Tutorial on a Probabilistic Measurement Model based on Landmark Range and Bearing Information

Jonas Schwertfeger (js@cs.brown.edu)

November 10, 2007

1 Preamble

The measurement model explained below is very similar to the landmark measurement model described in [1]. The purpose of this document is to give students of Brown's introductory robotics class CSCI1480 a more comprehensive tutorial on this particular type of measurement model.

2 Introduction

Recall the purpose of the measurement model in MCL: Estimate the likelihood of a hypothetical pose of the robot given a map of the environment and some sensor measurements. For example, we would like to know how likely is it that a robot is positioned at $x = 2\text{m}$, $y = 1.5\text{m}$, pointing into the direction $\theta = 90°$ given what the robot is "seeing" right now.

One particular simple measurement model assumes that we have knowledge of certain objects in the robot's environment, so-called *landmarks*, that can help us with localization. In our case these are colored fiducials. For each of them we are given their x and y coordinates as well as their type ("pink", "green-over-red", "red-over-green", etc.). Thus we can say a landmark is completely specified by the triple (x, y, t) where x, y are the location coordinates and $t \in \{1, 2, \ldots, n_t\}$ is the type. The map of the robot's environment is then simply a vector of such triples,

$$M = \{m_1, m_2, \ldots, m_{n_m}\} \tag{1}$$

$$= \left\{ \begin{bmatrix} x_1 \\ y_1 \\ t_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ y_2 \\ t_2 \end{bmatrix}, \ldots, \begin{bmatrix} x_{n_m} \\ y_{n_m} \\ t_{n_m} \end{bmatrix}, \right\} \tag{2}$$

where n_m is the total number of landmarks.

If the robot spots such a fiducial in its camera's field of view it can immediately determine the angle at which it sees the fiducial. Furthermore, based on how wide or tall the fiducial appears and physically is the robot can estimate the fiducial's distance. Let us call a spotted landmark a *feature* and denote it by the triple (r, ϕ, t) where r is the distance (range) from the robot to the fiducial, $\phi \in (-\pi, \pi]$ is the angle (bearing) at which the fiducial was spotted and $t \in \{1, 2, \ldots, n_t\}$ is its type. The bearing is 0 if the fiducial is right in the middle of the robot's field of view. If the fiducial is to the left the bearing is positive, if it is to the right the bearing is negative. At a particular time instant our sensor measurements z are now summarized by the vector of observed features,

$$f(z) = \{f_1, f_2, \ldots\} \tag{3}$$

$$= \left\{ \begin{bmatrix} r_1 \\ \phi_1 \\ t_1 \end{bmatrix}, \begin{bmatrix} r_2 \\ \phi_2 \\ t_2 \end{bmatrix}, \ldots \right\} \tag{4}$$

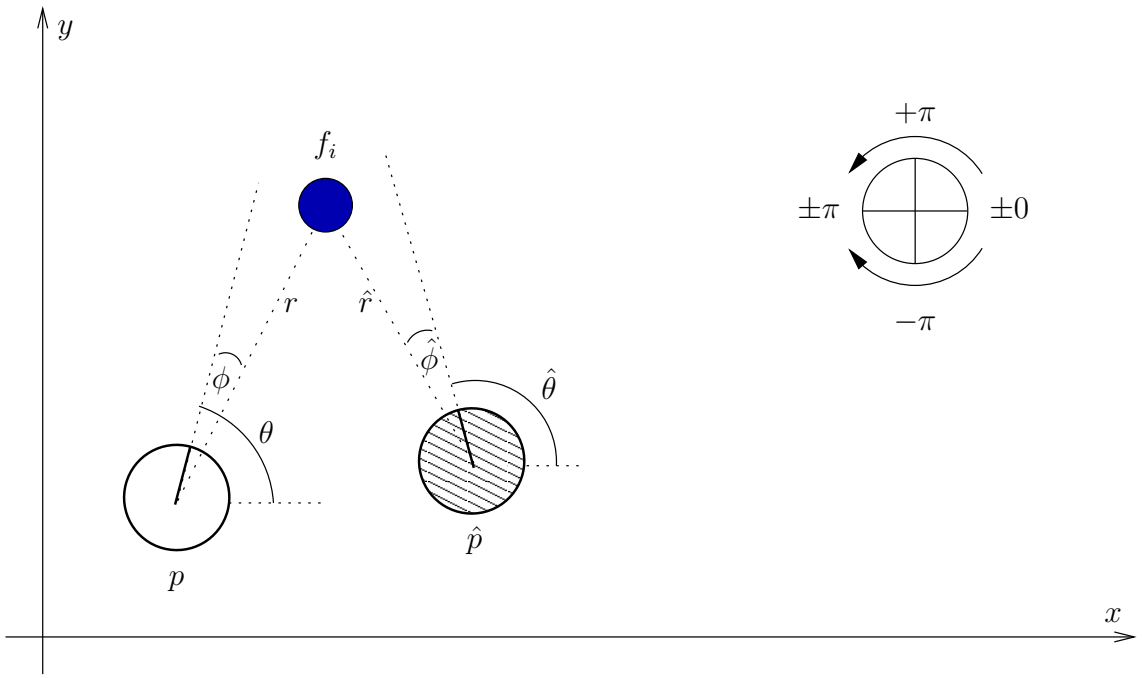

Figure 1: Range and bearing between the true robot pose p and a feature f_i, as well as between a hypothetical pose $\hat{p}$ and the same feature.

3 Some Planar Geometry

Our goal is to test how similar the feature vector at a specific hypothetical pose looks to the actual observed feature vector. Suppose we spotted a "green-over-red" and a "pink" fiducial with the robot's blob finder. This gives us the two features f_1 and f_2 (or f_i for $i \in \{1, 2\}$) for which we can easily compute the range and bearing from the blob information. Both features correspond to a particular landmark in our map. Let $c(f_i, M)$ be a *correspondence* function that, given a feature and the map, returns the index j of the landmark to which the feature corresponds. In our case this function simply looks a the type of the feature (e.g. "pink") and searches the list of landmarks for a landmark of type "pink" and then returns its index. Let us now compute the range and bearing between a hypothetical pose $\hat{p}$ and a landmark m_j that we spotted. The range is given by the Euclidean distance

$$\hat{r}_i = \sqrt{(m_{j,x} - \hat{p}_x)^2 + (m_{j,y} - \hat{p}_y)^2}\,, \tag{5}$$

where $m_{j,x}$ denotes the x coordinate of the j^{th} landmark, and the bearing by

$$\hat{\phi}_i = \text{atan2}(m_{j,y} - \hat{p}_y, m_{j,x} - \hat{p}_x) - \hat{p}_\theta \tag{6}$$

where atan2 computes the arctangent of $(m_{j,y} - \hat{p}_y)/(m_{j,x} - \hat{p}_x)$ with a range of $(-\pi, pi]$ [1]. Fig. 1 illustrates the range and bearing as measured by the robot's real pose p as well as the range and bearing computed for a hypothetical pose $\hat{p}$ [2]. The hypothetical pose is a likely pose if, for each observed feature, the range and bearing is similar to the real range and bearing we measured.

[1] The function atan2 is predefined in most programming languages; have a look at `http://en.wikipedia.org/wiki/Atan2` to understand what it does.

[2] The upper right corner of the figure shows that, by definition, angles are measured in radians and the two extremes 0 and $\pm\pi$ lie on the abscissa (x-axis). In addition, the coordinate origin is in the lower left corner. These two definitions are slightly different than the ones we used in room 404 so far but are more common in robotics and make life easier when dealing with atan2.

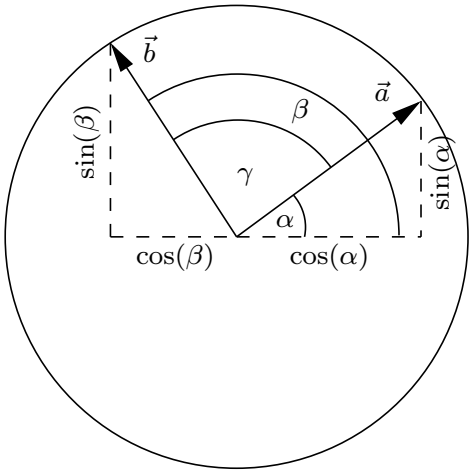

Figure 2: Computations on the unit circle.

The next step is to compute the similarity between the measured range/bearing and the hypotehtical pose's range/bearing. In case of the range this is simply

$$\Delta r_i = r_i - \hat{r}_i \ . \tag{7}$$

Computing the difference in bearing is not as straight forward as the range because of the discontinuity of angles at $\pm\pi$ [3]. However, if we define unit vectors according to each of the two angles we can use the dot product between these vectors to compute the angle between them. Fig. 2 depicts a unit circle with the two vectors $\vec{a}$, with angle α, and $\vec{b}$, with angle β. $\vec{a}$ is given by $(\cos(\alpha), \sin(\alpha))$ and $\vec{b}$ by $(\cos(\beta), \sin(\beta))$. The angle γ is then

$$\begin{aligned} \gamma &= \arccos(\vec{a} \cdot \vec{b}) \\ &= \arccos(\cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)) \ . \end{aligned}$$

Thus the bearing difference is

$$\Delta\phi_i = \arccos(\cos(\phi_i)\cos(\hat{\phi}_i) + \sin(\phi_i)\sin(\hat{\phi}_i)) \ . \tag{8}$$

4 Turning Geometry into Probabilities

At this point we know how to calcuate the difference between the features as observed by the robot and the features that the robot would see if it were in a hypothetical pose. But since we will most probably see multiple features (fiducials) at the same time, how do we combine these multiple differences and how do we express the combination as a probability? If you are interested in these questions you can follow the derivation below. However, if you simply want to know how to implement the model you can skip this section and proceed to Sec. 5.

To answer the above questions let us first take a step back and formalize what we want to know. We want to know the probability of all the observed features given a hypothetical pose and the map, which can formally be expressed as follows:

$$p(f(z) \mid \hat{p}, M) \tag{9}$$

First we note that Eq. (9) can be written as the joint likelihood of all features by substituting $f(z)$ with the right hand side of Eq. 3:

$$p(f(z) \mid \hat{p}, M) = p(f_1, f_2, \ldots \mid \hat{p}, M) \tag{10}$$

[3] For example, even though the two angles $3\pi/4$ and $-3\pi/4$ have an effective difference of $\pi/2$, the difference of their values is either $3\pi/2$ or $-3\pi/2$.

If we make the assumption that the likelihood of each feature does not depend on the likelihood of the other features we are allowed to write Eq. (10) as a product of likelihoods:

$$p(f(z) \mid \hat{p}, M) = \prod_i p(f_i \mid \hat{p}, M) \tag{11}$$

This is convenient because now all we have to do is to come up with the probability for a single feature,

$$p(f_i \mid \hat{p}, M) = p(r_i, \phi_i, t_i \mid \hat{p}, M) \ . \tag{12}$$

By making another independence assumption (namely that the likelihood of the range, the bearing, and the type, given $\hat{p}$ and M, are independent) we can write Eq. (12) as

$$p(f_i \mid \hat{p}, M) = p(r_i \mid \hat{p}, M) \cdot p(\phi_i \mid \hat{p}, M) \cdot p(t_i \mid \hat{p}, M) \ . \tag{13}$$

By substituting M with the right hand side of Eq. (1) we get

$$p(f_i \mid \hat{p}, M) = p(r_i \mid \hat{p}, m_1, \ldots, m_{n_m}) \cdot p(\phi_i \mid \hat{p}, m_1, \ldots, m_{n_m}) \cdot p(t_i \mid \hat{p}, m_1, \ldots, m_{n_m}) \tag{14}$$

If we are guaranteed that $\sum_{k=1}^{n_m} p(t_i \mid \hat{p}, m_k) = 1$ we can interpret $p(t_i \mid \hat{p}, m_k)$ as a "weighting factor" expressing the certainty with which the feature type t_i corresponds to the landmark type $m_{k,t}$. We can then write Eq. (14) as a weighted sum:

$$p(f_i \mid \hat{p}, M) = \sum_{k=1}^{n_m} p(r_i \mid \hat{p}, m_k) \cdot p(\phi_i \mid \hat{p}, m_k) \cdot p(t_i \mid \hat{p}, m_k) \tag{15}$$

We define the probability of the range, $p(r_i \mid \hat{p}, m_k)$, and the bearing, $p(\phi_i \mid \hat{p}, m_k)$, by imposing a zero-mean Gaussian distribution on the geometrically computed range and bearing differences. In other words, if the difference is small, then we get back a high probability but as the difference becomes larger the probability falls off pretty quickly. How quickly it falls off is determined by the normal distribution's standard deviation σ_r and σ_ϕ, respectively, which we have to set to reasonable values. Even though the above formulation of $p(t_i \mid \hat{p}, m_k)$ would allow us to incorporate uncertainty with respect to the type of a fiducial, we do not make use of it and replace it with the Kronecker delta function $\delta_{t_i, m_{k,t}}$ which returns 1 if $t_i = m_{k,t}$ and 0 otherwise. Thus, Eq. (15) simplifies to

$$p(f_i \mid \hat{p}, M) = \sum_{k=1}^{n_m} p(r_i \mid \hat{p}, m_k) \cdot p(\phi_i \mid \hat{p}, m_k) \cdot \delta_{t_i, m_{k,t}} \tag{16}$$

$$= p(r_i \mid \hat{p}, m_{c(f_i,M)}) \cdot p(\phi_i \mid \hat{p}, m_{c(f_i,M)}) \tag{17}$$

5 Putting it All Together

The implementation of this model is much easier than it might seem. Table 1 contains a function in pseudo-code that computes the measurement likelihood of a particular measurement z given a hypothetical pose $\hat{p}$ and a map M. First, the total likelihood is initialized (line 2). Then, for each feature, that is each fiducial, the function `computeFeatureLikelihood` is called and its output is multiplied with the total likelihood and stored as the new total likelihood (line 4). This corresponds to Eq. (11).

Table 2 contains the `computeFeatureLikelihood` function which computes the likelihood of a single feature given the pose and the map. In line 2 it calls the correspondence function $c(\cdot)$ which returns the index j of the landmark that corresponds to the feature f. Lines 3 to 6 compute the range and bearing differences and line 7 returns the product of the two difference probabilities. Line 7 implements Eq. (12). Finally, Table 3 contains the Gaussian function `normPdf`.

```
1 function computeMeasurementLikelihood(z, p̂, M)
2   l = 1
3   for i in f(z)
4     l = l · computeFeatureLikelihood(f_i, p̂, M)
5   return l
```

Table 1: Computes the likelihood of a measurement z given a pose $\hat{p}$ and a map M. Outputs the probability $p(f(z) \mid \hat{p}, M)$.

```
1 function computeFeatureLikelihood(f, p̂, M)
2   j = c(f, M)
3   r̂ = sqrt((m_{j,x} − p̂_x)^2 + (m_{j,y} − p̂_y)^2)
4   φ̂ = atan2(m_{j,y} − p̂_y, m_{j,x} − p̂_x) − p̂_θ
5   Δr = r − r̂
6   Δφ = arccos(cos(φ) cos(φ̂) + sin(φ) sin(φ̂))
7   return normPdf(Δr, σ_r) · normPdf(Δφ, σ_φ)
```

Table 2: Computes the likelihood of a single feature f given a pose $\hat{p}$ and a map M. Outputs the probability $p(f \mid \hat{p}, M)$.

```
1 function normPdf(v, σ)
2   return 1/sqrt(2πσ^2) exp(−v^2/(2σ^2))
```

Table 3: The probability density function of a univariate zero-mean normal distribution. Takes as input the value for which the probability shall be evaluated and the standard devation $\sigma > 0$.

```
1  function computeMultiLandmarkFeatureLikelihood(f, p̂, M)
2    J = c(f, M)
3    l = 0
4    for j in J
5      r̂ = sqrt((m_{j,x} − p̂_x)^2 + (m_{j,y} − p̂_y)^2)
6      φ̂ = atan2(m_{j,y} − p̂_y, m_{j,x} − p̂_x) − p̂_θ
7      Δr = r − r̂
8      Δφ = arccos(cos(φ) cos(φ̂) + sin(φ) sin(φ̂))
9      l = l + normPdf(Δr, σ_r) · normPdf(Δφ, σ_φ)
10   return l/ | J |
```

Table 4: Computes the likelihood of a single feature f given a pose $\hat{p}$ and a map M, assuming there might be several landmarks corresponding to the type of the feature. In this case the probabilities are averaged. Outputs the probability $p(f \mid \hat{p}, M)$.

6 Remarks

6.1 Ambiguous Feature-Landmark Correspondence

The function `computeFeatureLikelihood` in Table 2 assumes that for a particular feature there is only one corresponding landmark on the map. In room 404 however, there are multiple fiducials with the same color. Thus, they share the same type and consequently the map contains multiple markers that can correspond to an observed feature. This requires a few changes to the logic in that the correspondence function has to return a set of indices instead of just a single index and in that the likelihood between the feature and each of corresponding landmarks has to be computed. The output of the feature likelihood function is then the probability averaged over the set of landmarks. The function `computeMultiLandmarkFeatureLikelihood` in Table 4 reflects these changes. Line 10 divides the sum of likelihoods by the number of indices in J.

6.2 Choosing Standard Deviations for Range and Bearing

The two parameters σ_r and σ_ϕ determine how "harsh" the likelihood function acts when it judges range and bearing differences. For bearing a reasonable value seems to be $10°$ ($\sigma_\phi = \pi/18$). The standard deviation for range seems harder to specify. How about $\sigma_r = 0.5$m?

References

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* MIT Press, 2005.