Vision Based Mobile Robot Pose Estimation and Mapping

Yogesh A. Girdhar

1 Introduction

The problem of localization and pose estimation, are closely related to generic object recognition and even to ego-motion estimation, but they differ in several important ways. The long term research focus that this literature review relates to, deals with fully autonomous robot systems, in which sensing, localization, and specifically vision-based localization can be expected to be critical generic components.

This review explores large scale state-of-the-art localization systems [1], one of the fundamental algorithmic mechanisms for uncertainty modeling [3, 2], vision-based modeling of positions and objects [4, 6] , and the most classic approaches to computing usable features for recognition systems [5]. Papers [7] and [8] deal with two of the classical (i.e. well established) mathematical and algorithmic methods used for geometric modeling [7], and for learning-based feature selection[8].

2 Overview

The general problem of state estimation corresponds to estimating values of state paramaters of a given system, given the observation data. In mobile robotics, pose estimation is a more specific instance of this problem where state is defined as location and orientation of the robot.

If the robot does not have a map of the world around it, then it must simultaneously build the map and localize itself on it as it explores the world. This problem is referred to as *Simultaneous Localization And Mapping* (SLAM). In this instance of state estimation problem, the state parameters not only include the robot pose, but also location of various landmarks or obstacles in the the world.

3 Recursive State Estimation

Let $\mathbf{x}$ be the state variable we would like to estimate. We would like to compute the posterior distribution $p(\mathbf{x}_t|Z^t)$ representing the estimated current state $\mathbf{x}_t$ of the robot given the measurements made so far $Z^t = \{\mathbf{z}_1 \cdots \mathbf{z}_t\}$.

In other words, for each location, what is the probability that the current set of measurements came from that location. This can be expanded using Bayes' theorem as following:

$$\begin{aligned} p(\mathbf{x}_t|Z^t) &= p(\mathbf{x}_t|\mathbf{z}_t, Z^{t-1}) & (1) \\ &= \frac{p(\mathbf{z}_t|\mathbf{x}_t, Z^{t-1})p(\mathbf{x}_t|Z^{t-1})}{p(\mathbf{z}_t|Z^{t-1})} & (2) \end{aligned}$$

Given that we know $\mathbf{x}_t$, we generally assume measurement $\mathbf{z}_t$ to be independent of all previous measurements Z^{t-1}.

$$p(\mathbf{z}_t|\mathbf{x}_t, Z^{t-1}) = p(\mathbf{z}_t|\mathbf{x}_t) \tag{3}$$

Hence posterior density over $\mathbf{x}_t$ can be computer recursively as:

$$p(\mathbf{x}_t|Z^t) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|Z^{t-1})}{p(\mathbf{z}_t|Z^{t-1})} \tag{4}$$

where the prior is computer by integrating over all possible previous states.

$$p(\mathbf{x}_t|Z^{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|Z^{t-1})d\mathbf{x}_{t-1} \tag{5}$$

Computing the posterior distribution above recursively from previous state estimates is also known as *Bayes Filtering*. A straightforward application of Bayes Filtering as described above is not tractable in all but the most trivial cases. However if we make a few assumptions or approximations, its possible to come up with a reasonably good solution. Kalman Filtering and Particle Filtering are two such techniques.

The Kalman Filter is an analytical and optimal solution. It however works under the assumption that the state variable, measurements and noise can all be modeled by multivariate normal distributions, and that state evolution is a linear function.

Particle Filtering is a newer simulation based technique which gets over these limitations of Kalman Filtering. It can model random variables which have non-gaussian, multi-modal probability density function. This however comes at the cost of additional computational complexity. The main idea behind Particle Filters is that a random state variables are modeled by a set of particles, each being a possible value of the variable. Together, these particles can be thought of as a discrete representation of the PDF of the random variable.

For the purposes of pose estimation, Particle Filtering is in general more suited since assumptions made by Kalman Filtering are frequently not true.

3.1 Particle Filters

Typically our model consists of :

- $p(\mathbf{x}_0)$ representing initial condition.
- $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ representing stochastic dynamics of the system model used to estimate how the state variable is expected to evolve. For the case of a mobile robot, we could include an additional variable u_t representing the control signal. In this case our evolution model can be written down as: $p(\mathbf{x}_t|\mathbf{x}_{t-1}, u_t)$
- $p(\mathbf{z}_t|\mathbf{x}_t)$ the measurement or observation model.

In Particle Filtering [2], instead of explicitly representing the probability density function, we instead approximate it with a set on N samples $S_t = s_t^i$ (a.k.a particles) and their weights π_t^i. At time step t we apply our dynamics model to each particle s_{t-1}^i to get s_t^i by drawing s_t^i from $p(\mathbf{x}_t|\mathbf{z}_{t-1})$. Now to take into account the new measurement $\mathbf{z}_t$, we recalculate weight of each sample using our measurement likelihood model $\pi_t^i = p(\mathbf{z}_t|\mathbf{x}_t = s_t^i)$.

At any given time we can get an estimated value of our state by either calculating a weighted average of the particles or choosing the particle with the maximum weight or maybe a locally weighted mean.

If we if continue with the algorithm described above, we will notice that after a few time steps, most of the samples have weight close to zero since they have drifted away from the correct estimate of the state. As a result they become useless. To fix this we can *resample* at each time step to discard samples with low weights. This technique is sometimes known as *Sequence Importance Resampling* [2].

Overall, the one iteration of the algorithm at a given time step t can be summarized as following:

- For each sample s_{t-1}^i:
  - *Predict:* Compute s_t^i by sampling from $p(\mathbf{x}_t|\mathbf{x}_{t-1} = s_{t-1}^i)$.
  - *Measure:* Compute weight of the sample by looking at the measurements. $\pi_t^i = p(\mathbf{z}_t|\mathbf{x}_t = s_t^i)$
- *Estimate:* Output our prediction $\mathcal{E}[\mathbf{x}_t]$. This could either be weighted mean of $\mathbf{x}_t$, or the $\mathbf{x}_t^i$ corresponding to the highest weight, or some other technique depending on the application.
- *Resample:* Pick samples by drawing randomly with probability proportional to their weights. Give equal weight to these samples.

3.2 Localization using Particle Filters

One of the first uses of Particle Filters for vision based localization was in [2]. In the paper, the authors use a very simple vision sensor to measure the brightness of the ceiling directly above the robot and use that as the measurement model $p(\mathbf{z}|\mathbf{x})$. The system state is defined as robot's 2D position and its orientation.

$$\mathbf{x}_t = [p_{x_t}, p_{y_t}, \theta_t]$$

. The system dynamics model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ estimates the new position of the robot given the robot control input $\mathbf{u}_{t-1}$.

Considering this really simple vision sensor, the results of this approach are impressive. The robot was able to globally localize itself in a museum in 126 iterations and drawing 40,000 samples from a uniform probability density function as the initial conditions. Tracking the position of the robot however required a lot fewer samples (1000), and robot was able to end within 10cm of the correct end position after traversing for 200 meters. This is much better than previous results on this problem.

3.3 Simultaneous Localization and Mapping using Particle Filters

For the problem of SLAM, our goal is to have an estimate of the trajectory of the robot as well as the map of the region, at any given time t. Hence we can define our state variable x_t as:

$$\mathbf{x}_t = [R^t, \mathbf{m}_t] \tag{6}$$

where $R^t = \{\mathbf{r}_1, ..\mathbf{r}_t\}$ is the trajectory of the robot up till time t, and m_t represents the map at time t. For a robot moving in a 2D world, $\mathbf{r}_t$ could be modeled as $\mathbf{r}_t = [p_{x_t}, p_{y_t}, \theta_t]$. Map $\mathbf{m}_t$ can be modeled as a vector of position of all the observed landmarks. Now given a control signal $U^t = \{\mathbf{u}_1..\mathbf{u}_t\}$ and landmark measurements $Z^t = \{\mathbf{z}_1..\mathbf{z}_t\}$ at time t, our goal is to predict:

$$p(\mathbf{x}_t|Z^t, U^t) = p(R^t, \mathbf{m}_t|Z^t, U^t) \tag{7}$$

In other words, we would like to predict the robot's trajectory and map landmark locations at time t, given all the landmark measurements and control signals in the past.

A naïve application of particle filers to the SLAM problem, where we try to sample the entire state space directly, will lead to disastrous results. For particle filters to be able to effectively model the probability density function of a random variable, the number of particles needed grows exponentially with the dimensionality of the random variable. Hence the number of samples needed to model the SLAM problem state defined above is intractably high.

3.3.1 Rao-Blackwellisation

The sampling problem described above can mitigated by making by following some simplifying assumptions. We can only sample parts of our random variable and compute distribution of other parts analytically thereby reducing the dimension of the state space which we need to sample [6]. Here is how:

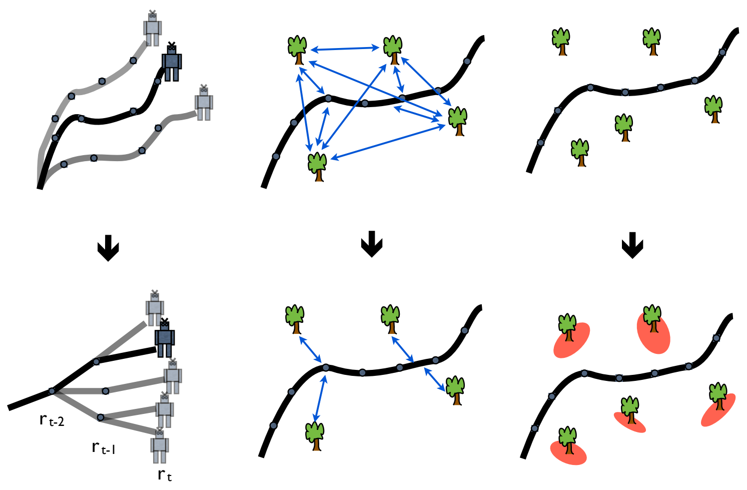

Figure 1: Simplifying assumptions made by Rao-Blackwellized Particle Filters. [LEFT] We only sample the next step in the trajectory at each time step instead of the entire trajectory. [MIDDLE] Location of landmarks is assumed to be mutually independent of each other. [RIGHT] We model location of each map landmark with a gaussian update by EKF.

1. **Constant Trajectory History** In each iteration of the algorithm, we only sample for position of the robot at the next time step $\mathbf{r}_t$ and not the entire trajectory R^t. This leads to a tree-like trajectory sampling, and makes number of samples needed tractable.

$$\begin{aligned} \mathbf{r}_t &\sim p(\mathbf{r}_t|\mathbf{r}_{t-1}, u^t) && (8) \\ R^t &= \{R^{t-1}, \mathbf{r}_t\} && (9) \end{aligned}$$

2. **Mutual Independence of Landmarks** Without any loss of generality, we can expand equation 7 as:

$$p(R^t, \mathbf{m}_t|Z^t, U^t) = p(R^t|Z^t, U^t)p(\mathbf{m}_t|R^t, Z^t, U^t) \quad (10)$$

Now if we make the assumption that location of each landmark m_t^k is independent of other landmarks, then we can rewrite equation 10 as:

$$p(R^t, \mathbf{m}_t|Z^t, U^t) = p(R^t|Z^t, U^t)\prod_k p(m_t^k|R^t, Z^t, U^t) \quad (11)$$

this will reduce the size of sampling space drastically, from exponential in number of landmarks to linear.

3. **Tracking Landmark Location with a Kalman Filter** The two approximations above reduces expressive power of our system to describe relative location of each landmark. To mitigate this problem, instead of fixing the location of each landmark in each particle, we describe it with a normal distribution. We can then update the distribution of each landmark analytically using Kalman Filters.

A visualization of the above simplifying assumptions is shown in Figure 1. This version of particle filters is also known as *Rao-Blackwellised Particle Filters (RBPF).* In [6], Sim presents current state of the art in SLAM. In this work RBPF is used along with SIFT based visual landmarks (see Section 4.3.1, 4.4.1) to solve the problem.

4 Feature Based Location Recognition

Ability to compare images of the environment and tell if they are *similar* is an essential for many vision based pose estimation algorithms. This problem is ill-posed since many times the same image can come from a variety of different locations and vice versa. Hence most of the techniques settle for an approximate solution. A simple technique could be to keep a database of images taken at different locations in the world and then compare with the incoming image to try to directly localize the robot. Another technique could be to use this matching to compute observation model $p(\mathbf{z}|\mathbf{x})$ and feed it into a more elaborate localization algorithm described in previous sections.

Images can be be compared in two ways; globally or feature based local matching. Global matching involves comparing the contents of the entire image. A popular global image matching technique is to use *Principal Component Analysis* (PCA) to compare images in a lower dimensional space.

4.1 Principal Component Analysis

PCA gives us an optimal orthogonal linear transformation to a new coordinate system such that direction of maximum variance of data corresponds to the first dimension of this space, and second greatest direction of variance corresponds to second dimension and so on.

The data matrix Y corresponding to our training(map) images can be represented by a huge matrix where each row corresponds to one *flattened* image. The covariance matrix of this data is YY^T. Now to compute basis vectors for the new space are just the eigenvectors of this covariance matrix sorted by their eigenvalues.

We can now represent an image optimally by projecting it onto the first few eigenvectors computed above. Images can be compared by finding the Euclidean distance between them in this lower dimensional space.

This approach although effective for some applications like face recognition, fails for our purpose. This is because the matching is *not* invariant to occlusion and changes in scale, shift, rotation and light direction. For this reason majority of current vision based pose estimation techniques use local feature based matching.

4.2 Local Matching

The general idea behind feature based recognition is that instead of trying to match the entire image, we extracts salient regions of the image and then only compare those. This local matching not only gives us invariance to occlusion, shift and light direction, but depending on the exact implementation also give us invariance to rotation and scaling. Feature based recognition of a scene image can be broken down into three sub-problems: identifying salient regions in the image to be used as features, describing and comparing these features to other features, doing location recognition by combining results of feature matching. We will now look at these subproblems in greater detail.

4.3 Identifying Salient Regions

4.3.1 SIFT Features

SIFT[5] is a popular algorithm to find location of salient features or keypoints in the image by looking for blob like regions of any scale. We know that a difference of Gaussian (DOG) function when convolved with an image, gives us a high response at points of local minima or maxima in the image. So if we convolve our input measurement image with DOG functions corresponding to several different sigmas, this would give us feature location corresponding to features of different sizes. Orientation of this feature is defined by direction of maximum image gradient. This is computed by building a image gradient histogram and choosing the peak direction. This ability to define an orientation of the feature gives is rotation invariance. A major advantage of selecting features using SIFT is that it gives us feature locations which are reasonably invariant to the robot location and lighting conditions.

4.3.2 Selecting Features by Maximizing Mutual Information

Mutual Information is an information theory concept representing the amount of information shared between two random variables. While doing feature based recognition, it makes sense to chose features which describe a particular pose or location well. In other words, we would like to chose features z such that there is high mutual information between them and the robot state $\mathbf{x}$.

More formally, mutual information $I(\mathbf{x}; z)$ between random variables $\mathbf{x}$ and z is defined as reduction in uncertainty about $\mathbf{x}$ when z has been observed or vice versa.

$$\begin{aligned} I(z;\mathbf{x}) &= H(z) - H(z|\mathbf{x}) && (12) \\ &= H(\mathbf{x}) - H(\mathbf{x}|z) = I(\mathbf{x}; z) && (13) \end{aligned}$$

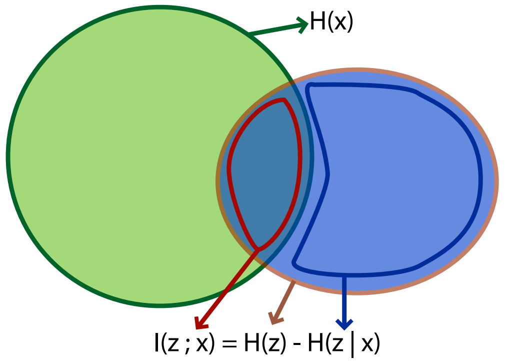

Figure 2: Mutual Information of two random variables $I(\mathbf{z}, \mathbf{x})$ is the amount of information shared between them.

Here H is the entropy of the random variable.

A features that has high mutual information with the robot pose are good, however its possible that this feature has high mutual information with several different poses. In this case our feature is not so useful in discriminating between different poses. Hence we should only consider features for which $p(z|x)/p(z|!x)$ is high.

4.4 Feature Description and Matching

4.4.1 SIFT Feature Description and Matching

Apart from giving the location of the features (Section 4.3.1), in [5], Lowe also descibes a technique to do do viewpoint and lighting invariant matching of the SIFT features.

To be able to do viewpoint and lighting invariant matching of features, we need a keypoint descriptor with similar properties. To do this, SIFT algorithm first compute image gradient in 16x16 cell array covering the region around the feature. Then it creates 4x4 gradient orientation histograms with 8 bins forming a 128 dimensional scale and rotation invariant feature descriptor which can be used for matching.

Figure 3 shows a visualization of a smaller 4x4 gradient array being converted to a smaller 2x2 descriptor histogram.

4.4.2 Principal Components in Frequency Domain

Dudek and Jugessur [4] demonstrate another interesting rotation invariant technique to compare features. It is known that Fourier transform of an image is invariant to shift when we looking at its amplitude spectrum. Also rotation in an image corresponds to shift when

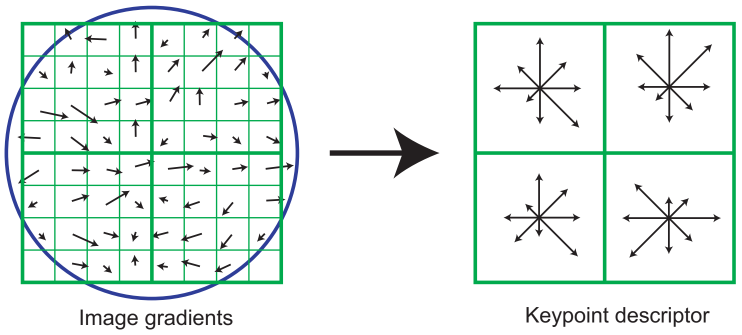

Figure 3: To compute the a SIFT descriptors, we first compute image gradients and then build gradient histograms.(figure source: [5])

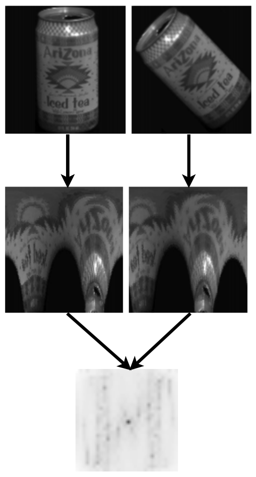

Figure 4: A sample image and its rotated version and their corresponding images in polar coordinates. We see that rotation corresponds to shift in polar coordinates. Both polar images give us the same amplitude spectrum image.(Can images taken from [4])

we look at it in its polar coordinates (see Figure 4). Combining these two idea we can see that if we first convert our images to polar coordinates and then compute their amplitude spectrum by taking their Fourier transform, we get a features descriptor which is invariant to rotation. These features can now be compared in a lower dimensional space using PCA.

4.4.3 Mutual Information Based Matching

In [8], Viola and Wells used mutual information to align different images. The approach was to find the relative transformation between the images for which mutual information between them is maximum. We can use the same concept of mutual information, to match our features to the images. The advantage of doing this over direct intensity based comparisons is that it can even match features when there is a non-linear transformation between them, which for example might arise due to illumination changes.

4.5 Location Recognition

4.5.1 Voting

Once we have capabilities to identify and match features, the simplest way to recognize a location could just be a voting scheme. In [4], features in the incoming image is matched to features corresponding to all locations in the database. Each match then contributes to a vote inversely proportional to the Euclidean distance between the two feature in their lower dimensional eigenspace (Section 4.4.2).

4.5.2 Observation Likelihood Models

When we have a mobile robot whose motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, u_t)$ is known, then it makes more sense to use the bayesian framework setup in Section 3. We can then use the feature matching techniques described above to build a observation model $p(\mathbf{z}_t|\mathbf{x}_t)$. The observation model gives us the probability of observing given set of features at a particular location or pose. We can hence define our measurement $\mathbf{z}$ to be a set of binary numbers, each representing visibility of a a feature.

$$\mathbf{z} = \{z_1, ..., z_n\} \tag{14}$$

Here n is the total number of features we have in our database. We can then write our observation likelihood distribution as a joint over individual features.

$$p(\mathbf{z}|\mathbf{x}) = p(z_1, ..., z_n|\mathbf{x}) \tag{15}$$

Now to learn this distribution, we would need number of samples exponential in n. Cummins in [1] suggests the use of a Chow Liu tree to come up with an approximation expressed as joint of first order conditionals.

$$p(\mathbf{z}|\mathbf{x}) = p(z_1, ..., z_n|\mathbf{x}) \approx p(z_{a_1}|\mathbf{x})p(z_{a_2}|z_{b_2}, \mathbf{x}), ..., p(z_{a_n}|z_{b_n}, \mathbf{x}) \tag{16}$$

Here for any given feature z_{a_i}, the corresponding feature z_{b_i} is chosen so that these two have maximum mutual information. This can be done by first creating a complete graph of mutual information, where each node is a feature and edge is the mutual information between them, and then finding the maximum weight spanning tree. Mutual information I can calculated by counting the joint occurrence of two features during the training phase.

$$I(z_a, z_b) = \sum_{z_a, z_b \in \{0,1\}} p(z_a, z_b) \log \frac{p(z_a, z_b)}{p(z_a)p(z_b)} \tag{17}$$

5 Stereo Vision

Till now we have seen how to localize a robot by just analyzing the camera images in a 2D fashion. We do not consider the geometry of the world. It is however possible to extract 3D information about the world if we have two or more cameras on the robot. A good overview of this topic can be found in [7].

5.1 Simple Stereo

First let us consider the case when we have two pinhole cameras with focal length f are separated by a distance T and have parallel view axis. To calculate the depth of a point or feature in the world, visible form both the cameras, first we need to calculate its *disparity*. Disparity is the difference in image position of a point corresponding to the two cameras. See Figure 5 for an illustration of disparity. Once the disparity d of a given point is known, we can calculate the depth Z of the point by using similar triangles.

$$\frac{T-d}{Z-f} = \frac{T}{Z} \tag{18}$$

We can calculate disparity value of each pixel in the image, or we could do it only for a sparse set of features extracted from the left and right images. For the case of pixel based matching, we could define a window around the first point and for all possible windows in the second image, compute the sum of squared differences. Window with the minimum difference will give us the corresponding point. Similarly for feature based matching we could minimize the Euclidean distance between the feature descriptors.

In either case, one might think that we have to consider all possible pair of pixels or features in the left and right images to find the corresponding matches. However if we consider *Epipolar Gemoetry* of stereo, we can find constraints which convert this problem from a 2D search to a 1D search.

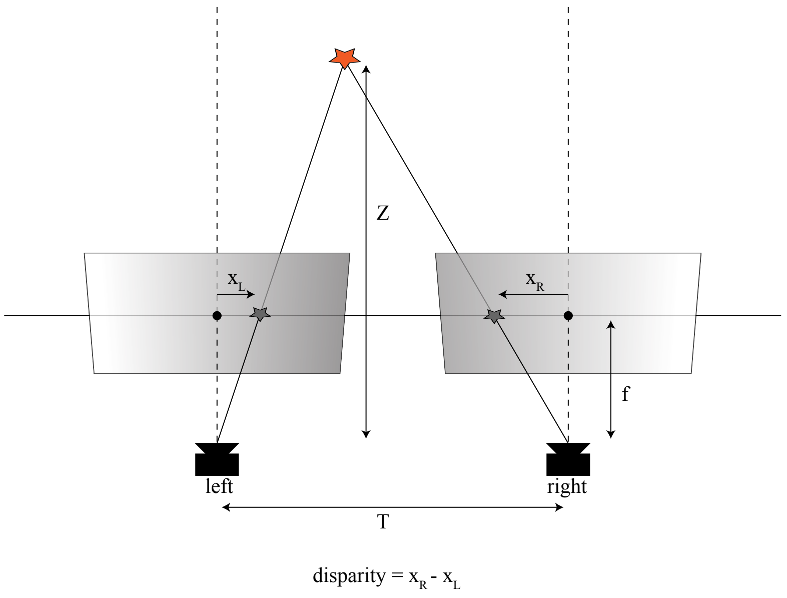

Figure 5: Disparity in a simple stereo system where the view axis of the two cameras are parallel to each other.

5.2 Epipolar Geometry

The general case of stereo known as Epipolar Geometry is show in Figure 6. Here we have a point P in the world with position vector P_l and P_r corresponding to the left and right camera frames. vectors p_l and p_r correspond to the projection of this point onto the projection plane. The main point to note here is that no matter what the depth of point P is in the left camera, its projected image p_r on the right camera plane is guaranteed to lie on the epipolar line shown in the figure.

The plane formed by O_l, O_r, P is known as epipolar plane. The equation of this plane can be written by writing the coplanarity condition:

$$(P_l - T) \cdot T \times P_l = 0 \tag{19}$$

The above equation can rewritten in the form:

$$\overline{p}_r^T F \overline{p}_l = 0 \tag{20}$$

Where $\overline{p}_r$ and $\overline{p}_l$ are the pixel coordinates of the corresponding points. F is called the *fundamental matrix* and it relates the two corresponding points in epipolar geometry. We can estimate F by using eight or more known correspondences to solve for entries of F. This is called the eight point algorithm.

Once we have F, we can now interpret $F\overline{p}_l$ as the epipolar line on which $\overline{p}_r$ must lie.

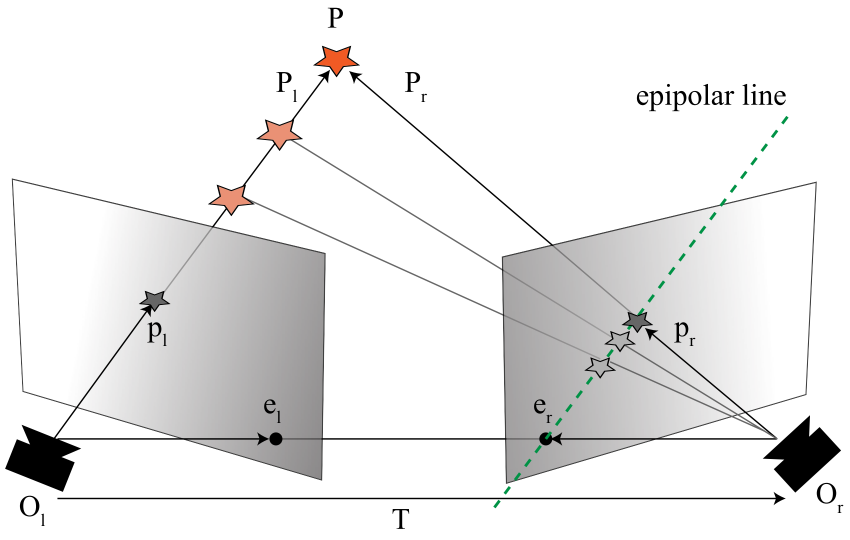

Figure 6: Epipolar Geometry

6 Discussion

In this literature review we look at several topics related to the problem of mobile robot pose estimation and mapping. We first looked at the bayesian framework typically used for state estimation, and then how we can use particle filters to do the estimation in practice. We saw how we can use Rao Blackwellized particle filters to do simultaneous localization and mapping.

With these state estimation tools in hand, we then saw how to plug in vision based measurements into our state estimation algorithms to recognize locations. We discussed the idea of feature based techniques, and how they are more suited to this problem than global matching techniques. We discussed different techniques to describe and match these features in a rotation, scale and light direction independent manner. Finally we looked at how a simple stereo vision system works and how we can extract 3D information about the world.

This paper hopefully presents a good picture of the current literature dealing with the problem of vision based mobile robot pose estimation and mapping.

References

[1] Mark Cummins and Paul Newman. Probabilistic appearance based navigation and loop closing. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'07)*, Rome, April 2007.

[2] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 1999.

[3] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[4] Gregory Dudek and Deeptiman Jugessur. Robust place recognition using local appearance based methods. *IEEE International Conference on Robotics and Automation (ICRA)*, april 2000.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision(IJCV)*, 60(2):91–110, 2004.

[6] Robert Sim, Pantelis Elinas, Matt Griffin, Alex Shyr, and James J. Little. Design and analysis of a framework for real-time vision-based slam using rao-blackwellised particle filter. In *3rd Canadian Conference on Computer and Robotic Vision (CRV 2006)*, 2006.

[7] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3D Computer Vision*, chapter 7 Stereopsis. Prentice Hal, 1998.

[8] Paul Viola and William M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision(IJCV)*, pages 137–154, 1997.