16.485: Visual Navigation for Autonomous Vehicles (VNAV) Fall 2021

Lecture 18-19: Optimization on Manifolds

Instructor: Luca Carlone *Scribe: Alan Papalia*

Important: These notes lean heavily on the material on Lie groups and optimization that we discussed in previous lectures. Make sure you review what we previously discussed on Lie groups, Riemannian manifolds, and optimization. We will try to use the "**Recall:** " tag to denote sections where we are drawing on previous lectures but we still recommend refreshing the previous notes.

This lecture centers around general topics in **optimization for state-estimation over Lie groups** and primarily discusses:

- representing probability distributions over Lie groups;
- expressing maximum likelihood estimation problems over Lie groups;
- performing optimization given that our variables belong to Lie groups.

Recall: Optimization is a key building block in modern robotics state estimation; however we point out that the state of a robot is typically not just a vector, but includes rotations or poses, which do not live in Euclidean space. Therefore, it is apparent that the unconstrained optimization techniques we previously discussed, where variables are vectors in $\mathbb{R}^n$, cannot be immediately applied to our state estimation problems. To address problems involving rotations and poses, we have to acknowledge that these geometric objects form Lie Groups, SO(d) and SE(d), respectively, and use the structure these groups provide to express and solve the resulting optimization problems.

Other resources which cover this content in further depth and mathematical rigor include [1, 2]. The contents of these notes were largely drawn from these sources as well as some excellent references listed in Section 19.4.1.

19.1 Living on Manifolds

Again, remember that the key variables of interest in robot state estimation possess Lie Group structure. We will often say that these variables **live on a manifold**, with Lie Groups being special instances of manifolds. With this shift in paradigm (i.e. the "on-manifold" paradigm), comes many key insights about how the state estimation problem can be formulated and solved. We list the following major insights, which we will elaborate on within these notes:

- the variables of interest live on-manifold (e.g. poses $\in$ SE(3));
- the measurements relating our variables will also often be on-manifold (e.g. $\tilde{T}_2^1 \in$ SE(3));
- the noise models for our measurements should adhere to the manifold structure;
- all steps in the (iterative) algorithms we use for optimization should keep the variable estimates on-manifold, to ensure the result is a valid estimate.

We will assume that the reader is at this point comfortable with the first point above. If not, we encourage you to refer back to previous notes and refresh this topic. We review the other three points in the rest of this document.

19.1.1 On-Manifold Measurements and Noise Models

The measurement models we have used in previous lectures were often of the form:

$$\text{measurement} = \text{true value} \; + \; \textit{noise}. \tag{1}$$

It is not immediately obvious how these noise models, e.g., with additive Gaussian noise, can be applied to manifold-valued variables. Gaussian variables are inherently vector-valued, and as such only certain state representations even allow this addition. Even when possible, the addition will almost certainly not result in a valid element on the manifold. For instance, consider the case where we measure an unknown rotation $R_{true} \in \mathrm{SO}(3)$:

$$\tilde{R} \triangleq R_{true} + R_\epsilon \notin \mathrm{SO}(3) \tag{2}$$

where $\tilde{R}$ is the measurement and R_ϵ is the measurement noise.

To circumvent this issue, it is convenient to consider noisy perturbations which are element of the Lie group, and use the closure property of the group to add a noisy perturbation in a way which will guarantee the result is still an element of the Lie group. For instance:

$$\tilde{R} \triangleq R_{true} R_\epsilon \in \mathrm{SO}(3) \tag{3}$$

We review potential instantiations of the model (3) below.

19.1.2 (Wrapped) Gaussian Distributions Over Lie Groups

We know that the Gaussian distribution is a distribution over vectors in Euclidean space, and therefore to use the Gaussian distribution (and many other distributions) we need some linear representation of the domain of our variable. Thankfully, we know something that will exactly do that for us!

Recall: The manifolds we work with are generally matrix Lie groups and matrix Lie groups have associated Lie algebras. These Lie algebras define the tangent spaces of our manifold. Finally, each tangent space is a (local) linear representation around a point on the manifold. Effectively, the Lie algebra provides us a (local) linear domain to express probability distributions over!

We can use the tangent space defined by the Lie algebra to express the Gaussian distribution over and then given a point (i.e., vector) in the tangent space we can then use the exponential map to project that point back onto our manifold. This can be intuitively thought of as taking the Gaussian distribution from the tangent space and then "wrapping" the distribution over the manifold.

For instance, we can define a (wrapped) Gaussian distribution over the group of rotations as:

$$\begin{aligned} \epsilon &\sim \mathcal{N}(\mu, \Sigma) \in \mathbb{R}^3 \\ R_\epsilon &\sim \mathrm{Exp}(\epsilon) \in \mathrm{SO}(3) \end{aligned} \tag{4}$$

and the same can be done for all the Lie groups we have seen in VNAV!

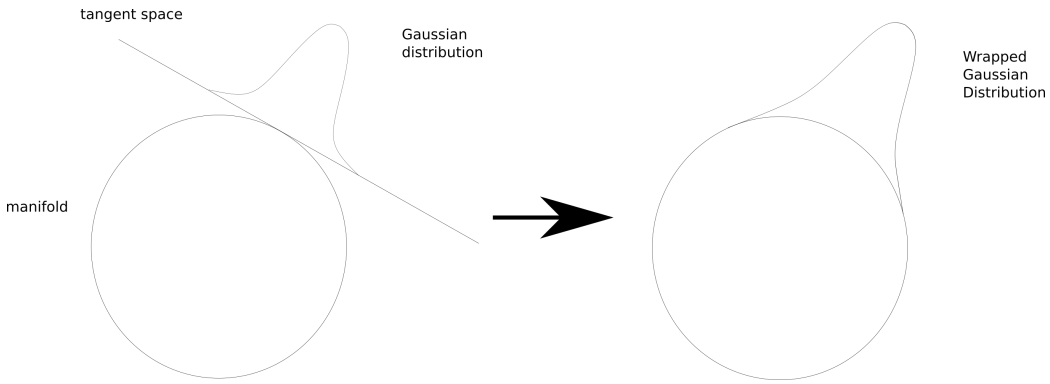

Figure 1: The wrapped Gaussian distribution over the unit circle.

Note: while for low-covariance measurements the wrapped Gaussian approach can suffice, there is a problem in that the exponential map is generally not a one-to-one map. This is because if you travel far enough in the tangent space you will eventually wrap around the entire manifold. As a result, for long-tailed distributions there can be substantial probabilities which are "wrapped" around but likely not properly counted when computing probability.

Note: other distributions can similarly be wrapped over the manifold in just the same manner, but we only present the Gaussian case.

19.1.3 Langevin Distributions Over Rotations

In addition to extending our standard probability distributions to manifolds, we can also define novel probability distributions directly on the Lie groups. In this section, we review the definition of the *Langevin* distribution for the Special Orthogonal group.

A random variable $R_\epsilon \in \mathrm{SO}$ drawn from the Langevin distribution with mode R_μ and concentration parameter κ is described as:

$$\begin{aligned} R_\epsilon &\sim \text{Langevin}(R_\mu, \kappa) \\ \mathbb{P}(R_\epsilon) &\triangleq \tfrac{1}{c(\kappa)} \exp\{\kappa \operatorname{tr}(R_\mu^\top R_\epsilon)\} \end{aligned} \tag{5}$$

We spare the derivation, but refer interested readers to [3] for further details on its derivation and estimation with Langevin distributions.

Note: to simplify the presentation we use the Langevin distribution in which a single concentration parameter, κ, is used but the more general form of the Langevin distribution allows for a full matrix, K to be used [3]. This matrix can be thought of similarly to the inverse of a covariance matrix.

19.2 On-Manifold Maximum Likelihood Estimation

To demonstrate how these probabilities work into a maximum likelihood estimation (MLE) formulation we will focus on a problem known as rotation averaging. This problem is important to a wide array of computer vision and robotic state estimation tasks. Basically, this problem seeks to estimate a single rotation R given a set of noisy observations $\tilde{R}_i$.

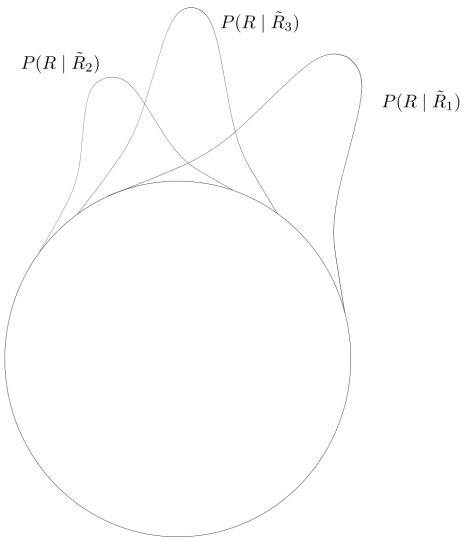

Figure 2: Representation of rotation averaging of three noisy measurements.

Recall that computing the MLE is equivalent to minimizing the negative log-likelihood of the original joint probability distribution:

$$\max_{R\in \mathrm{SO}(d)} \prod_i \mathbb{P}(\tilde{R}_i|R) = \max_{R\in \mathrm{SO}(d)} \sum_i \log(\mathbb{P}(\tilde{R}_i|R)) = \min_{R\in \mathrm{SO}(d)} \sum_i -\log(\mathbb{P}(\tilde{R}_i|R)) \tag{6}$$

Now let's develop the resulting optimization problem, under common assumptions for the probability distributions in (6).

Estimation with Wrapped Gaussian Distribution. In the case of wrapped Gaussian noise models, the MLE can be developed as follows:

$$\min_{R\in \mathrm{SO}(d)} \sum_i -\log(\mathbb{P}(\tilde{R}_i|R)) \tag{7}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma_i)}} \exp\left(-\frac{1}{2}\|\mathrm{Log}(R^\top \tilde{R}_i)\|^2_{\Sigma_i}\right)\right) \tag{8}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma_i)}}\right) + \frac{1}{2}\|\mathrm{Log}(R^\top \tilde{R}_i)\|^2_{\Sigma_i} \tag{9}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i \|\mathrm{Log}(R^\top \tilde{R}_i)\|^2_{\Sigma_i} \tag{10}$$

Estimation with Langevin Distribution. In the case of Langevin noise, the MLE can be developed as follows:

$$\min_{R\in \mathrm{SO}(d)} \sum_i -\log(\mathbb{P}(\tilde{R}_i|R)) \tag{11}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\log\left(\frac{1}{c(\kappa_i)}\exp\{\kappa_i \operatorname{tr}(R^\top \tilde{R}_i)\}\right) \tag{12}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\log\left(\frac{1}{c(\kappa_i)}\right) - \kappa_i \operatorname{tr}(R^\top \tilde{R}_i) \tag{13}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\kappa_i \operatorname{tr}(R^\top \tilde{R}_i) \tag{14}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i -\kappa_i 2d + \kappa_i \operatorname{tr}(R^\top \tilde{R}_i) \tag{15}$$

$$= \min_{R\in \mathrm{SO}(d)} \sum_i \kappa_i \|R - \tilde{R}_i\|_F^2 \tag{16}$$

19.3 Riemannian Optimization

Riemannian optimization consists in performing optimization over Riemannian manifolds, i.e., when the variables of the problem are constrained to remain on a given manifold. In robot state estimation, we primarily consider optimization over Lie groups (a special case of Riemannian optimization).

These notes will attempt to address general Riemannian optimization, but clarify where the properties of Lie groups become useful. While the differences between optimization over matrix Lie groups and general Riemannian manifolds are few, the additional structure of matrix Lie groups can provide substantial benefits through the exponential and logarithm maps.

19.3.1 Advantages of Riemannian Optimization

When we consider optimization problems with variables belonging to Lie groups, it is easy to realize we can simply write the optimization problems as constrained optimization problems. For example, we can write the requirement $R \in \mathrm{SO}(3)$ as the set of constraints $R^\top R = \boldsymbol{I}, \det(R) = +1$. There are general-purpose solvers which can solve such constrained optimization problems, therefore we could perform state estimation with our constraints written this way. Compared to this naive constrained optimization approach, Riemannian optimization reformulates the optimization problems as unconstrained problems, which confers both theoretical and practical advantages.

In terms of practical advantages, unconstrained optimization algorithms are often more efficient and scalable because the natural geometry of the problem is leveraged. In addition, the algorithms and insights from our previous notes on unconstrained optimization can more easily be translated to our manifold optimization setting.

In terms of theoretical advantages, the formulation becomes much more elegant. It can be shown how this form of optimization is a generalization of the optimization techniques covered in our previous notes. Additionally, the lens of Riemannian optimization and manifold structures can allow for more straightforward theoretical results on topics such as convexity and convergence.

19.3.2 Mathematical Terms and Notation

Assume that we have a given Riemannian manifold $\mathcal{M}$, with a tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ defined for every point $\mathbf{x} \in \mathcal{M}$ on the manifold. We will refer to a general vector in the tangent space as $\boldsymbol{\delta} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$.

Similarly, assume we have a *retraction* operator $\mathcal{R}_{\mathbf{x}}\mathcal{M} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$, which maps from a given point in the tangent space back onto the manifold. This can be thought of as taking a point in the tangent space and projecting it back onto the manifold. The *exponential map* we have seen in previous lectures is an example of retraction that can be used when the manifold is also a matrix Lie group. In particular, for a matrix Lie group we can define:

$$\mathcal{R}_{\mathbf{x}}(\boldsymbol{\delta}) = \mathbf{x} \cdot \mathrm{Exp}(\boldsymbol{\delta}) \tag{17}$$

where $\mathrm{Exp}(\cdot)$ is the exponential map for the Lie group.

However, there are many ways to define retractions. In general, a retraction must satisfy two properties: a retraction of the zero element must map back to the manifold point the tangent space was defined at ($\mathcal{R}_{\mathbf{x}}\mathcal{M}(\mathbf{0}) = \mathbf{x}$), and the derivative of the retraction at the zero element must be the identity map. This is effectively saying that if no steps are taken in the tangent space then the retraction operator should map exactly back to where the tangent space came from.

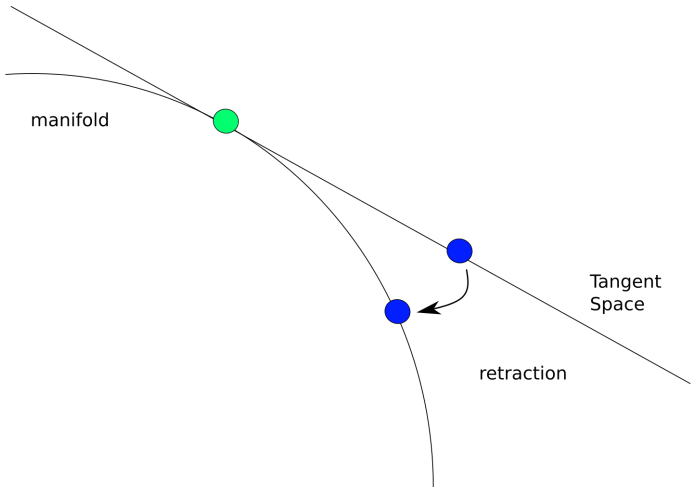

Figure 3: Retracting a point in the tangent space back onto the manifold.

In our earlier notes on unconstrained optimization in Euclidean space, we used ∇f and $\nabla^2 f$ to denote the gradient and Hessian of a function of a variable living in Euclidean space. In contrast, we will use $\mathrm{grad}\, f$ and $\mathrm{Hess}\, f$ to denote the Riemannian gradient and Hessian of a function over a manifold. We will refrain from deriving the Riemannian gradient and Hessian but provide many reference resources on Riemannian optimization and these derivations in Section 19.4.1.

In general, it suffices to know that $\mathrm{grad}\, f$ is the projection of ∇f into the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ and that the Riemannian Hessian is somewhat more complex to derive but bears a similar interpretation. That is, these notions can, at a high level, be thought of as analogs of the Euclidean gradient and Hessian that everyone is already comfortable with, just with extra steps to ensure that $\mathrm{grad}\, f$ and $\mathrm{Hess}\, f$ are entirely within the tangent space.

19.3.3 Iterative Algorithms for Riemannian Optimization

These mathematical definitions are sufficient to give a high-level description of a general Riemannian optimization algorithm. Effectively, the tangent space provides a mathematically elegant way to convert the manifold optimization problem to an unconstrained optimization problem, as in the Euclidean case. Retractions then give us a way to convert the result from tangent space optimization back into an on-manifold value.

Given understanding of the content in Section 19.3.2, the general form of a (local) Riemannian optimization algorithm is fairly straightforward. We write out the general form in Algorithm 1 and attempt to illustrate the steps in Figure 4.

Algorithm 1 General Riemannian Optimization

1: Given initial guess $\mathbf{x}$
2: **while** Convergence Criteria not Satisfied **do**
3: obtain grad f and Hess f at $\mathbf{x}$
4: perform optimization and determine step in tangent space: $\boldsymbol{\delta}^\star \in \mathcal{T}_{\mathbf{x}_k} \mathcal{M}$
5: retract solution back onto manifold: $\mathbf{x} \leftarrow \mathcal{R}_{\mathbf{x}}(\boldsymbol{\delta}^\star)$

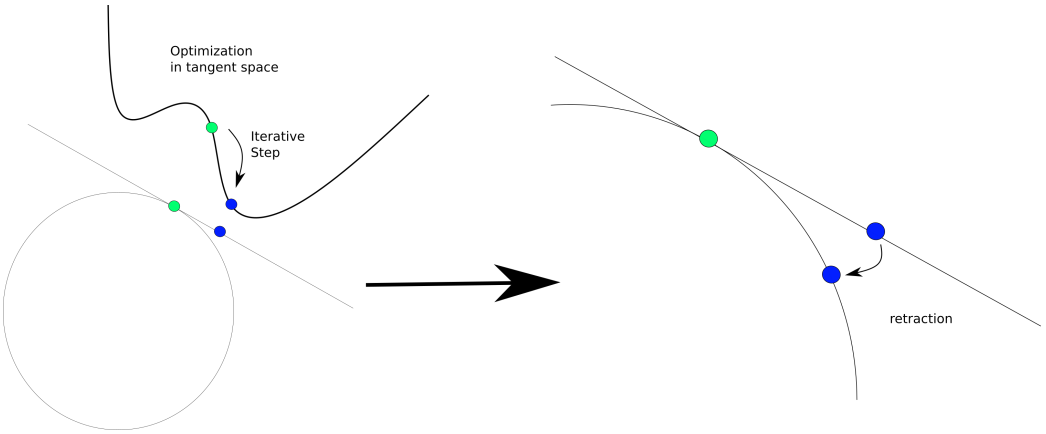

Figure 4: Iterative Riemannian Optimization Algorithm: first performing optimization and deciding a step in the tangent space, and then retracting that step onto the manifold.

Note: At no point does the algorithm explicitly discusses any potential constraints that may be imposed by a given manifold! This is one of the beauties of Riemannian optimization. Because the tangent space is a linear space, optimization in the tangent space does not need to adhere to any constraints. The retraction operation then enforces the constraints of the manifold (e.g. $R^\top R = I, det(R) = 1$). As a result Riemannian optimization enforces constraints at every iteration in a mathematically elegant manner. Particularly, in the case of matrix Lie groups the Lie algebra and exponential maps give principled ways of enforcing the constraints. This can pay huge dividends in terms of algorithmic performance when compared to naive constrained optimization.

19.3.4 A Practical View

Here we provide a more practical description on how to perform optimization on manifold. Consider the following generic optimization problem:

$$\min_{\mathbf{x}\in\mathcal{M}} \sum_{i=1}^{N} \|r_i(\mathbf{x})\|^2 \tag{18}$$

One way to understand iterative optimization on manifold is to imagine that, given an initial guess $\bar{\mathbf{x}} \in \mathcal{M}$, at each iteration, the algorithm performs the following three steps:

1. *Pull-back to the tangent space:* locally reparametrize the optimization as a function of a tangent-space perturbation around the current solution guess $\bar{\mathbf{x}}$:

$$\min_{\boldsymbol{\delta}\in\mathbb{R}^n} \sum_{i=1}^{N} \|r_i(\mathcal{R}_{\bar{\mathbf{x}}}(\boldsymbol{\delta}))\|^2 \tag{19}$$

   Note that this first step reformulated the original (on-manifold) problem into a problem over a vector in Euclidean space! Therefore, in the second step, we can use existing algorithms for Euclidean optimization to compute a suitable step $\boldsymbol{\delta}^\star$.

2. *Compute step in tangent space:* this step simply consists in computing a suitable step $\boldsymbol{\delta}^\star$ by treating (19) as a standard optimization problem in Euclidean space. For instance, we can compute steps using the recipes provided by the three techniques seen in the last lecture (Newton method, Gauss Newton, Levenberg-Marquardt).

3. *Apply retraction:* The final step consists in using $\boldsymbol{\delta}^\star$ to update our solution guess. In particular, we simply use the retraction function and update our guess $\bar{\mathbf{x}}$ as:

$$\bar{\mathbf{x}} \leftarrow \mathcal{R}_{\bar{\mathbf{x}}}(\boldsymbol{\delta}^\star) \tag{20}$$

These steps are iterated until a suitable stopping condition is met. For instance, one can stop when $\| \boldsymbol{\delta}^\star \|$ is below a threshold (i.e., when the updates to our variables become negligible), of when the change between objective function values at two consecutive operations becomes negligible.

19.4 Extra Notes

19.4.1 Resources on Riemannian Optimization

There are excellent resources describing the basics of Riemannian optimization. For a thorough understanding of the mathematical foundations we direct the interested to the following texts [1, 2]. Importantly, the projection operators, Riemannian gradient and Hessian for a number of common manifolds are contained in Chapter 7 of [2]. We also note that a nice, succinct description of the Riemannian gradient and Hessian can be found in Section 4.1.3 of [4].

Beyond these texts, there is also a growing collection of online resources. We attempt to catalog a few which we have found useful in the past, while noting this list is certainly far from complete.

- Blog Post: Optimizing in Smooth Waters by Nicolas Boumal

- Manopt Tutorial
- Youtube Video: Optimization on Manifolds by Nicolas Boumal
- Youtube Video: An Introduction to Optimization on Smooth Manifolds by Nicolas Boumal

References

[1] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. 2009. ISBN: 9780691132983. DOI: 10.1515/9781400830244.

[2] N. Boumal. *An introduction to optimization on smooth manifolds*. Available online. Aug. 2020. URL: http://www.nicolasboumal.net/book.

[3] A. Chiuso, G. Picci, and S. Soatto. "Wide-sense estimation on the special orthogonal group". In: *Communications in Information and Systems* 8.3 (2008), pp. 185–200.

[4] D. Rosen et al. "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group". In: *International Journal of Robotics Research* 38.2-3 (2017), pp. 95–125. ISSN: 17413176. DOI: 10.1177/0278364918784361. arXiv: arXiv:1612.07386v2.