MINIMISING THE EFFECTS OF CONTROL-SPACE QUANTISATION ON MOBILE ROBOT MOTION

R. ARAÚJO* and A. T. DE ALMEIDA*

*Institute for Systems and Robotics (ISR), and Electrical Engineering Department, University of Coimbra; Pólo II, Póvoa de Varzim; 3030 Coimbra, Portugal; Telephone: +351 (39) 7606276/00. Fax: +351 (39) 35672. E-mail: rui@isr.uc.pt

Abstract. This article presents two algorithms, that can be used for minimising the effects of mobile robot control-space quantisation errors. The first algorithm can be used to perform pure rotations (no translation) of the mobile robot while minimising 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. Simulation results demonstrating the effectiveness of the algorithm will be presented.

Key Words. Quantisation errors, mobile robots, error control.

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, their effects can be minimised, by devising some algorithms that take advantage of having knowledge about the particular quantisation and control system characteristics, of the system under consideration.

In general, digital control systems are characterised by having a quantised set of control commands, that 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 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.

This article presents two algorithms, for minimising the effects that control-space quantisation errors have on motion inaccuracies, in the specific case of a Khepera mobile robot (Mondada et al., 1993), (Floreano and Mondada, 1996). The first algorithm can be used to perform pure rotations (no translation) of the mobile robot while minimising 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.

2. KINEMATICS OF THE KHEPERA

This work is made around the Khepera miniature mobile robot (Floreano and Mondada, 1996). 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 on 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, ∆r = 52.5 mm, and rw = 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 α1 = 2π/N. The corresponding wheel advancement is obtained by:

\[ l = \frac{2\pi}{N} r_w \]  

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

Both the millimetre (mm) and the “increment”, will be used in this paper as units for measuring lengths. The increment is defined as equivalent to the advancement of the robot wheel, that corresponds to a rotation of one encoder’s pulse, as given in equation (1). The subscript “inc” is used when the length of a variable is expressed in increments. For example, in increments, the distance between the wheels is given by:

\[ \Delta r_{inc} = \Delta r/l \]  

which gives \( \Delta r_{inc} \approx 626.67 \) increments.
can be easily calculated from the speeds, associated encoder-variation vectors, period of robot is being digitally controlled with a sampling interval: \( p_R^W = \begin{bmatrix} x_R^W & y_R^W \end{bmatrix}^T \). The orientation of the robot’s front, at the beginning of a sampling interval, may be represented by a unit vector \( \mathbf{d} \) or with an angle \( \theta_R \) (figure 2):

\[
\begin{align*}
\mathbf{d} &= [d_0 \quad d_1]^T \quad (3) \\
\theta_R &= \text{atan2}(\pm d_1, d_0) \quad (4) \\
d_0 &= \cos(\theta_R) \quad ; \quad d_1 = \pm \sin(\theta_R) \quad (5)
\end{align*}
\]

In all this paper, unless otherwise stated, whenever ‘±’ or ‘‘ 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” (Michel, 1996).

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:

\[
d_{sR} = s_R' - s_R = [ds_{0R} \quad ds_{1R}]^T
\]
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 \( \{R1\} \) – see figure 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 figure 3):

\[
\begin{align*}
p_{R1}^R &= \begin{bmatrix} x_{R1}^R & y_{R1}^R \end{bmatrix}^T \quad (6) \\
\theta_{R1}^R &= \theta - \theta_R \quad (7)
\end{align*}
\]

If \( r \) is defined to be the robot centre’s, trajectory rotation-radius, then from figure 3 it can also be seen that \( x_{R1}^R \) and \( y_{R1}^R \) can be calculated as follows:

\[
\begin{align*}
x_{R1}^R &= \begin{cases} r \cdot s_\theta, \quad \text{if}\ ds_{0R} \neq ds_{1R} \\
l \cdot ds_{0R}, \quad \text{if}\ ds_{0R} = ds_{1R}
\end{cases} \quad (8) \\
y_{R1}^R &= \begin{cases} r \cdot (1 - c_\theta), \quad \text{if}\ ds_{0R} \neq ds_{1R} \\
0, \quad \text{if}\ ds_{0R} = ds_{1R}
\end{cases} \quad (9)
\end{align*}
\]

In this paper, the notation \( c_\theta = \cos(\alpha) \), and \( s_\alpha = \sin(\alpha) \) 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 \( \mathbf{d}' \), or the angle \( \theta_{R1}^W \). The objective is to calculate these variables. This can be done using the following two equations:

\[
\begin{align*}
(p_{R1}^W)' &= p_{R1}^W + R_{R1}^W \cdot p_{R1}^R \quad (10) \\
\mathbf{d}' &= R_{R1}^W \cdot \mathbf{d} \quad (11)
\end{align*}
\]

and equation (7). \( R_{R1}^W \), the rotation matrix of the robot frame with respect to the world frame, and \( R_{R1}^{W1} \), the incremental rotation matrix, may be written as follows:

\[
\begin{align*}
R_{R1}^W &= \begin{bmatrix} d_0 & -d_1 \\
d_1 & d_0 \end{bmatrix} \quad ; \quad R_{R1} = \begin{bmatrix} c_\theta & \pm s_\theta \\
\pm s_\theta & c_\theta \end{bmatrix} \quad (12)
\end{align*}
\]

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

\[
\begin{align*}
l \cdot ds_{1R} &= r_1 \cdot \theta \\
l \cdot ds_{0R} &= (r_1 + \Delta r) \cdot \theta
\end{align*}
\]

By subtracting (if \( ds_{0R} \neq ds_{1R} \)) and adding the equations of (12), and expressing \( r \) and \( \Delta r \) in increments, the following is obtained (note \( \theta = 0 \), \( r = \infty \), if \( ds_{0R} = ds_{1R} \)):

\[
\begin{align*}
\theta &= \frac{ds_{0R} - ds_{1R}}{\Delta r_{\text{inc}}} \quad (13) \\
r_{\text{inc}} &= \frac{\Delta r_{\text{inc}}}{2} \cdot \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}} \quad (14)
\end{align*}
\]
3. ROBOT ROTATION

In this section, an algorithm for performing pure rotation of the Khepera mobile robot is devised. The objective 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. Since a pure robot rotation is desired, then the two wheels must have symmetrical velocities. If $\theta$ is the desired rotation angle, then:

$$ds_{0R} = ds_{1R}$$

Substituting in equation (13) yields:

$$\theta = \frac{2ds_{0R}}{\Delta r_{inc}}$$

Thus, the required total number of increments will be,

$$ds_{0R} = \frac{\Delta r_{inc}}{2} \theta$$

It will be assumed that the rotation will take place in $N_T$ sampling intervals. The duration $N_T$ 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 $v_0(k)$ ($k = 1, \ldots, N_T$) for wheel 1 the velocities have opposite sign. The velocities $v_0(k)$ will be calculated such that the evolution of the angle $\theta$ will be, as closely as possible, linear. For that purpose, start 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 $v_0(i)$ ($i = 1, \ldots, k$).

$$ds'_{0R}(k) = ds'_{0R}(k-1) + v_0(i) \frac{T}{0.01}; k = 1, \ldots, N_T$$

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

For approximating the linear evolution of angle $\theta$, $v_0(k)$ ($k = 1, \ldots, N_T$) will be calculated in the following way.

$$\begin{align*}
  ds_{0R}(k) &= \frac{k}{N_T} \cdot ds_{0R} \\
  v_0(k) &= \text{round}\left(\frac{ds_{0R}(k) - ds'_{0R}(k-1)}{T/0.01}\right)
\end{align*}$$

The ‘round( )’ function rounds to the nearest integer.

4. STRAIGHT-LINE MOTION

This section presents 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.

It will be assumed that the motion takes place in $N_T$ sampling intervals. The duration $N_T$ may be calculated from a predefined velocity of the wheels.

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.

Consider figure 4 where $P_{ini} = P(0)$, and $P_f = P(N_T)$ 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, $P(N_T)$. $\rho_2$, $\rho_3$ – 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 $\theta = |\Delta s_{rl}| / \Delta r_{inc}$ with $|\Delta s_{rl}| = |ds_{0R} - ds_{1R}| = 2 \cdot T/0.01$ pulses per sampling interval. $\rho_1$ – the other line closest to $\rho_1$ that was attainable if $|\Delta s_{rl}| = |ds_{0R} - ds_{1R}| = 1 \cdot T/0.01$ 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 s_{rl}| = 2 \cdot |ds_{0R}| \cdot T/0.01 – an even number. Note that, as is easily proved, among these four lines, $\rho_2$ and $\rho_3$, are those that are “slope-closest” to $\rho_1$, and are “slope-distant” from each other $T/0.01$ pulses of difference per sampling interval.

Taking into account figure 4, define $\sigma(k) = \text{sgn}(x \cdot d(k))$, where $\text{sgn}(a) = a/|a|$ except $\text{sgn}(0) = 1$, “$\cdot$” denotes the vector product, and “$\cdot$” denotes the Z component of the vector.

Following an ideal straight-line trajectory would imply that, after starting in $P_{ini}$, at the end of each sampling interval $k$ ($k = 1, \ldots, N_T$), the robot would be in point $P(k)$, where we define

$$\begin{align*}
P_{ini} &= (x(0), y(0)) = P(0) \\
P(k) &= (x(k), y(k)), \quad k = 1, \ldots, N_T \\
P_f &= (x(N_T), y(N_T))
\end{align*}$$

Fig. 4. Straight-line approximation.
and
\[ x(k) = x(0) + \frac{k}{N_{TI}} \cdot (x(N_{TI}) - x(0)) \]
\[ y(k) = y(0) + \frac{k}{N_{TI}} \cdot (y(N_{TI}) - y(0)) \]

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:
\[ e(k) = \|P(k) - P'(k)\|, \quad k = 1, \ldots, N_{TI} \quad (15) \]

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 figure 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 \( ds_{0R} = ds_{1R} \). With the second option, besides the translation, there is a slight change of the slope angle \( \theta_{R} \). This change of this angle is such that the robot always has the slope of either line \( \rho_{2} \) or line \( \rho_{3} \) (figure 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 \( |\Delta d_{sR}| = |d_{s0R} - d_{s1R}| = 1 \cdot T/0.01 \) pulses per sampling interval. The algorithm chooses from those two options, the one that leads to attaining of 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. It is seen that, with this method, it is obtained, 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 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.

Starting by Problem 1. The task to be solved with this problem is to use a straight-line to go from \( P'(k) \) to \( P(k) \) on sampling interval \( k \). At the end of the interval the attained point will be, 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 it is given \( P(k), P'(k-1), P'(k) \), and \( d(k-1) \); and it is asked to calculate \( P'(k) \), \( e(k) \), and the velocities of the wheels \( v_{x_{R}}'(k) = v_{y_{R}}'(k) \) in pulses/10ms. At instant \( k \) vector \( d \), takes the value \( d(k) \), where \( d \) was defined in equations (3)–(5) in section 2 (see also figure 2). Also, for notational convenience, it will be defined \( \alpha(k) \), as the value of angle \( \theta_{R} \) at instant \( k \) (see equation (4) and figure 2).
\[ \alpha(k) = \theta_{R}(k) \quad (16) \]

Note that in this Problem 1 the value of \( \alpha(k) \) does not change.

Starting by determining the equation of the straight-line that passes on \( P'(k-1) \), and has a slope of \( \alpha(k-1) \). The following parametric equation, in parameter “\( \lambda \)”, can be written.
\[
\begin{align*}
    x &= x'(k-1) + \lambda \cos(\alpha(k-1)) \\
    y &= y'(k-1) + \lambda \sin(\alpha(k-1))
\end{align*}
\]

Multiplying the equations of this system, respectively by \( \sin(\alpha(k-1)) \), and \( \cos(\alpha(k-1)) \), and subtracting the results, the equation of the straight-line is obtained in its general form.
\[ Ax + By + C = 0 \quad (17) \]
\[
\begin{align*}
    A &= \sin(\alpha(k-1)) = \pm d_{1}(k) \\
    B &= \mp \cos(\alpha(k-1)) = \mp d_{0}(k) \\
    C &= \pm y'(k-1) \cos(\alpha(k-1)) - \ldots \\
    &- x'(k-1) \sin(\alpha(k-1))
\end{align*}
\]

Call this straight-line \( r'(k-1) \), and observe figure 6(a). Regarding this figure note the following:
- \( P''(k) \) is the point of line \( r'(k-1) \) 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 it is defined as \( v_{R}'(k) \).
- \( P'(k) \) and \( P''(k) \) are the points of line \( r'(k-1) \) closest to \( P(k) \), that can be attained with the integer velocities available on the robot wheels.
- \( P'(k) \) is the point of line \( r'(k-1) \) 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 \( \left\lfloor v_{R}'(k) \right\rfloor \).

OVERALL ALGORITHM

1. IF \( \sigma > 0 \) THEN (robot has the slope of \( \rho_{2} \))
   1.1 Choose among the next two options, the one leading to smallest error, \( e(k) \).
   1.1.1 Translation preserving angle \( \theta_{R} \).
   1.1.2 Decrease robot angle \( \theta_{R} \) by enforcing \( \Delta d_{sR} = d_{s0R} - d_{s1R} = -1 \cdot T/0.01 \) pulse per sampling interval. Robot will have same slope as line \( \rho_{3} \), at the end of the interval.
2. ELSE (robot has same slope as \( \rho_{3} \))
   2.1 Choose among the next two options, the one leading to smallest error, \( e(k) \).
   2.1.1 Translation preserving angle \( \theta_{R} \).
   2.1.2 Increase robot angle \( \theta_{R} \) by enforcing \( \Delta d_{sR} = d_{s0R} - d_{s1R} = +1 \cdot T/0.01 \) pulse per sampling interval. Robot will have same slope as line \( \rho_{3} \), at the end of the interval.
3. IF steps 1.1.2 or 2.1.2 were used THEN
   3.1 Let \( \sigma := -\sigma \).

Fig. 5. Algorithm used in each sampling interval.
Therefore, in general terms, it is required to determine the point that belongs to the line given by equation (17), 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 \(P''(k)\), may be obtained using the Lagrange multipliers method for minimising the distance function subject to restriction (17). The following solution is obtained,

\[
\begin{align*}
x''(k) &= \frac{B^2x(k) - ABy(k) - CA}{A^2 + B^2} \\
y''(k) &= \frac{A^2y(k) - ABx(k) - CB}{A^2 + B^2}
\end{align*}
\]

(19)

where \(A\), \(B\), and \(C\) are given by equation (18).

Next the real-valued velocity (in pulses/10ms) that is needed to attain \(P''(k)\), can be calculated as follows,

\[
v''(k) = \frac{\|P''(k) - P'(k-1)\|}{l \cdot T/0.01}
\]

(20)

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,

\[v'(k) = \text{round}(v''(k))\]

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

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

\[\lambda_p = v'(k) \cdot l \cdot T/0.01\]

The resulting error-distance can be calculated by equation (15).

Let Problem 2 be now analysed. 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: \(\Delta ds_R = |ds_{0R} - ds_{1R}| = 1 \cdot T/0.01\) pulses per sampling interval, and \(\text{sgn}(\Delta ds_R) = \text{sgn}(\sigma)\). At the end of the interval it is attained 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, \(e(k)\). In this problem it is given \(P(k), P'(k-1)\), and \(d(k-1)\); and it is asked to calculate \(P'(k), e(k)\), and the velocities of the wheels \(v'_{00}(k)\) and \(v'_{11}(k)\) (in pulses/10ms).

Define \(\rho'(k-1)\) as the set of all the points that can be attained by varying the wheel’s velocities \(v'_{00}(k)\) and \(v'_{11}(k)\), while still satisfying the conditions of the problem. However surprising it may seem to intuition, the fact is that, as will shown shortly, \(\rho'(k-1)\) is a straight-line. Having this in mind, figure 6(b) can be observed. 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 they are called \(v''_{00}(k)\) and \(v''_{11}(k)\).
- \(P'(k)\) 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, \(v'_{00}(k)\) and \(v'_{11}(k)\), as is easily seen, can be computed with the following equations,

\[
v'_{11}(k) = \begin{cases} 
\text{round}(v''_{11}(k)), & \text{if } \sigma > 0 \\
v''_{00}(k) - 1, & \text{if } \sigma < 0
\end{cases}
\]

(21)

\[
v'_{00}(k) = \begin{cases} 
\text{round}(v''_{00}(k)), & \text{if } \sigma > 0 \\
v''_{11}(k) - 1, & \text{if } \sigma < 0
\end{cases}
\]

(22)

From equation (13), and the hypothesis of this problem, the angle of rotation of the robot frame can be calculated as follows.

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

From equation (14), and the hypothesis of the problem, it is clear that the rotation radius of the robot’s centre is given by:

\[
r'' = \frac{\Delta r}{2} \cdot \frac{ds_{0R} + ds_{1R}}{2} = \text{sgn}(\sigma) \cdot \frac{\Delta r}{2} \cdot [2v''_{11}(k) - 1]
\]

(23)

where \(v''_{11}(k)\) is defined as follows,

\[
v''_{11}(k) = \begin{cases} 
v''_{11}(k), & \text{if } \sigma > 0 \\
v''_{00}(k), & \text{if } \sigma < 0
\end{cases}
\]

(24)

Taking into account equations (3), (6), (8)–(11), the following equations can be written, for the coordinates of the points that, with the available robot wheels velocities, can be attained at the end of sampling interval \(k\), when starting from point \(P'(k-1)\).

\[
\begin{align*}
x &= x''(k-1) + r'' \cdot a_x \\
y &= y''(k-1) + r'' \cdot a_y
\end{align*}
\]

(25)
\[
\begin{align*}
    a_x &= d_0(k-1) \sin(\theta'') - d_1(k-1) [1 - \cos(\theta'')] \\
    a_y &= d_1(k-1) \sin(\theta'') + d_0(k-1) [1 - \cos(\theta'')]
\end{align*}
\]

From (25) it results,
\[
\begin{align*}
    x' \cdot a_y &= x'(k-1) \cdot a_y + r'' \cdot a_x \cdot a_y \\
    y' \cdot a_x &= y'(k-1) \cdot a_x + r'' \cdot a_x \cdot a_y
\end{align*}
\]

Subtracting the two equations of system (26) yields,
\[
[x - x'(k-1)] \cdot a_y = [y - y'(k-1)] \cdot a_x
\]

Equation (27) confirms the statement previously made, that the set of points attainable, \(P'(k-1)\), constitutes a straight-line. From (27), the equation of this straight-line can equivalently be written on the form of equation (17), where,
\[
\begin{align*}
    A &= a_y \\
    B &= -a_x \\
    C &= y'(k-1) \cdot a_x - x'(k-1) \cdot a_y
\end{align*}
\]

Now, to obtain \(P''(k)\), an approach similar to the one that was used for solving Problem 1 can be applied. The coordinates of \(P''(k)\) can thus be calculated by equation (19), with \(A, B,\) and \(C\) given by (28). After that, velocity \(v'_0(k)\) is calculated using equation (20). Using equations (24), and (21)–(22) we calculate \(v'_0(k)\) and \(v'_1(k)\), that lead the robot to point \(P'(k)\).

Similarly to equations (23) and (24), the effective rotation radius, \(r'\), when we go to \(P'(k)\), can be calculated as follows,
\[
r' = \frac{\Delta r}{2} \cdot \frac{ds_{0R} + ds_{1R}}{ds_{0R} - ds_{1R}} = \text{sgn}(\sigma) \cdot \frac{\Delta r}{2} \cdot [2v'_0(k) - 1]
\]

where \(v'_0(k)\) is defined as follows,
\[
v'_0(k) = \begin{cases}
    v'_1(k), & \text{if } \sigma > 0 \\
    v'_0(k), & \text{if } \sigma < 0
\end{cases}
\]

With equation (25), the coordinates of \(P''(k)\) can now be calculated, by using \(r'\) instead of \(r''\):
\[
\begin{align*}
    x'(k) &= x'(k-1) + r' \cdot a_x \\
    y'(k) &= y'(k-1) + r' \cdot a_y
\end{align*}
\]

Finally, the resulting error-distance can be calculated by equation (15).

5. SIMULATION RESULTS

In this section simulation results are presented that demonstrate the effectiveness of the algorithms introduced in sections 3 and 4. The results presented here, were achieved using the “Khepera Simulator Version 2.0” (Michel, 1996). 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.

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 \((x(N_T), y(N_T)) = (800, 400)\) was then requested. Figure 7 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 4 was not used, it is seen 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 4, we were 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.

6. CONCLUSION

This paper has presented original algorithms designed for performing pure rotations, and straight-line motions of a mobile robot, while minimising errors arising from robot control-space quantisation. Simulation results were presented demonstrating the effectiveness of the approaches.

7. REFERENCES

