Journal of Intelligent and Robotic Systems **21:** 197–219, 1998.

Reducing Motion Inaccuracies on a Mobile Robot

RUI ARAÚJO and A. T. DE ALMEIDA
Institute of Systems and Robotics (ISR), and Electrical Engineering Department, University of Coimbra, Pólo II, Pinhal de Marrocos, 3030 Coimbra, Portugal; email: rui@isr.uc.pt

(Received: 29 November 1996; accepted: 1 September 1997)

Abstract. In this article we present two algorithms, for reducing the effects of control-space quantisation errors on a Khepera mobile robot. Specifically, we consider that control-space quantisation is present, when there is only a finite set of available robot wheels velocities. Thus the velocities of the robot wheels can not be chosen from any point in a continuous set. The first algorithm can be used to perform pure rotations (no translation) of the mobile robot while reducing the effects of these errors. The second algorithm can be used to perform robust straight-line motions, between the mobile robot current position, and a predefined goal position in its working environment. Simulation results demonstrating the effectiveness of the algorithm will be presented.

Key words: control quantisation errors, mobile robot.

1. Introduction

Keeping track of position is important in many mobile robot navigation methods. The estimation of mobile robot position, based on odometry and other dead-reckoning methods, has been frequently considered in the literature (e.g., [1, 2]). This problem may be seen as focused from a position-estimation perspective. However, this paper deals with a problem seldom considered in the literature, i.e., the effects of discretisation of both the time and control space on the behaviour of a robot. The problem's interest goes beyond the application discussed. Works related to this problem, in a broad sense, are those of G. Chirikjian et al. on discretely actuated robots [3–5]. In this article we present two algorithms, for reducing motion errors that are caused by control-space quantisation, in the specific case of a Khepera mobile robot [6, 7]. Error reduction is obtained by appropriately choosing the sequence of robot motion manoeuvres. The first algorithm can be used to perform *pure rotations* (no translation) of the mobile robot while reducing the effects of these errors. The second algorithm can be used to perform *straight-line motions*, between the mobile robot current position, and a predefined goal position in its working environment. On the Khepera mobile robot, there are two motion control variables: the velocities of the two wheels. Advantage is taken from the knowledge about the characteristics of control-space quantisation errors that are induced by the finite set of available wheels velocities. Thus, the problem we face in this paper, may be seen as being more focused from a control (actuation) perspective. Borenstein and Feng [8] present a systematic calibration

method for reducing odometry errors that arise from other nonidealities that are present on differential-drive mobile robots, such as the uncertainty about the effective wheelbase, and unequal wheel diameters.

The quantisation-error problem of digital systems is a long-standing problem (e.g., [9]). Control-space quantisation means that commands to the system can not be chosen from a continuous set, but can only take values from a set with a finite number of elements. These errors between the desired and actual commands, imply the appearance of state and output errors on the system. If care is not taken, the effects of these errors can accumulate, and after a series of commands, the actual final state of the system can be quite different from the desired state. The effect of input quantisation is that of reducing the reachable set of the system to a countable (not finite in general) subset of the state space. This set may or may not be dense, i.e., there may be a minimum unavoidable error to reach a given goal, or the error may be made arbitrarily small by long enough manoeuvres. For instance, in the problem of rotating the robot, if the minimum step of rotation is irrational in π, then the robot can reach any orientation with arbitrarily small errors, at the cost of possibly turning around many times before reaching that approximation. However, in the methods considered in this paper, there is an interest to keep the number of input steps on a reasonable number. The number of intervals is supposed to be prespecified. Thus the method for first obtaining it, is considered outside the scope of the paper.

The organisation of the paper is as follows. In Section 2, for completeness, we formulate the kinematic model of the Khepera mobile robot. In Section 3 we present the algorithm for robot rotation. Section 4 presents the algorithm for straight-line motion of the mobile robot. In Section 5 we present simulation results. Finally, in Section 6 we make some concluding remarks.

2. Kinematics of the Khepera Mobile Robot

This work is made around the Khepera miniature mobile robot [6, 7]. The circular shaped Khepera mobile robot (Figure 1) has two actuating wheels, each controlled by a DC motor that has an incremental encoder and can rotate in both directions.

An external computer can command each motor, to take a speed ranging from -10 to $+10$. The unit is the (encoder pulse)/10 ms that corresponds to 8 millimetres per second. The distance between the robot wheels, and the wheels's radius are given, respectively, by $\Delta r = 52.5$ mm, and $r_{\mathrm{w}} = 8$ mm. The number of pulses per revolution of the wheel is $N = 600$ (pulses/rev.). Each robot wheel has an associated up/down counter that accumulates the resulting number of pulses that were seen since the last (counter) reset. The rotation angle of a wheel per counter pulse is given by $\alpha_1 = 2\pi/N$. The corresponding wheel advancement is obtained by:

$$l = \frac{2\pi}{N} r_{\mathrm{w}} \tag{1}$$

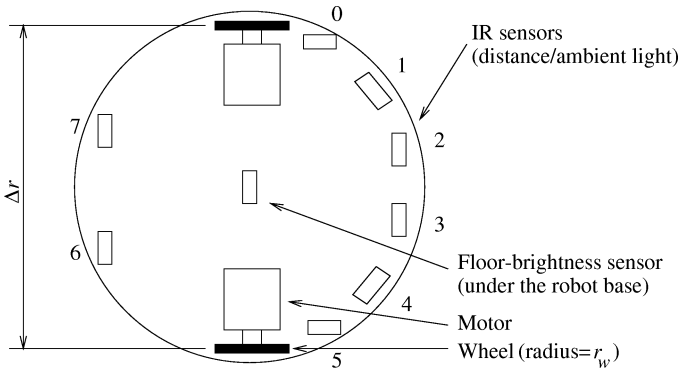

Figure 1. The Khepera miniature mobile robot.

which gives $l \approx 0.0837$ mm. The corresponding number of pulses, per millimetre of wheel advancement is given by $N_{\text{mm}} = 1/l \approx 11.94$.

In this paper, besides using the millimetre (mm) as a unit for measuring lengths, lengths are sometimes expressed in "increments". A distance (or a length) is expressed in increments, when it is specified by the number of encoder pulses required for the corresponding wheel to traverse such a distance. Thus, when expressed expressed in increments, a length has no unit of measure. It is just a number. However, this number is equal to the numerical value of the distance, if the unit used for measuring this distance was the length l, as defined above. For example, in increments, the distance between the wheels is given by:

$$\Delta r_{\text{inc}} = \Delta r / l \tag{2}$$

which gives $\Delta r_{\text{inc}} \approx 626.67$ increments.

In all this paper we assume that the mobile robot is being digitally controlled with a sampling period of T (expressed in seconds), and will make use of variable T_{10} to denote a time of 10 ms, i.e., $T_{10} = 10$ ms. Define $\mathbf{p}_R^W$ as the position of the robot's centre relative to the fixed world frame (Figure 2), at the beginning of a sampling interval:

$$\mathbf{p}_R^W = \left[\begin{array}{cc} x_R^W & y_R^W \end{array} \right]^{\mathrm{T}}. \tag{3}$$

The orientation of the robot's front, at the beginning of a sampling interval k, may be represented by a unit vector $\mathbf{d}(k)$, or with an angle $\theta_R = \theta_R(k) = \alpha(k)$ (Figure 2):

$$\mathbf{d} = \mathbf{d}(k) = \left[\begin{array}{cc} d_0(k) & d_1(k) \end{array} \right]^{\mathrm{T}}, \tag{4}$$

$$\theta_R = \theta_R(k) = \alpha(k) = \operatorname{atan2}\big[\pm d_1(k), d_0(k) \big], \quad d_0(k)^2 + d_1(k)^2 = 1, \tag{5}$$

$$d_0(k) = \cos\big[\theta_R(k)\big], \qquad d_1(k) = \pm \sin\big[\theta_R(k)\big]. \tag{6}$$

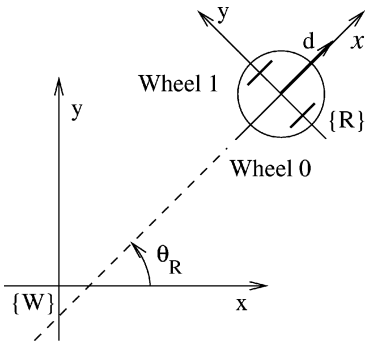

Figure 2. Mobile robot frame and world frame.

In all this paper whenever $\pm$ or $\mp$ signs appear, the upper sign applies to the normal case, and the lower sign applies when the world-frame Y-axis points in the direction opposite to that illustrated in Figures 2 and 3. The second case is useful, for example, in conjunction with the "Khepera Simulator Version 2.0" [10].

Assume that, at the beginning and at the end of a sampling interval k, the values of the encoder counters are, respectively, given by vectors $\mathbf{s}_R = [s_{0R}\ s_{1R}]^{\mathrm{T}}$, and $\mathbf{s}'_R = [s'_{0R}\ s'_{1R}]^{\mathrm{T}}$. The associated encoder-variation vector

$$\mathbf{d}_{sR} = \mathbf{s}'_R - \mathbf{s}_R = \left[\ ds_{0R}\ \ ds_{1R}\ \right]^{\mathrm{T}} \tag{7}$$

can be easily calculated from the speeds, v_i, of the two wheels on the sampling interval: $ds_{iR} = v_i T/T_{10}$.

On a sampling interval k, the rotation of the wheels imply a change on the robot's centre-frame (from $\{R\}$ to $\{R1\}$ – Figure 3). This change can be represented by a displacement vector of the robot's centre described on its own original frame, $\mathbf{p}^R_{R1}$; and by a change on the robot's frame rotation angle, $\theta^R_{R1} = \theta$ (Figure 3):

$$\mathbf{p}^R_{R1} = \left[\ x^R_{R1}\ \ y^R_{R1}\ \right]^{\mathrm{T}}, \qquad \theta^R_{R1} = \theta = \theta_{R1} - \theta_R. \tag{8}$$

If we define r to be the robot centre's trajectory rotation-radius, then from Figure 3 we can also see that x^R_{R1}, and y^R_{R1} can be calculated as follows:

$$x^R_{R1} = \begin{cases} r \cdot \sin\theta, & \text{if } ds_{0R} \neq ds_{1R}, \\ l \cdot ds_{0R}, & \text{if } ds_{0R} = ds_{1R}, \end{cases} \tag{9}$$

$$y^R_{R1} = \begin{cases} r \cdot (1 - \cos\theta), & \text{if } ds_{0R} \neq ds_{1R}, \\ 0, & \text{if } ds_{0R} = ds_{1R}. \end{cases} \tag{10}$$

At the end of a sampling interval k, the position of the robot is given by the vector $(\mathbf{p}^W_R)' = \mathbf{p}^W_{R1}$, and the orientation is given by vector $\mathbf{d}' = \mathbf{d}(k+1)$, or

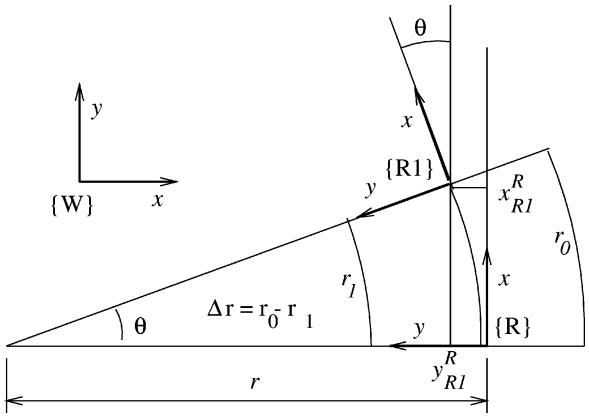

Figure 3. Robot frame change.

the angle θ_{R1}. The objective is to calculate these variables. This can be done using (8), and the following two equations:

$$(\mathbf{p}_R^W)' = \mathbf{p}_{R1}^W = \mathbf{p}_R^W + \mathbf{R}_R^W \cdot \mathbf{p}_{R1}^R, \qquad \mathbf{d}' = \mathbf{R}_{R1}^R \cdot \mathbf{d}. \tag{11}$$

$\mathbf{R}_R^W$, the rotation matrix of the robot frame with respect to the world frame, and $\mathbf{R}_{R1}^R$, the incremental rotation matrix, may be written as follows:

$$\mathbf{R}_R^W = \begin{bmatrix} d_0(k) & -d_1(k) \\ d_1(k) & d_0(k) \end{bmatrix}, \qquad \mathbf{R}_{R1}^R = \begin{bmatrix} \cos\theta & \mp\sin\theta \\ \pm\sin\theta & \cos\theta \end{bmatrix}. \tag{12}$$

At this point only r and θ remain to be calculated. Observing Figure 3, the following two equations can be written:

$$l \cdot ds_{1R} = r_1 \cdot \theta, \qquad l \cdot ds_{0R} = (r_1 + \Delta r) \cdot \theta, \tag{13}$$

where, as motivated by Figure 3, $r_0 = r + \Delta r/2$, and $r_1 = r - \Delta r/2$. The absolute value of r_0 (r_1) is easily interpreted as the rotation radius of the trajectory of wheel 0 (1). Note that Equations (13) are valid regardless of the value of ds_{0R} and ds_{1R}. By subtracting (if $ds_{0R} \neq ds_{1R}$) and adding the equations of (13), and expressing r and Δr in increments we finally obtain (note $\theta = 0$, $r = \infty$, if $ds_{0R} = ds_{1R}$):

$$\theta = \frac{ds_{0R} - ds_{1R}}{\Delta r_{\text{inc}}}, \tag{14}$$

$$r_{\text{inc}} = \frac{\Delta r_{\text{inc}}}{2} \cdot \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}}. \tag{15}$$

3. Robot Rotation

In this section we devise an algorithm for performing pure rotation of the Khepera mobile robot. It is an objective of the algorithm to reduce the effects of control-space quantisation errors, that arise due to the finite set of, equally spaced, motor velocities that are available as commands to the robot. We wish to perform a pure robot rotation, thus the two wheels must have symmetrical velocities. If θ is the desired rotation angle, then we have:

$$ds_{0R} = -ds_{1R} \tag{16}$$

Substituting in Equation (14) we get the total required number of increments:

$$ds_{0R} = \frac{\Delta r_{\text{inc}}}{2}\,\theta. \tag{17}$$

We will assume that the rotation will take place in $N_{T\theta}$ sampling intervals. The duration $N_{T\theta}$ may be calculated from a predefined velocity for the robot body or for the wheels.

Let the velocity of wheel 0, in each sampling interval, be denoted by $v_\theta(k)$ ($k = 1, \ldots, N_{T\theta}$) – for wheel 1 the velocities have opposite sign. The velocities $v_\theta(k)$ will be calculated such that the evolution of the angle θ will be, as closely as possible, linear. For that purpose we will begin by defining the variable $ds'_{0R}(k)$ that represents the total cumulative number of pulses of motor zero, since the beginning of the rotation, and thus taking into account velocities $v_\theta(i)$ ($i = 1, \ldots, k$):

$$ds'_{0R}(k) = ds'_{0R}(k-1) + v_\theta(i) \cdot \frac{T}{T_{10}}, \quad k = 1, \ldots, N_{T\theta}, \tag{18}$$

with $ds'_{0R}(0) = 0$.

For approximating the linear evolution of the angle θ, $v_\theta(k)$ ($k = 1, \ldots, N_{T\theta}$) will be calculated in the following way, where the "round()" function rounds to the nearest integer.

$$\begin{cases} ds_{0R}(k) = \dfrac{k}{N_{T\theta}} \cdot ds_{0R}, \\ v_\theta(k) = \text{round}\Big(\dfrac{ds_{0R}(k) - ds'_{0R}(k-1)}{T/T_{10}}\Big). \end{cases} \tag{19}$$

4. Straight-line Motion

In this section we present an algorithm for performing straight-line motion of the Khepera mobile robot while reducing the effects of control-space quantisation errors, that arise due to the finite set of, equally spaced, wheels velocities commands available. The objective of the method is to ensure that, at the end of

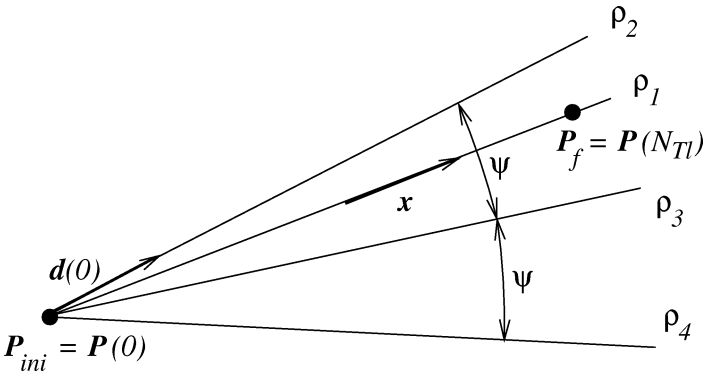

Figure 4. Straight-line approximation.

each sampling interval, the robot position is, as close as possible to the position it ideally would have if the motion had a constant velocity along the straight line joining the initial and desired final positions of the robot. The resulting trajectory will be close but not exactly a straight line. The straight-line motion can be divided on two phases. First, a pure rotation of the mobile robot, such that the front of the robot points to the final point as precisely as possible. Second, a motion to the final point using a path as close as possible to a straight line. For the pure rotation of phase 1, the algorithm presented in Section 3 can be used. In the rest of this section, the straight-line motion algorithm of phase 2 will be discussed.

We will assume that the second phase of the motion will take place in N_{Tl} sampling intervals. The duration N_{Tl} may be calculated from a predefined velocity of the wheels. The duration of the rotation phase is also prespecified as discussed in Section 3.

Consider Figure 4, where $\mathbf{P}_{\text{ini}} = \mathbf{P}(0)$, and $\mathbf{P}_f = \mathbf{P}(N_{Tl})$ are the initial and final points of the motion, respectively. In this figure, the following four straight lines are defined. Line ρ_1 – line connecting the initial point, $\mathbf{P}(0)$, to the final point, $\mathbf{P}(N_{Tl})$. Lines ρ_2, ρ_4 – respectively, the lines closest and second-closest to ρ_1, that are possible to achieve, at the end of the pure rotation motion of phase 1. This motion has an angle-resolution of $\theta = |\Delta ds_R|/\Delta r_{\text{inc}}$, with $|\Delta ds_R| = |ds_{0R} - ds_{1R}| = 2 \cdot T/T_{10}$ pulses per sampling interval. Line ρ_3 – the other line closest to ρ_1 that would be attainable if $|\Delta ds_R| = |ds_{0R} - ds_{1R}| = 1 \cdot T/T_{10}$ pulses per sampling interval. The slope of this line would only be achieved from ρ_2, with a robot motion that also included a translation component. This is because, pure rotation implies $|\Delta ds_R| = 2 \cdot |ds_{0R}| \cdot T/T_{10}$ – corresponding to an even number on the difference between the velocities of the two wheels. Note that, as is easily proved, among these four lines, ρ_2 and ρ_3, are those that are "slope-closest" to ρ_1, and are "slope-distant" from each other T/T_{10} pulses of difference per sampling interval.

Following an ideal straight-line trajectory would imply that, after starting in $\mathbf{P}_{\text{ini}}$, at the end of each sampling interval k ($k = 1, \ldots, N_{Tl}$), the robot would be in point $\mathbf{P}(k)$:

$$\mathbf{P}_{\text{ini}} = (x(0), y(0)) = \mathbf{P}(0), \tag{20}$$

$$\mathbf{P}_{\text{f}} = (x(N_{Tl}), y(N_{Tl})), \tag{21}$$

$$\mathbf{P}(k) = (x(k), y(k)), \quad k = 1, \ldots, N_{Tl}, \tag{22}$$

$$\begin{aligned} x(k) &= x(0) + \frac{k}{N_{Tl}} \cdot (x(N_{Tl}) - x(0)) \\ &= x(k-1) + \frac{1}{N_{Tl}} \cdot (x(N_{Tl}) - x(0)), \end{aligned} \tag{23}$$

$$\begin{aligned} y(k) &= y(0) + \frac{k}{N_{Tl}} \cdot (y(N_{Tl}) - y(0)) \\ &= y(k-1) + \frac{1}{N_{Tl}} \cdot (y(N_{Tl}) - y(0)). \end{aligned} \tag{24}$$

Because of trajectory errors that arise due to control-space quantisation, at the end of interval k, the robot is not in point $\mathbf{P}(k)$, but is in point $\mathbf{P}'(k) = (x'(k), y'(k))$ ($k = 1, \ldots, N_{Tl}$). We define the *trajectory error* in interval k in the following way:

$$e(k) = \|\mathbf{P}(k) - \mathbf{P}'(k)\|, \quad k = 1, \ldots, N_{Tl}, \tag{25}$$

where $\|(a, b)\| = \sqrt{a^2 + b^2}$ is the Euclidean norm of vector (a, b).

The overall algorithm for straight-line motion is based on the following ideas. At every sampling interval the algorithm chooses one of two options it has for motion. In the first option the robot does only a straight-line translation, maintaining its slope angle θ_R. This implies $ds_{0R} = ds_{1R}$. With the second option, the robot trajectory is a circular arc tangent to the trajectories of the adjacent intervals. In this way, besides an overall translation at the end of the sampling interval, there is a slight continuous change of the slope angle, θ_R, during the interval. The change of this angle is such that, after the advancements, the robot always has the slope of either line ρ_2 or line ρ_3 (Figure 4). Note that, the slopes of lines ρ_2 and ρ_3 are the ones closest to the slope of ρ_1, that can be attained. This second option implies $|\Delta ds_R| = |ds_{0R} - ds_{1R}| = 1 \cdot T/T_{10}$ pulses per sampling interval. The algorithm chooses from those two options, the one where we can attain a point $\mathbf{P}'(k)$ as close as possible to $\mathbf{P}(k)$. Note that the direction of slope change must be opposite to the previous slope change that occurred. We see that, with this method, we obtain not only positions but also slopes as close, as possible to the desired ones. This implies an overall trajectory with less errors and close to the desired straight-line. The two options for motion will be studied in Problems 1 and 2 below.

Let us start by Problem 1. The task to be solved with this problem is to use a straight-line to go from $\mathbf{P}'(k-1)$ to $\mathbf{P}(k)$ on sampling interval k. At the end of the interval we will achieve not $\mathbf{P}(k)$, but $\mathbf{P}'_1(k)$. $\mathbf{P}'_1(k)$ is the point with the least error, $e_1(k)$, that can be achieved in straight line with the available velocities of the robot. In this problem we are given $\mathbf{P}(k)$, $\mathbf{P}'(k-1)$, and $\mathbf{d}(k-1)$; and are asked to calculate $\mathbf{P}'_1(k)$, $e_1(k)$, and the velocities of the wheels ${}_1v'_{l0}(k) = {}_1v'_{l1}(k)$ (in pulses/10 ms). Note that in this Problem 1, vector $\mathbf{d}(k)$, and angle $\theta_R(k) = \alpha(k)$ are constant (see Equations (4)–(6) and Figure 2).

Let us start by determining the equation of the straight line $\rho'_1(k-1)$ that passes on $\mathbf{P}'(k-1)$, and has a slope of $\alpha(k-1)$. We can write the following parametric equation on λ (see also Equations (4)–(6) and Figure 2)

$$\begin{cases} x = x'(k-1) + \lambda \cos(\alpha(k-1)), \\ y = y'(k-1) \pm \lambda \sin(\alpha(k-1)). \end{cases} \tag{26}$$

Multiplying the two equations of (26), respectively, by $\sin(\alpha(k-1))$ and $\cos(\alpha(k-1))$, and subtracting the resulting equations, we obtain the equation of the straight line in its general form:

$$Ax + By + C = 0, \tag{27}$$

where

$$\begin{cases} A = \sin(\alpha(k-1)) = \pm d_1(k), \\ B = \mp \cos(\alpha(k-1)) = \mp d_0(k), \\ C = \pm y'(k-1)\cos(\alpha(k-1)) - x'(k-1)\sin(\alpha(k-1)). \end{cases} \tag{28}$$

Call this straight line $\rho'_1(k-1)$, observe Figure 5, and note the following: (1) $\mathbf{P}''_1(k)$ is the point of line $\rho'_1(k-1)$ that is closest to $\mathbf{P}(k)$. The velocity, ${}_1v''_l(k)$, required for going from $\mathbf{P}'(k-1)$ to $\mathbf{P}''_1(k)$ is in general real-valued. (2) $\mathbf{P}'_1(k)$ and $\mathbf{P}'''_1(k)$ are the points of line $\rho'_1(k-1)$ closest to $\mathbf{P}(k)$, that can be attained with the integer velocities available on the robot wheels. (3) $\mathbf{P}'_1(k)$ is the point of line $\rho'_1(k-1)$ that is closest to $\mathbf{P}(k)$ while also able to be attained with the available integer robot velocities. As is easily proved, this velocity is given by round$({}_1v''_l(k))$. Therefore, we wish to determine the point, $\mathbf{P}''_1(k)$, that belongs to the straight line given by Equation (27), and is at a minimum distance of point $\mathbf{P}(k) = (x(k), y(k))$. After obtaining the closest point, it is trivial to calculate the minimum distance.

The coordinates of point $\mathbf{P}''_1(k)$, may be obtained using the Lagrange multipliers method for minimising the distance function subject to restriction (27). We get,

$$\begin{cases} x''_1(k) = \dfrac{B^2 x(k) - ABy(k) - CA}{A^2 + B^2}, \\ y''_1(k) = \dfrac{A^2 y(k) - ABx(k) - CB}{A^2 + B^2}, \end{cases} \tag{29}$$

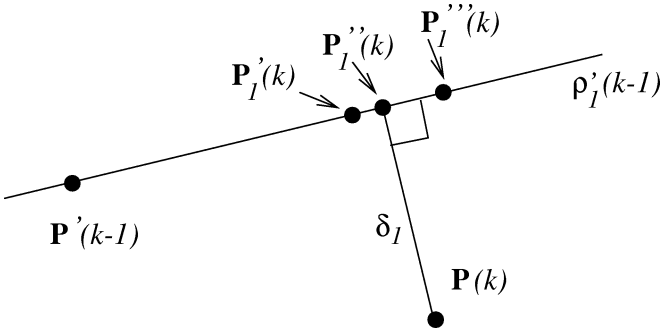

Figure 5. Equal velocities (Problem 1).

where A, B, and C are given by Equation (28). Next we can calculate the real-valued velocity (in pulses/10 ms) that is needed to attain $\mathbf{P}_1''(k)$, as follows,

$$ {}_1v_l''(k) = \frac{\|\mathbf{P}_1''(k) - \mathbf{P}'(k-1)\|}{l \cdot T/T_{10}}, \tag{30}$$

where l is given by Equation (1). The integer velocity corresponding to the closest point, $P'(k)$, that is possible to attain with the available robot velocities is given by,

$$ {}_1v_l'(k) = \text{round}({}_1v_l''(k)) \tag{31}$$

The coordinates of point $\mathbf{P}_1'(k) = (x_1'(k), y_1'(k))$, can now be obtained from x, and y in Equation (26), of line $\rho_1'(k-1)$, if the following value of λ is used:

$$ \lambda = \lambda_\rho = {}_1v_l'(k) \cdot l \cdot T/T_{10}. \tag{32}$$

The resulting error-distance, $e_1(k)$, can be calculated using an equation similar to (25).

Taking into account Figure 4, define $\sigma(k) = \text{sgn}(\mathbf{x} \times \mathbf{d}(k))_z$ where $\text{sgn}(a) = a/|a|$ except $\text{sgn}(0) = 1$, $\times$ denotes the vector product, and $)_z$ denotes the z component of the vector. Vector $\mathbf{x}$ is given by:

$$ \mathbf{x} = \frac{\mathbf{P}_\text{f} - \mathbf{P}_\text{ini}}{\|\mathbf{P}_\text{f} - \mathbf{P}_\text{ini}\|}, $$

and $\mathbf{d}(k)$ is a unit vector with the same direction of the x axis of the robot frame on time interval k.

Let us now analyse Problem 2. The objective of this problem is to move the robot from $\mathbf{P}'(k-1)$ to $\mathbf{P}(k)$ on sampling interval k, while enforcing the following conditions on the velocities of the robot's wheels: $|\Delta ds_R| = |ds_{0R} - ds_{1R}| = 1 \cdot T/T_{10}$ pulses per sampling interval, and $\text{sgn}(\Delta ds_R) = \text{sgn}[\sigma(k-1)]$. At the end of the interval we will attain not $\mathbf{P}(k)$, but $\mathbf{P}_2'(k)$. From all the points that can be reached in those conditions, $\mathbf{P}_2'(k)$ is the one with the minimum associated

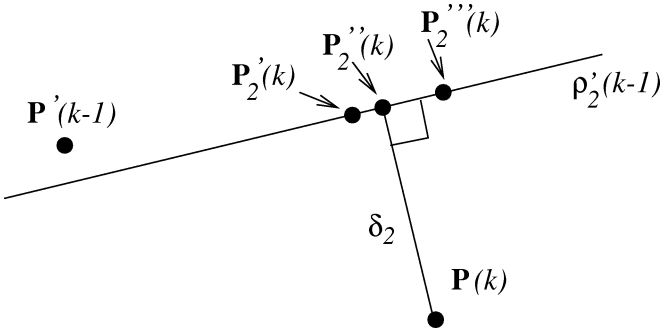

Figure 6. Velocities differing by one (Problem 2).

error, $e_2(k)$. In this problem we are given $\mathbf{P}(k)$, $\mathbf{P}'(k-1)$, and $\mathbf{d}(k-1)$; and are asked to calculate $\mathbf{P}'_2(k)$, $e_2(k)$, and the velocities of the wheels ${}_2v'_{l0}(k)$ and ${}_2v'_{l1}(k)$ (in pulses/10 ms).

Let $\rho'_2(k-1)$ be the locus of the points attainable from $\mathbf{P}'(k-1)$ after time T, by varying the wheel's velocities ${}_2v'_{l0}(k)$ and ${}_2v'_{l1}(k)$, while still satisfying the conditions of the problem. However surprising it may seem to our intuition, the fact is that, as we will show shortly, $\rho'_2(k-1)$ is a straight line. Having this in mind, we can observe Figure 6. Note that whereas in Figure 5 line $\rho'_1(k-1)$ included the robot trajectory, here in Figure 6 line $\rho'_2(k-1)$ is distinct from the robot trajectory. Regarding Figure 6 note the following: (1) $\mathbf{P}''_2(k)$ is the point of line $\rho'_2(k-1)$ that is closest to $\mathbf{P}(k)$. The required velocities for going from $\mathbf{P}'(k-1)$ to $\mathbf{P}''_2(k)$ are in general real-valued, and here we call them ${}_2v''_{l0}(k)$ and ${}_2v''_{l1}(k)$; (2) $\mathbf{P}'_2(k)$ and $\mathbf{P}'''_2(k)$ are the points of line $\rho'_2(k-1)$ closest to $\mathbf{P}(k)$, that can be attained with the integer velocities available on the robot wheels; (3) $\mathbf{P}'_2(k)$ is the point of line $\rho'(k-1)$ that is closest to $\mathbf{P}(k)$ while also able to be attained with the available integer robot velocities. These velocities, ${}_2v'_{l0}(k)$ and ${}_2v'_{l1}(k)$, as is easily seen, can be computed with the following equations,

$$ {}_2v'_{l1}(k) = \begin{cases} \text{round}\big({}_2v''_{l1}(k)\big), & \text{if } \sigma(k-1) > 0, \\ {}_2v'_{l0}(k) - 1, & \text{if } \sigma(k-1) < 0, \end{cases} \tag{33} $$

$$ {}_2v'_{l0}(k) = \begin{cases} {}_2v'_{l1}(k) - 1, & \text{if } \sigma(k-1) > 0, \\ \text{round}\big({}_2v''_{l0}(k)\big), & \text{if } \sigma(k-1) < 0. \end{cases} \tag{34} $$

Let us suppose for a moment that velocities ${}_2v''_{l0}(k)$ and ${}_2v''_{l1}(k)$ were applied to the robot wheels. Thus the attained point would be $\mathbf{P}''_2(k)$. From Equations (14) and (15), and the hypothesis of this problem, the angle and radius of rotation of the robot's centre frame can respectively be calculated as follows:

$$ \theta'' = \frac{ds_{0R} - ds_{1R}}{\Delta r_{\text{inc}}} = \frac{T \,\text{sgn}[\sigma(k-1)]}{T_{10} \Delta r_{\text{inc}}}, \tag{35} $$

$$r'' = \frac{\Delta r}{2} \cdot \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}} = \text{sgn}\,[\sigma(k-1)] T_{10} \frac{\Delta r}{2} [2\ {}_2v_l''(k) - 1], \tag{36}$$

where we define ${}_2v_l''(k)$ as follows,

$${}_2v_l''(k) = \begin{cases} {}_2v_{l1}''(k), & \text{if } \sigma(k-1) > 0, \\ {}_2v_{l0}''(k), & \text{if } \sigma(k-1) < 0. \end{cases} \tag{37}$$

Taking into account Equations (4), (8)–(12), we can write the following equations for the coordinates of the points that, with the available robot wheels velocities, can be attained at the end of sampling interval k, if we have started on point $\mathbf{P}'(k-1)$:

$$\begin{cases} x = x'(k-1) + r'' \cdot a_x, \\ y = y'(k-1) + r'' \cdot a_y, \end{cases} \tag{38}$$

where

$$\begin{cases} a_x = d_0(k-1) \cdot \sin(\theta'') - d_1(k-1) \cdot [1 - \cos(\theta'')], \\ a_y = d_1(k-1) \cdot \sin(\theta'') + d_0(k-1) \cdot [1 - \cos(\theta'')]. \end{cases} \tag{39}$$

Multiplying the two equations of (38), respectively by a_y and a_x, and subtracting the resulting equations, we obtain

$$[x - x'(k-1)] \cdot a_y = [y - y'(k-1)] \cdot a_x. \tag{40}$$

Equation (40) confirms the statement previously made, that the set of points attainable, $\rho_2'(k-1)$, constitutes a straight line. From (40), we can equivalently write the equation of this straight line on the form of Equation (27), where,

$$\begin{cases} A = a_y, \\ B = -a_x, \\ C = y'(k-1) \cdot a_x - x'(k-1) \cdot a_y. \end{cases} \tag{41}$$

Now, to obtain $\mathbf{P}_2''(k)$, we can apply an approach similar to the one that was used for solving Problem 1. Thus, similarly to Equation (29), the coordinates of $\mathbf{P}_2''(k)$ can be calculated using the following equation, with A, B, and C given by (41).

$$\begin{cases} x_2''(k) = \dfrac{B^2 x(k) - AB y(k) - CA}{A^2 + B^2}, \\ y_2''(k) = \dfrac{A^2 y(k) - AB x(k) - CB}{A^2 + B^2}. \end{cases} \tag{42}$$

After that, velocity ${}_2v_l''(k)$ is calculated in the following way:

$${}_2v_l''(k) = \frac{\|\mathbf{P}_2''(k) - \mathbf{P}'(k-1)\|}{l \cdot T / T_{10}}. \tag{43}$$

Using Equations (37), and (33)–(34) we calculate ${}_2v'_{l0}(k)$ and ${}_2v'_{l1}(k)$, that lead the robot to point $\mathbf{P}'_2(k)$.

Similarly to Equations (36) and (37), we can calculate the effective rotation radius, r', when we go to $\mathbf{P}'_2(k)$ as follows

$$r' = \frac{\Delta r}{2} \cdot \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}} = \text{sgn}[\sigma(k-1)]\ T_{10}\ \frac{\Delta r}{2}\ [2\ {}_2v'_l(k) - 1], \tag{44}$$

where,

$${}_2v'_l(k) = \begin{cases} {}_2v'_{l1}(k), & \text{if } \sigma(k-1) > 0, \\ {}_2v'_{l0}(k), & \text{if } \sigma(k-1) < 0. \end{cases} \tag{45}$$

The coordinates of point $\mathbf{P}'_2(k) = (x'_2(k), y'_2(k))$, can now be obtained from x, and y, in Equation (38), of line $\rho'_2(k-1)$, if r'' is substituted by r'. Finally, the resulting error-distance, $e_2(k)$, can be calculated by using an equation similar to (25).

According to the overall algorithm, if $e_1(k) \leqslant e_2(k)$, velocities ${}_1v'_{l0}(k) = {}_1v'_{l1}(k)$ are chosen. In this case the point attained at the end of the sampling interval is $P'(k) = P'_1(k)$. Otherwise, velocities ${}_2v'_{l0}(k)$ and ${}_2v'_{l1}(k)$ are chosen. In this case the robot attains point $\mathbf{P}'(k) = \mathbf{P}'_2(k)$. The resulting error is thus $e(k) = \min(e_1(k), e_2(k))$. Note that $\mathbf{P}'(k)$ will be used by the algorithm on the next sampling interval. At that stage, point $\mathbf{P}'(k)$ can be estimated by the approach we have taken in this section. However, $\mathbf{P}'(k)$ can also be obtained from a sensing system that provides feedback from the real world (e.g., odometry).

Experiments with this basic two-option algorithm revealed some sub-optimalities, on two types of situations, that motivated the introduction of two corresponding improvements.

Before analysing the two situations, we will introduce some preliminary definitions that will be used on the discussion. Let $\mathbf{P} = (x, y)$ be a point, and define $\text{side}(\mathbf{P}) = \text{sgn}[\mathbf{x} \times (\mathbf{P} - \mathbf{P}(0))]_z$, where $\mathbf{x}$ is a unit vector with the direction of ρ_1 (the line that passes through $\mathbf{P}_{\text{ini}}$ and $\mathbf{P}_{\text{f}}$ – see Figure 4), $\times$ denotes the vector product operation, and $]_z$ denotes the Z component of the vector. As can easily be seen, side($\mathbf{P}$) is positive if and only if, point $\mathbf{P}$ belongs to the left semiplane defined by the ideal trajectory line, when we advance from the initial to the final point. Similarly, $\text{side}(\mathbf{P}) = -1$ if point $\mathbf{P}$ belongs to the right semiplane.

Suppose that, in terms of slope, line ρ_2 is closer than ρ_3, to the ideal trajectory line ρ_1 (as may be easily seen, the two improvements that will be introduced, remain valid even if the opposite case occurs). Line ρ_2 (ρ_3) is called the *most (least) similar line*. The *most (least) similar slope*, τ_2 (τ_3), is defined as the slope of the most (least) similar line. Additionally, let α_2 (α_3) be the angle between this line and line ρ_1. If $\mathbf{d}_0^*$ is a unit vector with the direction of the most similar line ρ_2 (see Figure 4), we define σ_0 as,

$$\sigma_0 = \text{sgn}(\mathbf{x} \times \mathbf{d}_0^*)_z. \tag{46}$$

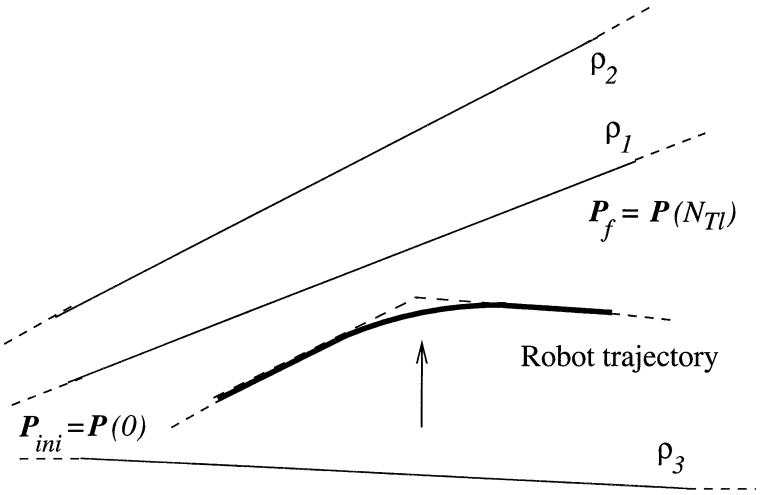

Figure 7. Example illustrating the first undesirable situation.

The first undesirable observed situation is the following. The robot is approaching line ρ_1, through a line of most similar slope (thus side$(P'(k)) \cdot \sigma_0 < 0$ and $\sigma(k) \cdot \sigma_0 > 0$), and a minimum curvature advancement was performed – see Figure 7. This corresponds to choosing $|\Delta ds_R| = 1 \cdot T/T_{10}$, because the solution of Problem 2 was the most favourable. This leads the robot trajectory to start driving away from the ideal trajectory line ρ_1. In this way, although during some of the following sampling intervals (the number may vary), the minimum error option is chosen, the long term tendency for error decrease will be weaker. What is happening on the sampling interval under consideration is that, although the error would be greater if the approach to the ideal trajectory were maintained, choosing this option would enable the achievement of lower errors on future sampling intervals. Clearly the robot trajectory slope should change only if it is driving away from line ρ_1, and as a consequence of the change it shall start approaching ρ_1.

In order to avoid this first situation, we only allow a minimum curvature advancement (using the solution of Problem 2) on interval k if, on that interval, the robot is driving away from line ρ_1 (see Figure 4). Formally, the following condition was used to detect the situation when the robot is allowed to curve:

$$\begin{aligned}&\text{Condition 1} = \\ &\quad \sigma(k-1) > 0 \wedge \text{side}[\mathbf{P}'(k-1)] > 0 \wedge \text{side}[\mathbf{P}'_1(k)] > 0 \wedge \text{side}[\mathbf{P}'_2(k)] > 0 \\ &\quad \vee \\ &\quad \sigma(k-1) < 0 \wedge \text{side}[\mathbf{P}'(k-1)] < 0 \wedge \text{side}[\mathbf{P}'_1(k)] < 0 \wedge \text{side}[\mathbf{P}'_2(k)] < 0,\end{aligned} \tag{47}$$

where $\wedge$ and $\vee$ denote the logical "AND" and "OR" operations, respectively. To clearly avoid the situation, Condition 1 ensures that, (i) the robot is currently

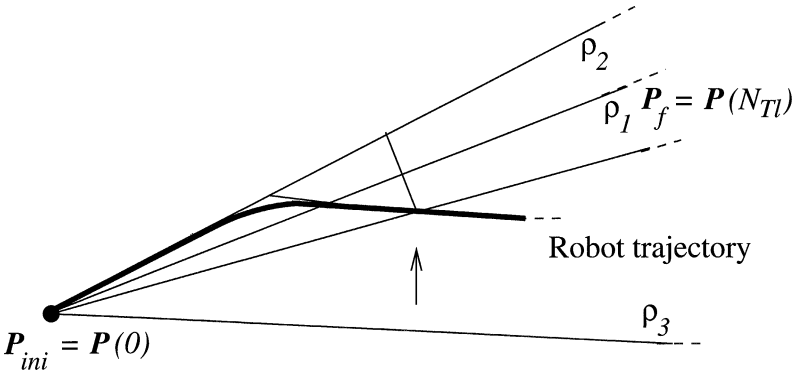

Figure 8. Example illustrating the second undesirable situation.

driving away from ρ_1, and (ii), the present robot position $\mathbf{P}'(k-1)$, and the next two possible robot positions, $\mathbf{P}'_1(k)$ and $\mathbf{P}'_2(k)$, are all on the same side of ρ_1.

The second situation observed is the following. Suppose that, after the pure rotation phase that precedes the real straight-line motion, the front-slope of the robot was the most similar, τ_2. Then, during the motion, on the sampling intervals that preceded some interval k, the robot was performing a straight-line motion with least similar slope. As a result of the motion on interval k, the robot attained a new position where the error is greater than it would be, if from the very beginning, the robot used the motion along the most similar line ρ_2 (see Figure 8). This situation is unfavourable on every interval k, where it may occur. It becomes particularly undesirable when occurring sufficiently close to the last sampling interval, to invalidate any later recovery of the error until the end of the motion.

To avoid the effects of this second situation, a modification based on the following idea, was introduced on the algorithm: on sampling interval k, force the robot trajectory slope to switch (i.e., use the solution of Problem 2), whenever the best solution of Problems 1 and 2, results on a point for which the distance to $\mathbf{P}(k)$ (point of the ideal trajectory) is greater than the distance, $e_b(k)$, from $\mathbf{P}(k)$ to a bounding line $\rho_b(k)$. Switching enforcement only takes place when the robot is driving away from the ideal trajectory. Formally, the switching-enforcement condition is the following:

$$\text{Condition 1 is TRUE} \wedge \min\left(e_1(k), e_2(k)\right) > e_b(k). \tag{48}$$

The bounding line, $\rho_b(k)$, passes through point $\mathbf{P}_b(k)$, and makes with line ρ_1, an angle α_b, satisfying $|\alpha_b| = |\alpha_2|$. Thus, in absolute value, both lines $\rho_b(k)$ and ρ_2, make the same angle with line ρ_1. Point $\mathbf{P}_b(k)$ initially coincides with $\mathbf{P}_{\text{ini}} = \mathbf{P}(0)$, and is subsequently changed, zero or more times, during the motion. On sampling interval k_2, it is changed to $\mathbf{P}_b(k_2) = \mathbf{P}(k_1 - 1)$. The sampling interval k_2 is such that, the robot changed its trajectory slope from the most similar, τ_2, to the least similar, τ_3. Interval k_1 ($k_1 < k_2$) depends on k_2, and

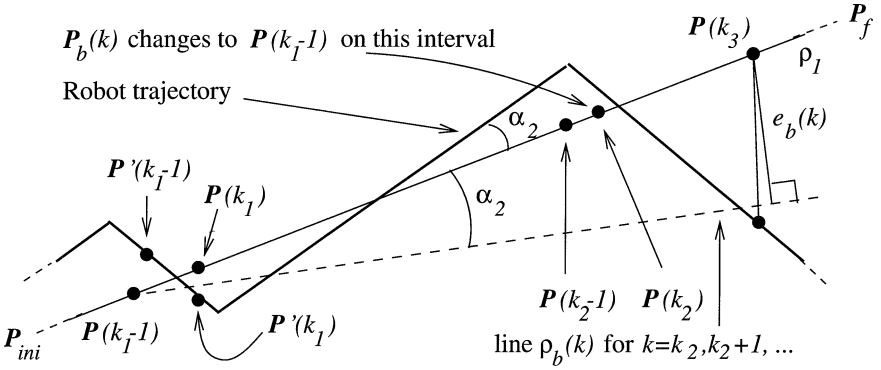

Figure 9. Illustration of the procedure used for overcoming the second situation.

corresponds to the most recent interval where the robot's trajectory crossed, with the least similar slope, τ_3, the ideal trajectory, ρ_1. Figure 9 illustrates this procedure. Minimum curvature advancements correspond to sampling intervals with circular arc trajectories. However, in order to simplify Figure 9, those arcs were not depicted, but are approximated by the continuation of the adjacent straight line trajectories until their intersection.

In order to formally define $\mathbf{P}_b(k)$, let us start by defining point $\mathbf{P}_a(k)$:

$$\mathbf{P}_a(k) = \begin{cases} \mathbf{P}_{\text{ini}}, & k = 1, \\ \mathbf{P}(k-1), & \text{Condition 2 is TRUE}, \\ \mathbf{P}_a(k-1), & \text{otherwise}, \end{cases} \tag{49}$$

where

$$\text{Condition 2} = k > 1 \ \wedge \text{side}[\mathbf{P}'(k-2)] \cdot \text{side}[\mathbf{P}'(k-1)] < 0 \wedge \sigma_0 \cdot \sigma(k-1) < 0. \tag{50}$$

Let us also define a logical function $\text{np}(k)$, that indicates if the value of point $\mathbf{P}_a(k)$, should be assigned to point $\mathbf{P}_b(k)$, when the next switch from the most similar trajectory slope, τ_2, to the least similar slope, τ_3, takes place.

$$\text{np}(k) = \begin{cases} \text{TRUE}, & \text{Condition 2 is TRUE}, \\ \text{FALSE}, & \text{Condition 3 is TRUE}, \\ \text{np}(k-1), & \text{otherwise}. \end{cases} \tag{51}$$

$$\text{Condition 3} = k > 1 \ \wedge \sigma_0 \cdot \sigma(k-1) > 0 \wedge \sigma(k-1) \cdot \sigma(k) < 0 \wedge \wedge \ \text{np}(k-1) = \text{TRUE}. \tag{52}$$

Point $\mathbf{P}_b(k)$ is then formally defined as:

$$\mathbf{P}_b(k) = \begin{cases} \mathbf{P}_{\text{ini}}, & k = 1, \\ \mathbf{P}_a(k), & \text{Condition 3 is TRUE}, \\ \mathbf{P}_b(k-1), & \text{otherwise}. \end{cases} \tag{53}$$

OVERALL ALGORITHM

1. IF $k = 1$ THEN

   1.1. Calculate the unit vector $\mathbf{d}_0^*$ which has the direction of either line ρ_2 or line ρ_3 whichever forms the lowest angle with line ρ_1.

   1.2. Calculate σ_0 according to Equation (46).

   1.3. Let $\mathbf{P}_b := \mathbf{P}_{\text{ini}}$; np := FALSE; PrevCorr := FALSE.

2. ELSE

   2.1. IF side$[\mathbf{P}'(k-2)] \cdot$ side$[\mathbf{P}'(k-1)] < 0$ AND $\sigma_0 \cdot \sigma(k-1) < 0$ THEN

      2.1.1. Let $\mathbf{P}_a := \mathbf{P}(k-1)$.

      2.1.2. Let np := TRUE.

3. Solve Problem 1 obtaining $\mathbf{P}'_1(k)$, $e_1(k)$, and ${}_1v'_{l0}(k) = {}_1v'_{l1}(k)$ as the results.

4. Solve Problem 2 obtaining $\mathbf{P}'_2(k)$, $e_2(k)$, ${}_2v'_{l0}(k)$, and ${}_2v'_{l1}(k)$ as the results.

5. Let $e_b(k) :=$ distance from $\mathbf{P}(k)$ to $\rho_b(k)$.

6. IF $\Big[e_2(k) < e_1(k)$ OR $\min(e_1(k), e_2(k)) > e_b(k)$ AND PrevCorr = TRUE$\Big]$ AND $\sigma(k-1) \cdot$ side$[\mathbf{P}'(k-1)] > 0$ AND $\sigma(k-1) \cdot$ side$[\mathbf{P}'_1(k)] > 0$ AND $\sigma(k-1) \cdot$ side$[\mathbf{P}'_2(k)] > 0$ THEN

   6.1. IF $\sigma_0 \cdot \sigma(k-1) > 0$ AND np = TRUE THEN

      6.1.1. Let $\mathbf{P}_b := \mathbf{P}_a$.

      6.1.2. Let np := FALSE.

   6.2. Apply to the robot wells the following velocities that were obtained from the solution of Problem 2: $v_0(k) = {}_2v'_{l0}(k)$ and $v_1(k) = {}_2v'_{l1}(k)$. As a result of those applied velocities the new robot position will be: $\mathbf{P}'(k) := \mathbf{P}'_2(k)$.

   6.3. Let PrevCorr := TRUE.

7. ELSE

   7.1. Apply to the robot wells the following velocities that were obtained from the solution of Problem 1: $v_0(k) = {}_1v'_{l0}(k) = v_1(k) = {}_1v'_{l1}(k)$. As a result of those applied velocities the new robot position will be: $\mathbf{P}'(k) := \mathbf{P}'_1(k)$.

Figure 10. Overall straight-line motion algorithm.

The straight-line motion method of this section is summarised on the overall algorithm of Figure 10, to be used on each sampling interval of a straight line motion. On this algorithm, variables $\mathbf{P}_b$, $\mathbf{P}_a$, and np were used to represent, on sampling interval k, the values of $\mathbf{P}_b(k)$, $\mathbf{P}_a(k)$, and $\text{np}(k)$, as defined on Equations (53), (49), and (51), respectively.

The algorithm of Figure 10 uses a variable called "PrevCorr", with the following purpose. Due to the limited set of available robot wheels velocities, it is probable that, on the initial intervals of motion, the second situation is detected with the condition of Equation (48) (also used on algorithm of Figure 10). In an attempt to overcome this situation, the algorithm would force a minimum curvature advancement (use the solution of Problem 2) that leads to position error increase. Since we are at the beginning of a motion, this increase will be high in relative terms. For this reason the algorithm only tries to detect this situation, after having made a minimum curvature advancement not induced by the second situation. This invalidates the detection of this situation on the initial sampling intervals of a motion.

5. Experiments

This section presents results of experiments demonstrating the effectiveness of the algorithms presented in Sections 3 and 4, with the Khepera mobile robot. The results presented, were achieved using the "Khepera Simulator Version 2.0" [10]. The default sampling period of the simulator, $T = 62.7$ ms, was used. To test the approach, it was enforced on the simulator, that only quantisation errors are present.

The experimental simulation study was conducted in the following way. There were 100 different straight-line motions performed. In all of those motions the initial robot position was always the same: $\mathbf{P}(0) = (x(0), y(0)) = (100, 100)$, with an orientation angle, θ_R, of zero degrees (x-axis aligned). On motion i ($i = 0, \ldots, 99$) the robot was requested to move to a new point ${}^i\mathbf{P}_\text{f} = \mathbf{P}(0) + {}^i\Delta\mathbf{P}$, where the total displacement is expressed in polar coordinates as ${}^i\Delta\mathbf{P} = ({}^ir, {}^i\theta)$. The (ideal) trajectory length ${}^ir$ was held constant on all the motions with ${}^ir = 760$ mm. The (ideal) trajectory angle was swept along a set of equally spaced values between ${}^0\theta = 112.3$ and ${}^{99}\theta = 113.446$ degrees, i.e., ${}^i\theta = {}^0\theta + (k/99)({}^{99}\theta - {}^0\theta)$. The procedure and reasoning for choosing the values of ${}^0\theta$ and ${}^{99}\theta$ was the following. The angle resolution available on the initial pure rotation, corresponds to the use of the minimum velocity on one sampling interval. From Equation (16), this resolution is given by $\theta_\text{res} = 2T/(T_{10}\Delta r_\text{inc}) \approx 1.146$ degrees. Due to the finite set of robot angles available, the algorithm behaviour will repeat for every additional of θ_res on the required (ideal) trajectory angle. For this reason, it suffices to study an angular range of approximately 1.146 degrees. This is precisely the amplitude $({}^{99}\theta - {}^0\theta)$ of the angle range that we used. Note that ${}^0\theta \approx 98 \cdot \theta_\text{res}$ and ${}^{99}\theta \approx 99 \cdot \theta_\text{res}$.

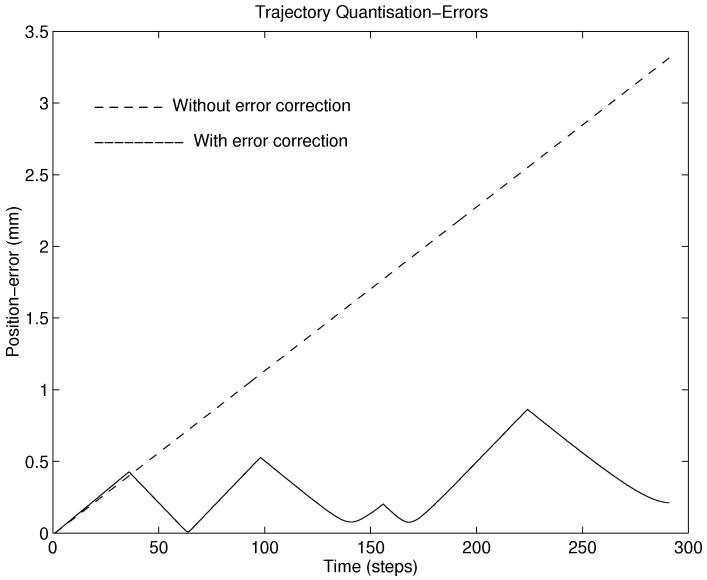

Figure 11. Comparison of position-errors time-evolution when moving along one straight-line motion. Two situation as are compared: using and not using the algorithm presented in this paper.

Figure 11 presents the evolution of position-error on one straight line motion. Two situations are compared. The dashed line represents the error when not using the presented algorithm, i.e., when moving along the slope-closest line, ρ_2, that was attained at the end of the initial pure rotation phase (see Section 4). It is a plot of the distance of the ideal intermediate point, $\mathbf{P}(k)$, to line ρ_2. Note that this distance is somewhat optimistic because it does not take into account the finite set of available wheel velocities. The solid line represents the position-error resulting from the application of the proposed algorithm. In this case, on interval k, the position-error is given by the distance from the ideal point, $\mathbf{P}(k)$, to the attained point, $\mathbf{P}'(k)$. It is clearly observed on Figure 11, that the algorithm presented in this article enabled a substantial reduction on position-error. The error improvement can also be observed on Figure 12. This figure presents the last 10 sampling intervals of the ideal robot trajectory, and the trajectories that result, when using, and not using, the proposed algorithm.

Final position-error improvement was not restricted to this particular motion, but was observed on the whole experiment composed of 100 straight-line motions. Figure 13 presents an histogram of the final position-errors, obtained when not using the presented algorithm, i.e., when moving along the slope-closest line, ρ_2 (Section 4). The final errors are calculated as the distances from the desired final points ${}^i\mathbf{P}(N_{Tl})$, to the corresponding lines ${}^i\rho_2$. Note that this is optimistic, since it ignores the restrictions on the robot wheels velocities. The 100 samples have a mean final position error of $m = 4.08$ mm and a standard deviation of

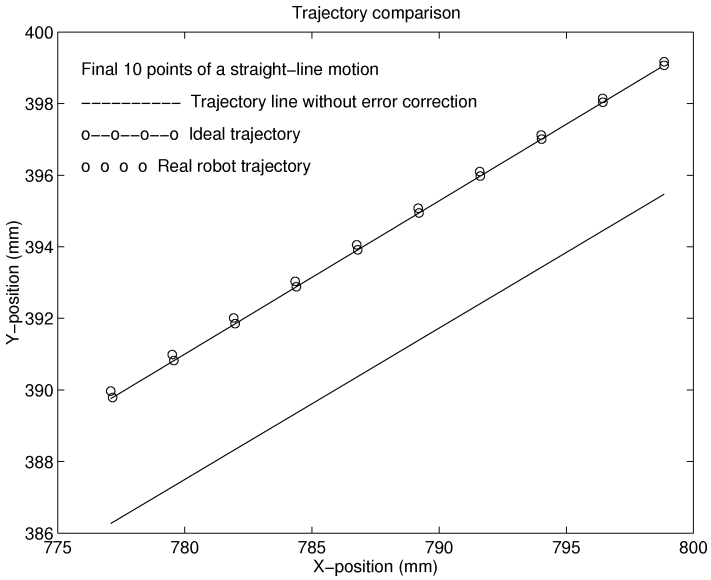

Figure 12. Robot path on the final 10 sampling intervals of motion: ideal trajectory, and trajectories using and not using the algorithm.

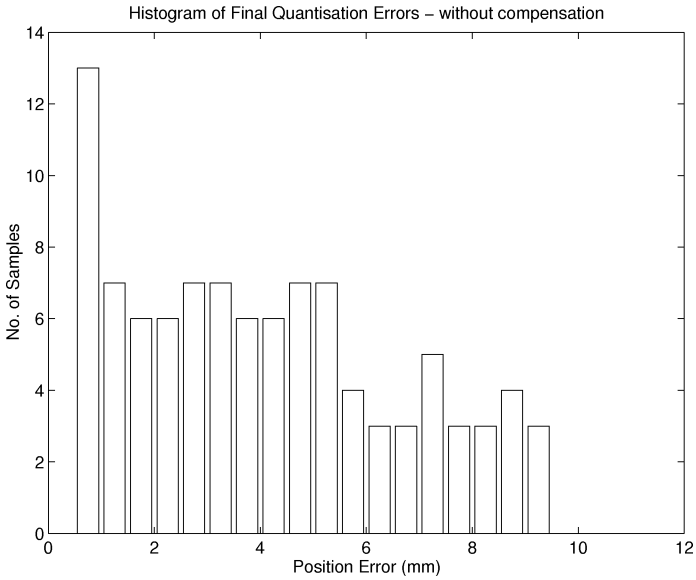

Figure 13. Distribution of final-errors on a universe of 100 different straight-line motions that did not use the proposed algorithm. The mean error was $m = 4.08$ mm with a standard deviation of $\sigma = 2.59$ mm.

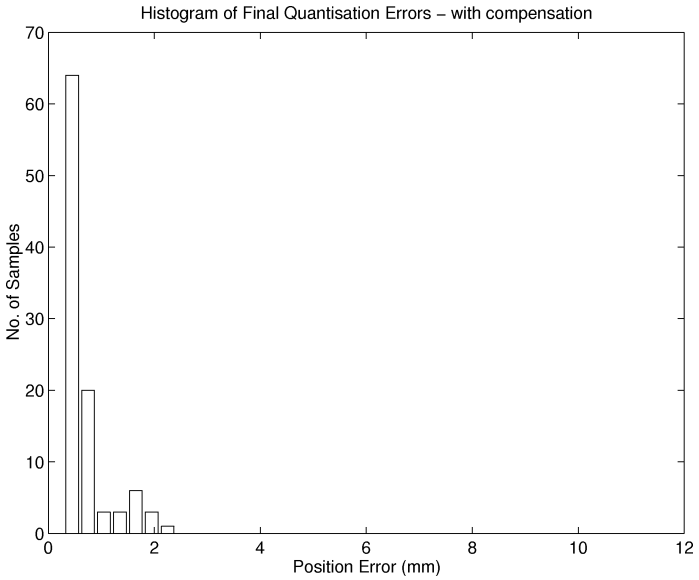

Figure 14. Distribution of final-errors on a universe of 100 different straight-line motions using the proposed algorithm. The mean error was $m = 0.59$ mm with a standard deviation of $\sigma = 0.49$ mm.

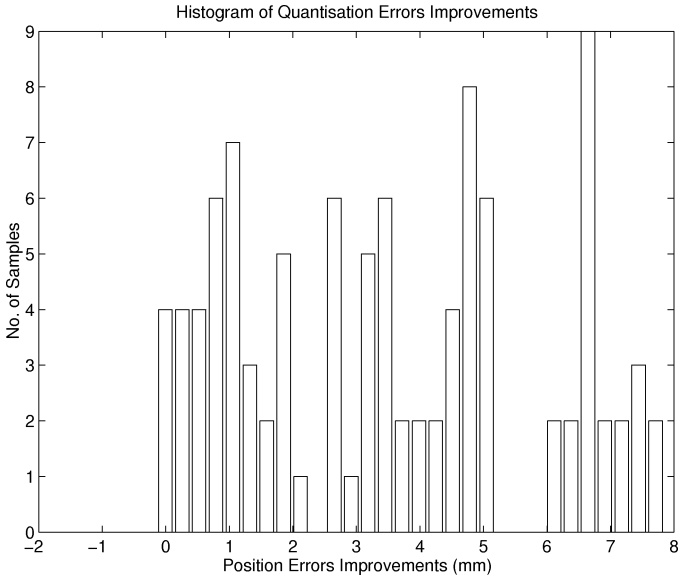

Figure 15. Distribution of final-error improvement, obtained on a universe of 100 different straight-line motions, when the proposed algorithm was introduced. The mean improvement was $m = 3.49$ mm with a standard deviation of $\sigma = 2.32$ mm.

$\sigma = 2.59$ mm. Figure 14 presents an histogram of the final position-errors, that resulted when the proposed algorithm was used. In this case the final errors are calculated as the distances from points ${}^{i}\mathbf{P}(N_{Tl})$, to the point really attained at the end of the corresponding motion, ${}^{i}\mathbf{P}'(N_{Tl})$. Those 100 samples have a mean final position error of $m = 0.59$ mm and a standard deviation of $\sigma = 0.49$ mm. By comparing Figures 13 and 14, it is clearly seen that the use of the proposed algorithm, induced a substantial transfer of error samples, to lower error values.

Figure 15 presents an histogram of error decreases that were achieved when the proposed algorithm was introduced. The distribution is clearly composed of positive decreases. This demonstrates that the error improvement was general to all the samples (it is not simultaneously composed of better and worse samples). The distribution of improvements has a mean of $m = 3.49$ mm and a standard deviation of $\sigma = 2.32$ mm.

6. Conclusion

In this paper we faced the problem of motion errors that arise when there is control-space quantisation on a mobile robot. We presented two algorithms that can be used for reducing motion errors that are due to this quantisation. The first, presented algorithm can be used to perform pure rotations (no translation) of the robot, and the second algorithm can be used to perform straight-line motions. Simulation results were presented that demonstrate the effectiveness of the approach, with the Khepera mobile robot. By adapting the ideas of the proposed methods, in order to use them in conjunction with mobile robot localisation algorithms, one of the components of the overall position error could be reduced, thus improving the overall performance.

References

1. Tonouchi, Y., Tsubouchi, T., and Arimoto, S.: Fusion of dead-reckoned positions with a workspace model for a mobile robot by Bayesian inference, in: *Proc. IEEE Int. Conf. on Intell. Robots and Systems (IROS '94)*, Munich, Germany, 1994, pp. 1347–1354.
2. Komoriya, K. and Oyama, E.: Position estimation of a mobile robot optical fiber gyroscope (OFG), in: *Proc. IEEE Int. Workshop on Intell. Robots and Systems (IROS '94)*, Munich, Germany, 1994, pp. 143–149.
3. Ebert-Uphoff, I. and Chirikjian, G. S.: Inverse kinematics of discretely actuated hyper-redundant manipulators using workspace densities, in: *Proc. IEEE Conf. on Robotics and Automation (ICRA '96)*, Minneapolis, MN, April 1996, pp. 139–145.
4. Lees, D. S. and Chirikjian, G. S.: An efficient method for computing the forward kinematics of binary manipulators, in: *Proc. IEEE Conf. on Robotics and Automation (ICRA '96)*, Minneapolis, MN, April 1996, pp. 1012–1017.
5. Lees, D. S. and Chirikjian, G. S.: A combinatorial approach to trajectory planning for binary manipulators, in: *Proc. IEEE Conf. on Robotics and Automation (ICRA '96)*, Minneapolis, MN, April 1996, pp. 2749–2754.
6. Mondada, F., Franzi, E., and Ienne, P.: Mobile robot miniaturisation: A tool for investigation in control algorithms, in: *Experimental Robotics III, Proc. 3rd Int. Symp. on Experimental Robotics*, Kyoto, Japan, October 1993, Springer, London, pp. 501–513.

7. Floreano, D. and Mondada, F.: Evolution of homing navigation in a real mobile robot, *IEEE Trans. Systems Man Cybernet., Part B: Cybernetics* **26**(3) (1996), 396–407.
8. Borenstein, J. and Feng, L.: Measurement and correction of systematic odometry errors in mobile robots, *IEEE Trans. Robotics and Automat.* **12**(6) (1996), 869–880.
9. Åström, K. J. and Wittenmark, B.: *Computer-Controlled Systems Theory and Design*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
10. Michel, O.: Khepera Simulator Version 2.0 user manual, Freeware mobile robot simulator written at the University of Nice, Sophia Antipolis (March 1996), Downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`.