DÉPARTEMENT D'INFORMATIQUE
LABORATOIRE DE MICROINFORMATIQUE
LAMI - DI INF
CH- 1015 LAUSANNE

Philip Mächler Email: maechler@di.epfl.ch
%021 - 693 39 07 Téléfax 021 - 693 52 63

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Submitted to MCPA '97 Pisa, Italy

Robot Odometry Correction Using Grid Lines on the Floor

Philip Mächler (maechler@di.epfl.ch)
LAMI-DI, Swiss Federal Institute of Technology, Lausanne

Abstract

This paper presents an algorithm to correct the odometry error of an autonomous mobile robot only by using a painted grid on the floor. The supposed robot position is calculated by odometry and is matched with the grid lines, whose existence is known to the robot. A new "position probability function" was developed and used by the correction algorithm.
The correction of the odometry error is also based on interactive trajectory modification, in order to reduce the error when crossing a line.

Keywords

Autonomous mobile robots, Localization, Navigation, Classification, Odometry.

1 Introduction and Overview

For autonomous robots, navigating and interacting with their environment are fundamental skill. Depending on their task, robots need different techniques to work and survive in their environment. For example, autonomous vacuum cleaners for swimming pools do their job quite perfectly with a simple strategy: turning over when touching a wall. On the other hand, an apparently similar and simple job like having an autonomous robot moving inside an office becomes a technical challenge. The major difficulty for the robot is to have a spatial representation of the environment, and a strategy to cover the work-area in a safe and exhaustive manner [Knieriemen91] [Pau90].

Various navigation systems can provide robot with its absolute position, like laser range finder, ultrasonic distance measurement, Global Positioning System or optical triangulation [Welch92]. These systems are sophisticated and expensive.

A robot needs to keep constant track of its position (x and y) and its direction (α). In most cases this is done by odometry (local positioning), integrating the velocity. Relative positioning, using odometry is inexpensive, but inaccurate due to slipping wheels and other influences.

This paper describes an approach to localize the position of a robot in a very straightforward manner by combining the local navigation system (odometry) with a very basic global navigation aid (grid lines).

To correct the odometry error, a fixed grid is used, whose features are known to the robot. This grid allows to synchronize the position each time the robot crosses a grid line. The combination of odometry and grid lines results in a final corrected position. This technique includes a new "position probability model" of the robot location and an interacting pilot system which actively influences the robot trajectory to take better advantage of the grid lines.

These two additional features ("position probability model" and pilot) allows the robot to figure out its exact position with simple means.

2 Odometry correction

This section describes the basic technique to correct the odometry error with the grid on the floor.

2.1 One-dimensional correction

The robot is equipped with a single light sensor on its bottom side which detects the painted grid line on the floor. We assume that the light sensor provides an ideal short pulse when crossing a grid line. This pulse will be used to synchronize the supposed position of the robot (calculated by odometry) with the real position (indicated by the grid lines). The upper part of fig. 2-1 shows the robot passing some vertical grid lines. While running, the robot calculates its position by odometry. This supposed position, associated with a gaussian increasing error, is represented by a dotted diagonal line in the diagram. The solid line represents the real x position. Checking the probability of the corrected error is a good way to recognize abnormal conditions and systematic drifts (allowing continuous re-calibration).

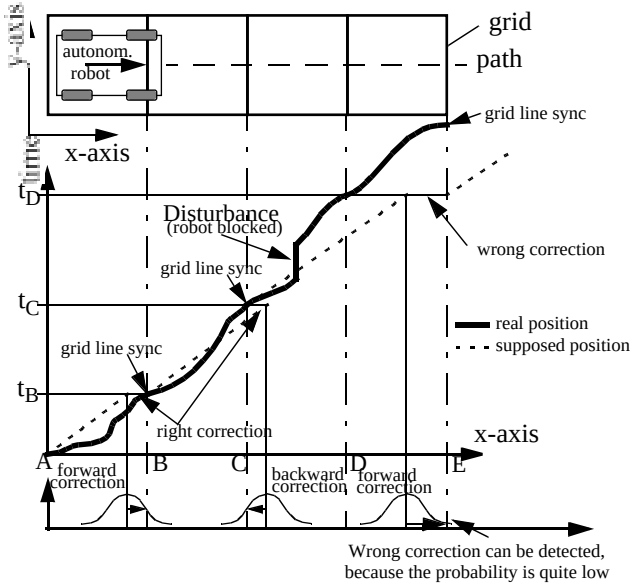


Fig. 2-1: Fundamental technique to correct odometry error

Explanation of fig. 2-1: The robot (solid line) starts at position A and passes over the first grid line B at time t_B. The supposed position (dotted line) is a little bit behind the grid line B, which means that the robot moves faster than expected. The supposed position will therefore be corrected forwards. During the next sequence (B to C), the robot moves unintentionally slower and therefore crosses the grid line C later than expected. The supposed position has to be corrected backwards, because point C is closer to the supposed position than point D. The last sequence shows a wrong correction due to a blockage of the robot. The light sensor sends a signal event at time t_D. Unfortunately, the supposed position is closer to point E than to point D, which results in a wrong correction. The robot lost its position because the odometry error was too big. This wrong correction can normally be suspected, because the probability of the obtained point E is quite low.

2.2 Two-dimensional correction

In the previous example (one dimensional), the direction of correction was defined by a basic linear membership function, indicating the closest grid line (see fig. 2-1). Such a technique cannot be used in a two dimensional environment with a right-angle grid network for synchronization. The reason is shown in fig. 2-2.

A robot follows a horizontal grid line in a zigzag movement. The y position of the robot is therefore constantly well calibrated (in contrast to its x position, which is comparatively rarely corrected). The probability, that the last signal event (■) is produced by the vertical grid line is much higher (even if the horizontal grid line is closer), because the x position is much more uncertain than the y position. The following approaches addresses this problem.

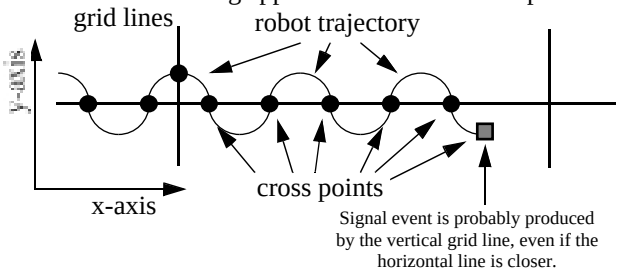


Fig. 2-2: Example: The closest grid line is not always the best.

2.2.1 The ellipse approach

Due to the error in the distance measurement, the position of the robot belongs to some certain “Position Probability Distribution” (PPD). A gaussian distribution (see right illustration in fig. 2-3) is often used to model the PPD [Piasecki95][Pruski96]. The width of this bell curve represents the position probability (depending on the covered path) and other parameters. The circles in fig. 2-3 show a horizontal cut through the bell curve at a certain altitude (top view). The chosen altitude corresponds to 95% of the probability of being inside the area (circle, ellipse).

When the robot passes a horizontal grid line (see fig. 2-3), the bell curve will shrink in the corresponding y dimension because the y position has just been calibrated. This is shown by several ellipses, representing the same horizontal cut through the distorted bell curve.

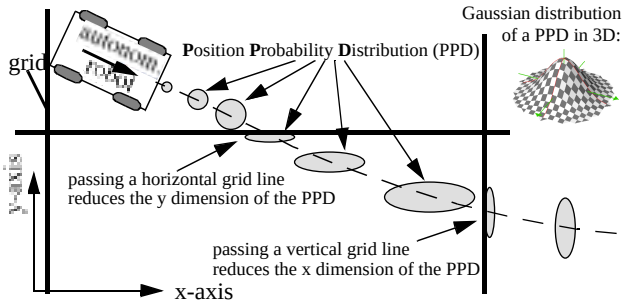


Fig. 2-3: Passing grid lines change the shape of the PPD

The ellipses will grow continuously in both dimensions between the grid lines.

However, the shape of the PPD is not only determined by grid lines. Mechanical attributes of the robot have also some influences on the shape as well. This will be considered in the next approach.

2.3 The “banana” approach

In the case of a 2-wheel robot (like the Khepera®[1]), the fundamental PPD is far from being a circle or an ellipse. The shape of the PPD (see section 3 "Calculation of a PPD model") looks more like a banana. The fig. 2-4 gives a first idea of the PPD-shape and the expected effect.

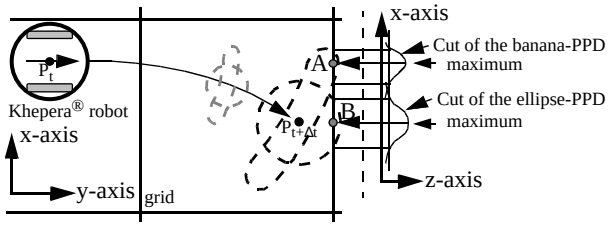


Fig. 2-4: The PPD-model influences strongly the result. The banana and the ellipse model result in different correction

A Khepera® robot starts at point P_t and turns slowly to the right. After a certain time (Δt), a signal is supplied by the light sensor, indicating a grid line below the robot. The robot assumes its position at point $P_{t+\Delta t}$ with a certain position probability distribution (PPD). The 3 dimensional PPD is cut by the grid lines, which are known by the robot. The resulting 2 dimensional cut is shown on the right side in fig. 2-4 for each PPD model (banana and ellipse). The highest point (maximum) of the cut indicates the position with the highest probability. This intuitive example shows the importance of choosing the correct PPD model. This knowledge can also be used to influence the trajectory of the robot in order to put its PPD in a suitable position.

3 Calculation of a PPD model

The following section discusses how to calculate the shape of the PPD. The approach is based on the two wheel Khepera® robot. The odometry error is attributed by the given speed of the wheels (*V) and its deviations (ΔV). Therefore the real speed of the wheels can be described by $V_x = {}^*V_x \pm \Delta V_x$ ($_x$ indicates the left ‘L’ or right ‘R’ wheel).

3.1 Geometrical derivation

We assume a rotation of the robot around a fixed point (R). The deviation of the two wheels are not equal ($\Delta V_L \neq \Delta V_R$) but constant. The radius (r) of the rotation depends therefore of the wheel speed difference ($|\Delta V_L - \Delta V_R|$) and the wheelbase (L). A lateral drift of the wheels will be neglected (see fig. 3-1).

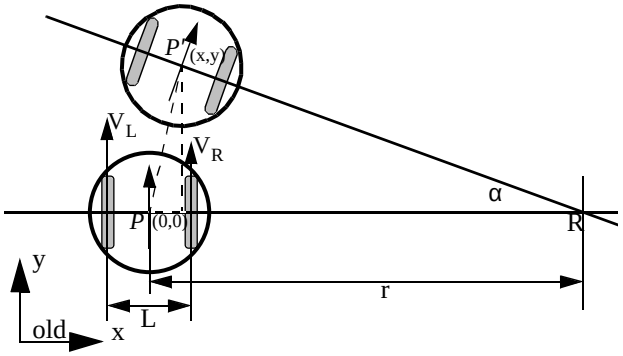


Fig. 3-1: Derivation of a PPD model

The following calculation derives an equation for x, y and α, depending only on V_L, V_R and L. The resulting x and y is the position and α the direction of the robot.

The relation between the wheel speed and the angle can be described as:

$$V_{circumference}\,t = r\alpha \longrightarrow \frac{V_L t}{r+\frac{L}{2}} = \frac{V_R t}{r-\frac{L}{2}} = \alpha$$

This double equation allows the extraction of r and α as functions of V_L, V_R and L:

$$V_L t\left(r-\frac{L}{2}\right) = V_R t\left(r+\frac{L}{2}\right) \longrightarrow \boxed{r = \frac{L(V_L+V_R)}{2(V_L-V_R)}}$$

$$\alpha = \frac{V_R t}{r-\frac{L}{2}} = \frac{V_R t}{\frac{L(V_L+V_R)}{2(V_L-V_R)}-\frac{L}{2}}$$

$$= \frac{V_R t \;\; 2(V_L-V_R)}{L(V_L+V_R)-L(V_L-V_R)} \longrightarrow \boxed{\alpha = \frac{V_L-V_R}{L}t}$$

y and x can now be calculated by trigonometry:

$$y = r\,\sin(\alpha) \qquad \boxed{y = \left(\frac{L(V_L+V_R)}{2(V_L-V_R)}\right)\sin\left(\frac{V_L-V_R}{L}t\right)}$$

$$r-x = r\,\cos(\alpha) \qquad \boxed{x = \left[\frac{L(V_L+V_R)}{2(V_L-V_R)}\right]\left[1-\cos\left(\frac{V_L-V_R}{L}t\right)\right]}$$

The resulting equations supply the position (x and y) of the robot, depending on the constant wheel speed (V_L, V_R), the wheelbase (L), and the time (t).

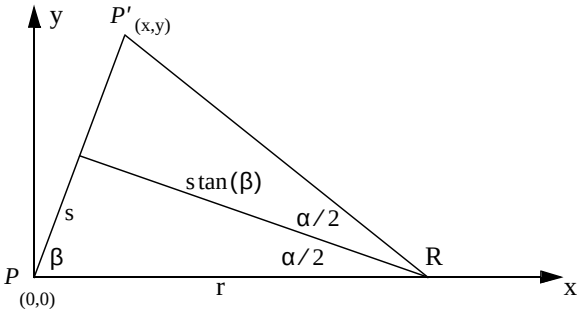
3.2 Calculation and display of the PPD shape

In order to illustrate the distribution of the final position of the Khepera, we have to transform the received equations into the following form:

$$\begin{aligned} x &= g(V_L,V_R) \\ y &= h(V_L,V_R) \end{aligned} \longrightarrow \begin{aligned} V_L &= k(x,y) \\ V_R &= l(x,y) \end{aligned}$$

1. The Khepera robot was designed by LAMI EPFL Lausanne and is now marketed by K-Team SA Switzerland (http://www.k-team.com/)

We assume a constant wheel speed (Vx) and use a geometrical model to transform the variables:



The relation between (V_L, V_R) and (r, α) is already calculated (see chapter 3 "Calculation of a PPD model", page 3). We have now to find a geometrical relation between (x, y) and (r, α):

$$r = \frac{s}{\cos(\beta)} = \sqrt{s^2 + s^2\tan^2(\beta)} = s\sqrt{1 + \tan^2(\beta)}$$

Replacement of s and β: $s = \frac{\sqrt{x^2 + y^2}}{2}$ $\frac{y}{x} = \tan(\beta)$

$$\frac{\sqrt{x^2 + y^2}}{2}\sqrt{1 + \frac{y^2}{x^2}} = \frac{\sqrt{x^2 + y^2}}{2}\frac{1}{x}\sqrt{x^2 + y^2} = \boxed{\frac{x^2 + y^2}{2x} = r}$$

$$\alpha = 2\left(\frac{\pi}{2} - \beta\right) = \boxed{\pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right) = \alpha}$$

We can now calculate the relation between velocity (V_L, V_R) and the position (x, y):

$$\alpha = \frac{V_L - V_R}{L}t = \pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right)$$

$$\longrightarrow \text{A:}\quad V_L - V_R = \frac{L}{t}\left(\pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right)\right)$$

$$r = \frac{L}{2} + \frac{V_R L}{V_L - V_R} = \frac{L(V_L + V_R)}{2(V_L - V_R)} = \frac{x^2 + y^2}{2x}$$

$$\longrightarrow \text{B:}\quad \frac{V_L + V_R}{V_L - V_R} = \frac{x^2 + y^2}{Lx}$$

$$\longrightarrow \text{B' =A B:}\quad V_L + V_R = \frac{x^2 + y^2}{tx}\left(\pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right)\right)$$

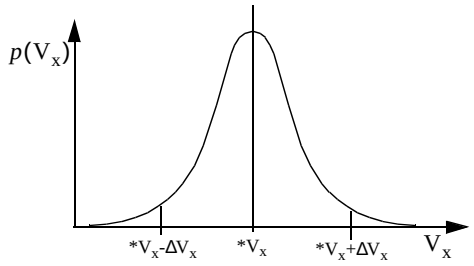
A+B': $$\boxed{V_L(x,y) = \frac{1}{2t}\left(\pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right)\right)\left(\frac{x^2 + y^2}{x}\right) + L}$$

A–B': $$\boxed{V_R(x,y) = \frac{1}{2t}\left(\pi - 2\,\mathrm{atan}\left(\frac{y}{x}\right)\right)\left(\frac{x^2 + y^2}{x}\right) - L}$$

The last equations computes a fix velocity (V_L, V_R) for every robot position (x,y) under the following conditions:

- The wheel speed is constant.
- All positions on the y-axis (x=0) result in an infinite solution, because of the geometrical transformation.

The resulting velocity is not fixed. It has a gaussian distribution (see fig. 3-2), which variation will create the banana. We assume a constant wheel speed V_x distributed according to a bell curve between $*V_x - \Delta V_x$ and $*V_x + \Delta V_x$.



Gaussian distribution: $p(V_x) = \frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{(V_x-\mu)^2}{2\sigma^2}}$ $\mu = *V_x$ $\sigma = \frac{1}{2}\Delta V_x$

Fig. 3-2: Gaussian distribution of the wheel speed

μ is the average wheel speed and σ represents the standard deviation. We choose interval from $*V_x - \Delta V_x$ until $*V_x + \Delta V_x$ covering 95% of the entire gaussian-surface:

$$\boxed{\sigma = \frac{1}{2}\Delta V_x} \quad \text{because} \quad \int_{-2\sigma = -\Delta V_x}^{2\sigma = \Delta V_x}\left[\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2}{2\sigma^2}}\right]dx \approx 95\%$$

In order to calculate the PPD of the final robot location (which is a function of the gaussian distributed velocities), we must multiply the velocity probability function by the Jacobian determinant of the velocities with respect to the x and y locations:

$$f_{xy}(x,y) = f_{V_L}(V_L(x,y))f_{V_R}(V_R(x,y)) \cdot \left|J_{V_L V_R}(X(V_L,V_R), Y(V_L,V_R))\right|$$

Jacobian Matrix: $J_{V_L V_R}(x,y) = \begin{bmatrix} \frac{\partial V_L}{\partial x} & \frac{\partial V_L}{\partial y} \\ \frac{\partial V_R}{\partial x} & \frac{\partial V_R}{\partial y} \end{bmatrix}$

Inverting the product and the Jacobian:

$$f_{xy}(x,y) = \frac{f_{V_L V_R}(V_L(x,y), V_R(x,y))}{\left|J_{xy}(V_L(x,y), V_R(x,y))\right|}$$

Inverted Jacobian Matrix: $J_{xy}(V_L, V_R) = \begin{bmatrix} \frac{\partial x}{\partial V_L} & \frac{\partial x}{\partial V_R} \\ \frac{\partial y}{\partial V_L} & \frac{\partial y}{\partial V_R} \end{bmatrix}$

This final equation represents the probability of each position of the banana, located on the position (x,y).

We use Mathematica®[1] to calculate the equations and to visualize the PPD. The following image (see fig. 3-3) shows the PPD produced by a Khepera® robot, moving straight ahead for 3 meters (see the scale beside the graphic

1. Mathematica® is a commercial software product by Wolfram Research Inc.

in mm) with a wheel speed ($*V_x$) of 100mm/sec and a deviation of ±1% (ΔV_x =1mm/sec, σ=0.5) for each wheel. The wheelbase (L) is 52mm. These values correspond to a real Khepera®.

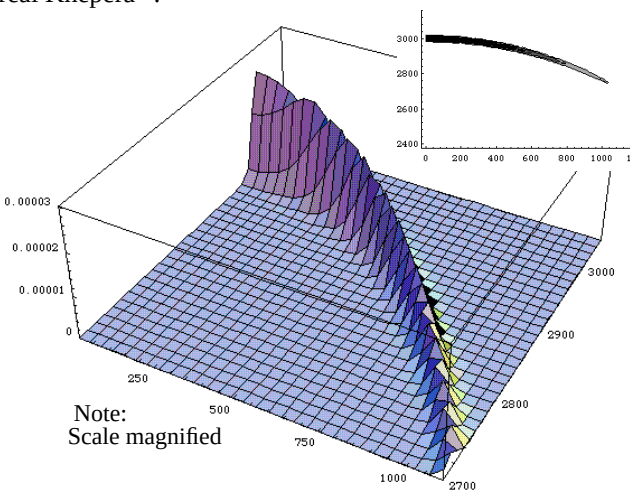


Fig. 3-3: 3D and 2D presentation of the right side of the PPD shape. Robot trajectory 3m; straight ahead; wheel slip 1%

It is remarkable, that the angular deviation is much bigger that the forward deviation. The robot has a lateral error of about 1 meter, as compared to a forward error of about 30 mm. The banana is like a part of a very thin arc, evenly decreasing to the end.

3.3 Deformation of the PPD by curves and corners

Robots move rarely straight ahead. Therefore it is important to analyze in which manner their PPD-shape changes when they take curves and corners. Unfortunately, the just derived equations for calculating the PPD-shape depend on constant wheel speeds (V_L and V_R). Therefore, corners and variable-radius curves cannot be calculated. The following iterative approach will be used to calculate this kind of PPD-shape (see section 3 "Calculation of a PPD model"):

$$y = \frac{L(V_L + V_R)}{2(V_L - V_R)} \sin\left(\frac{V_L - V_R}{L} t\right)$$

$$x = \frac{L(V_L + V_R)}{2(V_L - V_R)} \left[1 - \cos\left(\frac{V_L - V_R}{L} t\right)\right]$$

We use Mathematica® to sweep V_x from ($*V_x - \Delta V_x$) to ($*V_x + \Delta V_x$) and for the following visualization of the PPD. There is no significant difference between the PPD shape calculated by the derived equation or by the iterative calculation because of the high numerical precision of Mathematica®. In order to visualize the path covered by the robot, we use only the iterative calculation for all further simulations.

4 Examples

The following examples illustrate the behavior of the PPD in curves and corners. The graphic shows the same PPD, but in top view. The dotted line indicates the ideal robot trajectory without any slip. All examples show the PPD 4 times (for each meter).

4.1 Curve

The following simulation (see fig. 4-1) shows a robot constantly turning right for 4 meters. The left wheel is turning 4.08% faster than the right one (left: 102mm/sec, right: 98mm/sec) with 1% slip.

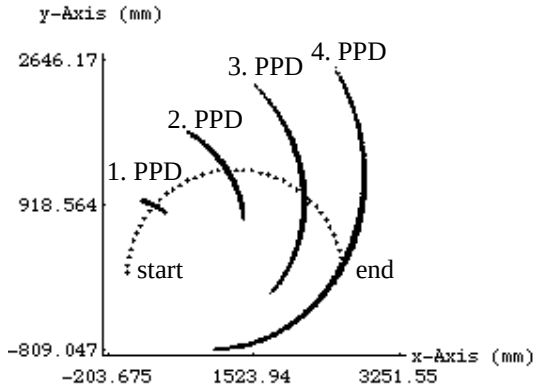


Fig. 4-1: Example 1: PPD behavior in a curve.

Note: It is interesting to observe, that the orientation of the PPD changes only slowly and has not at all the same direction than the robot.

4.2 Turning back after one meter

This simulation (see fig. 4-2) shows the PPD of a robot moving straight ahead, but turning back (turn of 180°) after one meter. The wheel slip is also 1%.

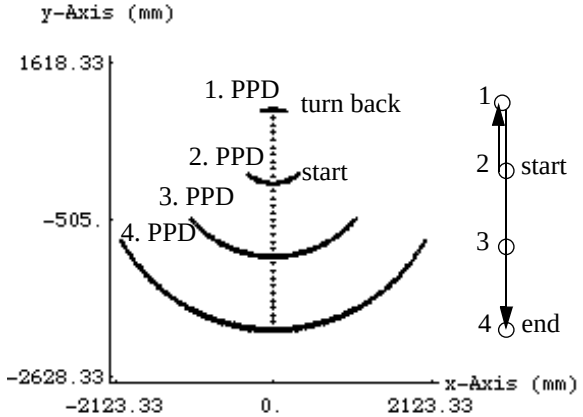


Fig. 4-2: Example 2: PPD behavior after a turn back.

Note: The robot will reach again its starting point after 2 meters. Nevertheless, the corresponding second PDD is bend (downward), like after a move in the direction of the negative y-axis. This illustrates, that the PDD is more influenced by the covered trajectory than by its end position.

4.3 Turning back after 2 meters

The last example (see fig. 4-3) shows the same main movement like in example 3, but the robot turns back after 2 meters. Wheel slip 1%.

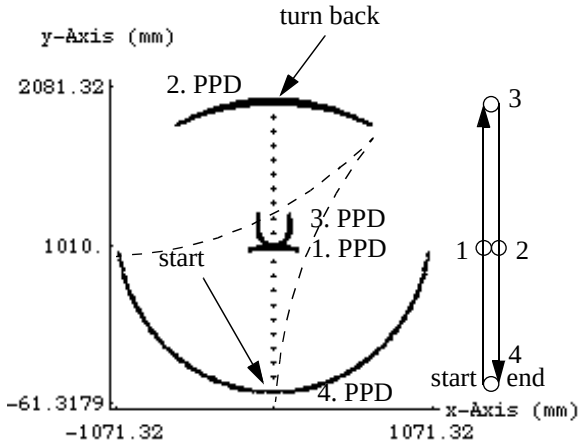


Fig. 4-3: Example 3: PPD decrease after a turn back

Note: The third PPD is much smaller than previous PPD. It seems, that the size of a PPD can also decrease. Nevertheless, the PPD is strongly bent, indicating that the orientation of the robot is quite uncertain in contrast to its position.

4.4 Conclusion of the examples:

It is interesting to notice that the shape of the PPD seems to remain a segment of an arc (condition: the wheel drift has to be constant). The difference consists only in radii and orientations. The thickness of the line seems also to be independent of the trajectory.
The latest example shows also that a certain motion strategy can shrink the PPD. This result is surprising but nevertheless understandable. To better understand, follow the dotted line in fig. 4-3 which indicates the trajectory of the robot with a constant drift to the right side.

5 Statement of the problems

The proposed grid-solution is an easy way to correct odometry errors with limited additional expenses. Nevertheless, this simplicity hides some problems which will be described in the following section.

5.1 Angular drift, caused by inaccuracy α

Because the robot knows the geometry of the painted grid on the floor, an independent correction of the x and y position while crossing a grid-line seems to be an easy way to continuously correct the corresponding dimension of the robot position (see fig. 5-1). Unfortunately, this approach neglects the angle α of the robot, whose estimation accuracy will decline with time, causing a strong angular drift of the robot, which can not be recognized by this algorithm. The robot will get lost.

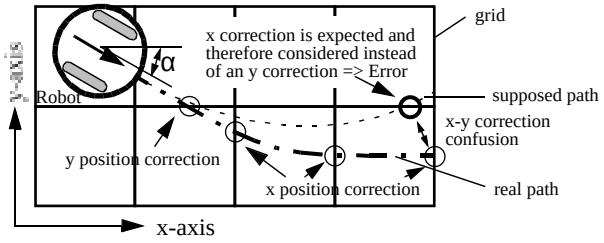


Fig. 5-1: Wrong correction because of confused grid lines

We notice that this angular drift is the main reason for the position error. The distance and speed error while moving straight ahead are comparatively small (see section 3 "Calculation of a PPD model").

5.2 Information distribution of a grid

Another problem of this approach is the information distribution on the grid. The supplied information depends on the region, which can be divided into 4 zones (see fig. 6-3):

- V and H zones supply information about the x and y position of the robot.
- C zones are difficult to interpret and should be therefore avoided by the robot.
- N zones supply no direct information, but are still necessary to distinguish between V and H zones.

It is clear that the robot should pass alternating V and H zones, separated by N zones. At the same time, C zones should be avoided because they provides several cross points with the PPD and the grid lines. The C zone will not be further considered in this paper.

To take advantage of the grid information, a “grid fitter” will be implemented changing the planned trajectory of the robot. The resulting path is only slightly influenced by the grid fitter, but the resulting advantage is relevant and will be illustrated by an example (see fig. 6-3).

6 Strategy for odometry correction

The strategy for odometry correction has to be separated into two sections. The first describes how to use the knowledge of the PPD to correct the estimated position of the robot in a passive manner.
The second section describes how to influence the trajectory of the robot in order to improve the result of the passive correction system.

6.1 Passive correction, using the PPD

The robot position is obtained fundamentally by odometry. The proposed algorithm corrects the odometry errors by knowing the evolution of the PPD motion and by taking advantage of the thin PPD-shape.
The following conditions have to be satisfied:

- The initial position and starting direction of the robot have to be known.
- The robot has to be equipped with a light sensor fixed on the underside in order to detect the painted grid.
- The robot knows the characteristics of the painted grid (organization, grid distance).

6.1.1 Correction of the position

The robot starts from a known position and updates its PPD and therefore its position (P_i) continuously by odometry (see fig. 6-1). When the robot crosses a grid line (P_r) in the real world (left image), it will choose the intersection position between its PPD and the grid line as its new corrected position (see P_c on the right image). In case of several matching points, the cross-point with the highest PPD-distribution wins (see also fig. 2-4 for other examples of intersections between PPD's and grid lines).

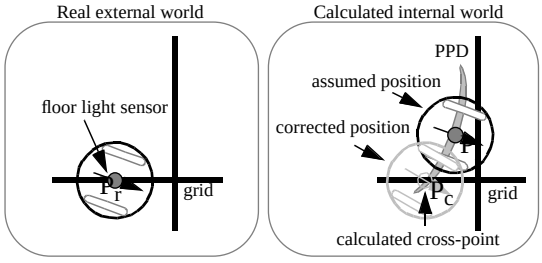


Fig. 6-1: Correction of the assumed position by matching the PPD with the grid

Note: Although the robot on position P_i is much closer to the vertical grid line, the final corrected position will be placed on the horizontal grid line (and moreover quite far away from the robot). Less carefully computed PPD or a intuitive solution would lead to a wrong position on the vertical line.

6.1.2 Correction of the angular error

As already discussed, the angular error is the most significant factor of odometry error. Therefore, the direction correction is very important. The matching point (P_c) of the PPD (see fig. 6-1) is not only the new robot position, but contains also information about the angular error. This means that every point on the PPD depends on a certain wheel speed difference $V_L - V_R$ (see section 3 "Calculation of a PPD model") and therefore on a certain angular drift $d\alpha$.

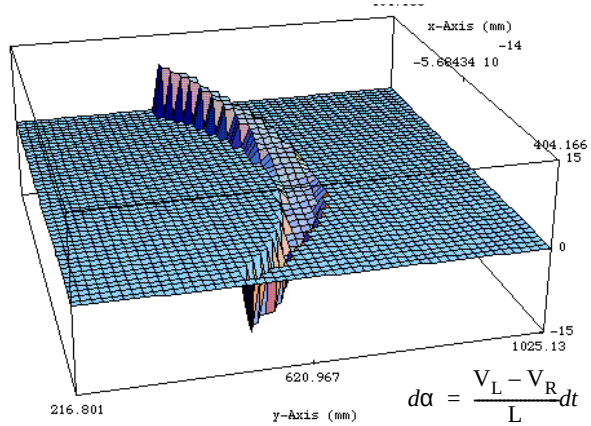


Fig. 6-2: Each point of the PPD indicates the angular drift $d\alpha$

The fig. 6-2 shows a plot of the wheel speed difference $V_L - V_R$ in the PPD. An intersection point of the PPD with a grid line supplies the average wheel speed difference and therefore the average angular drift, under the condition that the wheelbase (L) and the traveling time (t) are known.
Note: The speed is constant.

6.2 Active correction

As discussed, the influence of a grid fitter to the actual path will improve the odometry correction drastically (see section 5.2 "Information distribution of a grid"). This chapter describes, how the grid fitter works and how it has to be inserted in a standard navigation system.

6.2.1 Grid fitter

To take advantage of the grid information (see section 5.2 "Information distribution of a grid"), a "grid fitter" influences the planned trajectory of the robot. The resulting path is only slightly modified by the grid fitter, but the resulting advantage is relevant and will be illustrated by an example (see fig. 6-3).

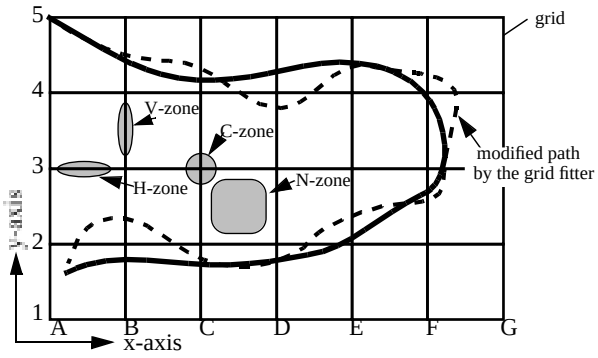


Fig. 6-3: The modified path (dotted line) makes a better use of the different zones than the intended path (solid line).

The robot is asked to follow the solid line, starting at position A5. While driving to the right side (to F4), the robot passes 4 V-zones, which allows to adjust 4 times the x position. During this time, the y position and the direction are neglected in favor of the freshly calibrated x position. After that, the robot will turn right, but unfortunately passing a C zone containing no useful information. Close to the point F3, the robot crosses finally a H-V zone, supplying significant position information. The way back to the left side of the scene (E2-A1) is similar to the first trajectory segment (A5-F4). The lack of V-H transitions prevent a sufficient odometry correction.

The fig. 6-4 illustrates the events produced by the intended path (upper diagram) and the corrected path (lower diagram). It can be seen that the corrected path contains more useful V-H transitions (compare with fig. 6-3).

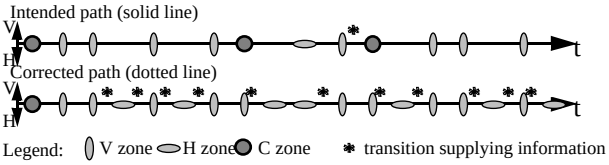


Fig. 6-4: Zone events, caused by the intended and corrected way

6.2.2 Including the grid fitter into a navigation system

The navigator steers the robot to a chosen destination, considering known objects in a map and some unexpected events. The navigator can normally be separated into a path planner and a pilot (see fig. 6-5):

Path planner (global navigator):

The path planner calculates the path defined by the task describer (a), considering the obstacles indicated in the environment map (b). The path planner will calculate the result (c) in advance and hand it over to the pilot step by step (for example left, right, slightly right, stop, ...).

Pilot (local navigator):

The pilot has no knowledge neither about the environment, nor about the aim of the task. It reacts to different motion suggestions submitted by the path planner (c), grid fitter (e) and obstacle avoider (d). Each unit supplies a motion suggestion (like right, left, strong right), combined with a certain priority (urgent, when convenient, mandatory). The pilot has to select or to combine these suggestions by selecting or fusion.

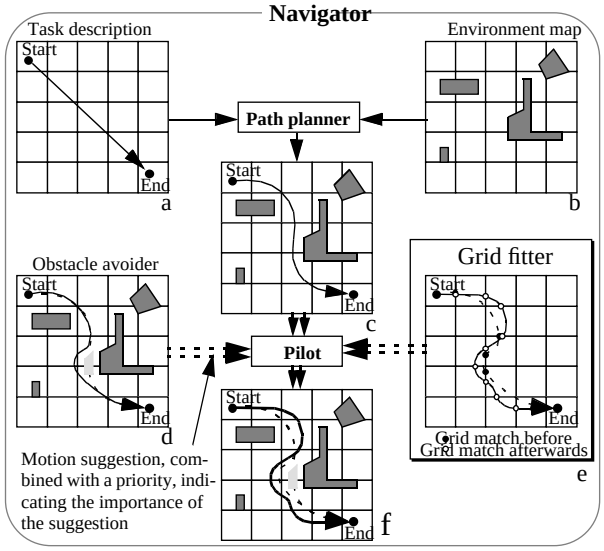


Fig. 6-5: Standard navigation system, containing a path planner and a pilot, influenced by the grid fitter

It is important to understand, that the grid fitter is only included into the navigator in order to improve the odometry correction algorithm. The illustration (see fig. 6-5) shows, that the “grid fitter” module has the same structure as the other modules (path planner and obstacle avoider) influencing the pilot. The integration of the grid fitter into the navigator produces no complications, if the interface of these modules is well defined. A call from the pilot to the grid fitter (and other modules) could be defined as follows:

```
void GridFitter(int actual_position,int actual_direction,int
   actual_speed,int *sug_direction,int *sug_priority);
```

The grid fitter module gets the actual position, direction and speed of the robot from the pilot and returns the suggested direction with a certain priority. Of course, the grid fitter has a local knowledge about the grid.

6.2.3 Grid fitter implemented by potential fields

The grid fitter optimizes the odometry correction algorithm by guiding the robot into interesting V and H zones (see section 5.2 "Information distribution of a grid"). This task can be easily modeled by potential fields and the following rules (see fig. 6-6):

- Single (isolated) grid lines (V and H zones) attract the robot (striped circles).
- Grid cross points (C zone) repel the robot (black circles).
- The speed and the direction of the robot are also considered in the potential field. This means that attractive zones behind the robot have a much lower influence than attractive zones in front of the robot. The robot can be seen as a moving mass with a certain speed and direction. Of course, the potential field does not change the absolute speed of the robot, which would be the case for a moving mass.
- V and H zones have to be crossed over alternately, thus when the robot has crossed some vertical lines, it has to be attracted in a stronger way by the horizontal ones.

The arrows on fig. 6-2 indicate the attractive and repulsive power of the zones.

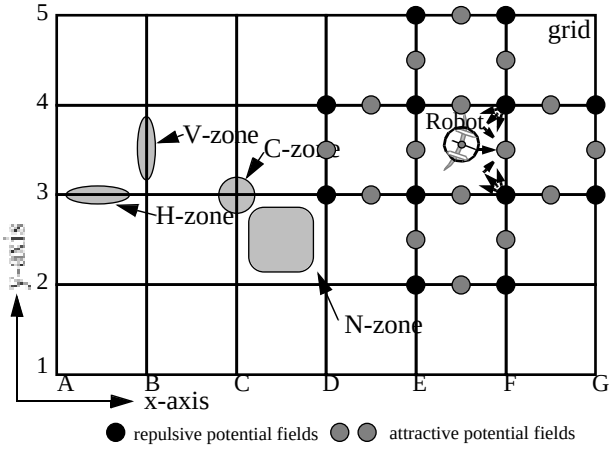


Fig. 6-6: Potential fields guide the robot over V and H zones, containing information

6.2.4 Other tasks of the grid fitter

The robot is not simply attracted and repelled by the potential fields. In order to improve the correction of the angular drift(see section 6.1.2 "Correction of the angular error"), the PPD should have a certain angle to the grid line (horizontal and vertical) while crossing it. This situation improves the recognition of angular errors. The following illustration (see fig. 6-7) shows the difference, if the PPD is crossing the grid line parallel (left image) or with a certain angle (right image).

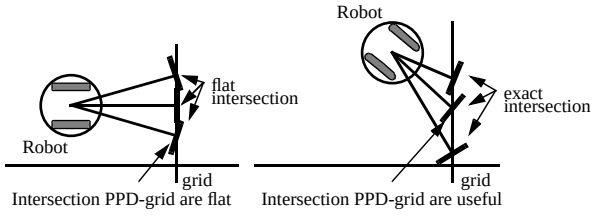


Fig. 6-7: Inclined movement to the grid improves the recognition of angular errors

The left image shows the PPD passing the grid line right-angled. The intersection (grid line and PPD) is flat and supplies no exact cross point. In the right image, the intersection supplies an exact cross point, because the PPD has a certain angle to the grid line.

Please note that the orientation of the PPD is important and not the orientation of the robot. In certain circumstances, the PPD could have a good angle to the grid line caused by its “inertia” (see section 4 "Examples"), even if the robot crosses it right-angled (see fig. 6-8).

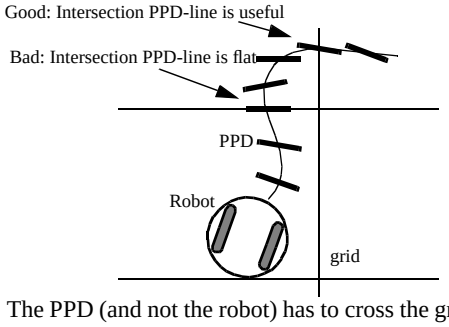


Fig. 6-8: The PPD (and not the robot) has to cross the grid line in a certain angle to get exact intersection points

Tasks of the grid fitter:

- Avoiding bad and passing good zones (see section 6.2.3 "Grid fitter implemented by potential fields").
- Crossing grid lines with a certain angle in order to improve the quality of the intersection.
- Minimizing the size of the PPD by adequate motion maneuvers (see section 4.3 "Turning back after 2 meters").

7 Practical Tests

The described algorithm was implemented on a Khepera robot, equipped with a Motorola 68311 Microprocessor, 128KRam and 256KRom. The aim was to program an autonomous robot running on a grid-field performing odometry correction without any external support like navigation devices or linked work stations.

7.1 Simplification of the algorithm

Because of the low CPU power of the Khepera robot (68311 @ 16MHz), neither the shape of the PPD nor the 3D cut with the grid line could be calculated in real rime. Therefore, the PPD was approached by several sample points of the PPD. These sample points are updated about 10 times per second. Connecting these points with a line gives a useful approximation of the PPD. Moreover, calculating the intersection of this line with grid lines uses little CPU power.

7.2 Description of the experiment

The experiment was done in two stages (see fig. 7-1); first without PPD correction (left image), then with correction (right image) to highlight the effect of the algorithm. The robot starts on a known position and direction. While running, the robot calculates its position continually simply by odometry and draws its position to a Tektonix-Terminal (left image). While moving on a random trajectory, the robot plots continuously its calculated position (×) and the real measured position(▫) on the screen. An increasing drift to the down-left side can be recognized. After one minute, the robot reaches an absolute error of more than a half square unit (one square is 10cm).

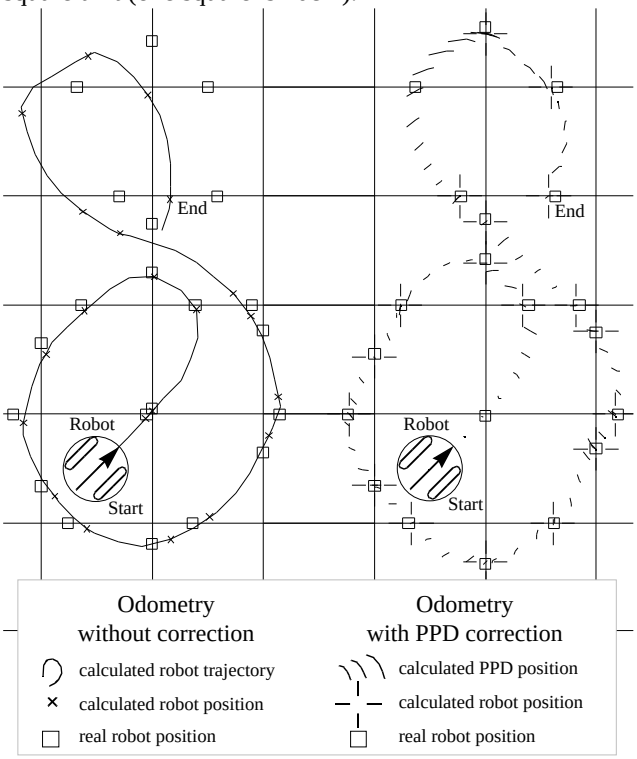


Fig. 7-1: Experiment without and with correction

The right illustration shows more or less the same robot trajectory, but corrected by the described algorithm. The path is marked by continuous plotted PPD's. The PPD is represented by a simple line connecting the sample points. When the robot receives a signal from the light sensor, meaning it is crossing a grid line, the intersection point is marked by a cross (-¦-), which is obviously always on a grid line. The corresponding real position is shown by a little square (▫).

The size of the PPD is reduced by two every time an intersection has been founded. Between these events, the PPD grows corresponding to the drift of 1% per wheel. The average size of the PPD keeps constant and is an indicator for the reliability of the robot position.

Note:

The direction of the PPD is not always right-angled to the robot trajectory. After turn a hard corner, the PPD can have the same direction as the robot trajectory for a certain time. This fact can be interesting for the Pilot (see section 6.2.4 "Other tasks of the grid fitter").

8 Conclusion

The present algorithm allows robots to efficiently correct their odometry simply by detecting the lines of a grid painted on the floor. Robots can thus navigate with minimum external help.

This algorithm demonstrates also an optimal exploitation of available information by using a new model of position probability distribution (PPD) and an interacting pilot (Grid Fitter).

9 Acknowledgments

I want to thank Dr. Paolo Ienne for his support while developing the equations in a very kind and humorous manner and J.B. Billeter for his impulses and advises.

10 References

[Piasecki95] Marek Piasecki, "Global Localization for Mobile Robots by Multiple Hypothesis Tracking", Robotics and Autonomous Systems Nr. 16, 1995, 12 pages (93-104)

[Pau90] L.F. Pau, "Mapping and spatial modeling for navigation", 1990, 355 pages, ISBN 3-540-52711-7

[Knieriemen91] Thomas Knieriemen, "Autonome mobile Roboter; Sensordateninterpretation und Weltmodellierung zur Navigation in unbekannter Umgebung", 1991, 258 pages, ISBN 3-411-15031-9

[Welch92] Sharon S. Welch, "Sensors and sensor systems for guidance and navigation II", 1992, 323 pages, ISBN 0-8194-0859-X

[Pruski96] Alain Pruski, "Robotique mobile, la planification de trajectoire", 1996, 236 pages, ISBN 2-86601-549-5