Mobile Robots - localization

Gaëtan GARCIA
Ecole Centrale de Nantes
Emaro I - ASP I
1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France
gaetan.garcia@ec-nantes.fr

June 7, 2016

Foreword

This document is not a self-contained book covering the material of the Localization part of the Mobile Robots course. It is meant as a help in reviewing the material for exams, and requires the slides and your personal notes.

The document is still in draft form. I specifically need to work on including the bibliography, which is currently extremely sparse and cites documents which may not be easy to find.

Chapter 1

Introductory material

1.1 Definitions

The *pose* of a rigid solid is its position and orientation in $3D$, which can be characterized by six independent parameters. *Localization* is the process of determining some or all variables of the pose. In our context, localization is applied to a vehicle, which we will often call robot, although it may not be the case in practice. A vehicle is not a rigid solid, but localization applies to its chassis or, in case of an articulated chassis, one of the solids that consitute the chassis.

It may not always be necessary to determine the six degrees of freedom of the vehicle. How many variables of the pose should be determined depends on the type of motion the vehicle is undergoing, on the task it has to perform, etc. A very common, less general situation is that of a vehicle moving on a plane. In such a case, only $2D$ position, and possibly heading, need to be determined. The situation is so common that the vector of position and heading in a plane has its own name: the *posture*.

There exist of course more general problems, for example articulated vehicles, for which internal position parameters may have to be calculated, such as the angle between the front and rear parts of the vehicles [Bouvet and Garcia, 2000], but also situations where speed and rotation speed must also be estimated.

One important practical consideration is the choice of a representation for the pose, especially in $3D$, where multiple solutions exist: homogeneous transformations, position vector plus Euler angles, quaternions, etc. The reader is referred to the course "Modeling and Control of Manipulators" of Wisama Khalil for more details or to his book [Khalil and Dombre, 2004]. The question is important because experience shows that a good choice of representation can greatly simplify the equations, and sometimes the solution of a localization problem. We will illustrate this in the course.

Finally, we will be here mostly interested in what may be called *self localization*. In a self localization problem, the mobile embeds the sensors and software necessary to determine its own location.

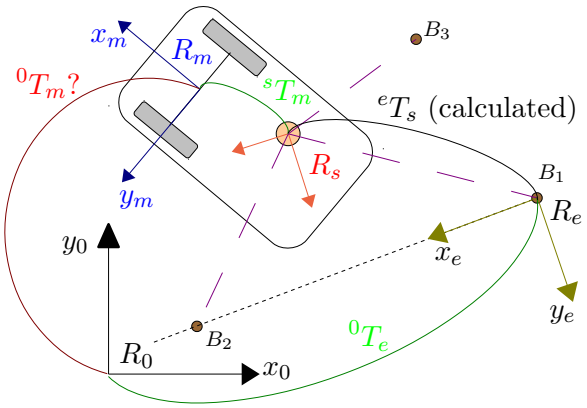


Figure 1.1: The classical frames of a localization problem

1.2 The classical frames in a localization problem

Figure 1.1 shows the classical frames and transformations that are taken into account in a localization problem.

- R_m is the vehicule (mobile) frame, usually attached to the chassis. For $(2,0)$ robots[1], it is classical to define its x axis perpendicular to the axle of the fixed wheels because this is the direction of the speed vector under the non-slipping assumption.

- R_0 is the absolute frame, in which the user wants to know the pose of the vehicle. On figure 1.1, the pose is represented by transformation matrix ${}^0T_m$. It is the unknown that should be determined, hence the question mark on the figure.

- R_e (environment frame) is attached to the known reference points of the environment, which the localization system uses to perform measurements (distances to the points, azimuth and/or elevation angles of the points, etc.) These points are supposed to be known in the absolute reference frame R_0. Usually they are stationary in R_0, but in some cases they may move in a known way (*e.g.* GPS satellites). The assumption in this course is that ${}^0T_e$ is known. Whenever possible, R_e is chosen in such a way that it simplifies calculations.

- R_s is a frame attached to the localization sensor(s). We will always suppose that ${}^mT_s$ is a known transformation. Generally, the localization system calculates ${}^eT_s$: pose of the localization sensor(s) frame with respect to the environment frame. In some cases, the required precision is such that ${}^mT_s$ can be neglected. In other cases, it may be necessary to calibrate it. Its orientation may (goniometric sensor) or may not (distance sensor) be important.

As stated above, in many cases, the sensor determines its own location ${}^eT_s$ with respect to the environment frame R_e. The actual unknown ${}^0T_m$ is then obtained by the following matrix product:

$$ {}^0T_m = {}^0T_e \cdot {}^eT_s \cdot {}^sT_m \tag{1.1}$$

[1] See the Modeling part of the course.

1.3 The various classes of localization algorithms

A first dichotomy in localization algorithms concerns the vehicle motion. Does the algorithm allow the vehicle to move during the process or not:

- A *static* localization algorithm requires that the vehicle be motionless during the time interval necessary to gather all sensor data used to localize it. In practice, the important point is that *all measurements correspond to the same pose of the vehicle.*
- A *dynamic* localization algorithm is able to handle the case when the vehicle moves while the data is gathered. The various measurements used by the system correspond to different poses of the vehicle.
- A *quasi-static* localization algorithm corresponds to the case when the robot moves while taking the various measurements used to localize it, but the motion amplitude is negligible. In practice, it means that the motion amplitude is much smaller than the required precision of localization. The only difference with the static algorithm is that, when implementing such an algorithm, one should always make sure that the assumption actually holds.

Another dichotomy exists between the so-called *relative* and *absolute* localization systems. But in order to define these classes, we first need to recall what *proprioceptive* and *exteroceptive* sensors are:

- An *exteroceptive* sensor performs a measurement which is relative to a known point or feature of the environment. There are essentially two types of such sensors: *distance-meters* or *range finders* measure distances; *goniometers* measure angles.
- A *proprioceptive* sensor does not need any known reference in the environment to perform measurements. In this class of sensors we find wheel encoders, gyrometers, Doppler radars, accelerometers, inclinometers...

Let's now define relative and absolute localization algorithms:

- An *absolute localization algorithm* determines the pose of the vehicle with respect to a frame attached to the environment.
- A *relative localization algorithm* determines the pose of the vehicle with respect to its initial location.

Relative localization is often refered to as *dead reckoning*, which is defined in [Wikipedia, 2011b] as: "the process of estimating one's current position based upon a previously determined position, or *fix*, and advancing that position based upon known or estimated speeds over elapsed time, and course".

Absolute localization requires exteroceptive sensors. Relative localization uses proprioceptive sensors. The principle is that the sensors allow the speed (or elementary displacements) of the vehicle to be calculated. The elementary displacements in turn are used in an integration loop to determine the current pose of the vehicle. As the reader can now see, the frames of figure 1.1 in fact apply to absolute localization problems, not to relative localization.

Absolute and relative localization algorithms have very different and complementary characteristics. The characteristics of relative localization algorithms are:

- Since they do not require any reference to known features of the environment, they can be used in all conditions (day/night, fog/clear weather...)
- It is possible to freely choose the sampling frequency of the integration (within reasonable bounds of course), and the period between calculations can be *fixed* (synchronous processing).
- The processing time is usually short.
- Precision degrades with time and travelled distance.

The characteristics of absolute localization algorithms in turn are:

- By definition, localization is performed with respect to a fixed frame of the environment.
- The frequency of sensor readings is often low, especially as compared to relative localization sensors.
- Sensor data are often available *asynchronously* (*i.e.* not at fixed time instants), due to sensor technology (*e.g.* rotating sensor), vehicle motion, etc.
- With some exteroceptive sensors, especially when based on vision, the raw data requires processing before the actual measurement is available, creating time delays: the measurement corresponds to a passed position of the vehicle. The distance travelled during the processing time may not be negligible.
- The number of external measurements available may vary according to vehicle position, since the number of reference points visible to the sensors may vary.
- The precision of the localization process depends on the configuration of the reference points, and also on the position of the vehicle with respect to the reference points.

As can be seen from these lists of characteristics, the two types of localization algorithms are indeed complementary. Localization errors increase as a relative localization algorithm is used. This phenomenon is called *drift* and can only be maintained within finite bounds by the use of exteroceptive measurements. In the simplest case, relative localization is used as long as errors are considered to be reasonable, then external measurements are taken to perform an absolute localization. This can be called *sequential relative and absolute localizations*. Figure 1.2 illustrates this solution as traditionally used by sailors:

- Repeated relative localizations are performed using a *log* and a *compass*:
  - The *log* is a ship instrument which measures the speed of the boat with respect to water[2]. Together with a map of currents, it allows to determine the speed of the ship with respect to the ground.
  - The *compass* measures the heading of the boat with respect to magnetic north.

[2]See [Wikipedia, 2011a] for the historical instrument and [Wikipedia, 2011c] for modern systems

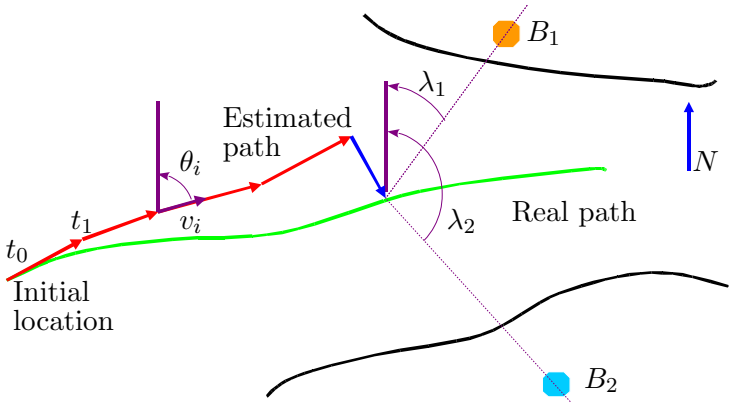


Figure 1.2: The (traditional) sailor's method: repeated relative localizations followed by absolute localization

  - The speed and heading of the boat are supposed constant over a period (time between two relative localizations). It is then possible to draw the displacement vector with a pen directly on the map, from the previous location. The direction of the vector is known with respect to north. The method is pictured in figure 1.2.

- Sailor know by experience how long they can afford to use relative localizations. It highly depends on how far the ship is from potential dangers. When necessary, they perform an absolute localization. We will consider the simple case when he is not too far from the shore and can see reference points like beacons or remarkable features of the coast (a cape for example).

  - Using a marine compass, measure the *bearing angle* of a beacon, *i.e.* the horizontal angle from north to the reference point.
  - Draw a line with the corresponding angle with respect to north and passing through the beacon. The ship is somewhere on this line.
  - Repeating the process for a second remarkable point localizes the ship at the intersection of the two lines. The method is pictured in figure 1.2.
  - Whenever possible, use a third measurement. The three lines drawn usually do not intersect at a single point. The size of the obtained triangle gives an idea of the precision of the process.

There is yet another type of localization algorithm, called *hybrid localization algorithm*. This type of algorithm repeatedly uses relative localization steps and "as soon as" an external measurement is available, it is used to improve the current location estimate. The essential difference between the hybrid and sequential algorithms is that the hybrid algorithm can use the data provided by a single measurement which, taken alone, would not be sufficient for an absolute localization.

This will be made clearer further into the course, but we can write the general form of a hybrid algorithm:

The calculation of the initial location can be a standard absolute localization algorithm. We later present a commonly used mathematical tool which allows a hybrid algorithm to be implemented.

Now that we have the basic vocabulary, we can present the contents of the remaining chapters of the course.

Algorithm 1 Hybrid localization algorithm

```
Calculate initial location
loop {fast}
  Read proprioceptive sensors (e.g. encoders)
  Calculate relative localization
  if new external sensor data available then
    Use external data to improve estimated location
  end if
end loop
```

- Chapter 2 is dedicated to *absolute localization*. We first study a few *absolute static localisation* methods and give an example of estimating the precision of the localization result as a function of the precision of the measurements. *Absolute dynamic localization* algorithms are then covered. Clues are given about what to do when the motion of the vehicle between two measurements while not being measured, is not negligible. It is shown that the only way to solve such a problem is to assume the motion has some regularity, *e.g.* constant speed over a certain time interval.

- Chapter 3 is devoted to *odometry*, a *relative localization method* based on measuring elementary rotations of a robot's wheels. Odometry is very often used in industrial and research applications. The chapter aims at helping you capture the main advantages and limitations of odometry. It also shows how to evaluate the performances of odometry for a given robot.

- In chapter 4, an example of sequential use of relative and absolute localizations applied to a robot is given. It is based on an existing industrial solution which uses magnets located in the ground and can be used as a low infrastructure equivalent to wire-guided systems.

- Chapter 5 corresponds to the technique which is considered state of the art in localization. It consists in the use of state estimation techniques. We concentrate on the most popular of all estimators for non linear systems, the extended Kalman filter. The full theory of the Kalman filter is not recalled. Instead, we recall its equations and apply it to the same localization problem already solved in chapter 2 as a static absolute localization method and a dynamic localization method. We believe that by studying the various types of solutions for the same problem, the important ideas of each can be grasped by the student.

- Finally, chapter 6 aknowledges the fact that localization is essentially a state observation problem and that, as such, studying the observability of the system is a problem of interest. In this chapter, we show that the study of system observability can help understand the situations in which the localization system may fail. We also show that the effort can be rewarded with finding potentially problematic situations which were not easy to predict.

1.4 Application examples

Application examples are given in the slide show which has been mailed to you for self-study. It is in no way exhaustive and insists on applications where the author has personal experience. Surfing the internet with the proper keywords will yield further examples if you wish to broaden your knowledge of applications.

Chapter 2

Absolute localization methods

2.1 Static absolute localization examples

2.1.1 The surveyor's problem

The surveyor's problem consists in determining the $2D$ (x, y) position of a point using two previously known points as references. To perform this task, the surveyor uses a goniometer to measure the angular separation between two points. The goniometer is moved from one known point to the next, so this is not self-localization. The problem is illustrated by figure 2.1.

Angle λ_1 is measured by setting the goniometer above A and measuring the separation angle between B and the unknown point P. The instrument is then set at B to measure the separation angle λ_2 between A and P. In order to solve the localization problem, it is convenient to (see figure 2.2):

- Define a local frame with its origin at A and X axis along AB, directed from A to B. The coordinates of P are determined in this frame.
- Use the distances from A to P and B to P as intermediate unknowns.

The local frame corresponds to R_e in figure 1.1. It simplifies calculations because the coordinates of A and B in this frame are simple. The technique which consists in using distances as intermediate variables instead of angles is very common and usually simplifies the calculations when dealing with goniometric systems. The distance d between the known points is called the *baseline*. It can be shown that:

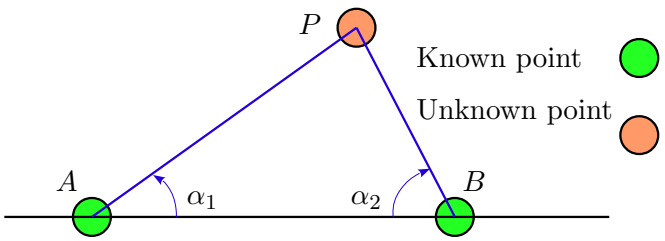


Figure 2.1: The sureyor's problem: localise P using known points A and B and a goniometer.

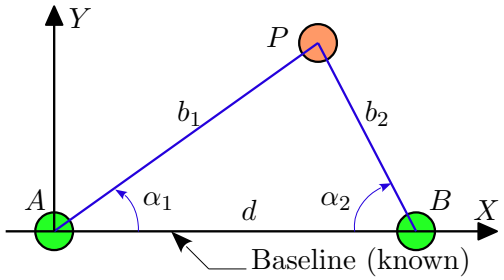


Figure 2.2: Local frame and intermediate unknowns in the surveyor's problem.

$$\begin{aligned} b_1 &= d\,\frac{\sin\alpha_2}{\sin(\alpha_1+\alpha_2)} \qquad b_2 = d\,\frac{\sin\alpha_1}{\sin(\alpha_1+\alpha_2)} \\ x &= d\,\frac{\cos\alpha_1\sin\alpha_2}{\sin(\alpha_1+\alpha_2)} \qquad y = d\,\frac{\sin\alpha_1\sin\alpha_2}{\sin(\alpha_1+\alpha_2)} \end{aligned} \tag{2.1}$$

Exercise 1 (Using distances as intermediate variables)**.** *Prove the equations of system 2.1.*

From these equations and the data, it is possible to calculate the numerical values of the coordinates of P. But as always, it is not very meaningful to give a result without giving an idea of its precision. How can we represent and calculate the precision of the obtained position? Classically, the precision on P is given by its $(2*2)$ covariance matrix. In order to calculate it, we need the following result:

Rule 1. *Let X and Y be two random vectors of dimensions $n*1$ and $p*1$, for which the linear relation $Y = KX$ holds (K is $p*n$). Then their covariance matrices satisfy the relation: $P_Y = KP_XK^T$.*

In our case, we are interested in the relation given by equations 2.1:

$$\begin{bmatrix} x \\ y \end{bmatrix} = f\left(\begin{bmatrix} d \\ \alpha_1 \\ \alpha_2 \end{bmatrix}\right) \tag{2.2}$$

Function f of equation (2.2) is not linear, so it is not possible to directly use rule 1. In that case, the standard solution is to linearize f around the values of d, α_1 and α_2, and then to apply the result.

In addition, d is not a measurement, but a value calculated using the coordinates of the known points A and B. If the covariance matrices of the corresponding coordinates are known, then it is possible to calculate the variance of d by the same method: linearize the relation $d = g(x_A, y_A, x_B, y_B)$ around the current values of x_A, y_A, x_B, y_B and apply the above result. We thus obtain the variance σ_d^2 of d. In the sequel, we suppose that it is known.

Let M be the "measurement" vector $M = [d, \alpha_1, \alpha_2]^T$. We assume that α_1, α_2 are independent random variables, unbiased (their mean error is zero) and hence characterized by their common variance σ_α^2. The variance is the same because α_1 and α_2 are measured using the same instrument. The random variables are independent because they are the result of two independent measurements. We further consider that d is independent from α_1 and α_2. This is a little more debatable,

as d may be determined using the coordinates of points A and B which have themselves been obtained using the same goniometer. But if d is measured directly as a distance, the hypothesis certainly holds. Under these reasonable assumptions, the covariance matrix of M can be written as:

$$C_M = \begin{bmatrix} \sigma_d^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$$

Let J be the Jacobian matrix of f with respect to M:

$$J = \begin{bmatrix} \dfrac{\partial x}{\partial d} & \dfrac{\partial x}{\partial \alpha_1} & \dfrac{\partial x}{\partial \alpha_2} \\ \dfrac{\partial y}{\partial d} & \dfrac{\partial y}{\partial \alpha_1} & \dfrac{\partial y}{\partial \alpha_2} \end{bmatrix}$$

Then applying rule 1, we obtain C_p as: $C_p = J\, C_M\, J^T$

Exercise 2 (Calculating a Jacobian matrix)**.** *Calculate the analytic expression of J.*

2.1.2 Static 2D localization using distances to known points

We consider here the problem of $2D$ localization, but the principle remains the same in $3D$. As illustrated by figure 2.3, measuring the distances d_1 and d_2 to two known points $B_1 = [x_1, y_1]^T$ and $B_2 = [x_2, y_2]^T$, the coordinate vector of the unknown point $[x, y]^T$ is defined as the solution of the system of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2 \end{cases} \tag{2.3}$$

When only two measurements are used, the system yields two solutions, of which one corresponds to the desired point. Two situations arise: either the approximate location of the point is known beforehand, allowing disambiguation, or it is completely unknown. In the latter case, a third measurement, if available, defines a unique location. In practice, due to measurement errors, the three circles will not intersect at a common location and the system will have to be solved in the least squares sense. The residual error then gives an idea of the precision of the localization process, just as explained for the traditional sailor's method.

Exercise 3 (Solving the static 2D problem in a local frame)**.** *Define a local frame R_B such that B_1 is the origin and B_1B_2 is the X axis, and solve for ${}^BP$ (coordinates of P in R_B). Express the $3*3$ transformation matrix ${}^0T_B$ and give the positions ${}^0P$ as a function of ${}^0T_B$ and ${}^BP$.*

Exercise 4 (Solving the static 3D problem in a local frame)**.** *Define a local frame R_B such that B_1 is the origin, B_1B_2 is the X axis, B_1, B_2, B_3 define the XY plane, with B_3 having a positive y coordinate, and solve for ${}^BP$ (coordinates of P in R_B). Express the $4*4$ transformation matrix ${}^0T_B$ and give the positions ${}^0P$ as a function of ${}^0T_B$ and ${}^BP$.*

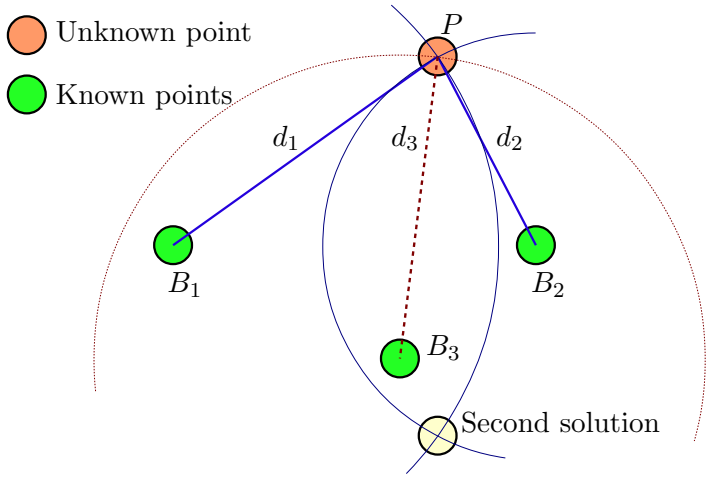


Figure 2.3: localization of a point by measuring distances to known locations.

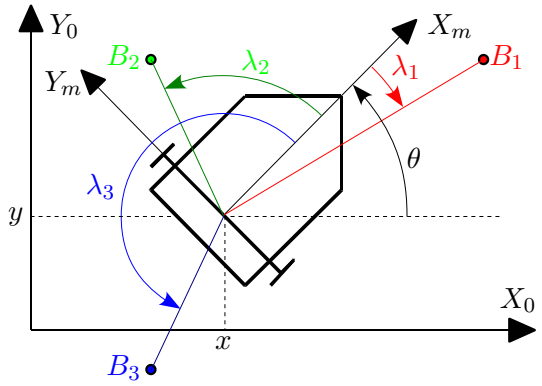


Figure 2.4: The vehicle and its sensor measuring the azimuth angles of three beacons B_1, B_2 and B_3.

2.1.3 Static 2D localization using azimuth angles

Let us now turn our attention to a slightly more complex situation, depicted by figure 2.4.

The vehicle (which is here motionless) is equipped with a sensor which measures the azimuth (or bearing) angle of several beacons. The azimuth angle is the angle between an axis attached to the vehicle passing through the sensor center and the axis from the sensor center to the beacon (see λ_1 to λ_3 on figure 2.4).

The system measures the azimuth angle of three beacons at known locations B_1, B_2 and B_3 and the problem is to determine the vehicle's posture vector $[x, y, \theta]^T$.

A few remarks first:

- The problem is significantly different from the sailor's problem: it cannot be solved by drawing lines on a map where the beacons are marked. The reason is that the angles are not measured with respect to a direction attached to the Earth, but to the vehicle, and the heading θ of the vehicle is unknown.
- The system can use artificial beacons, set in the environment for the purpose of localization, or natural landmarks, provided the system is able to identify them.
- Many hardware setups can provide the measurements used in this example:
  - A camera can detect one or more beacons at a time and provide their azimuth angles (and could also measure elevation angles, see later).
  - A rotating laser beam can be used to detect passive catadioptric beacons using the light returned to the sensor.
  - A rotating linear camera can also be used and would also provide elevation angles (see later).

Let us now see a method to calculate the posture of the vehicle in this problem, which was first presented in [Le Corre and Peyret, 1990]. We will be using several classical techniques. Firstly, we will use the distances $b_i = dist(M, B_i)$ as *intermediate unknowns*. We have already said in the surveyor's problem that it was often convenient when dealing with goniometric sensors. Secondly, we will use *invariants*, namely the distances between two beacons, which are independent of the frame in which the distance is calculated. This will lead to a fairly simple, although non linear system of equations in the new unknowns b_i.

Step 1 Express the coordinates of the beacons B_i in the mobile frame R_m. The coordinates of beacon number "i" in R_m will be denoted as $[{}^m x_i, {}^m y_i]^T$. They are obtained by projecting the vector MB_i onto axes X_m and Y_m of R_m.

$$\begin{cases} {}^m x_i = b_i \cos \lambda_i \\ {}^m y_i = b_i \sin \lambda_i \end{cases}$$

Step 2 Write the squared inter-beacon distances d_{ij}^2 in frame R_m. Of course, the values d_{ij}^2 are known since the positions of the beacons are assumed to be known in R_0. The d_{ij}^2 can be shown to be equal to:

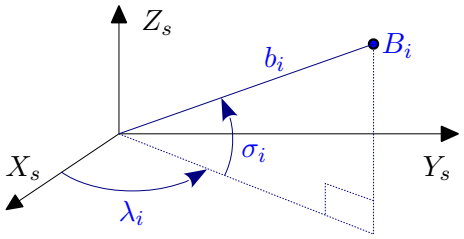


Figure 2.5: Azimuth (λ) and elevation (σ) angles of beacon B_i in sensor frame R_s.

$$d_{ij}^2 = b_i^2 + b_j^2 - 2b_i b_j \cos(\lambda_i - \lambda_j)$$

Step 3 Solve for b_1, b_2 and b_3.

In the case of this problem, we have a system of three equations corresponding to d_{12}^2, d_{13}^2 and d_{23}^2. The system has three unknowns: b_1, b_2 and b_3 (the λ_i are measurements).

The system can be solved analytically (see [Khalil and Murareci, 1993], in French), but the expressions are very complex. It is far more common to solve that type of system numerically, although this requires an initial value of the unknowns. It is also possible to solve without such initial values, for example using interval analysis [Jaulin and Walter, 1993], but such methods are beyond the scope of this course.

Step 4 Final determination of ${}^0T_m$.

Once b_1, b_2 and b_3 have been determined, the coordinates of the beacons in R_m are also known, by replacing b_1, b_2 and b_3 by their values in the equations of step 1. If a frame R_b has been attached to the beacons, we then have ${}^mT_b$, and hence ${}^0T_m$ using equation (1.1).

Please note that R_b is the equivalent of R_e in figure 1.1. Typically, it is defined with B_1 as origin, and B_1B_2 as axis X_b.

Exercise 5 (Advantage of using distances as intermediate variables)**.** *Write the system of equations in x, y and θ for this problem. Compare to the system we obtained above.*

2.1.4 Static 3D localization using azimuth and elevation angles

We now consider the case when the vehicle (still motionless) is equipped with a sensor which can measure both the azimuth and elevation (or height) angle of a beacon, as depicted in figure 2.5.

The new problem to solve is to find the full pose (position and attitude) of the sensor frame R_s (and hence of the mobile frame R_m) knowing the azimuth and elevation angles of three beacons at know locations B_1, B_2 and B_3. This solution is presented in [Le Corre and Peyret, 1990].

Figure 2.6 shows a possible implementation of such a sensor. The sensor is made of a linear CCD camera which rotates around an axis equipped with an optical encoder. The optic center of the camera lies on the rotation axis. The camera detects any light source located in the plane defined by the CCD sensor and the optic center. In the case depicted in the figure, the beacon is a dual light source, in order to be distinguishable from other light sources like the sun for example. Experimental results using this sensor are given later in the document.

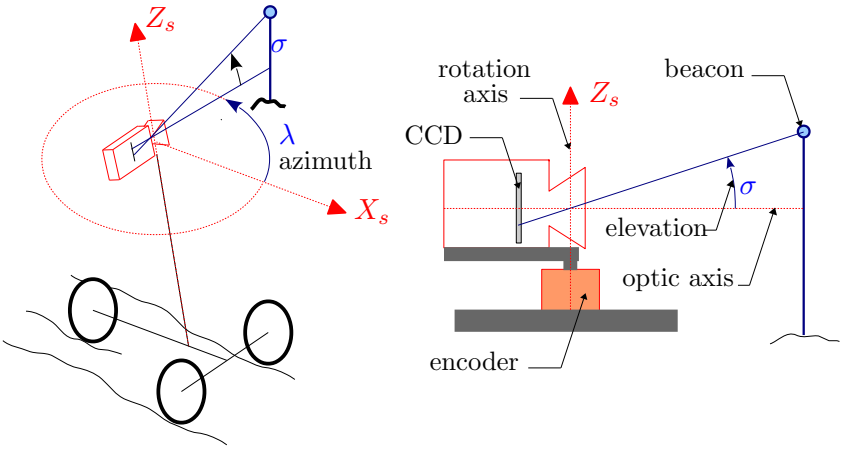


Figure 2.6: Implementation of the azimuth and elevation sensor using a rotating linear CCD camera.

The steps to solve for the position and attitude of the sensor frame (and hence the mobile frame) are similar to those of the $2D$ case.

In the following solution, we assume that R_m and R_s coincide.

Step 1 $3D$ Coordinates of the beacons in the mobile frame R_m.

The coordinates of beacon number "i" in R_m are denoted as $[{}^m x_i, {}^m y_i, {}^m z_i]^T$. They are obtained by projecting vector MB_i onto axes X_m, Y_m and Z_m of R_m.

$$\begin{cases} {}^m x_i = b_i \cos\sigma_i \cos\lambda_i \\ {}^m y_i = b_i \cos\sigma_i \sin\lambda_i \\ {}^m z_i = b_i \sin\sigma_i \end{cases}$$

Step 2 Write the squared inter-beacon distances in frame R_m.

$$d_{ij}^2 = b_i^2 + b_j^2 - \alpha_{ij} b_i b_j$$

With:

$$\alpha_{ij} = 2\left(\cos\sigma_i \cos\lambda_i \cos\sigma_j \cos\lambda_j + \cos\sigma_i \sin\lambda_i \cos\sigma_j \sin\lambda_j + \sin\sigma_i \sin\sigma_j\right)$$

The term α_{ij} is a constant for a given set of measurements, so *the equations have exactly the same form as in the* $2D$ *case!*

Step 3 Solve for b_1, b_2 and b_3.

Same as in the $2D$ case.

Step 4 Final determination of ${}^0T_m$.

Same as in the $2D$ case.

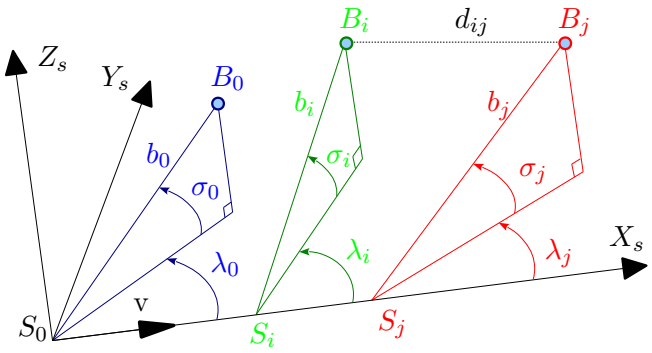


Figure 2.7: Azimuth and elevation measurements in the dynamic 3D case.

This is where you should be convinced of the interest of using distances as intermediate unknowns. We end up solving a system of *three* equations with *three* unknowns only, when the initial problem has *six* unknowns (three for position and three for attitude). Moreover, the equations are simple. The overall task is incredibly simpler than by using, say, Cartesian coordinates for position and Euler angles for attitude. If you still have any doubt, try!

2.2 An example of dynamic absolute localization (deterministic algorithm)

The solution to the so-called "dynamic" problem was first presented in [Le Corre and Garcia, 1992].

It should be noted that, if no other sensor than the azimuth and elevation sensor is available, and the vehicle moves, we are bound to use hypotheses about the vehicle motion. The simplest choice is to make the following assumptions:

- The vehicle moves on a surface which can be considered as locally planar.
- Locally, the vehicle can be assumed to move on a straight line at constant speed.

In the previous cases, with the vehicle motionless, we did not have to care about the position of the sensor frame with respect to the mobile frame. Here, we assume that it is located in such a way that the speed vector of the vehicle is directed along axis X_s of the sensor frame R_s. In the case of a $(2,0)$ robot for example, this could be achieved by setting the origin of the sensor frame at the midpoint of the robot wheelbase, with axis X_m perpendicular to the axle of the wheels and in a plane parallel to the rolling surface.

In the present case, assuming the sensor is a rotating sensor, the various beacons will not be detected at the same vehicle location. With three beacons, the sequence of detections could be B_1, B_2, B_3, and B_1 seen from a new location, etc. Figure 9 shows the measurements of several beacons along a (local) straight line path. S_0 is the sensor frame at instant t_0, which is defined as the instant of the first of the set of measurements used to solve the localization problem. How many are necessary is determined later. The first beacon of the sequence is renumbered B_0 for convenience. The beacons can always be renumbered in such a way that they are detected in the order B_0, B_1, $B_2 \ldots$ At some point, B_3 can be the same as B_0 for example.

How many unknowns do we have? The position and attitude at t_0 completely define the speed direction. In addition to these variables, there remains only the velocity v to determine. So the number of unknowns is seven.

How many beacons do we need to detect? Three is not enough, because we have only six independent measurements for seven unknowns. Four is enough, and we even get more equations than strictly required, which will allow the system to be solved in a least squares sense. Again, this does not mean that four different beacons are required: detecting one of three beacons from two different locations is equally valid.

Now that we know how many measurements we need, we can also make clearer the adverb "locally" used in the assumptions: the assumptions should be valid during the time interval necessary to detect four beacons. We will not detail the equations. The solution, which is presented in [Le Corre and Garcia, 1992], is very similar to the static case. The first step is to express the coordinates of the beacons in the sensor frame as of time t_0. For B_0, there is no change with respect to the previous case. For other beacons, a term $v(t_i - t_0)$ is added to the X coordinate. Then the squared inter-beacon distances are calculated. There are six of them and we solve for five unknowns, namely b_0, b_1, b_2, b_3 and v. The rest is the same as in the static case.

So far, so good. But what are the problems?

- The motion assumptions are very simple and may not correspond to reality.
- If we take a more general model for the motion, it will be described by a larger number of parameters. Hence, the model will have to be valid over a longer time interval, since it will require detecting a larger number of beacons to solve for the unknowns.
- In any case, the need for a motion model limits the motion dynamics that the system will be able to handle.

We will see later that it is far more efficient to add proprioceptive sensors to the vehicle in order to measure the actual displacements between two time instants.

Chapter 3

A relative localization method: odometry

Odometry is the most popular relative localization method in mobile robotics. Basically, it consists in expressing the translational and rotational speed of the robot as functions of the rotation speed of some of the robots' wheels. The corresponding equations are then put into discrete form to yield relations between elementary translations and rotations of the robot frame and elementary rotations of the wheels.

We study here the case of a robot of type $(2, 0)$ but it is possible to consider other robot types once their kinematic model has been established.

3.1 Equations of odometry

The kinematic model of the robot is the following (See the Modeling part of the Mobile Robots course. The only difference is that the left and right radii are distinguished):

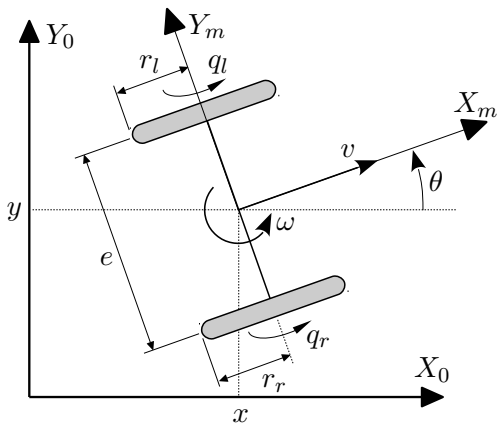


Figure 3.1: Notations for the $(2, 0)$ robot.

$$\begin{cases} v = (r_r \dot{q}_r + r_l \dot{q}_l)/2 \\ \omega = (r_r \dot{q}_r - r_l \dot{q}_l)/e \\ \dot{x} = v \cos\theta \\ \dot{y} = v \sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{3.1}$$

The equations of odometry are obtained by sampling the continuous model. We will note Δq_k the elementary rotation of a wheel between time instants t_k and t_{k+1}: $\Delta q_{r,k}$ and $\Delta q_{l,k}$ for the right and left wheels respectively. One possible set of equations for odometry is:

$$\begin{cases} \Delta D_k = (r_r \Delta q_{r,k} + r_l \Delta q_{l,k})/2 \\ \Delta\theta_k = (r_r \Delta q_{r,k} - r_l \Delta q_{l,k})/e \\ \theta_{k+1} = \theta_k + \Delta\theta_k \\ x_{k+1} = x_k + \Delta D_k \cos(\theta_k + \Delta\theta_k/2) \\ y_{k+1} = y_k + \Delta D_k \sin(\theta_k + \Delta\theta_k/2) \end{cases} \tag{3.2}$$

There is no single way to sample a non linear system. In other books or papers, you may find the following forms for the last two equations.

$$\begin{cases} x_{k+1} = x_k + \Delta D_k \cos\theta_k \\ y_{k+1} = y_k + \Delta D_k \sin\theta_k \end{cases} \tag{3.3}$$

Or:

$$\begin{cases} x_{k+1} = x_k + \Delta D_k \cos\theta_{k+1} \\ y_{k+1} = y_k + \Delta D_k \sin\theta_{k+1} \end{cases} \tag{3.4}$$

The first form is exact if the robot motion between time instants t_k and t_{k+1} is a circular arc, while the second version would be exact if the robot were to first perform a translation of length ΔD and then a rotation of angle $\Delta\theta$. The first form proves to be a little more precise when optimal conditions to apply odometry are met (*cf. infra*).

3.2 Expression of posture error

When analysing the performance of a localization algorithm, it is common to display the errors: differences between the estimated and real positions. It is convenient to project the error vector onto the robot frame, as it leads to interpretation of x errors as longitudinal errors (the estimated position being ahead or behind the real position) and of y errors as transverse or lateral errors (the estimated position being to the left or to the right of the real position). There is no such interpretation if the errors are calculated in a fixed frame.

The posture error in the absolute reference frame is simply:

$${}^0E = {}^0P_{real} - {}^0P_{est} = \begin{bmatrix} x_{real} - x_{est} \\ y_{real} - y_{est} \\ \theta_{real} - \theta_{est} \end{bmatrix}$$

${}^{est}E$ is obtained by the following equations:

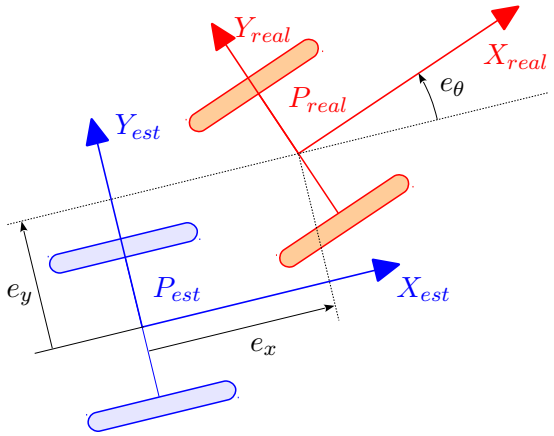


Figure 3.2: Posture error in robot frame.

$$ {}^{est}E = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = {}^{est}R_0\left(\theta_{est}\right)\left[{}^0P_{real} - {}^0P_{est}\right] $$

$$ {}^{est}E = \begin{bmatrix} \left(x_{real} - x_{est}\right)\cos\theta_{est} + \left(y_{real} - y_{est}\right)\sin\theta_{est} \\ -\left(x_{real} - x_{est}\right)\sin\theta_{est} + \left(y_{real} - y_{est}\right)\cos\theta_{est} \\ \theta_{real} - \theta_{est} \end{bmatrix} \tag{3.5} $$

3.3 Analysis of simulation results

There are several sources of errors in the odometry process. Simple simulations can help understand which ones have a dominant effect on the performance of odometry. In the following, we study the influence of errors on geometrical parameters of the robot, of the sampling period and of the resolution of wheel encoders.

In all the simulations shown, the robot always follows the same circular trajectory, centred at the origin and of unit radius. The simulation program generates the corresponding wheel encoder values along time, taking into account the real wheel radii and track gauge. The simulated robot has wheels of radius 10 cm and a track gauge of 40 cm.

The relative localization algorithm in turn applies the odometry equation at a given sampling rate, and with possibly erroneous wheel radii and track gauge. The odometry estimated position differs from the real position, and the posture error is calculated as indicated above.

In no case do we simulate here wheel slippage or the complex phenomena due to the fact that actual wheel-ground contact, contrary to the model, is not punctual. These can be very influential errors, but our aim here is to evaluate the relative influences of the radius and track gauge errors, sampling rate and encoder resolution.

3.3.1 Odometry in a near-perfect world

Given the fact that we don't take slippage into account, if there are no errors on geometrical parameters, if the sampling frequency is high and the encoders have a very high resolution, only

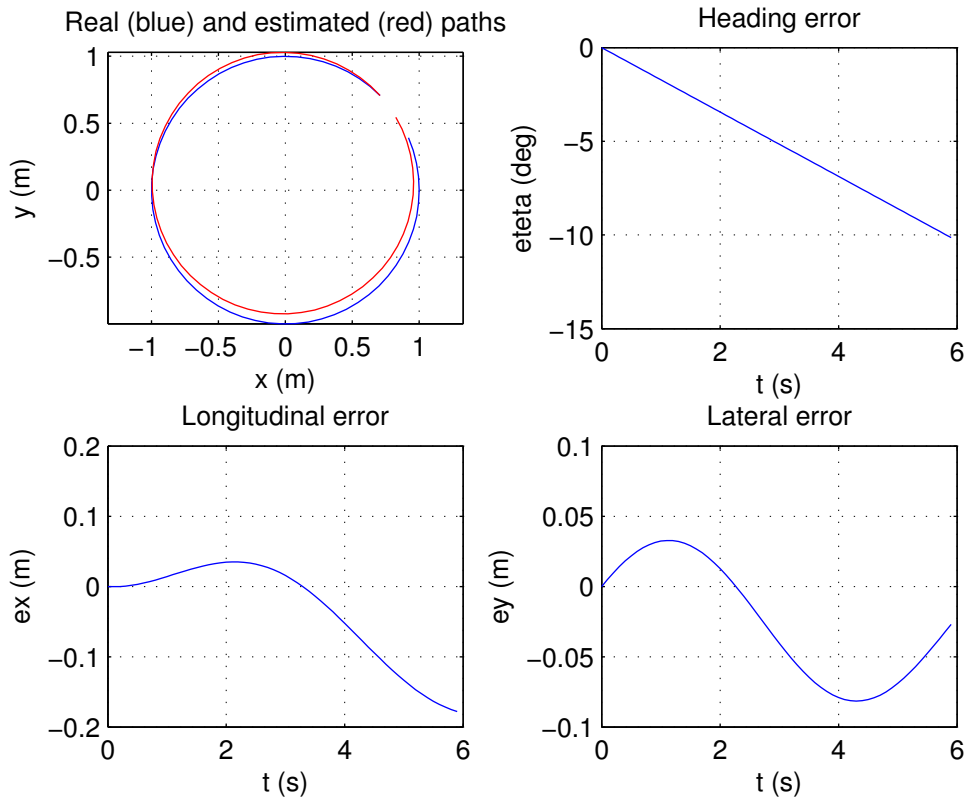


Figure 3.3: Influence of 1% error on right wheel radius

numerical errors remain, and they generate very little drift of the relative localization. But reality is nowhere close to this idealized situation.

3.3.2 Influence of errors on robot parameters

This example shows the effect of a 1% wheel radius error. Sampling rate is high, encoder resolution simulated as "infinite" (no quantization error introduced, only numerical errors).

The error is very severe, generating more than 10°of error on the robot heading and close to 20 cm of position error for a 6.28 m trajectory, or around 3% of the travelled distance. For tyres, variations of tyre pressure and wear can easily generate such an order of magnitude of error, and more.

Track gauge errors generate similar problems. Also remember that, for wide tyres, it is not easy to decide where "the" contact point is. If an "equivalent contact point" is considered, its location is probably influenced by static (*e.g.* cargo) and dynamic (inertial forces) load distribution.

Practical notes:

- When plotting results, always indicate x and y units.
- Plotting the estimated and real paths is not sufficient: you have to plot errors. Otherwise, only lateral errors are visible, not longitudinal errors. Plus, the overall scale can prevent you

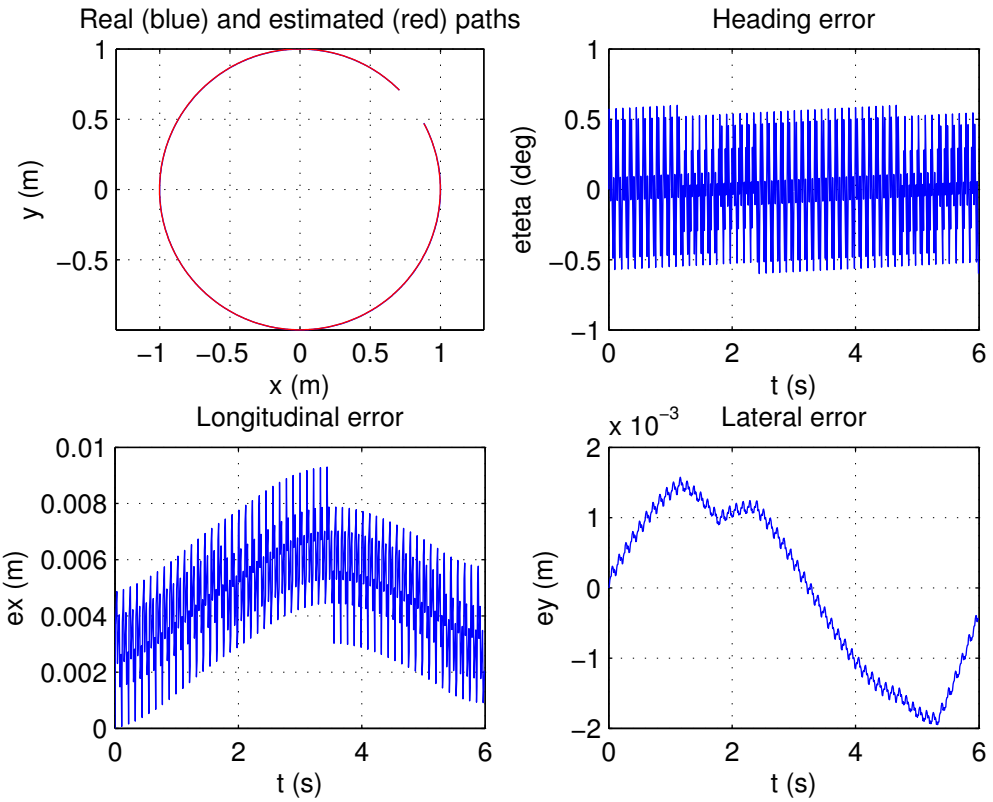


Figure 3.4: Influence of a low (100 dots per wheel revolution) encoder resolution.

from seeing the order of magnitude of the errors.

3.3.3 Influence of encoder resolution

This simulation (figure 3.4) shows that a low resolution encoder is sufficient for odometry. Here, encoders have 100 dots per wheel revolution. Thus, one dot corresponds to an elementary translation of approximately 6 mm for the wheel. With respect to the order of magnitude of errors on the geometrical parameters, the influence is negligible.

3.3.4 Influence of the sampling rate

Of course, the sampling rate is related to the dynamics of the robot: maximum speed, maximum rotation speed... Altogether, odometry does not require high sampling frequencies. Moreover, the corresponding equations represent a low computational cost, so that odometry is not a computer intensive process.

The reason for using higher resolution encoders on a mobile robot is more related to the control. Some control algorithms make use of the current speed of the robot. Low resolution encoders generate noise on the speed estimation, which may be detrimental to the control.

3.4 Classes of odometry errors

In [Borenstein and Feng, 1995], odometry errors can be classified into systematic and non systematic errors. *Systematic errors* are errors which have permanent effect, while *non systematic errors* correspond to accidents which happen at particular time instants.

Systematic errors include the effect of finite resolution encoders and sampling frequency, but we have seen that these sources of error are not dominant. They also include errors on the geometrical parameters of the robot, like wheel radius and track gauge. Also important are inconsistencies between the kinematic model and reality, such as those created by misalignment of wheel axes.

Typical non systematic errors are caused by floor unevenness (including unexpected objects) which results in a travelled horizontal distance different from the measured (non horizontal) distance. Other examples include slippage, which may be due to slippery floor, overacceleration and fast turning resulting in skidding, external forces applied to the mobile, internal forces (*e.g.* generated by castor wheels), non-point wheel-to-floor contact...

Systematic errors are very serious because they accumulate over time. They will typically dominate in most indoor applications. Non systematic errors can play a major role on rough, irregular terrains. They are very difficult to handle because they are unexpected.

Applications which use odometry commonly use error evolution models based on worst case analysis, typically to determine inter-landmark distances. Non systematic errors can cause such models to fail. The error can increase very fast, in a way not consistent with the model. As a result, a robot controlled using the estimated position may not be able to reach the next landmark.

An interesting idea to correct errors induced by accidental slippage using a low-cost gyro can be found in [Borenstein and Feng, 1996]: in essence, the gyro information is taken into account only when odometry-calculated rotation speed and gyro output do not agree. Elsewhere, odometry-calculated rotation speed is preferred because of the high noise in the low-cost gyro output.

3.5 Evaluation of odometry performance

Odometry performance is usually characterized by measuring the difference between the actual and estimated robot positions after a certain travelled distance. The performance is expressed as a percentage of the travelled distance, and the tests are repeated a certain number of times to obtain a statistically meaningful result.

The figures and methods presented in this part are taken from [Borenstein and Feng, 1995].

3.5.1 Unidirectional square path test

The principal of the square path test is the following:

- Make the robot perform a nominally close trajectory (here a square).
- Compare the actual end position to the end position calculated by odometry.
- Perform this test a number of times to obtain a statistically meaningful set of return position errors.

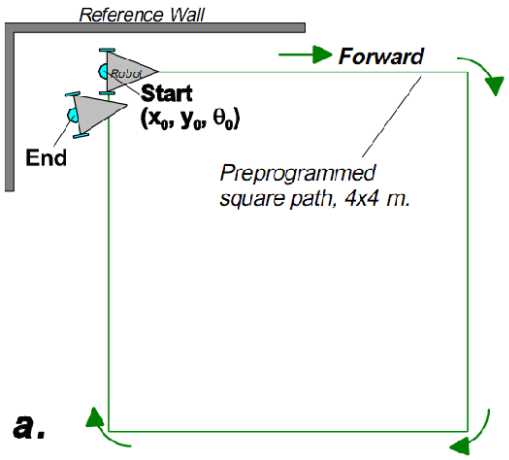


Figure 3.5: Unidirectional square path test

Practical remarks:

- The end position should not be compared to the *desired* end position but to the *actual* end position. Otherwise, the result may include errors due to the control algorithm not being able to precisely reach the desired position.

- It is fairly simple to materialize a frame using two perpendicular walls. Measuring the distances of three points of the robot (known in robot frame) to the walls allows the position and heading of the robot to be determined with respect to the frame.

- The reason for using a square path is that, while a given robot may not be able to perform complex paths, performing straight line motions and rotations on the spot is the minimum any reasonable robot should be able to do, though some robots like (1,1) robots with limited steering angle, may not be able to rotate "on the spot".

The (unidirectional) square path test is illustrated in figure 3.5.

The unidirectional square path test is not a good test of odometry performance. The reason is that two different sources of error can yield the exact same error. Or that two sources of error can cancel each other out, yielding the illusion of a perfect result, as examplified by figure 3.6.

Of course, if an absolute localization system were used in real time during the motion, the errors along the path would show. The idea of the square path test is precisely that it does not require such an equipment.

3.5.2 Bidirectional square path test

A simple way to avoid the previous undesirable phenomenon is to perform the path in the two directions, rotating clockwise and counter-clockwise. If errors compensate in one direction, they sum up in the opposite direction, as shown in figure 3.7.

The bidirectional square path test has been introduced by Borenstein and Feng in [Borenstein and Feng, 199 and is also known as *University of Michigan Benchmark* or *UMBmark*.

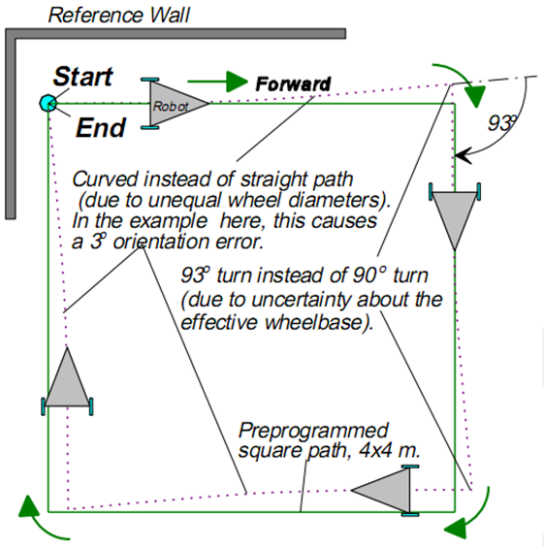


Figure 3.6: Influences of track gauge and wheel radius errors can cancel each other out.

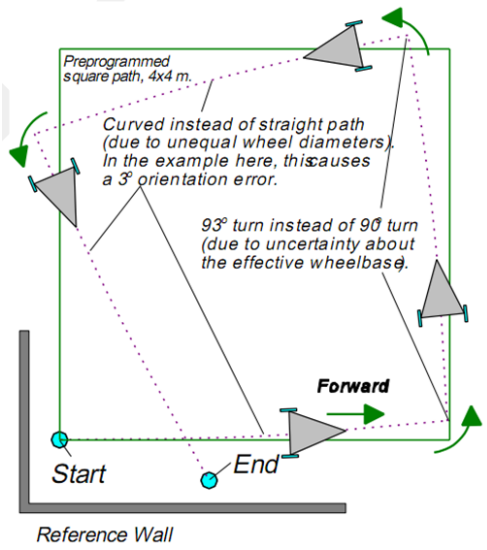


Figure 3.7: The effects of the two sources of error sum up when rotating in the other direction.

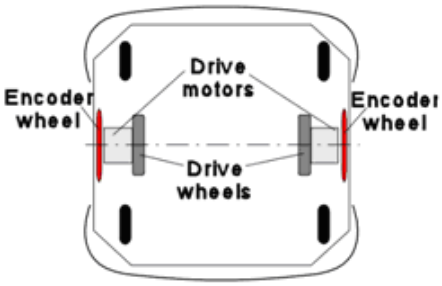


Figure 3.8: Auxiliary wheels for odometry.

3.6 Practical considerations

- Orientation errors grow faster for vehicles with narrow track gauge.
- Castor wheels can induce slippage when reversing direction, especially when they bear a large part of the load.
- Odometry wheels should be thin and not compressible for good determination of wheel radius and track gauge. This may not be compatible with load bearing, but is sometimes possible for lightweight robots with little power.
- Odometry can use auxiliary wheels. The idea is to add a pair of thin, non load bearing wheels, typically made of metal with a thin rubber band, as shown in figure 3.8.

3.7 Experimental results with an outdoor robot

The results presented hereafter were obtained with the outdoor mobile robot "Melody" shown in picture 3.9. Melody is a fairly large robot used to test $6D$ localization systems designed for paving applications (see applications examples).

The graph of figure 3.10 shows the actual and estimated paths of the robot, the estimation being performed using odometry. The robot was slowly moving on a wet, even muddy lawn.

The final error is 1.15 m for a path length of 26 m, or approximately 4.4% of the travelled distance.

There is a very important difference with the performance tests presented above, though. In the square path test, the robot's initial posture is by definition $(0, 0, 0)$ and the goal posture is defined with respect to this origin. As a consequence, the initial error is null. But in the case presented here, the robot calculates its initial posture using the algorithm presented in paragraph 2.1.3. Hence, there is an initial error, including an initial heading error. So this is more demanding than the square path test.

3.8 Odometry on non planar surfaces

Provided some measurement (or estimation) of attitude angles with respect to the horizontal is available, it is possible to perform $3D$ odometry. The idea is the same as in $2D$: measure the

Figure 3.9: The outdoor mobile robot "Melody".

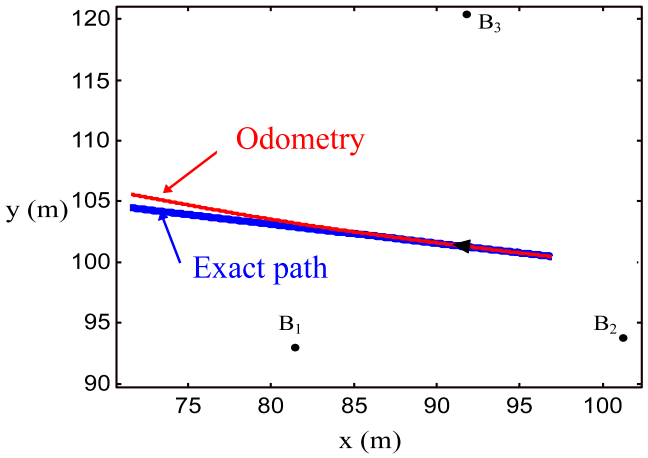


Figure 3.10: Actual and odometry estimated paths of the robot "Melody".

elementary translations and rotations of the robot between two time instants, then project them onto the axes of a fixed reference frame using attitude information.

We show here an example of using two inclinometers to measure the attitude of the robot. We assume here that the inclinometers directly measure angles with respect to the horizontal. This is only the case if accelerations can be considered as negligible. Indeed, inclinometers are actually sensitive to accelerations.

The inclinometers which were used in the implementation (on the robot "Melody" mentioned earlier) are based on a pendulum. The angle of the pendulum with respect to the instrument frame is measured. It is 90 deg when the sensor frame is horizontal. The pendulum moves in a fluid, the viscosity of which can be tuned to low-pass filter the accelerations. Alternatively, the pendulum can be actively controlled to remain perpendicular to the sensor frame. In this case, the torque is measured and yields the angle of inclination, provided accelerations other than gravity are negligible.

For the equations of odometry on non planar surfaces, please refer to the slides.

Chapter 4

Sequential use of relative and absolute localizations

This chapter describes an example of a localization system for robots, which relies on the same idea as the traditional sailor's method: use relative localization for a while and periodically recalculate an absolute localization, before the error of relative localization becomes too large.

4.1 System description and principle

In the case described here, the mobile robot uses odometry and recalculates full posture information (position and heading) when it passes over a landmark made of a set of two permanent magnets inserted in the floor at known locations. The set of two magnets is called a *station*. Of course, the admissible distance between stations depends on the performance of odometry for the particular type of robot being used.

The robot uses a linear magnet detector. This detector measures the position of the magnet along the detector at the moment when the detector passes over the magnet. Several implementations of such a sensor can be used:

- A set of solenoids: due to the robot motion, the magnetic flux in the solenoids varies, which generates an electrical current in the solenoid(s) located above the magnet.

- A set of reed switches along the sensor: when a reed switch is close enough to the magnet, it switches on. The state of the sensors can be read as a binary n-bit word, where n is the number of reed switches.

As indicated in the figure below, we assume that the sensor is aligned with the axis common to the driving wheels of the $(2, 0)$ robot. This assumption is not restrictive and slightly simplifies the equations.

Without loss of generality, we also suppose that the sensor measures the magnet position with respect to its center and that the center coincides with the reference point M of the robot.

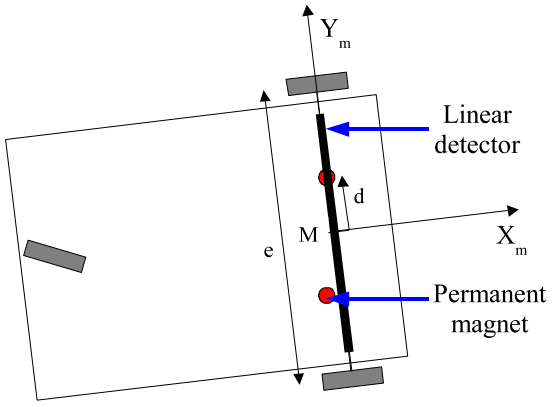


Figure 4.1: Robot, sensor and permanent magnets.

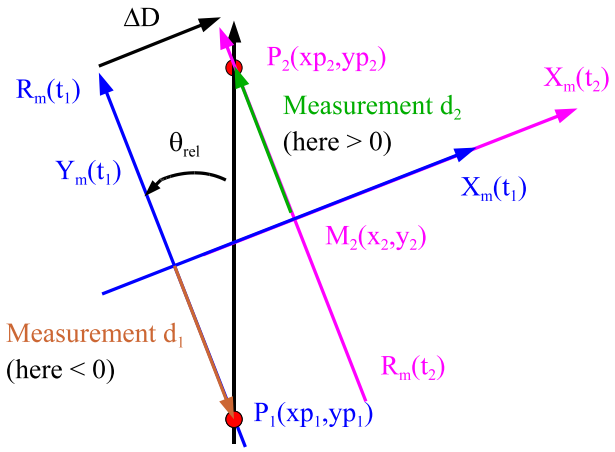


Figure 4.2: Robot frame and measurements when passing over a station. The robot frame is represented at the two time instants t_1 and t_2 of passing over magnets at known locations P_1 and P_2 in the absolute reference frame, yielding measurements d_1 and d_2.

4.2 Absolute localization equations

In the original systems using this principle, the stations were typically located perpendicular to the nominal path of the robot. Hence, the linear detector would be near parallel to the station when passing over it. As a consequence, the time instants of crossing both magnets were very close. In addition, the stations were always installed at locations where the robot's nominal path was a straight line. That's why the equations hereafter are established under the assumption that the motion of the robot between the time instants of passing over each magnet is a straight line.

On figure 4.2, the relative orientation of the robot with respect to the station is θ_{rel} and is normally small. ΔD is the distance travelled between t_1 and t_2. Our goal is to calculate the posture of the robot at time t_2: (x_2, y_2, θ_2).

$$\sin(\theta_{rel}) = \frac{\Delta D}{dist(P_2, P_1)}$$

Then the absolute orientation of the robot is easily obtained from its relative orientation and

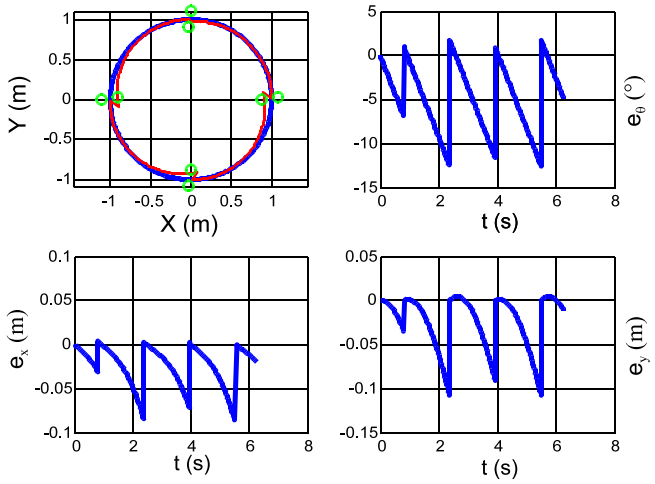


Figure 4.3: Simulation results of the magnet based localization system.

the orientation of vector P_1P_2.

$$\theta = \arctan 2\,(yp_2 - yp_1, xp_2 - xp_1) + \theta_{rel} - \frac{\pi}{2}$$

The coordinates of the point of the linear detector which is above P_2 can be expressed as a function of x_2, y_2, θ and d_2, and these coordinates are equal to (xp_2, yp_2):

$$P_2 = (x_2 - d_2 \sin\theta\ ,\ y_2 + d_2 \cos\theta) = (xp_2, yp_2)$$

Finally the posture of the robot at time instant t_2 is given by:

$$\begin{cases} x_2 = xp_2 + d_2 \sin\theta \\ y_2 = yp_2 - d_2 \cos\theta \\ \theta_2 = \arctan 2\,(yp_2 - yp_1, xp_2 - xp_1) + \theta_{rel} - \dfrac{\pi}{2} \end{cases} \tag{4.1}$$

4.3 Sample simulation results

Figure 4.3 shows simulation results of the previously described system. The simulated robot and real trajectory are the same as in the simulation of odometry in paragraph 3.3. The green circles represent the magnets. They have been positioned in such a way that the robot is not perfectly perpendicular to the station when it crosses it. Otherwise, instants t_1 and t_2 would always be equal. This is dual to the actual case, when stations are perpendicular to the nominal path and the robot does not follow the nominal path exactly.

Here, errors have been purposely exaggerated by introducing a 10% error on the radius of the right wheel. The robot starts at point $(\sqrt{2}, \sqrt{2})$ and arrives at $(0, 1)$ slightly before $t = 1s$ with an error of approximately 3 cm in x and y and 7°in heading. Crossing the first station causes a new absolute location to be calculated (previous odometry estimate is discarded): errors are not perfectly cancelled due to the various sources of errors (*e.g.* magnetic sensor resolution).

In this simulation, an additional source of error has been introduced: the equations to solve for the posture have been established under the assumption that the motion of the robot was a straight line, and this is not the case. By setting all other error sources to zero, it is possible to check the influence of a disturbance to this assumption. Increasing the circle radius reduces the disturbance and the errors after crossing the station, especially the orientation error.

You have received (or will receive soon) a Matlab program of this simulation. You are invited to manipulate the sources of error to visualize their respective influences.

Chapter 5

Hybrid localization method using a Kalman filter

The Kalman filter is the screwdriver of localization systems. It has been successfully used in many applications. It is reasonably computationally efficient and easy to program. We do not present the theory of Kalman filtering. We only give the equations and try to grasp their logic.

5.1 Extended Kalman filter equations

A non linear discrete system can be written in state-space form as:

$$\begin{cases} X_{k+1} = f(X_k, U_k) \\ Y_k = g(X_k) \end{cases} \tag{5.1}$$

The first equation of system (5.1) is generally called the *evolution equation*, while the second is called *observation equation*. X is the *state vector*, f is the *state function*. U is the system *input* and Y the system *output*. X, Y and U may have different dimensions, respectively n, p and m.

The observation equation expresses the outputs Y as a function of the state. Or, as it is sometimes presented "What you see as a function of where you are".

The equations are only a mathematical model of the real system. The differences between the behavior of the real system and the equations is classically represented by additive noises: α for the evolution equation and γ for the observation equation, characterized by their covariance matrices Q_α $(n * n)$ and Q_γ $(p * p)$:

$$\begin{cases} X_{k+1} = f(X_k, U_k) + \alpha_k \\ Y_k = g(X_k) + \gamma_k \end{cases} \tag{5.2}$$

Before proceeding with the equations, let us define a few notations. X_k denotes the state vector at a certain time instant t_k. The "hat" symbol, as in $\widehat{X}_k$, means an estimated value. As such, it could apply to other variables of the estimation process, not only the state. For example, we will calculate covariance matrices which are also *estimated* matrices. Still, we reserve the "hat" symbol to the state vector in order not to make notations uselessly complex.

We also use double index notations, as in $\widehat{X}_{k+1/k+1}$. The meaning is "estimation of X at time t_{k+1} using all information available until instant t_{k+1}". We also use $\widehat{X}_{k+1/k}$ for "estimation of X at time t_{k+1} using all information available until instant t_k". Essentially, the difference is that $\widehat{X}_{k+1/k}$ does not make use of the last measurement Y_k (it is called a *predicted* state), while $\widehat{X}_{k+1/k+1}$ does and is called an *estimated* state.

The equations of the extended Kalman filter are:

Prediction phase:

$$\begin{cases} \widehat{X}_{k+1/k} = f\left(\widehat{X}_{k/k}, U_k\right) \\ P_{k+1/k} = A_k \cdot P_{k/k} \cdot A_k^T + Q_\alpha \end{cases} \tag{5.3}$$

With:

$$A_k = \frac{\partial f}{\partial X}\left(\widehat{X}_{k/k}, U_k\right) \tag{5.4}$$

Notice that A_k is calculated using the latest evaluation of the state X.

When calculating the matrices, you need to recall how Jacobian matrices are defined.

Rule 2. *Let f be a real vector function $f(X) = [f_1(X) \dots f_p(X)]^T$ of the real vector $X = [x_1 \dots x_n]^T$. The Jacobian matrix of f with respect to X, denoted $\frac{\partial f}{\partial X}$, is the $p * n$ matrix defined as:*

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_p}{\partial x_1} & \cdots & \frac{\partial f_p}{\partial x_n} \end{bmatrix}$$

Estimation phase:

$$\begin{cases} \widehat{X}_{k+1/k+1} = \widehat{X}_{k+1/k} + K_k\left[Y_k - g\left(\widehat{X}_{k+1/k}\right)\right] \\ P_{k+1/k+1} = (I - K_k \cdot C_k)\, P_{k+1/k} \end{cases} \tag{5.5}$$

With: h

$$\begin{cases} C_k = \frac{\partial g}{\partial X}\left(\widehat{X}_{k+1/k}\right) \\ K_k = P_{k+1/k} \cdot C_k^T \left(C_k \cdot P_{k+1/k} \cdot C_k^T + Q_\gamma\right)^{-1} \end{cases} \tag{5.6}$$

Let us now shortly comment these equations.

The prediction phase in system (5.3) predicts the state at time t_{k+1} from the last estimated state $\widehat{X}_{k/k}$ and the known input U_k to the system. This corresponds to the numerical integration of a differential equation. That's why if we were to use only repeated prediction steps, $\widehat{X}_{k/k}$ would eventually drift with respect to the real state X_k. This is reflected by the second equation of the system. To get a simple picture of its meaning, imagine $n = m$, $A = I_{n*n}$ and $Q_\alpha = diag(\sigma_1^2, \dots, \sigma_n^2)$.

The estimation phase in system (5.5) states that the correction between $\widehat{X}_{k+1/k}$ and $\widehat{X}_{k+1/k+1}$, which is the effect of taking into account the measurement Y_k, is proportional to the difference

between the measurement Y_k and its expected value $\widehat{Y} = g(\widehat{X}_{k+1/k})$. The expected measurement $\widehat{Y}$ is the measurement we would obtain if the actual state X were equal to the predicted state $\widehat{X}_{k+1/k}$. Logically, if the difference is zero, the measurement does not produce any correction.

The term $C_k P_{k+1/k} C_k^T + Q_\gamma$ is the covariance matrix of $(Y - \widehat{Y})$. The covariance of the difference of two random variables is the sum of their covariances. Q_γ is the variance of Y. $C_k P_{k+1/k} C_k^T$ if the variance of $g(\widehat{X}_{k+1/k})$ as obtained by rule (1). It is logical that the gain be inversely proportional to the variance of this term. The gain being proportional to $P_{k+1/k}$ is also logical: if your predicted state is uncertain, you allow the measurement to have greater effect on it. On the other hand, if you are very confident on your predicted state, you won't let the measurement influence it too much. As to the term C_k^T, we will show its effect in the detailed example.

5.2 Extended Kalman filter with noisy input

In the previous paragraph, the input U to the system is considered as deterministic. It is sometimes necessary or simply convenient to consider the input as a random signal, *e.g.* when the input is measured with a certain amount of noise. We call U^* the noisy measurement of the input U. Classically, we consider that U^* is U disturbed by an additive noise β characterized by its $(m * m)$ covariance matrix Q_β.

In this case, the equations of the extended Kalman filter can be modified as follows:

$$\begin{cases} X_{k+1} = f\left(X_k, U_k\right) + \alpha_k \\ U_k^* = U_k + \beta_k \\ Y_k = g\left(X_k\right) + \gamma_k \end{cases} \tag{5.7}$$

The prediction phase becomes:

$$\begin{cases} \widehat{X}_{k+1/k} = f\left(\widehat{X}_{k/k}, U_k^*\right) \\ P_{k+1/k} = A_k \cdot P_{k/k} \cdot A_k^T + B_k \cdot Q_\beta \cdot B_k^T + Q_\alpha \end{cases} \tag{5.8}$$

With A_k and B_k defined as:

$$A_k = \frac{\partial f}{\partial X}\left(\widehat{X}_{k/k}, U_k^*\right) \quad and \quad B_k = \frac{\partial f}{\partial U}\left(\widehat{X}_{k/k}, U_k^*\right) \tag{5.9}$$

In the calculation of the predicted state and of matrices A and B, we have no other choice but to use the measured input, as the real input is unknown. The estimation phase is unchanged.

5.3 Application example

5.3.1 The system and its state-space equations

We again use the example of the mobile robot equipped with odometry and a rotating azimuth or bearing angle sensor, which we already used in paragraph 2.1.3. Previously, we did not use the wheel encoders. The robot, its sensor and the beacons are depicted on figure 5.1.

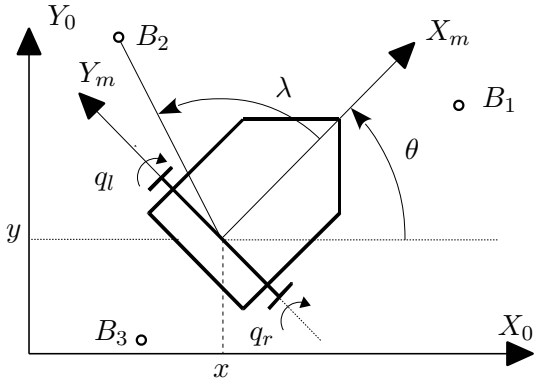


Figure 5.1: Robot, encoder wheels and goniometric sensor measuring the azimuth angle of a beacon.

The sensor rotates around an axis parallel to Z_0. It is not fast enough to neglect the robot motion during a full rotation of the sensor. As a consequence, it is not possible to use a quasi-static algorithm to perform the localization task.

We want to estimate the posture of the vehicule, which will be our state vector $X = [x, y, \theta]^T$. We use the discrete form of the equations that describe the kinematics of the vehicle, *i.e.* the odometry equations.

As stated in paragraph 3.1, several forms can be used. We use the form in system (3.3), which is slightly simpler. The corresponding equations are recalled here:

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \Delta D_k \cos\theta_k \\ y_k + \Delta D_k \sin\theta_k \\ \theta_k + \Delta\theta_k \end{bmatrix} = f(X_k, U_k) \tag{5.10}$$

With:

$$U_k = \begin{bmatrix} \Delta D_k \\ \Delta\theta_k \end{bmatrix} = \begin{bmatrix} (r_r \Delta q_{r,k} + r_l \Delta q_{l,k})/2 \\ (r_r \Delta q_{r,k} - r_l \Delta q_{l,k})/e \end{bmatrix} \tag{5.11}$$

It is equally valid to take the vector of elementary rotations of the wheels $[\Delta q_r, \Delta q_l]^T$ as input vector.

Note: any parameter of f which is neither in X nor in U must be a known constant. This is true in our case: only r_r, r_l and e, which are assumed known constants, have been left out. In some cases, unknown robot parameters can be identified by adding them to the state vector, as shown in [Garcia et al., 1995] for wheel radii.

We now need to write the observation equation. Recall that this equation expresses the outputs as a function of the state. In our case, it states what the goniometric sensor outputs Y would be, should the robot stand at X. In fact, *there are as many different observation equations as there are beacons*. To remind of that, we denote the observation function g_B for beacon $B = [x_B, y_B]^T$. From figure 5.1 we obtain:

$$\lambda + \theta = arctan2(y_B - y, x_B - x)$$

But the equation should not be left under that form. We want to obtain the form $Y = g(X)$, so we should have only λ on the lefthand side of the equality. In addition, we also indicate that the

equation corresponds to a certain time instant. We intentionally use a different index name, j, to remind that, due to the fact that we use a rotary sensor, there is not always a beacon in the line of sight of the sensor at each time instant t_i. Hence, we write the equation as:

$$Y_j = \lambda_j = arctan2(y_B - y_j, x_B - x_j) - \theta_j \tag{5.12}$$

Note: this means that not all prediction steps have a corresponding estimation step. If there is no measurement available, no estimation step can be performed. This corresponds to the general form of the hybrid localization algorithm presented in section 1.3.

We consider that the input is measured with noise. So the system is described by equations (5.7) and the prediction step by (5.8), while the estimation step is unchanged and defined by equations (5.5). The interest of this form is explained later.

5.3.2 The matrices of the Kalman filter

Matrices A and B, defined by equations (5.9) are equal to:

$$A = \frac{\partial f}{\partial X} = \begin{bmatrix} 1 & 0 & -\Delta D \sin\theta \\ 0 & 1 & \Delta D \cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

$$B = \frac{\partial f}{\partial U} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{5.14}$$

The only missing matrix now is C.

$$C = \frac{\partial g_B}{\partial X} = \begin{bmatrix} \dfrac{y_B - y}{(x_B - x)^2 + (y_B - y)^2} & -\dfrac{x_B - x}{(x_B - x)^2 + (y_B - y)^2} & -1 \end{bmatrix} \tag{5.15}$$

Here, we omitted the index in the matrices. In pratice, it is important to remember at which value of X and U a matrix is evaluated. But this is easy: it's always the latest estimation which is used. Hence, A_k is evaluated at $\widehat{X}_{k/k}$ which is the result of the previous prediction and estimation sequence of the filter. In our case, since not all prediction steps have a corresponding estimation step, $\widehat{X}_{k/k}$ may in fact be $\widehat{X}_{k/k-1}$ if no estimation took place in the previous cycle. The notations don't indicate that explicitly, but it's good to keep it in mind. For the record, C_k is calculated at $\widehat{X}_{k+1/k}$ and has the following expression:

$$C_k = \begin{bmatrix} \dfrac{y_B - \widehat{y}_{k+1/k}}{(x_B - \widehat{x}_{k+1/k})^2 + (y_B - \widehat{y}_{k+1/k})^2} & -\dfrac{x_B - \widehat{x}_{k+1/k}}{(x_B - \widehat{x}_{k+1/k})^2 + (y_B - \widehat{y}_{k+1/k})^2} & -1 \end{bmatrix} \tag{5.16}$$

The following example should help you grasp the importance of matrix C in the estimation process. Remember that, by definition, C expresses how the output varies when the robot moves around the posture X at which the function is calculated. In other words, it shows how sensitive the output is to local variations of the posture. In the example of figure 5.2, the robot moves straight to a beacon, with a heading angle of $\theta = 0$. So the matrix C will look like $C = [0 * *]$,

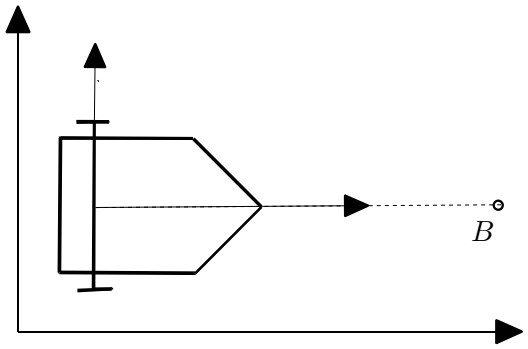


Figure 5.2: Robot moving straight to a beacon.

where "*" denotes a non-zero value. This means that the output is insensitive to motions of the robot parallel to the X_0 axis. As a consequence, the Kalman gain will be of the form $K = [0 * *]^T$: any measurement performed in this situation will have no effect on the x coordinate.

Remarks:

- Make sure you didn't forget everything about calculating derivatives of functions. You may need to do it in the exam ...
- Different authors use different naming/index conventions. Don't be bothered by that: stick to the meaning, and possibly rewrite using your favorite convention.

5.3.3 Coherence tests

Rule 3. *Let X and Y be two random vectors of dimensions n, the respective covariance matrices of which are P_X and P_Y. The scalar $d = [(X-Y)^T \cdot (P_X + P_Y)^{-1} \cdot (X-Y)]^{1/2}$ is a distance, called the* Mahalanobis *distance.*

The Mahalanobis distance can be used to test the coherence between a measurement vector Y and its predicted value $\widehat{Y}$. If the distance is lower than a predefined threshold d_{thresh}, the measurement is suitable for use in the estimation phase, otherwise it is rejected. We already saw in section 5.1 the expression of the covariance matrix of $Y - \widehat{Y}$, so we can write the Mahalanobis distance of Y and $\widehat{Y}$ as:

$$d^2 = (Y_k - \widehat{Y}_{k+1/k})^T \left(C_k \cdot P_{k+1/k} \cdot C_k^T + Q_\gamma\right)^{-1} (Y_k - \widehat{Y}_{k+1/k}) \tag{5.17}$$

In our case, $\widehat{Y}$ of course depends on which beacon we are considering, and we can calculate as many distances as there are beacons within range of the sensor. The different (squared) distances correspond to the following equations, parameterized by the coordinates of the beacons:

$$d^2 = (\lambda_k - g_B(\widehat{X}_{k+1/k}))^T \left(C_k \cdot P_{k+1/k} \cdot C_k^T + Q_\gamma\right)^{-1} (\lambda_k - g_B(\widehat{X}_{k+1/k})) \tag{5.18}$$

Note that when the measurement is a scalar, $C \cdot P \cdot C^T + Q_\gamma$ is a scalar, so there is no matrix inversion to perform.

The Mahalanobis distance provides a good tool to match a measurement to an unsigned beacon. The idea is to calculate the Mahalanobis distances associated to all beacons within reach. The following cases arise:

- No distance is lower than d_{thresh}: the measurement is rejected. If the sensor uses light beacons, it can happen if the sensor detects the reflection of a beacon on a specular surface rather than the beacon itself.
- A single distance is lower than d_{thresh}: perform the estimation phase using the matrix C corresponding to the beacon which yields this minimum distance.
- Two (or more) distances are lower than d_{thresh}: there is an ambiguity. This can happen in a situation where the sensor is close to the straight line through both beacons.

5.3.4 Filter tuning

For the filter to run, it is necessary to associate values to noise covariance matrices Q_α, Q_β and Q_γ.

Let us first consider the measurement noise Q_γ. It is important to realize that there is an underlying hypothesis to observation equation (5.12). We assume that the beacon is in the line of sight of the sensor a time instant t_{j+1}. But actually, the beacon is in the line of sight at time t in the interval $[t_j, t_{j+1}[$ and we decide to consider the measurement as corresponding to time t_{j+1}. This is acceptable if the travelled distance during a sampling period is small with respect to the desired precision of the localization system. This travelled distance, or rather the corresponding variation of the azimuth angle of the beacon, has to be taken into account when estimating Q_γ, in addition to the noise due to the sensor used to measure λ, *e.g.* an encoder. These two contributions to noise γ can be estimated knowing characteristics of the equipment, sensor rotation speed, robot speed and minimum distance to the beacons ...

What about Q_α and Q_β? What is the goal when tuning these parameters? The goal is to model the evolution of uncertainty corresponding to the use of odometry. But we are not just concerned with modelling this evolution over one sampling period, as would be the case if we had a measurement at each sampling period. The model should hold at least during the time interval between the detection of two beacons. In practice, we would like the model to hold somewhat longer, in order to be able to cope with short accidental maskings of the beacons.

Let us examine what happens when Q_β is zero, *i.e.* the input is a deterministic vector and we are back to system (5.2), with the Kalman filter governed by equations (5.3) and (5.5). Suppose for example that $Q_\alpha = diag(\sigma^2, \sigma^2, \sigma_\theta^2)$. The variances for variables x and y have been set equal as there is no reason to make any difference between them. If repeated odometry phases are performed, and provided $P_{0/0} = diag(\sigma_0^2, \sigma_0^2, \sigma_{\theta_0}^2)$, the uncertainty ellipse in the $x-y$ plane will be a circle of growing radius. This does not fit with how uncertainty grows when perfoming odometry: uncertainty tends to grow faster in the direction perpendicular to motion, and so the orientation of the uncertainty ellipse should change when the robot moves.

A more satisfactory behavior can be obtained by setting Q_α to zero and tuning Q_β only. Figure 5.3 shows a typical evolution of the covariance matrix P_k, graphically represented by the uncertainty ellipse in the $x-y$ plane. During prediction steps, the area of the ellipse increases and the orientation of the major axis of the ellipse varies along the path. Estimation phases decrease the area of the uncertainty ellipse, which may also be reoriented. Please also note the "step" produced by the estimation, which modifies $X_{k+1/k}$ into $X_{k+1/k+1}$.

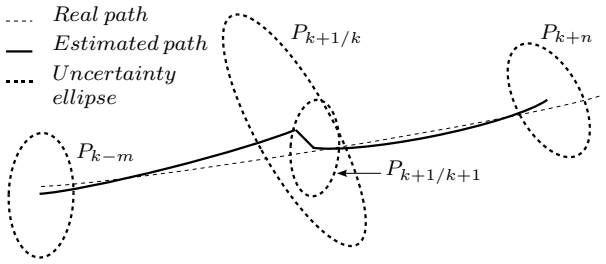


Figure 5.3: Typical evolution of the uncertainty ellipse.

How to tune Q_β? Equations (5.11) can be rewritten under matrix form as:

$$\begin{bmatrix} \Delta D_k \\ \Delta\theta_k \end{bmatrix} = \begin{bmatrix} \frac{r_r}{2} & \frac{r_l}{2} \\ \frac{r_r}{2e} & -\frac{r_l}{2e} \end{bmatrix} \begin{bmatrix} \Delta q_r \\ \Delta q_l \end{bmatrix} = M * \begin{bmatrix} \Delta q_r \\ \Delta q_l \end{bmatrix} \tag{5.19}$$

If we consider the origin of the noise β on U as noise on the vector of elementary rotations of the wheels, which we will call β_q, then applying rule (1):

$$Q_\beta = M \cdot Q_{\beta_q} \cdot M^T \tag{5.20}$$

In practice, there is no reason to consider the noise as being different for the right and left wheel, so $Q_{\beta_q} = \sigma_q^2 * I_{2*2}$ is a reasonable form. This is where the interest of this development lies: one ends up with σ_q as unique tuning parameter.

In practice, σ_q is tuned by trial and error. If you have access to localization errors (*e.g.* by using another localization system on the robot), then the tuning parameter is adjusted in such a way that errors are "nearly always" within the $+/-2\sigma$ interval: if errors exceed the $+/-2\sigma$ interval, increase σ_q; if they are much smaller than 2σ, decrease σ_q.

Such a trial and error method is easy to implement with a single tuning parameter, while it can be hard with two or more.

It is crucial to understand that noise β_q is *not* a simple wheel position measurement error. It is used to account also for all model errors in the odometry. Hence, using super-high resolution encoders will *not* make it smaller. Anyway, we already know from section 3.3 that encoder resolution is only a secondary source of error in odometry.

5.3.5 Experimental results

For experimental results, see the slides.

Chapter 6

Localisation and observability[1]

Some of the techniques presented in the localization course lead to a state-space representation and to a Kalman filter state observer. In the example we use in the course, both observations (angular measurements) and the measurement of the inputs of the system (speed and rotation speed) are used. Several questions arise: is state estimation always possible? Are there situations in which the observer diverges? We are going to see that there exist observability tests which help determine potentially dangerous situations.

6.1 Observability for non linear systems

6.1.1 Definitions

Observability of a dynamical system informs on the ability to reconstruct its state, given the outputs and inputs of the system. Let us consider a continuous-time system described by a differential system of the form:

$$\begin{cases} \dot{x} = f(x,u) \ , \ x \in \mathbb{R}^n \\ y = g(x) \ , \ y \in \mathbb{R}^p \end{cases} \tag{6.1}$$

The case when $n = p$ is a special case, where the expression of x as a function of y can simply be obtained by inverting the function g (if it is invertible). If the n outputs are well chosen, inversion is generally possible, with the possible exception of points belonging to a zero-measure set. But when the dimension of the measurement vector is strictly less than the dimension of the state vector ($p < n$), the measurement of the outputs only is not sufficient to determine the state: the problem is an observability problem *stricto sensu*.

There exist several ways to define observability. We use the definition of Diop and Fliess.

Rule 4. *A system is observable iff it is possible to express the state vector as a function of the input vector, the output vector and their time derivatives. Or:*

$$x = \varphi(y, \dot{y}, ..., y^{(k_1)}, u, \dot{u}, ..., u^{(k_2)}) \quad with\, k_1 \in \mathbb{N}\, and\, k_2 \in \mathbb{N} \tag{6.2}$$

[1]This document is taken for chapter 2 of the Ph.D. thesis of Philippe Bonnifait, which I supervised.

The system is said to be *generically* observable if the set of points where this condition is not met is a zero-measure set.

6.1.2 Definitions

Observability can be proved using a rank sufficient condition. But in order to express the rank condition for observability, we first need to introduce the Lie derivative.

Rule 5. *Let g be a differentiable* real *function and f a real* vector *function, then the Lie derivative of g with respect to f is the vector field :*

$$L_f g = \frac{\partial g}{\partial x} f \tag{6.3}$$

In order to grasp the interest of the Lie derivative for dynamical systems, let us consider the following single output system:

$$\begin{cases} \dot{x} = f(x,u) \\ y = g(x) \end{cases}$$

The derivatives of the output with respect to time are:

$$\begin{aligned} \dot{y} &= \frac{\partial g}{\partial t} = \frac{\partial g}{\partial x}\dot{x} = \frac{\partial g}{\partial x} f = L_f g \\ \ddot{y} &= \frac{\partial [L_f g]}{\partial t} = L_f^2 g \end{aligned}$$

When using $L_f g$ instead of $\dot{y}$, the evolution model appears, so that it will be taken into account in the rank condition.

We conventionally denote as dh the gradient vector of the scalar function h with respect to the state vector X:

$$dh = \left[\frac{\partial h}{\partial x_1} \cdots \frac{\partial h}{\partial x_n} \right]^T \tag{6.4}$$

Rule 6. *Let $g : \mathbb{R}^n \to \mathbb{R}^p$. Its components (which are real-valued functions) are written:*

$$g = \left[\begin{array}{ccc} g^1 & \dots & g^p \end{array} \right]^T \tag{6.5}$$

Let the observability matrix O be defined by:

$$O = \left[\begin{array}{ccccccc} dg^1 & dL_f g^1 & \dots & dL_f^{n-1} g^1 & \dots & dg^p & \dots & dL_f^{n-1} g^p \end{array} \right] \tag{6.6}$$

If the rank of O is full (equal to n), then the system is observable.

6.2 Continuous modelling of the localization problem

The previous condition is applicable to continuous systems. Hence, we use the continuous form of the robot evolution model. In addition, we also consider that the angular measurements of

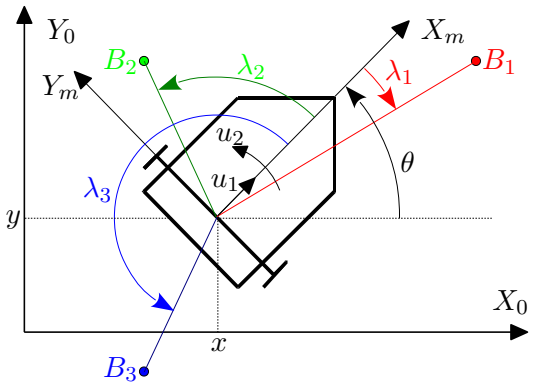


Figure 6.1: Azimuth angles with three beacons, state variables and input vector.

the various beacons are available continuously and simultaneously. This could be obtained with a sensor composed of several cameras, each continuously pointing to a given beacon.

The continuous state of the system is denoted by $X = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$. The observation equation corresponding to beacon B_k, the coordinates of which are (x_{B_k}, y_{B_k}), is written:

$$\lambda_k = atan2\left(y_{B_k} - y, x_{B_k} - x\right) - \theta = g_k\left(X\right) \;\; with: \;\; 1 \leq k \leq m \tag{6.7}$$

If there are m beacons, the m measurements are grouped into a single vector λ. For example, in the normal situation, m equals 3 and:

$$\begin{cases} \lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}^T \\ g(X) = \begin{bmatrix} g_1(X) & g_2(X) & g_3(X) \end{bmatrix}^T \end{cases}$$

In addition, the input vector of the system is $U = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T = \begin{bmatrix} v & \omega \end{bmatrix}^T$, where v is the robot speed and ω its angular speed.

The continuous evolution model (robot kinematic model) is:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{6.8}$$

Which can be rewritten as:

$$\dot{X} = f(X, U)$$

Finally, it should be noted that the observability analysis *does not take into account the noises*. U and λ are supposed to be perfectly known. The overall system is:

$$\begin{cases} \dot{X} = f(X, U) & X \in \mathbb{R}^3 \\ \lambda = g(X) & \lambda \in \mathbb{R}^m \end{cases}$$

6.3 Application to the system (robot + goniometric sensor + beacons)

6.4 Approach

We are going to apply the rank condition to the system, which comprises the robot, the goniometric sensor and the beacons. This system is characterised by the number of beacons: the systems (robot + sensor + three beacons) and (robot + sensor + two beacons) are different. We will simply suppose here that the dimension p of vector λ (which is equal to the number of beacons) is constant over a sufficiently long time interval $[\mathrm{t}_0, T]$:

$$For\, any\, t \in [t_0, T] \begin{cases} \dot{X}(t) = f(X(t), U(t)) & X \in \mathbb{R}^3 \\ \lambda(t) = g(X(t)) & \lambda \in \mathbb{R}^p\, with\, p \leq 3 \end{cases}$$

We are now going to successively consider the cases of three beacons, two beacons and one beacon and apply the observability test. Since the condition is sufficient and not necessary, the situations where the conditions are not satisfied will be called *potentially problematic*.

6.4.1 Three beacons

This case is particularly important, because it is the standard situation for the localization problem.

$$\begin{cases} \dot{X}(t) = f(X(t), U(t)) & X \in \mathbb{R}^3 \\ \lambda(t) = g(X(t)) & \lambda \in \mathbb{R}^3 \end{cases} \tag{6.9}$$

Observation as an inversion problem.

X can be expressed as a function of λ if g can be inverted. Let us consider the Jacobian matrix O_1:

$$O_1 = \begin{bmatrix} dg_1 & dg_2 & dg_3 \end{bmatrix} \tag{6.10}$$

$$dg_i = \begin{bmatrix} \frac{y_{B_i} - y}{D_i} & \frac{x - x_{B_i}}{D_i} & -1 \end{bmatrix}^T a \tag{6.11}$$

With: $D_i = (y - y_{B_i})^2 + (x - x_{B_i})^2$.

The function can be inverted if the determinant of O_1 is not equal to zero (O_1 has full rank). So we are interested in the solutions of equation:

$$\det(O_1) = 0$$

Calculations show that the solutions of this equation correspond to the (unique) circle (C) which passes through the three beacons, which we suppose are not aligned.

Since g is invertible for X not on (C), and since the input U has not been considered, we can conclude that the system is observable for any X out of (C) and for any speed. Since, in addition,

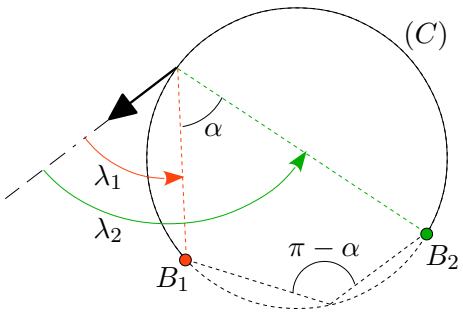


Figure 6.2: Locations which satisfy $\lambda_2 - \lambda_1 = \alpha$ correpond to a circular arc (solid line). The other part of the circle (dotted line) corresponds to $\lambda_2 - \lambda_1 = \pi - \alpha$.

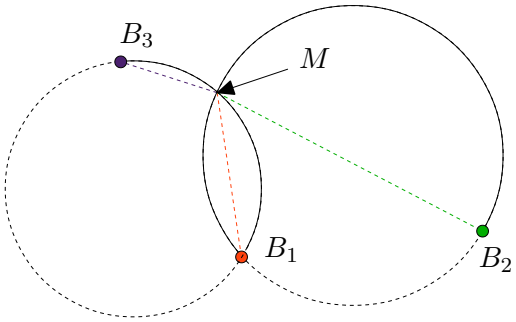


Figure 6.3: Assuming the mobile cannot coincide with a beacon, two measurements define its position M uniquely (but not its heading).

the set of singular configurations is a zero-measure set, we can say that the system is generically observable.

It is possible to give a geometrical interpretation of what happens on (C). Indeed, two azimuth measurements define a circular arc through the two corresponding beacons, on which the mobile is located. This circular arc is defined by the constraint $\lambda_2 - \lambda_1 = \alpha$ (figure 6.2). On the remaining part of the circle, $\lambda_2 - \lambda_1 = \pi - \alpha$

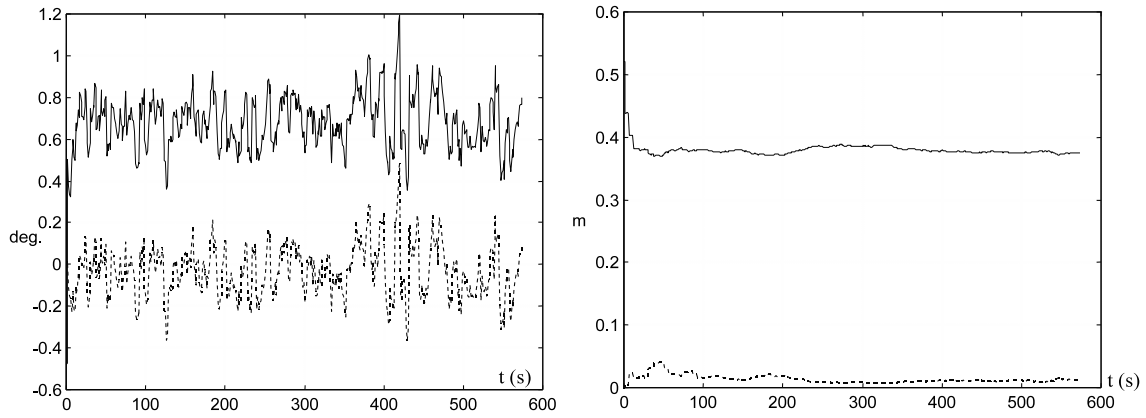
Using the angle difference $\lambda_3 - \lambda_1$ constrains the mobile reference point M to lie at the intersection of two such circular arcs, as shown by figure 6.3. One point of intersection of the two arcs is beacon B_1. Assuming the mobile cannot be at a beacon location, this gives its position. The circle defined by the angular difference $\lambda_3 - \lambda_2$ intersects the first two at the same point. On the other hand, it allows the determination of the absolute heading of the mobile.

But if point M of the robot is on (C), then the intersection of the three arcs is a circular arc of non zero length, and there are an infinite number of solutions.

Using the observability condition on the circle.

In the previous paragraph, the observability test was not used. We are now going to use the rank condition proper, which means that the evolution model (and hence the input) will be taken into account. The observability matrix is:

$$O = \left[\begin{array}{ccccccc} dg_1 & dL_f g_1 & dL_f^2 g_1 & \dots & dg_3 & \dots & dL_f^2 g_3 \end{array} \right] \in \mathbb{R}^{3*9} \tag{6.12}$$



Heading error Absolute position error

Figure 6.4: Filter errors for a motionless vehicle on (C) with three beacons, with (solid line) and without (dotted line) initial errors.

The first orfer Lie derivatives are the following:

$$L_f g_i(X) = v\left(\frac{y_{B_i} - y}{D_i}\cos\theta + \frac{x - x_{B_i}}{D_i}\sin\theta\right) - \omega$$

If $v = 0$, $L_f g_i(X) = -\omega$ and $dL_f g_i = dL_f^2 g_i = \vec{0}$ hence $O = \left[\begin{array}{ccccccc} dg_1 & dg_2 & dg_3 & \vec{0} & \dots & \vec{0} \end{array}\right]$.

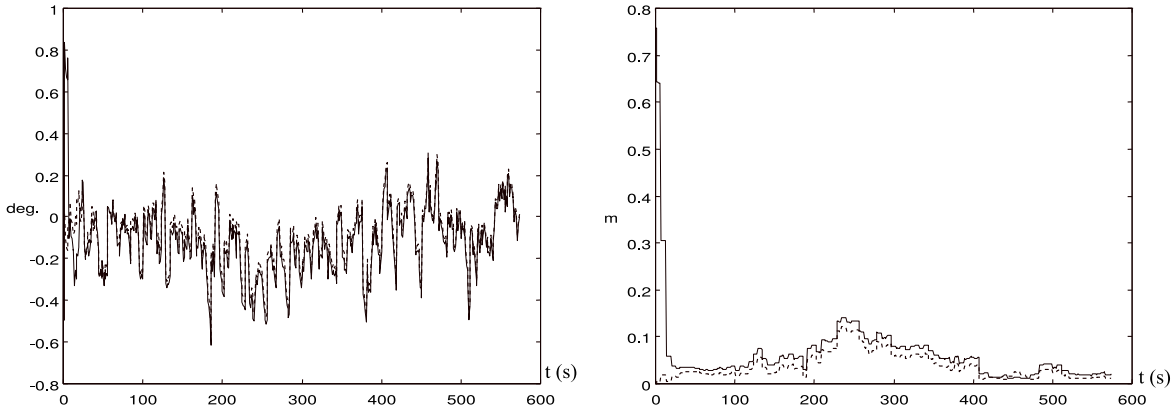
The rank of O is the same as the rank of $O1$ and is thus strictly lower than 3 on (C). We then have a potentially problematic situation.

On figure ** are shown simulation results, for a motionless robot on (C). The output of two filters are shown, operating on the same noisy data, with the difference that one of them also starts with an erroneous initial estimate. The behaviour of the filters is summarized by looking at the heading error and at the absolute position error. It should be noted that the absolute position error is always positive, so it is not possible to check whether it has zero mean, contrary to the heading error.

We can notice that if the vehicule is motionless on (C), the estimator is unable to bring the error to zero. What about the case when the robot moves along (C), in which case v and ω are non zero? Due to the complexity of the derivatives, an analytical determination of the rank of the observability matrix was not obtained. Alternatively, it is possible to test the behaviour of the filter for a vehicle moving along (C).

The results of figure 6.5 clearly show that the filter correct an initial error (heading errors are nearly superimposed after a short transient phase), even though the precision of localization is reduced with respect to other trajectories.

For this filter, the results indicate that the state is observable in this situation, due to the input and the evolution model or, in other words, due to the presence of odometry. This shows how the input can play an important part in the observability of non linear systems, contrary to linear systems.



Heading error Absolute position error

Figure 6.5: Filter errors for a vehicle moving along (C) with three beacons, with (solid line) and without (dotted line) initial errors.

6.4.2 Two beacons

In this section, we assume that, possibly because a beacon is hidden or too far, the sensor detects only two beacons. The system becomes:

$$\left\{ \begin{array}{c} \dot{X}(t) = f\left(X(t), U(t)\right) \\ \lambda(t) = g\left(X(t)\right) \end{array} \right. \quad \lambda \in \mathbb{R}^2 \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} \dot{X} & = & f(X,U) \\ \lambda_1 & = & g_1(X) \\ \lambda_2 & = & g_2(X) \end{array} \right. \tag{6.13}$$

Since $p < n$, state observation cannot be obtained using the azimuth measurements alone. The input, through the evolution model, has to be taken into account. The observability matrix is now:

$$O = \left[\begin{array}{ccccccc} dg_1 & dL_f g_1 & dL_f^2 g_1 & dg_2 & dL_f g_2 & dL_f^2 g_2 \end{array} \right] \in \mathbb{R}^{3*6} \tag{6.14}$$

If v is equal to zero, it is easy to check that:

$$O = \left[\begin{array}{cccccc} dg_1 & dg_2 & \vec{0} & \vec{0} & \vec{0} & \vec{0} \end{array} \right]$$

As a consequence, the rank of O is at most equal to two. This is of course intuitively clear, as two measurements are not enough to determine the three components of the position and heading of the robot.

In the sequel, we will suppose that v is not equal to zero. Let us consider the determinants, denoted s_i, of the C_3^6 sub-matrices of O. We are looking for trajectories which produce a non full rank, *i.e.* $(X,U) = (x, y, \theta, v, \omega)$ for which the s_i are all equal to zero. The corresponding system of equations is a non linear differential system. Calculations are complex and rather cumbersome. With the help of a formal calculation toolbox, we obtain the following potentially problematic situations. Of course, there may be others which we did not find. The situations we detected correspond to three lines, *whatever the speed and direction along these lines.*

We have taken a closer look to straight lines through B_1 and B_2. For straight lines other than Δ_1, Δ_2 and Δ_3, we have been able to extract sub-matrices with non zero determinants, thus proving

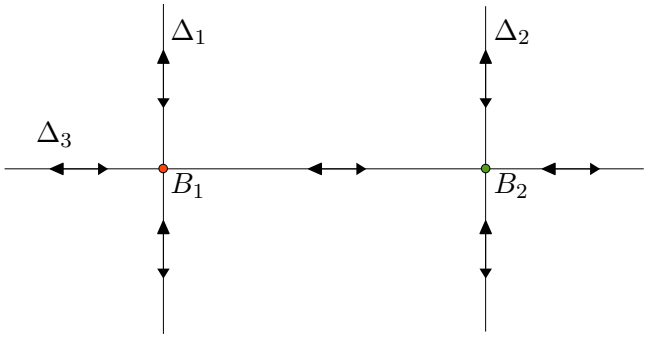
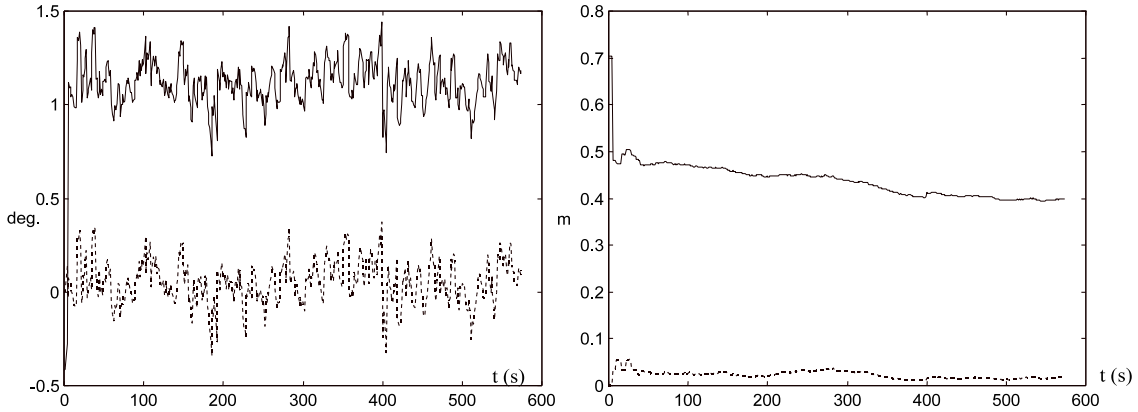


Figure 6.6: Potentially problematic paths with two beacons.



Heading error Absolute position error

Figure 6.7: Filter errors for a vehicle moving along Δ_1, with (solid line) and without (dotted line) initial errors.

observability (for non zero speed of course). In addition, the Kalman filter observer also runs fine when a motion on these other straight lines is simulated.

For a vehicle moving along Δ_3, the situation is the same as if it were seing a single beacon, a situation which we consider later in this document.

Let us now consider a vehicle moving along Δ_1 towards B_1.

The results of figure 6.7 show that the filter is unable to drive the initial error to zero. Note that a vehicle moving along Δ_1 measures a constant azimuth λ_1, the time derivatives of which are all equal to zero. In such a case, O can have full rank only if λ_2 is not constant over time, which implies that the vehicle moves. On the other hand, it is not intuitively clear why Δ_1 and Δ_2 are different from other straight lines through B_1 and B_2. For the sake of brevity, we will not give the explanation here.

6.4.3 One beacon

It is easy to check that the determinant of the observability matrix is equal to zero whatever the state and the input. The fact that the system is not observable is clear. For example, two straight lines at the same distance from the beacon can yield the same outputs for a given speed.

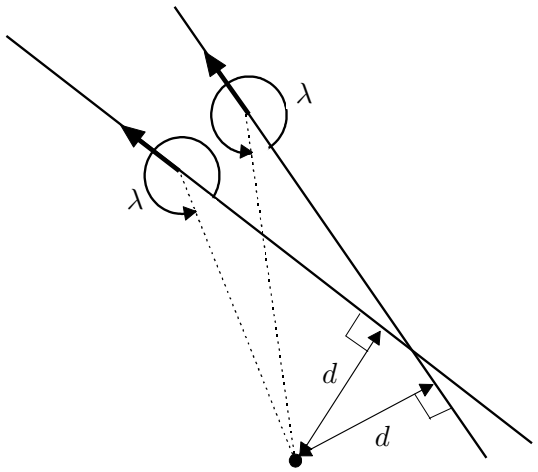


Figure 6.8: Example of trajectories which yield the same azimuth angles

6.5 Conclusion about the observability study

Studying the observability of the system allowed us to determine problematic trajectories for the vehicle, some of which were not obvious *a priori*. It is then necessary to check the behaviour of the observer on these cases using simulations. In addition, this study showed that it was not necessary for the sensor to permanently detect three beacons, since two are enough in most cases, provided that the vehicle moves. Hence, the system is able to withstand momentary maskings of a beacon. On the other hand, observability analysis does not inform on the influence on precision of the number of beacons and their relative (GDOP[2]). It is clear, though, that with two beacons instead of three, precision is not as good.

Unfortunately, in the case of two beacons, the rank condition is too difficult to check to guarantee that our study was exhaustive. Thus, in spite of the simulations that were conducted, it is impossible to be sure that there are no other problematic trajectories.

[2]Global Dilution Of Precision. It is a scalar value which informs about the precision of the localization process. It depends on the beacon configuration and on the position of the vehicle with respect to the beacons.

Bibliography

[Borenstein and Feng, 1995] Borenstein, J. and Feng, L. (1995). Umbmark: A benchmark test for measuring odometry errors in mobile robots. In *Proceedings of the 1995 SPIE Conference on Mobile Robots*.

[Borenstein and Feng, 1996] Borenstein, J. and Feng, L. (1996). Gyrodometry: a new method for combining data from gyros and odometry in mobile robots. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 423–428, Minneapolis, Minnesota.

[Bouvet and Garcia, 2000] Bouvet, D. and Garcia, G. (2000). Improving the accuracy of dynamic localization systems using RTK GPS by identifying the GPS latency. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 2525–2530, San Francisco, Californie.

[Garcia et al., 1995] Garcia, G., Bonnifait, P., and Le Corre, J.-F. (1995). A multisensor fusion localization algorithm with self-calibration of error-corrupted mobile robot parameters. In *Proceedings of the 1995 International Conference on Advanced Robotics*, Sant Feliu de Guixols, Spain.

[Jaulin and Walter, 1993] Jaulin, L. and Walter, E. (1993). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064.

[Khalil and Dombre, 2004] Khalil, W. and Dombre, E. (2004). *Modeling, Identification & Control of Robots*. Kogan Page Science.

[Khalil and Murareci, 1993] Khalil, W. and Murareci, D. (1993). Résolution polynomiale exacte des équations du sirem. Rapport interne 93.22, Laboratoire d'Automatique de Nantes.

[Le Corre and Garcia, 1992] Le Corre, J.-F. and Garcia, G. (1992). Real-time determination of the location and speed of mobile robots running on non-planar surfaces. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2594–2599, Nice, France.

[Le Corre and Peyret, 1990] Le Corre, J.-F. and Peyret, F. (1990). SIREM: the absolute location of civil-engineering equipment. *Mechatronic Systems Engineering*, 1:183–192.

[Wikipedia, 2011a] Wikipedia (2011a). Chip log — wikipedia, the free encyclopedia. [Online; accessed 17-April-2011].

[Wikipedia, 2011b] Wikipedia (2011b). Dead reckoning — wikipedia, the free encyclopedia. [Online; accessed 17-April-2011].

[Wikipedia, 2011c] Wikipedia (2011c). Pitometer log — wikipedia, the free encyclopedia. [Online; accessed 17-April-2011].