Material for IEEE/ION PLANS 2010

David D. Diel
Scientific Systems Company
Email: ddiel@alum.mit.edu

1 Trajectory Optimization Framework

In addition to providing standardized multi-sensor data, we have created a framework for developing and testing algorithms that aid navigation, where "navigation" is defined as continuous self-localization relative to a local or global frame. This framework supports our own data sets as well as user-supplied data. It serves two major purposes: i) To facilitate the development of new trajectory optimization algorithms with a focus on alternative sensors other than GPS; and ii) To assist system integrators who must test a variety of sensors and algorithms on a level playing field and combine them to achieve desired accuracy. Our approach is to define the broad structure of the navigation problem while leaving the details to be implemented differently for each application.

The framework consists of four interfaces (abstract classes) that divide the navigation problem into manageable components, as illustrated in Figure 1. A `DynamicModel` is an algorithm that relates a set of generic parameters to the rigid-body trajectory of a physical system such as a car or airplane. It also generates "costs", which are discussed later in this section. Sensor data comes into the system through a customizable hardware abstraction class labeled `DataContainer`. Each `Measure` relates sensor data to a trajectory hypothesis and generates additional costs. Finally, at the core of the framework is the `Optimizer`, which is an algorithm that minimizes a combination of costs by intelligently generating and modifying the parameters of one or more `DynamicModel` instances and passing them through zero or more `Measure` instances.

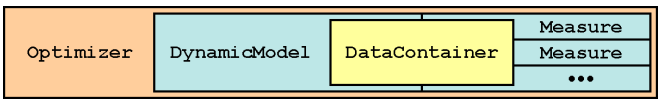

Figure 1: Block diagram of framework components.

A user interacts with the framework by specifying a set of components that derive from the framework classes. These components represent the physical system and define "optimal" for a specific application. The framework then sets up a scalar objective function to be evaluated and optimized. Further details of this approach are provided below, and framework code is available in both MATLAB and C++ online [1].

1.1 Dynamic Model

Consider the following canonical state transition models for continuous and discrete nonlinear systems:

$$\underbrace{\dot{\boldsymbol{x}}_t = \boldsymbol{f}\left(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{v}_t\right)}_{\text{continuous}} \qquad \underbrace{\boldsymbol{x}_{n+1} = \boldsymbol{f}\left(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{v}_n\right)}_{\text{discrete}}. \tag{1}$$

These equations have been established to model the dynamics of numerous physical systems, from Brownian particles to holonomic ground vehicles, fighter jets, and even animals. They are typically placed in an integration loop, such that the state $\boldsymbol{x}$ is computed incrementally at increasing time instants t, given an initial condition. The symbols $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{v}$ all represent vector-valued functions indexed by time. The input $\boldsymbol{u}$ represents data that is known, and the input $\boldsymbol{v}$ represents parameters whose probability distributions are known. If an interpolation method is specified, then both the continuous and discrete models can be written in *functional* form as follows:

$$\boldsymbol{x} = \boldsymbol{F}\left(\boldsymbol{v}; \boldsymbol{u}\right). \tag{2}$$

Our `DynamicModel` class standardizes the interface to $\boldsymbol{F}$. It requires that $\boldsymbol{x}$ contain a C^1 continuous 6-DOF rigid-body trajectory relative to an Earth-Centered Earth-Fixed (ECEF) frame. It accesses the data $\boldsymbol{u}$ implicitly, a detail that we notate using a semicolon (;) in the argument list. It requires that each parameter vector $\boldsymbol{v}_n$ consist of logical and/or integer parameters. And, it defines an indexing system that allows the domain of $\boldsymbol{x}$ to grow with the domain of $\boldsymbol{v}$ in a consistent manner as time moves forward. Finally, assuming that each parameter vector $\boldsymbol{v}_n$ is independently distributed, the `DynamicModel` associates a cost function r with the negative log likelihood of the normalized probability density of $\boldsymbol{v}_n$ as follows

$$c_n = r\left(\boldsymbol{v}_n\right) = -\ln\left(\frac{\mathrm{pdf}\left(\boldsymbol{v}_n\right)}{\left\|\mathrm{pdf}\left(\boldsymbol{v}_n\right)\right\|_\infty}\right). \tag{3}$$

1.2 Measure

Many sensors can be modeled by the canonical form

$$\boldsymbol{y}_n = \boldsymbol{g}\left(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{w}_n\right) \tag{4}$$

where each measurement $\boldsymbol{y}_n$ arises from the instantaneous state $\boldsymbol{x}_n$, the known data $\boldsymbol{u}_n$, and the stochastic vector-valued function $\boldsymbol{w}_n$. However, some measurements of interest cannot be

accurately modeled in this form, because they are not inherently instantaneous. A prime example is a feature match between two images that were acquired at different times. Regardless of whether one uses Optical Flow [4], SIFT [3], normalized cross-correlation [2], or another technique, the computed feature displacement will depend on the position and orientation of the sensor at each time. Another sensor example is a gyroscope, which can be modeled as if it measures instantaneous rotation rates, but is most accurately modeled as measuring changes in orientation over discrete time periods.

One approach to dealing with sensors of the type described above is to create a combining function $\boldsymbol{\psi}_{ab} = \boldsymbol{\gamma}(\boldsymbol{y}_a, \boldsymbol{y}_b)$, where the pair of indices labeled ab indicate a measurement arising from two times, $n = a$ and $n = b$, sorted such that $a \leq b$. This implies an indexing system that has a graph structure instead of a simple linear index. Each discrete time is associated with a node (or vertex), and each node pair forms an edge. In general, not all of the nodes are connected, making it an incomplete graph. This idea can be incorporated into Equation 4 by rewriting it in *functional* form

$$\boldsymbol{y}_{ab} = \boldsymbol{g}(\boldsymbol{x}, ab, \boldsymbol{u}, \boldsymbol{w}) \tag{5}$$

where the sensor accesses its arguments as functions. The functional $\boldsymbol{g}$ can evaluate the body trajectory at any time in its domain, which we assume includes the span $t \in [t_a, t_b]$. Likewise, it can evaluate its other arguments within their valid domains. As a generalization of Equation 4, it can also be used to simulate measurements. However, for the purpose of trajectory optimization, simulated measurements may not be necessary, as long as there is a function that can test a hypothetical trajectory against a set of sensor data. Therefore, we define a cost *functional* s associated with data from each algorithm or sensor m as follows:

$$c_{m,ab} = s_m(\boldsymbol{x}, ab; \boldsymbol{u}) = -\ln\left(\frac{\mathrm{pdf}_m(\boldsymbol{u}|\boldsymbol{x}, ab)}{\|\mathrm{pdf}_m(\boldsymbol{u}|\boldsymbol{x}, ab)\|_\infty}\right). \tag{6}$$

Our `Measure` class standardizes the interface to each s_m without requiring explicit computation of $\boldsymbol{y}$ or $\boldsymbol{w}$. This not only has the potential to reduce processor burden, but it also makes it possible to wrap a wide variety of sensors and algorithms with a uniform interface. For example, suppose s_1 represents a "smart camera" running a sparse feature tracker, and s_2 represents a GPS unit. Since they both posess the same interface, they can be tested and their performance can be compared objectively, a property that is especially helpful to system integrators.

1.3 Optimizer

Once cost functions have been defined, putting together the overall objective is straightforward. All costs are additive by definition, which implies the assumption of independent distributions for each parameter vector in the `DynamicModel` and for each edge contributed by each `Measure`. Therefore, the optimization problem boils down to the composition

$$\begin{aligned}\boldsymbol{v}^* &= \underset{\boldsymbol{v}}{\mathrm{argmin}}\left\{\sum_n c_n + \sum_m \sum_{ab} c_{m,ab}\right\} \\ &= \underset{\boldsymbol{v}}{\mathrm{argmin}}\left\{\sum_n r(\boldsymbol{v}_n) + \sum_m \sum_{ab} s_m(\boldsymbol{F}(\boldsymbol{v}; \boldsymbol{u}), ab; \boldsymbol{u})\right\}\end{aligned} \tag{7}$$

where the solution $\boldsymbol{v}^*$ is an optimal parameter hypothesis. To find the optimal trajectory $\boldsymbol{x}^*$, this needs to be inserted back through the dynamic model as follows:

$$\boldsymbol{x}^* = \boldsymbol{F}(\boldsymbol{v}^*; \boldsymbol{u}). \tag{8}$$

Our `Optimizer` class wraps algorithms that query r and s in order to search for global minima in Equation 7. In general, neither convexity nor uniqueness of solution are guaranteed; however, specific sets of components can be designed to offer these guarantees. To facilitate efficient optimization, the costs can be computed individually before the summation takes place. This formulation suggests automated learning of the relationship between time indexed parameters and time indexed graphs of costs, a subject beyond the scope of this paper. In future work, we plan to demonstrate that our framework generalizes a wide range of trajectory optimization techniques including EKF, UKF, gradient descent, genetic algorithm, simplex, and the pose graph method [5].

References

[1] Scientific Systems Company. Project "functionalnavigation" on google code. `http://code.google.com/p/functionalnavigation/`.

[2] J. P. Lewis. Fast normalized cross-correlation. *Vision Interface*, 1995.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[4] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence*, 1981.

[5] Edwin Olson, John Leonard, and Seth Teller. Fast iterative optimization of pose graphs with poor initial estimates. *ICRA*, pages 2262–2269, 2006.