An introduction to Gaussian Process Motion Planning

Sahit Chintalapudi

03/07/2018

Motion Planning

- *Path planning* answers what has to happen to get from point A to point B, *Motion planning* answers how to do it.
- Find trajectories of robot states that achieve a desired task.
  - Are these trajectories *feasible*?
  - Are these trajectories *optimal*?

Current schools of thought

- Sampling based motion planners
  - Building a tree through the "free space". E.g - RRTs
  - Pros: probabilistically complete
  - Cons: Inefficient in high dimensional spaces. Paths tend to be redundant calling for further optimization.
- Trajectory Optimization
  - Minimize a objective function that encourages feasibility and optimality
  - Pro: can be more computationally efficient
  - Cons: Local minima can lead to sub-optimal trajectories

Gaussian Processes

$$\theta : \mathbb{R} \to S$$

- Let θ be a function that represents our trajectories. It maps time to robot state. We can query the trajectory at *any* time.
- A Gaussian process is a probability distribution over the space of trajectories

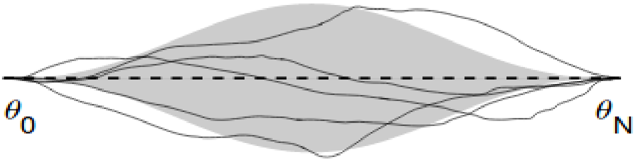

Figure 2. An example GP prior for trajectories. The dashed line is the mean trajectory $\mu(t)$ and the shaded area indicates the covariance. The 5 solid lines are sample trajectories $\theta(t)$ from the GP prior.

Problem Statement

Define our cost functional $\mathcal{F}[\theta(t)] = \mathcal{F}_{obs}[\theta(t)] + \lambda \mathcal{F}_{GP}[\theta(t)]$
$\mathcal{F}_{gp}$ is the *prior* for our Gaussian process.

$$p(\theta) \propto \exp\{-\frac{1}{2}||\theta - \mu||^2\}$$

minimize $\mathcal{F}[\theta(t)]$
subject to $\mathcal{G}_i[\theta(t)] \leq 0$
and $\mathcal{H}_i[\theta(t)] = 0$

Optimizing cost

- Our cost functional is non-convex, so we take an iterative, gradient based approach.
- Goal: find a perturbation $\delta\theta$

$$\delta\theta^* = argmin\{\mathcal{F}[\theta] + \nabla\mathcal{F}[\theta]\delta\theta + \frac{\eta}{2}||\delta\theta||^2\}$$

- : Update rule given by

$$\nabla\mathcal{F}[\theta(t)] = \frac{\partial v}{\partial\theta(t)} - \frac{d}{dt}\frac{\partial v}{\partial\dot{\theta}}(t)$$

Factor graphs and GPMP2

- GPMP2 reformulates this problem as one of inference, parameterizing θ in terms of desired events e.
- The optimal trajectory maximizes

$$\theta^* = argmax\{p(\theta)l(\theta; e)\}$$

- Factor graphs are a probabilistic graphical model (just like Bayes nets!)
- (If this stuff sounds cool come to my optimization talk where I'll go into more detail on Levenberg-Marquadt)

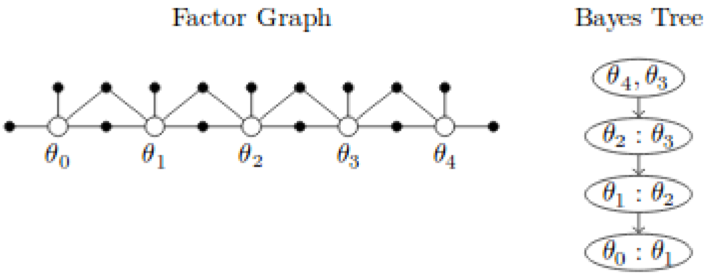

Figure 6. Example of a Bayes Tree with its corresponding factor graph.

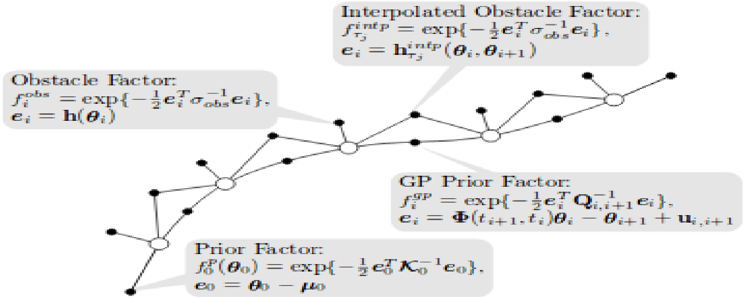

Figure 5. A factor graph of an example trajectory optimization problem showing support states (white circles) and four kinds of factors (black dots), namely prior factors on start and goal states, GP prior factors that connect consecutive support states, obstacle factors on each state, and interpolated obstacle factors between consecutive support states (only one shown here for clarity, any number of them may be present in practice).

GPMP-Graph

- A *homotopy class* is a set of trajectories that we can continuously deform between
- We construct different factor graph chains for different homotopy classes and optimize over them in parallel

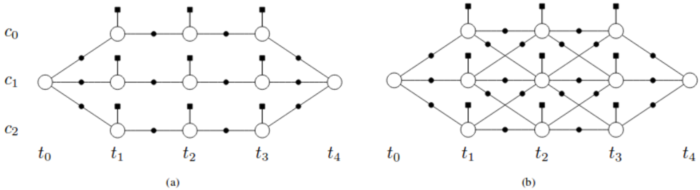

Fig. 3: Example graph-based trajectories constructed from three chains (c_0 to c_2) from start x_0^1 at (t_0, c_1) to goal x_5^1 at (t_5, c_1). Interconnections between chains are absent in (a) and present in (b). States (white circle) are connected temporally in t and spatially in c with GP prior factors (black circle). Collision factors (black square) are shown and GP interpolation factors present between states connected by edges are omitted for clarity.

Citations/Further Reading

"Continuous-time Gaussian process motion planning via probabilistic inference" - Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert and Byron Boots
"Motion Planning with Graph-Based Trajectories and Gaussian Process Inference" - Eric Huang, Mustafa Mukadam, Zhen Liu, and Byron Boots