Ideas for SfM-Inspired Path Planning

David Fridovich-Keil
dfk@eecs.berkeley.edu

January 7, 2016

1 Project description

The aim of this project is to design a robust path planner for a mobile outdoor robot. Since the primary motivation for exploration is the collection of sufficient data to create a highly detailed 3D environment map in post-processing, the design of the robot planner is predicated on that goal.

In particular, the robot in this project is a quadrotor platform with several pairs of low-resolution stereo grayscale images, and a single high-resolution color camera on a three-axis stabilized gimbal. Note that there are *no laser scanners*, and *no Kinect-style depth sensors*. In practice, this means that all depth estimation is done using stereo disparity maps and, more generally, **Structure from Motion**, or SfM.

In light of this, we will design the path planning algorithm around our knowledge of how SfM works. That is, SfM relies on matching corresponding **keypoints** between images, and from these correspondences triangulating 3D **landmarks** in the environment. A keypoint is simply a location in an image that is in some way special – for example, the most common type of feature is a corner of some sort, since corners are very distinctive. There are a whole host of feature detectors out there; at the end of the day, most of the popular ones are more or less equivalent in terms of performance, so we will not go into any further detail here. Of course, as soon as two keypoints are determined to match, it is possible to triangulate an approximate 3D position of the corresponding landmark in the environment. Doing this for a large number of landmarks allows the joint optimization of camera pose and landmark position; this is called **bundle adjustment**.

2 Big idea

Since our only sensors are cameras, the final model generation step will be one gigantic bundle adjustment problem. Of course, it is not hard to see that having more landmarks over a larger volume will improve reconstruction quality. The big contribution of this project to the great body of literature on path planning is a novel formulation of this intuitive idea within the existing framework of information-theoretic approaches to exploration.

3 Background

This section provides a (very) brief overview of the state of the art in path planning. For a more in-depth description of these topics, consult the appropriate references. For the most part, this is a summary of the corresponding section(s) in [1].

3.1 What is Exploration?

Exploration is defined as the process by which a robot constructs a map of its environment by moving and acquiring new sensor measurements. It is comprised of two distinct subtasks:

1. Identifying and scoring potential actions the robot can take in order to learn more about the environment

2. Autonomously choosing one of these candidate actions, updating the robot's position on the map, and receiving new sensor measurements and registering them to the map

There are two broad classes of algorithms to solve the exploration problem – geometric and information-theoretic methods – which are described in the following subsections.

3.2 Geometric Exploration Methods

Begin by considering a partially explored environment consisting of regions which are either occupied, unoccupied, or unknown. Formally, each point x in the environment is assigned a label, $L(x) \in \{\text{occupied, unoccupied, unknown}\}$. Perhaps the most obvious case for illustrative purposes is a two dimensional environment made up of discrete squares. That is, each point in the environment is a tuple of integers – i.e. $x = (i, j) : i, j \in \mathbb{Z}$. In this case, it is a simple matter to find the boundary between the explored and unexplored regions (e.g. a simple floodfill will do the job).

One approach that has been incredibly influential is to find these so-called *frontiers* and then, using some heuristic, choose one and plan a path towards it. This is based on the seminal work by Yamauchi, [2].

Of course, there are quite a lot of reasonable heuristics one might consider using. The simplest is probably just to move toward the nearest frontier, but that is not always the best choice. At the end of the day, this method is highly sensitive to the choice of hueristic, which makes it somewhat less general than we might like.

Moreover, it completely breaks down in three dimensions, where it is not such an easy task to even *find* the frontiers, let alone find a *good* frontier to go to. There are two reasons for this. First, in three dimensions algorithms like floodfill take exponentially longer to compute – this can be overcome by clever optimizations like caching results. Second, and worse, 3D environments tend to be more complex than 2D, with more varied shapes and occlusions, leading to far more frontiers, not all of which are worth visiting, yet any one of which might be chosen by an imperfect heuristic.

For these reasons, the community of late has started to gravitate toward a dramatically different formulation based on information theory.

3.3 Information-Theoretic Exploration Methods

Information-theoretic methods may seem dramatically different from purely geometric methods, but in essence they are simply a generalization with some added mathematical formalism.

3.3.1 What is Information Theory?

First off, information theory itself bears some explanation. Of course, this is an entire field, so the summary provided here is merely meant as a very rough guide. For a more in-depth background consult any standard textbook.

Information theory is the mathematical study of information – that is, information theory provides a formal way to understand and analyze what we mean by *information*. It is very closely linked to probability theory, in the sense that information is usually a property of a random variable, rather than a quantity of raw data.

Suppose that a discrete random variable X has a particular distribution, $\mathbb{P}\{X = x\} = p(x)$. The *information content* of X, also called **entropy**, can be calculated from the distribution $p(x)$. The most common form of entropy is called the Shannon entropy, and it is calculated as follows:

$$H_S = -\sum_x p(x) \log_2 p(x) \tag{1}$$

There are many other ways to calculate entropy however; we leave a more in-depth discussion for later sections.

3.3.2 Occupancy Grid

Now let us consider the same simple 2D voxel-grid we described above, in which each point $x = (i, j) : i, j \in \mathbb{Z}$ is assigned a label $L(x) \in \{\text{occupied, unoccupied, unknown}\}$. Of course, this last label, "unknown" is not a *true* label, in the sense that once the robot has explored the entire map, no voxel will be labelled "unknown."

This leads us to an elegant formalism: consider the map to be a collection of random variables X_{ij}, each of which independently takes a label $X_{ij} \in \{\text{occupied, unoccupied}\}$ with some probability. Of course, a priori those probabilities are equal, since we know nothing about the environment.

Elementary probability theory lets us write the joint distribution of a collection of voxels $m = \{X_{ij}\}_{(i,j)\in\mathcal{M}}$ for some set $\mathcal{M}$ as follows:

$$p(m) = \prod_{(i,j)\in\mathcal{M}} p(X_{ij}) \tag{2}$$

What we have just described is called an **occupancy grid**, and it is an especially popular way of representing a map. Of course, there are downsides: most obviously, that the world is not actually a voxel grid, and by reducing it to one introduces sampling artifacts.

3.3.3 Application to Exploration

Now that we have expressed the environment as a collection of binary (Bernoulli) random variables, and using our knowledge of entropy, we can devise a very elegant solution to the exploration problem. That is, for some entropy metric $H(m)$ and paths $r \in \mathcal{R}$, we can write down the optimization problem:

$$r^* = \arg\min_{r \in \mathcal{R}} H(m|r) \tag{3}$$

Intuitively we wish to minimize entropy (which is like the randomness left in the map) given that we take path r. Of course, this is in principle a very difficult problem – after all, how can we know for sure what the entropy will be after taking a particular path?

But the game is not over yet. Consider the following problem, which is only slightly different:

$$r^* = \arg\min_{r \in \mathcal{R}} \mathbb{E}[H(m|r)] \tag{4}$$

Here, we have taken the expectation of the entropy metric. This is usually something we can actually calculate, assuming that we have a good idea of how our robot senses the environment. Such a **sensor model** enables us to anticipate which voxels we will observe over a particular path. Note that the actual observations themselves – i.e. whether or not a particular voxel is observed to be occupied or not – is irrelevant to the calculation of entropy. Entropy only measures randomness, which is obviously decreased when we observe more voxels.

Before proceeding, let us clarify the notation used above, and make it slightly more precise. $H(m|r)$ is intended to mean "the entropy of map m after taking path r." This is silghtly confusing, of course, because the vertical bar is usually used for condiitioning a probability distribution. Fortunately, that is actually what we are doing here. Recall that m is actually a collection of independent random variables, and that a path and a sensor model define a set of possible observations. That is, each path r actually corresponds to a random variable z corresponding to the observations acquired by the robot. These observations follow a distribution which we can explicitly calculate from the sensor model and the existing map. Thus, it makes sense to talk about evaluating the expectation of the entropy of the map conditioned on these observations z, which are actually a function (albeit a random one) of the deterministic path r.

In the literature, this optimization problem is usually expressed slightly differently. Consider a reward function $\mathcal{J}(m, z_\tau(x_\tau))$ (think of this as how much of the map is known to the robot), for the map m and observations $z_\tau(x_\tau) = (z_{t+1}(x_{t+1}), \ldots, z_{t+\tau}(x_{t+\tau}))$ for robot configurations $x_\tau = (x_{t+1}, \ldots, x_{t+\tau})$. This set of robot configurations is equivalent to the path r used above. Using this notation, the problem becomes:

$$x_\tau^* = \arg\max_{x_\tau \in \mathcal{R}} \mathcal{J}(m, z_\tau(x_\tau)) \tag{5}$$

Typically the reward function $\mathcal{J}$ is chosen to be the **divergence** of the conditional distribution of the map m given observations z_τ. Divergence is a distance metric between

probability distributions – in this case, it is essentially the difference in entropy between the prior distribution of m and the posterior. Using the Shannon entropy, the associated Kullback-Leibler divergence of two distributions $(p(x), q(x))$ for a random variable X is:

$$D_{KL}(p||q) = \sum_x p(x) \log_2 \left(\frac{p(x)}{q(x)} \right) \tag{6}$$

As with entropy, there are many other definitions of divergence.

4 Adaptation to SfM

Occupancy grids are very common for general SLAM (**s**imultaneous **l**ocalization **a**nd **m**apping) applications because they are easy to implement and, with sufficient memory and the proper data structures, scale sufficiently well to large environments. However, structure from motion is a very special case of SLAM – for SfM to work, it needs to model the landmarks as *points*, not as voxels. Of course, it is easy to convert from a point cloud into an occupancy grid; unfortunately, the other direction is not really possible without significant quantization error.

At the end of the day, though, the occupancy grid is not the fundamental reason why we must devise a novel exploration scheme. Rather, the problem is that, as is, the information-theoretic approach is *volume*-based in the sense that it's figure of merit is essentially a measure of how much volume (or area in 2D) has been expored. In our project, we would like to be able to build a high-resolution 3D model of a building, which means that we require *lots of points*, not a large volume.

Thus, we must devise a way to reformulate the information-theoretic approach to exploration in terms of a point cloud, rather than an occupancy grid.

4.1 Toward a New Objective

The first order of business is to come up with a new objective function $\mathcal{J}$. Since structure from motion is all about landmarks, it makes sense to begin by introducing a new random variable N_τ to represent the number of new landmarks in the map aquired during time interval τ; i.e. $N_\tau = \text{card}(m_{t+\tau}) - \text{card}(m_t)$ where the map m is explicitly time-parameterized.

Naively, we might wish to simply maximize the expected value of N_τ, as follows:

$$x_\tau^* = \arg\max_{x_\tau} \mathbb{E}[N_\tau | z_\tau(x_\tau)] \tag{7}$$

Unfortunately, this is too simplistic and it will almost certainly fail. Fear not – the rest of this work is devoted to fixing it!

4.2 Modeling

Perhaps the most obvious problem with the aforementioned optimization is taking the expectation. Before, we could do this rather simply because entropy did not depend on *what*

we observed, only *that we observed it*. Moreover, and more importantly, we were working with a voxel grid that easily lent itself to a probabilistic model.

Here though, we are counting landmarks in a *continuous* environment. That means that we must be able to model the presence of landmarks conditioned on our anticipated field of view. One interesting approach is to model the occurrance of landmarks in the environment as an n-dimensional non-homogeneous Poisson process with rate as a function of the map. For example, one simple approach would be to maintain an occupancy grid (either implicitly or explicitly) and set the rate of the Poisson process to be some function of the current estimated probability of occupancy at each grid location.

With this model, we can very easily estimate the conditional expectation above in closed form.

4.2.1 Aside: Poisson Processes

Poisson processes are one of the most elegant stochastic processes, and they have some very nice properties. Typically, Poisson processes are conceived as arrivals in time; i.e. if $\{X_t\}$ is a Poisson process with rate λ, then

$$X_s \sim \text{Pois}(\lambda s) \tag{8}$$

$$\mathbb{P}\{X_s = k\} = e^{-\lambda s}\frac{(\lambda s)^k}{k!} \tag{9}$$

A non-homogeneous Poisson process is one with a non-constant rate. For example, if $\lambda = \lambda(t)$, then

$$X_s \sim \text{Pois}\left(\int_{\mu}^{s+\mu} \lambda(t)dt\right) \tag{10}$$

Finally, an n-dimensional Poisson process is one that has multiple "time" dimensions (of course, these need not actually be time and in our case they represent space). In this case, for rate $\lambda(x, y)$ in two dimensions (for simplicity), we have

$$X_{(x,y)} \sim \text{Pois}\left(\int_{\mu}^{x+\mu}\int_{\nu}^{y+\nu} \lambda(s,t)dsdt\right) \tag{11}$$

Think of this last version as modeling fish in the ocean. Over any given x-by-y rectangle of ocean, the probability of observing k fish is $\mathbb{P}\{X_{(x,y)} = k\}$ if fish are distributed as a 2D Poisson process with rate function $\lambda(x, y)$. Of course, it is a simple matter to replace fish in the ocean with landmarks in an arbitrary environment, and it is also not a stretch to consider $\lambda(x, y)$ as a stepwise function in x and y, as described above.

4.2.2 Formal Model Definition

Formally then, we can model the distribution of N_τ conditioned on robot configurations x_τ (corresponding to observed map region $\mathcal{O}_\tau$) as follows:

$$N_\tau | x_\tau \sim \text{Pois}\left(\int_{\theta \in \mathcal{O}_\tau} \lambda(\theta) d\theta\right) \tag{12}$$

for rate function $\lambda(\cdot)$ defined on the map (e.g. $\mathbb{R}^3$). This rate function is in turn estimated from the occupancy grid representation, $p(\theta)$ = estimated probability that each voxel is occupied, for some monotonic function f that scales probabilities to rates (e.g. by multiplying by the number of observed landmarks in each voxel):

$$\lambda(\theta) = f(p(\theta)) \tag{13}$$

One useful property of the Poisson distribution is that its mean is equal to its argument. This means that

$$\mathbb{E}[N_\tau | x_\tau] = \int_{\theta \in \mathcal{O}_\tau} \lambda(\theta) d\theta \tag{14}$$

which is simply a discrete sum in our case because $\lambda(\cdot)$ is voxelized.

4.3 Regularization

Often, we want the solution to optimization problems to have some particular property. For example, in regression analysis, we often want the solution vector to be sparse. To achieve this, we use some sort of **regularization**. Regularization is the addition of some other term(s) to the objective function that penalize (or reward) poor (or good) solutions. In the regression example, one common approach is to penalize the L_1 norm of the solution vector.

In path planning, it is quite common to penalize some particular measure of *cost*, like expected energy usage. That is a fine approach for our problem, however we can use regularization more immediately to ensure efficient exploration behavior.

Consider a toy example, in which the entire environment consists of two highly detailed object in space. As is, our objective function will drive the robot to explore one of these obejcts very carefully, but it will almost certainly not do a good job of exploring the space around the object. More importantly, it will almost certainly never even *find* the other object. In order to incentivize the planner to find the other object, we can reward it for finding landmarks that are far apart. One way to do this is to add on a term that is the sum of squared distances between all landmarks l in the map m, as follows:

$$\text{ssd}(m) = \sum_{i \neq j} \|l_i - l_j\|_2^2 \tag{15}$$

From the Poisson model, we can also write down an estimate for the expected locations of new landmarks: since the rate function is voxelized, the expected location of each landmark

is at the center of the corresponding voxel and can be weighted according to the rate at that voxel.

The full optimization problem is thus (note that m_{CM} is the current map's center of mass and $\hat{l}$ is an estimated landmark position):

$$x_\tau^* = \arg\max_{x_\tau} \mathbb{E}[N_\tau + \text{ssd}(m_\tau)|z_\tau(x_\tau)] \tag{16}$$

$$= \arg\max_{x_\tau} \sum_{\hat{l}\in\mathcal{O}_\tau} \lambda(\hat{l}) + \sum_{\hat{l}_1,\hat{l}_2\in\mathcal{O}_\tau} \lambda(\hat{l}_1)\lambda(\hat{l}_2)\|\hat{l}_1 - \hat{l}_2\|_2^2 + \sum_{\hat{l}\in\mathcal{O}_\tau} \lambda(\hat{l})\|\hat{l} - m_{CM}\|_2^2 \tag{17}$$

References

[1] Erik Nelson, "Environment Model Adaptation for Autonomous Exploration," *MS Thesis: Carnegie Mellon Robotics Institute*. Pittsburgh, PA: May 2015.

[2] Brian Yamauchi. "A frontier-based approach for autonomous exploration." In *Computational Intelligence in Robotics and Automation, 1997. CIRA97., Pro- ceedings., 1997 IEEE International Symposium on*, pages 146151. IEEE, 1997.