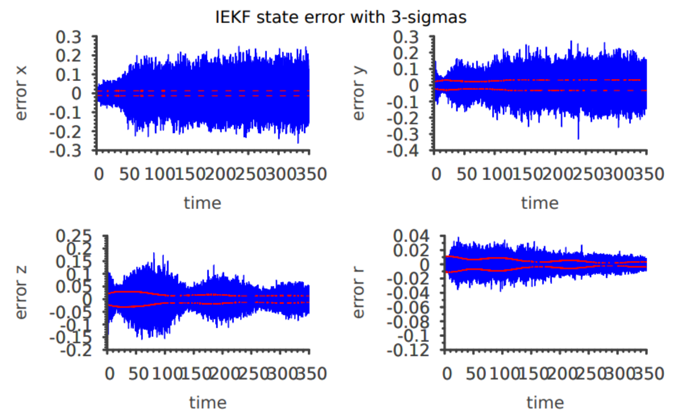

Fig. 6. IEKF estate error with 3-sigmas

or a drone moving around freely, for instance, and we want to make a 3D image of the scene that the camera shot, we need a way to do this. We have to Know the trajectory of the camera in motion Know the location each image was taken from. This is normally modeled with rigid body motion in SE(3), Known as Special Euclidean Transformation. to estimate the trajectory and location, the total motion, of the camera over all images to establish a 3D representation, we'll have to estimate nine parameters for the rotation by itself and that would have to be solved as a constrained optimization problem which is massively tedious and time consuming. Lie groups and algebra gives us a lower-dimensional linear representation for dealing with rigid body motion Infinitesimal rotations around the identity are locally only dependent on the three-dimensional space of symmetric matrices. This builds a tangent space, that is therefore, easier to work with because it's a linear space with three degrees of freedom and no additional constraints. This makes the problem of rigid body estimation and modeling computationally cheaper, and more efficient.

References

[1] C. C. Cossette, A. Walsh, and J. R. Forbes, "The complex-step derivative approximation on matrix lie groups," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 906–913, 2020.

[2] J. R. Martins, P. Sturdza, and J. J. Alonso, "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 3, pp. 245–262, 2003.

[3] M. Saroufim. How to move? lie group robotics. [Online]. Available: https://marksaroufim.medium.com/how-to-move-lie-group-robotics-67fc4f3959d1

[4] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[5] J. Deray and J. Solà, "Manif: A micro Lie theory library for state estimation in robotics applications," *Journal of Open Source Software*, vol. 5, no. 46, p. 1371, 2020. [Online]. Available: https://doi.org/10.21105/joss.01371