DECREASING THE EFFECTS OF ACTUATION-SYSTEM QUANTISATION-ERRORS ON A KHEPERA 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;
Phone: +351 (39) 7006276/00; Fax: +351 (39) 7006247; email: rui@isr.uc.pt

ABSTRACT

In this paper we analyse the effects of quantisation errors that are present on the actuation system of a typical "two-driving-wheel" mobile robot. This analysis will then enable us to devise an algorithm, that can be used for minimizing the effects of those quantisation errors on mobile robot motion. The algorithm can be used to perform straight-line motions, between the mobile robot current position, and a predefined goal position in its working environment. The presentation of the approach will be focused on a Khepera mobile robot. The effectiveness of the presented approach will be established by the presentation of simulation results.

Keywords: Mobile robot, control quantisation, errors.

1. Introduction

The problem of quantisation errors appears in many digital control systems with effects of different severities. Due to their nature, these errors can not be completely removed from the system. However, we can try to minimise their effects, by devising some algorithms that take advantage of having knowledge about the particular quantisation and control system characteristics, of the system we are dealing with. In general, digital control systems have two basic characteristics. First, they are discrete time, i.e. control commands to the controlled system, can only be applied, or supplied, or changed, at a discrete set of time instants. These discrete time points are separated by time intervals, that frequently have constant duration. The second characteristic is that, when the control commands are supplied they are quantised, i.e. they can not be chosen from a continuous set, but can only take values from a set with a finite number of elements. The values available for each component of the command-space, are usually but not always, equally spaced real numbers. By this second characteristic, the system's controller can't apply all the commands it ideally required but, must choose from a finite set of available commands. These errors between the desired and actual commands, imply the appearance of state and output errors on the system. These errors have the further attribute of being impossible to recover because of the information lost in the quantisation process. If care is not taken, 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. In this article we present an algorithm, that can be used for minimizing the effects of control-space quantisation errors, in the specific case of a Khepera mobile robot [1]. The algorithm can be used to perform straight-line motions, between the mobile robot current position, and a predefined goal position in its working environment.

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

2. Kinematics of the Khepera

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

Each motor can take a speed ranging from $-10$ to $+10$. The unit is the (encoder pulse)/10ms 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 $r_w = 8$ 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:

$$l = \frac{2\pi}{N} r_w \tag{1}$$

which gives $l \approx 0.0837$ mm. The corresponding number of pulses, per millimetre of wheel advancement is given by, $N_{mm} = 1/l \approx 11.94$. In this paper we will use as a unit for measuring the lengths, both the millimetre (mm) and the “increment”. We define the increment as equivalent to the advancement of the robot wheel, that corresponds to a rotation of one encoder’s pulse, as given by equation 1 above. When the length of a variable is expressed in increments, we use the subscript “inc”. For example, in increments, the wheels’s radius is given by $\Delta r_{inc} = \Delta r/l$, which gives $\Delta r_{inc} \approx 626.67$ increments.

In all this paper we assume that the mobile robot is being digitally controlled with a sampling period of $T$ (expressed in seconds). Define $p_R^W$ as the position of the robot’s centre relative to the fixed world frame (fig. 2), at the beginning of a sampling interval: $p_R^W = [x_R^W y_R^W]^T$. The orientation of the robot’s front, at the beginning of a sampling interval, may be represented by a unit vector $d$ or with an angle $\theta_R$ (fig. 2):

$$d = [d_0, d_1]^T \tag{2}$$

$$\theta_R = \text{atan2}(\pm d_1, d_0); \quad d_0^2 + d_1^2 = 1 \tag{3}$$
\[ d_0 = \cos(\theta_R); \quad d_1 = \pm \sin(\theta_R) \]  

In all this paper, unless otherwise stated, 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 point 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” [2].

Let us assume that, at the beginning and at the end of a sampling interval, the values of the encoder counters are respectively given by vectors \( s_R = \begin{bmatrix} s_{0R} & s_{1R} \end{bmatrix}^T \), and \( s'_R = \begin{bmatrix} s'_{0R} & s'_{1R} \end{bmatrix}^T \). The associated encoder-variation vector,

\[ ds_R = s'_R - s_R = \begin{bmatrix} ds_{0R} & ds_{1R} \end{bmatrix}^T \]

can be easily calculated from the speeds, \( v_i \), of the two wheels on the sampling interval: \( ds_{iR} = v_i T / 0.01 \). On a sampling interval, the rotation of the wheels imply a change on the robot's centre-frame (from \( \{ R \} \) to \( \{ R' \} \) – see fig. 3). This change can be represented by a displacement vector of the robot's centre described on it's own original frame, \( p_{R1}^R \); and by a change on the robot’s frame rotation angle, \( \theta_{R1}^R = \theta \) (see fig. 3):

\[ p_{R1}^R = \begin{bmatrix} x_{R1}^R & y_{R1}^R \end{bmatrix}^T; \quad \theta_{R1}^R = \theta = \theta_{R1} - \theta_R \]

If we define \( r \) to be, the robot centre's trajectory-rotation-radius, then from fig. 3 we can also see that \( x_{R1}^R \), and \( y_{R1}^R \) can be calculated as follows:

\[ x_{R1}^R = \begin{cases} r \cdot s_{0R}, & \text{if } ds_{0R} \neq ds_{1R} \\ l \cdot ds_{0R}, & \text{if } ds_{0R} = ds_{1R} \end{cases}, \quad y_{R1}^R = \begin{cases} r \cdot (1 - c_\theta), & \text{if } ds_{0R} \neq ds_{1R} \\ 0, & \text{if } ds_{0R} = ds_{1R} \end{cases} \]

In this paper the notation \( c_\theta = \cos(\theta) \), and \( s_\theta = \sin(\theta) \) is sometimes used. At the end of a sampling interval, the position of the robot is given by the vector \( (p_{R1}^W)' = p_{R1}^W \), and the orientation is given by vector \( d' \), or the angle \( \theta_{R1} \). The objective is to calculate these variables. This can be done using (5), and the following two equations:

\[ (p_{R1}^W)' = p_{R1}^W + R_{R1}^W \cdot p_{R1}^R; \quad d' = R_{R1}^R \cdot d \]

\( R_{R1}^W \), the rotation matrix of the robot frame with respect to the world frame, and \( R_{R1}^R \), the incremental rotation matrix, may be written as follows:

\[ R_{R1}^W = \begin{bmatrix} d_0 & -d_1 \\ d_1 & d_0 \end{bmatrix}; \quad R_{R1}^R = \begin{bmatrix} c_\theta & \mp s_\theta \\ \pm s_\theta & c_\theta \end{bmatrix} \]

At this point only \( r \) and \( \theta \) remain to be calculated. By observing figure 3, the following two equations can be written:

\[ l \cdot ds_{1R} = r_1 \cdot \theta; \quad l \cdot ds_{0R} = (r_1 + \Delta r) \cdot \theta \]

By subtracting \( (ds_{0R} \neq ds_{1R}) \) and adding the equations of (8), and expressing \( r \) and \( \Delta r \) in increments, the following is obtained (note \( \theta = 0, r = \infty, \text{if } ds_{0R} = ds_{1R} \)):

\[ \theta = \frac{ds_{0R} - ds_{1R}}{\Delta r_{inc}}; \quad r_{inc} = \frac{\Delta r_{inc}}{2} \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}} \]

3. Straight-line Motion

In this section we present an algorithm for performing straight-line motion of the Khepera mobile robot. The goal of the algorithm is to minimise 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. The objective of the method is to ensure that, at the end of 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. We will assume that the
motion will take place in $N_{TI}$ sampling intervals. The duration $N_{TI}$ may be calculated from a predefined velocity of the wheels. The straight-line motion can be divided on the following two phases: (1) Pure rotation of the mobile robot, such that the front of the robot points to the final point as precisely as possible; (2) Motion to the final point using a path as close as possible to a straight-line. For the pure rotation of phase 1 we use the algorithm presented in [3]. Note that in a pure rotation, the two wheels must have symmetrical velocities. Thus, if $\theta$ is the desired rotation angle, then $d\theta_{R} = -d\theta_{L}$. In the rest of this section we will present the straight-line motion algorithm of phase 2.

Consider figure 4 where $P_{ini} = P(0)$, and $P_{f} = P(N_{TI})$ are the initial and final points of the motion respectively. In this figure, the following four straight-lines are defined. $\rho_{1}$ - line connecting the initial point, $P(0)$, to the final point, $P(N_{TI})$. $\rho_{2}$, $\rho_{3}$ - respectively the lines closest and second-closest to $\rho_{1}$, that was possible to achieve, at the end of the pure rotation motion of phase 1. This motion has an angle-resolution of $\theta = |\Delta d_{R}|/\Delta_{inc},$ with $|\Delta d_{R}| = |d_{R0} - d_{R1}| = 2 \cdot T/0.01$ pulses per sampling interval. $\rho_{3}$ - the other line closest to $\rho_{1}$ that was attainable if $|\Delta d_{R}| = |d_{R0} - d_{R1}| = 1 \cdot T/0.01$ pulses per sampling interval. The slope of this line would only be achieved from $\rho_{2}$, with a robot motion that also included a translation component. This is because, pure rotation implies $|\Delta d_{R}| = 2 \cdot |d_{R0} - d_{R1}| \cdot T/0.01$ – an even number. Note that, as is easily proved, the following two statements are valid among these four lines: (1) Lines $\rho_{2}$ and $\rho_{3}$, are those that are closest to $\rho_{1}$, and are “slope-distant” from each other $T/0.01$ pulses of difference per sampling interval. (2) Lines $\rho_{2}$ and $\rho_{4}$, are the “slope-closest” to $\rho_{1}$, that we are able to attain with a pure rotation ($2 \cdot T/0.01$ pulses of difference per sampling interval). Taking into account figure 4, let us also define $\sigma(k) = \text{sgn}(x(k) \times d(0))_{z}$, where $\text{sgn}(a) = a/|a|$ except $\text{sgn}(0) = 1$, “$\times$” denotes the vector product, and “$\times_{z}$” denotes the $Z$ component of the vector.

Let us assume that after starting at point $P_{ini} = P(0) = (x(0), y(0))$, the desired final robot location is $P_{f} = (x(N_{TI}), y(N_{TI}))$. Following an ideal straight-line trajectory would imply that, at the end of each sampling interval $k$ ($k = 1, \ldots, N_{TI}$), the robot would be in point $P(k) = (x(k), y(k))$, with coordinates

\[
x(k) = x(0) + \frac{k}{N_{TI}} \cdot [x(N_{TI}) - x(0)] = x(k - 1) + \frac{1}{N_{TI}} \cdot [x(N_{TI}) - x(0)]
\]

\[
y(k) = y(0) + \frac{k}{N_{TI}} \cdot [y(N_{TI}) - y(0)] = y(k - 1) + \frac{1}{N_{TI}} \cdot [y(N_{TI}) - y(0)]
\]

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

\[
e(k) = \|P(k) - P'(k)\|, \quad k = 1, \ldots, N_{TI}
\]

where $\|(a, b)\| = \sqrt{a^2 + b^2}$ is the Euclidean norm of vector $\|(a, b)\|$. The overall algorithm for straight-line motion in each sampling interval (see fig. 5), 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 $\theta_{R}$. This implies $d_{R0} = d_{R1}$. With the second option, besides the translation, there is a slight change of the slope angle $\theta_{R}$. The change of this angle is such that, the robot always has the slope of either line $\rho_{2}$ or line $\rho_{3}$ (fig. 4). Note that, the slopes of lines $\rho_{2}$ and $\rho_{3}$ are the ones closest to the slope of $\rho_{1}$, that can be attained. This second option implies $|\Delta d_{R}| = |d_{R0} - d_{R1}| = 1 \cdot T/0.01$ pulses per sampling interval. The algorithm chooses from those two options, the one where we can attain a point $P'(k)$ as close as possible to $P(k)$. Note that the direction of slope change must be opposite to the previous slope change that occurred. We see that, with this method, we
<table>
<thead>
<tr>
<th>OVERALL ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. IF $\sigma &gt; 0$ THEN (robot has same slope as $\rho_3$)</td>
</tr>
<tr>
<td>1.1 Choose between the following two options the one that leads to smallest error, $e(k)$.</td>
</tr>
<tr>
<td>1.1.1 Translation preserving robot angle $\theta_R$.</td>
</tr>
<tr>
<td>1.1.2 Decrease robot angle $\theta_R$ by enforcing $\Delta d s_R = d s_{0R} - d s_{1R} = -1 \cdot T/0.01$ pulse per sampling interval. Robot will have same slope as line $\rho_3$, at the end of the interval.</td>
</tr>
<tr>
<td>2. ELSE (robot has same slope as $\rho_3$)</td>
</tr>
<tr>
<td>2.1 Choose between the following two options the one that leads to smallest error, $e(k)$.</td>
</tr>
<tr>
<td>2.1.1 Translation preserving robot angle $\theta_R$.</td>
</tr>
<tr>
<td>2.1.2 Increase robot angle $\theta_R$ by enforcing $\Delta d s_R = d s_{0R} - d s_{1R} = +1 \cdot T/0.01$ pulse per sampling interval. Robot will have same slope as line $\rho_3$, at the end of the interval.</td>
</tr>
<tr>
<td>3. IF steps 1.1.2 or 2.1.2 were used THEN</td>
</tr>
<tr>
<td>3.1 Let $\sigma := -\sigma$.</td>
</tr>
</tbody>
</table>

Figure 5: Algorithm used in each sampling interval.

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. Steps “1.1.1”, “1.1.2”, “2.1.1”, and “2.1.2” are not trivial. Steps “1.1.1” and “2.1.1” will be studied in Problem 1. Steps “1.1.2” and “2.1.2” will be studied in Problem 2. Both these problems will be formulated and solved in the remaining lines of this section.

Let us start by Problem 1. The task to be solved with this problem is to use a straight-line to go from $P'(k-1)$ to $P(k)$ on sampling interval $k$. At the end of the interval we will achieve not $P(k)$, but $P'(k)$. $P'(k)$ is the point with the least error, $e(k)$, that can be achieved in straight-line with the available velocities of the robot. In this problem we are given $P(k)$, $P'(k-1)$, and $d(k-1)$; and are asked to calculate $P'(k)$, $e(k)$, and the velocities of the wheels $v_{1R}(k) = v_{1R}(k)$ in pulses/10ms). At instant $k$ vector $d$, takes the value $d(k)$, where $d$ was defined in equations (2)–(4) in section 2 (see also fig. 2). For notational convenience we will also define $\alpha(k)$ as the value of angle $\theta_R$ at instant $k$ (see equation (3) and fig. 2): $\alpha(k) = \theta_R(k)$. Note that in this Problem 1 the value of $\alpha(k)$ does not change.

Let us start by determining the equation of the straight-line that passes on $P'(k-1)$, and has a slope of $\alpha(k-1)$. We can write the following parametric equation that uses “$\lambda$” as the parameter.

$$
\begin{align*}
  x &= x'(k-1) + \lambda \cos(\alpha(k-1)) \\
  y &= y'(k-1) \pm \lambda \sin(\alpha(k-1))
\end{align*}
$$

This is equivalent to,

$$
\begin{align*}
  x \sin(\alpha(k-1)) &= x'(k-1) \sin(\alpha(k-1)) + \lambda \cos(\alpha(k-1)) \sin(\alpha(k-1)) \\
  y \cos(\alpha(k-1)) &= y'(k-1) \cos(\alpha(k-1)) \pm \lambda \sin(\alpha(k-1)) \cos(\alpha(k-1))
\end{align*}
$$

Subtracting the two equations of the previous system, we obtain the equation of the straight-line in its general form.

$$
Ax + By + C = 0
$$

(11)

$$
\begin{align*}
  A &= \sin(\alpha(k-1)) = \pm d_1(k) \\
  B &= \mp \cos(\alpha(k-1)) = \mp d_0(k) \\
  C &= \pm y'(k-1) \cos(\alpha(k-1)) - x'(k-1) \sin(\alpha(k-1))
\end{align*}
$$

(12)

Call this straight-line $\rho'(k-1)$, and observe figure 6. Regarding figure 6 note the following: $P''(k)$ is the point of line $\rho'(k-1)$ that is closest to $P(k)$. The required velocity for going from $P'(k-1)$ to $P''(k)$ is in general real-valued, and here we define it as $v''_R(k)$. $P'(k)$ and $P''(k)$ are the points of line $\rho'(k-1)$ closest to $P(k)$, that can be attained with the integer velocities available on the robot wheels. $P'(k)$ is the point of line $\rho'(k-1)$ that is closest to $P(k)$ while also able to be attained with the available integer robot velocities. As is easily proved, this velocity is given by ‘round ($v''_R(k)$)’. Therefore, in general terms, we wish to determine the point that belongs to the straight-line given by equation (11), and is at a minimum distance of a point $(x_1, y_1)$. After obtaining the closest point, it is trivial to calculate the minimum distance.
The coordinates of point $P^{(k-1)}(k)$ may be obtained using the Lagrange multipliers method for minimising the distance function subject to restriction (11). We get,

$$
\begin{align*}
    x''(k) &= \frac{B^2 z(k) - A B y(k) - C A}{A^2 + B^2} \\
    y''(k) &= \frac{A^2 y(k) - A B z(k) - C B}{A^2 + B^2}
\end{align*}
$$

(13)

where $A$, $B$, and $C$ are given by equation (12). Next we can calculate the real-valued velocity (in pulses/10ms) that is needed to attain $P^{(k)}(k)$, as follows,

$$
u''(k) = \frac{||P''(k) - P'(k-1)||}{l \cdot T/0.01}
$$

(14)

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,

$$
u'(k) = \text{round}(\nu''(k))
$$

We can now calculate the coordinates of the new point, $P'(k)$, as follows,

$$
\begin{align*}
    x'(k) &= x'(k-1) + \lambda_p \cdot \cos(\alpha(k-1)) \\
    y'(k) &= y'(k-1) + \lambda_p \cdot \sin(\alpha(k-1))
\end{align*}
$$

$$
\lambda_p = \nu'(k) \cdot l \cdot T/0.01
$$

The resulting error-distance can be calculated by equation (10).

Let us now analyse Problem 2. The objective of this problem is to move the robot from $P'(k-1)$ to $P(k)$ on sampling interval $k$, while enforcing the following conditions on the velocities of the robot’s wheels: $|\Delta s_{sR}| = |s_{sR} - s_{sL}| = 1 \cdot T/0.01$ pulses per sampling interval, and $\text{sgn}(\Delta s_{sR}) = \text{sgn}(\sigma)$. At the end of the interval we will attain not $P(k)$, but $P'(k)$. From all the points that can be reached in this conditions, $P'(k)$ is the one with the minimum associated error, $\epsilon(k)$. In this problem we are given $P(k)$, $P'(k-1)$, and $d(k-1)$; and are asked to calculate $P'(k)$, $\epsilon(k)$, and the velocities of the wheels $\nu_{sR}(k)$ and $\nu_{sL}(k)$ (in pulses/10ms).

Define $\rho'(k-1)$ as the set of all the points that can be attained by varying the wheel’s velocities $\nu_{sR}(k)$ and $\nu_{sL}(k)$, while still satisfying the conditions of the problem. However surprising it may seem to our intuition, the fact is that, we will show shortly, $\rho'(k-1)$ is a straight-line. Having this in mind, we can observe figure 7. Regarding this figure note the following: $P'(k)$ is the point of line $\rho'(k-1)$ that is closest to $P(k)$. The required velocities for going from $P'(k-1)$ to $P'(k)$ are in general real-valued, and here we call them $\nu_{sR}''(k)$ and $\nu_{sL}''(k)$. $P'(k)$ and $P''(k)$ are the points of line $\rho'(k-1)$ closest to $P(k)$, that can be attained with the integer velocities available on the robot wheels. $P''(k)$ is the point of line $\rho'(k-1)$ that is closest to $P(k)$ while also able to be attained with the available integer robot velocities. These velocities, $\nu_{sR}(k)$ and $\nu_{sL}(k)$, as is easily seen, can be computed with the following two equations,

$$
\nu_{sR}'(k) = \begin{cases} 
    \text{round}(\nu_{sR}''(k)), & \text{if } \sigma(k-1) > 0 \\
    \nu_{sR}'(k) - 1, & \text{if } \sigma(k-1) < 0
\end{cases} \quad \nu_{sL}'(k) = \begin{cases} 
    \text{round}(\nu_{sL}''(k)), & \text{if } \sigma(k-1) > 0 \\
    \nu_{sL}'(k) - 1, & \text{if } \sigma(k-1) < 0
\end{cases}
$$

(15)

1710
Let us suppose for a moment that velocities $v''_0(k)$ and $v''_1(k)$ were applied to the robot wheels. Thus the attained point would be $P''(k)$. From (9), and the hypothesis of this problem, we have that the angle of rotation of the robot frame can be calculated as follows.

$$\theta'' = \frac{\Delta s_{0R} - \Delta s_{1R}}{\Delta r_{inc}} = \frac{T \cdot \text{sgn}(\sigma)}{0.01 \cdot \Delta r_{inc}}$$

From (9), and the hypothesis of the problem, we have that the rotation radius of the robot's centre is given by:

$$r'' = \frac{\Delta r}{2} \cdot \frac{\Delta s_{0R} + \Delta s_{1R}}{\Delta s_{0R} - \Delta s_{1R}} = \text{sgn}(\sigma) \cdot \frac{\Delta r}{2} \cdot [2v''_i(k) - 1] \quad (16)$$

where we define $v''_i(k)$ as follows,

$$v''_i(k) = \begin{cases} v''_i(k), & \text{if } \sigma > 0 \\ v''_0(k), & \text{if } \sigma < 0 \end{cases} \quad (17)$$

Taking into account (2), (5), and equations (6)–(7), 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 $P'(k-1)$.

$$\begin{cases} z = x'(k-1) + d_0(k-1) \cdot r'' \cdot \sin(\theta'') - d_1(k-1) \cdot r'' \cdot [1 - \cos(\theta'')] \\ y = y'(k-1) + d_1(k-1) \cdot r'' \cdot \sin(\theta'') + d_0(k-1) \cdot r'' \cdot [1 - \cos(\theta'')] \end{cases} \quad (18)$$

This is equivalent to,

$$\begin{cases} z = x'(k-1) + r'' \cdot a_z \\ y = y'(k-1) + r'' \cdot a_y \end{cases} \quad (19)$$

where

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

From (19) we get,

$$\begin{cases} x \cdot a_y = x'(k-1) \cdot a_y + r'' \cdot a_x \cdot a_y \\ y \cdot a_x = y'(k-1) \cdot a_x + r'' \cdot a_y \cdot a_x \end{cases} \quad (20)$$

Subtracting the two equations of system (20) we obtain,

$$[x - x'(k-1)] \cdot a_y = [y - y'(k-1)] \cdot a_x \quad (21)$$

Equation (21) confirms the statement previously made, that the set of points attainable, $P'(k-1)$, constitutes a straight-line. From (21), we can equivalently write the equation of this straight-line on the form of equation (11), where,

$$\begin{cases} A = a_y \\ B = -a_x \\ C = y'(k-1) \cdot a_x - x'(k-1) \cdot a_y \end{cases} \quad (22)$$

Now, to obtain $P''(k)$, we can apply an approach similar to the one that was used for solving Problem 1. The coordinates of $P''(k)$ can thus be calculated by equation (13), with $A$, $B$, and $C$ given by (22). After that, velocity $v''_i(k)$ is calculated using equation (14). Using equations (17), and (15) we calculate $v''_0(k)$ and $v''_1(k)$, that lead the robot to point $P'(k)$.

Similarly to equations (16) and (17), we can calculate the effective rotation radius, $r'$, when we go to $P'(k)$ as follows,

$$r' = \frac{\Delta r}{2} \cdot \frac{\Delta s_{0R} + \Delta s_{1R}}{\Delta s_{0R} - \Delta s_{1R}} = \text{sgn}(\sigma) \cdot \frac{\Delta r}{2} \cdot [2v'_i(k) - 1] \quad (23)$$

where we define $v'_i(k)$ as follows,

$$v'_i(k) = \begin{cases} v'_i(k), & \text{if } \sigma(k-1) > 0 \\ v'_0(k), & \text{if } \sigma(k-1) < 0 \end{cases} \quad (24)$$

With equation (19), we now calculate the coordinates of $P'(k)$ by using $r'$ instead of $r''$:

$$\begin{cases} x'(k) = x'(k-1) + r' \cdot a_x \\ y'(k) = y'(k-1) + r' \cdot a_y \end{cases} \quad (25)$$

Finally, the resulting error-distance can be calculated by equation (10)

1711
4. Simulation Results

In this section we present simulation results demonstrating the effectiveness of the algorithm presented in section 3. The results presented here, were achieved using the "Khepera Simulator Version 2.0" [2]. The algorithm was implemented using the C Programming Language, on a separate process that communicated with the simulator. A sampling period of $T = 62.7$ ms was used. This corresponds to the default sampling period of the simulator. To test the approach, it was enforced on the simulator, that only quantisation errors are present.

![Figure 8: Example of quantisation errors on a Khepera mobile robot straight-line motion.](image)

In the example presented here, the robot's initial position was $(x(0), y(0)) = (100, 100)$, with an orientation angle of zero degrees (X-axis aligned). A straight-line motion to point $(x(N_{fit}), y(N_{fit})) = (800, 400)$ was then requested. Figure 8 plots a graph of the resulting robot X-position, versus the robot's position errors in the Y coordinate due to quantisation of control space. In the normal case, where the error-correcting algorithm of section 3 was not used, we see that the error increases linearly as the robot moves along its trajectory. This could already be expected by observing figure 4. By using the algorithm of section 3, we are able to decrease control-space quantisation errors. Also it can be seen that these errors become bounded. This clearly demonstrates the advantage of the presented algorithm.

5. Conclusion

In this paper we treated the problem of errors that arise when there is mobile robot control-space quantisation. We presented an algorithm that can be used for minimising the errors that are due to this quantisation. The algorithm can be used to perform straight-line motions, between the mobile robot current position, and a predefined final position in its working environment. Simulation results were presented that demonstrate the effectiveness of the approach.

References


JORNADAS HISPANO-LUSAS DE INGENIERÍA ELÉCTRICA

Salamanca, del 3 al 5 de julio de 1997

TOMO III

UNIVERSIDAD DE SALAMANCA
SECCIÓN DE INGENIERÍA ELÉCTRICA

UNIVERSIDADE NOVA DE LISBOA
D.T.O. DE INGENIERÍA ELECTROTÉCNICA