Asynchronous Control of Rotation and Translation for a Robot Vehicle†

James L. Crowley
Patrick Reignier

LIFIA (IMAG) - I.N.P.G.
46 Ave. Felix Viallet
Grenoble, France

September 1988
Revised August 1992

Appeared in
Journal of Robotics and Autonomous Systems
February 1993

† This work has been funded by Transitions Research Corporation and by Project EUREKA EU 110 MITHRA.

Abstract

Robot arms require an "arm controller" to command joint motors to achieve a coordinated motion in an external Cartesian coordinate space. In the same sense, robot vehicles require a "vehicle controller" to command the motors to achieve a coordinated motion specified in terms of an external Cartesian coordinate space. This paper presents the design of a general purpose vehicle controller.

The vehicle controller is designed as a three layer structure. The top layer is an interpreter which assures a control protocol based on asynchronous commands and independent control of orientation and forward displacement. The middle layer is a control loop which maintains an estimate of the vehicle's position and orientation, as well as their uncertainties. The control loop generates estimates and commands translation and rotation in terms of a "virtual vehicle". The bottom layer is a translator between the "virtual vehicle" and whatever physical vehicle on which the controller is implemented.

Keywords

device controller, vehicle controller, position estimation, odometry, odometric error model, Kalman Filter

1 Introduction

Robot arms require an "arm controller" to command joint motors to achieve a coordinated motion in an external Cartesian coordinate space. In the same sense, robot vehicles require a "vehicle controller" to command the motors to achieve a coordinated motion specified in terms of an external Cartesian coordinate space.

This paper presents the organisation of a "standard vehicle controller" which provides asynchronous independent control of forward translation and orientation. The controller accepts both velocity and displacement commands, and can support a diverse variety of navigation techniques. In addition, the controller operates in terms of a standardized "virtual vehicle" which may be easily adapted to a large variety of vehicle geometries. All vehicle specific information is contained in the low level interface which translates the virtual vehicle into a specific vehicle geometry. Early versions of this controller have been developped for the CMU Terragator, the Denning DRV-1 and the TRC Labmate. This controller has been in every-day use in our laboratory on a RobotSoft Robuter since 1988.

This first section presents an overview of the vehicle controller as a three layer structure. This is followed by descriptions of the command interpreter and the techniques for estimating vehicle position and generating vehicle commands. A description is then given of the translator for a differentially steered vehicle. Versions of this controller have been in everyday use in a vehicle in our laboratory since 1988.

1.1 Overview: A Three Layer Architecture

Many early mobile robots [Nilsson 73], [Crowley 85], [Moravec 85], moved in a straight line "stop and go" manner. That is, the vehicle could travel in a straight line for some distance and then stop. While stopped, it could turn to a new heading. Once turning had finished, the vehicle was then free to make another straight line displacement. In the most cases, such vehicles received commands "synchronously". Once an order was received, other orders were blocked until completion of the current order. Such systems usually included an abort command for the case of detection of an obstacle. None-the-less, it was impossible to request the vehicle's estimated position during a displacement.

More recent vehicle controllers been built using radius of curvature as the control parameter [Thorpe et al. 88]. Such an approach is attractive because the calculation of the radius of curvature is quite simple for a number of common vehicle configurations. None the less, this parameter leads to unstable control solutions for many navigation tasks.

Work on road following [Wallace et. al. 85] has shown that stable trajectory control can be formulated in terms of independently controlling the first derivatives of forward displacement and orientation. Such parameters are attractive because they correspond to independent dynamic forces that operate on the vehicle during displacements. Kanayama [Kanayama 85] has shown that the path obtained by such a control parameters can be modelled in terms of the chlotoid function.

This paper describes an asynchronous vehicle controller which independently controls forward displacement and orientation. The controller is capable of responding to commands at any time using the protocol described below. New commands for displacement immediately replace previous commands. Position and orientation are modelled using an estimated value and an uncertainty, represented by a covariance. The control protocol includes a Kalman filter command to correct the estimated position and orientation and their uncertainty from external perception.

2 The System Organization

The standard vehicle controller is organised in three layers, as illustrated in figure 2.1. The top layer is a set of procedures which interpret command strings in order to set the value of control parameters or return information about the vehicle's position or velocity. Commands are directly translated into calls to a set of interface procedures. These interface procedures provide an alternative command interface for processes running on the same processor.

The inner layer is a control loop composed of two parts. One part updates an estimate of the vehicle's pose, covariance, and state vector. The second part compares the observed displacement to the control parameters and generates new orders for the motors of a "virtual vehicle".

The vehicle controller estimates and commands a state vector composed of the incremental translation and rotation (S, α) and their derivatives.

$$X = \begin{bmatrix} S \\ S' \\ \alpha \\ \alpha' \end{bmatrix}$$

The controller also maintains estimates of the absolute position and orientation (the pose) and its uncertainty, represented by a covariance. The "pose" of a vehicle is defined as the estimated values for position and orientation:

$$\hat{p} \equiv \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}$$

The uncertainty in pose is represented by a covariance matrix:

$$\hat{C}_p \equiv \begin{bmatrix} \sigma_x^2 & \sigma_{yx} & \sigma_{\theta x} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{\theta y} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{bmatrix}$$

The bottom layer, called the translator, provides a virtual vehicle interface for a particular vehicle geometry. This layer is composed of two parts. One part reads the values of the motor encoders and translates the incremental change into a change in position and orientation of the vehicle. The second part receives orders for changes in the velocity of orientation and position of the vehicle, and translates these into velocity commands for the motors.

The following sections describe each layer of the vehicle controller in detail.

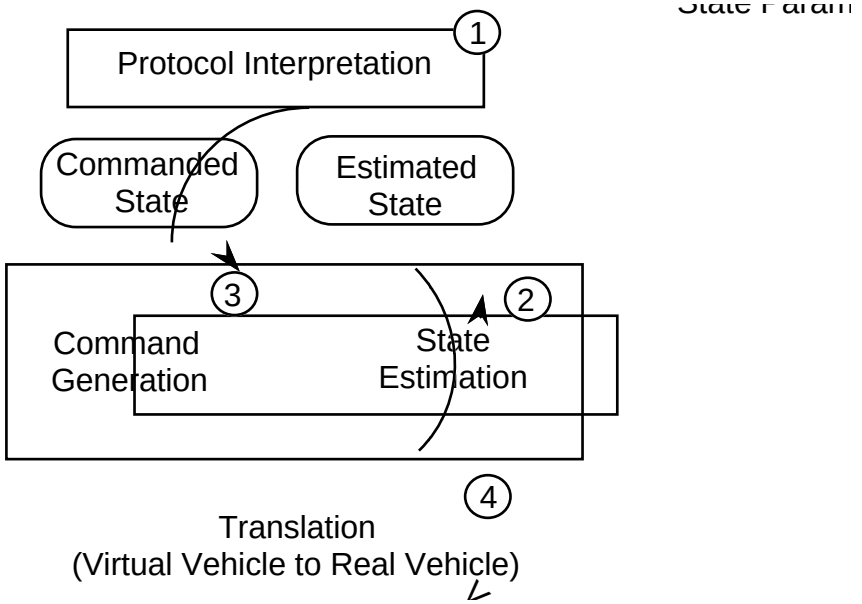

Figure 2.1 Organisation of the Standard Vehicle Controller. The bottom layer is a translator which converts a specific vehicle geometry into a "virtual vehicle". The middle layer is a control loop which estimates and commands the vehicle's translation and orientation. The top layer is a command interpreter .

3 External Interface Protocol

The vehicle controller responds to six commands. Three of these commands refer to movement of the vehicle.

m[ove] [d[, v[, a]]]	"Move" with speed v meters/sec, for a maximum distance of d meters, and an acceleration of a meters/sec^2.
t[urn] [d[, v[, a]]]	Turn with a angular speed of v degrees/sec, for a maximum rotation of d degrees, and an angular acceleration of a degrees/sec^2.
Stop	Stop all motion and hold the current position and orientation.

Brackets "[]" indicate optional parameters or letters. A command without parameters will take the last specified value as a default. A command to move or turn replaces the current reference values for the servo and thus takes effect immediately.

The vehicle control loop uses twin data structures for translation and rotation. Commands to move, turn or stop set new values for the commanded displacement, speed and acceleration within these structures. The finite state machines for move and for turn are independent, so that commands to one does not interfere with a command to the other. Both commands are cancelled by a command to stop. A command to move or stop will reset the accumulated translation and its derivatives (s, s') to zero, while a command to turn or stop will reset accumulated rotation and its derivative (α, α').

This protocol allows independent locomotion procedures for movement and steering to drive the vehicle. For example, a translation process initializes movement by specifying a move with a speed and maximum distance. Then at regular intervals, the process verifies that no obstacle is present and sends a command "m" without parameters, thereby resetting the accumulated distance. Independently, a steering process periodically measures the error between the desired and actual heading, and issues a turn command with the desired correction angle, and a velocity based on the maximum permissible curvature for the current speed.

The "pose" of the vehicle is its position and orientation. Commands which refer to the estimated position and velocity of the vehicle are:

GetEstPos	Return the estimated pose, (x, y, θ), and Covariance $\mathbf{C}_p$.

GetEstSpeed	Return the estimated speed of the vehicle v_s, v_α.
CorrectPos ΔP, $\mathbf{K}_p$	Correct the pose by ΔP using the Kalman filter gain matrix $\mathbf{K}_p$.
ResetPos P, $\mathbf{C}_p$	Set the pose and covariance to the specified values.

The commands to "get" the estimated position or velocity returns the current values for position, covariance, or velocity. The command to reset the position and covariance is useful for initialization and for testing the position correction. The command to correct the estimated position applies a Kalman filter to the position using the following equations.

$$\hat{P} := \hat{P} + \mathbf{K}\,\Delta P.$$

$$\hat{\mathbf{C}}_p := \hat{\mathbf{C}}_p - \mathbf{K}\,\hat{\mathbf{C}}_p$$

The Kalman Gain matrix is a 3 by 3 matrix which tells how much of the change in each parameter should be taken into account in correcting the other parameters [Crowley 89a]. When a sensor is available to give a complete correction in pose, P_o, with a covariance, $\mathbf{C}_o$, the correction and Kalman gain matrix may be calculated as:

$$\Delta P = P_o - \hat{P}$$
$$\mathbf{K} = \hat{\mathbf{C}}_p\,[\,\mathbf{C}_o + \hat{\mathbf{C}}_p\,]^{-1}$$

When a partial observation is available, an "extended" Kalman filter may be used to introduce the partial observation as a constraint. In this case, a model of the sensing process is required in the form of a row vector, ${}^oH_p$, which predicts the observation from the current estimate.

$$P_o^* = {}^oH_p\,\hat{P}$$

In this case the correction and gain are given by:

$$\Delta P = {}^oH_p^T\,(P_o - {}^oH_p\,\hat{P})$$

$$\mathbf{K} = \hat{\mathbf{C}}_p\,{}^oH_p\,[\,\mathbf{C}_o + {}^oH_p\,\hat{\mathbf{C}}_p\,{}^oH_p^T\,]^{-1}$$

Experiments in using an extended Kalman filter to estimate position using ultrasonic range sensors is described in [Crowley 89]. Experiments with the use of vision to measure the angle to a known vertical structure to correct position are described in [Chenavier et al 92]. A review of the use of Kalman filters for world modeling and estimation is given in [Crowley et al 92]. The process for estimating the vehicle's pose and uncertainty from odometry are described below.

4 The Inner Control Loop

The inner control loop is composed of two parts: Position estimation and generation of motor commands. The position estimation process maintains an estimate of the vehicle's current position and velocity, as well as their uncertainties. The command process generates velocity commands for forward displacement and orientation for a "virtual vehicle".

The inner control loop is actually composed of two independent copies of the same control process: one copy controls forward displacement while the second copy controls orientation. For both forward displacement and orientation displacement, the control loop is organized as a finite state machine with the four states, as illustrated in figure 4.1. These four state are:

HOLD:	Act so as to maintain the current displacement.
ACCEL:	Linearly increase the speed to a specified value.
DECCEL:	Linearly decrease the speed to a specified value.
MOVE:	Maintain a specified velocity

In the HOLD state, the process controls the value of a parameter. That is, the control process acts to maintain the accumulated forward displacement or accumulated orientation equal to zero. In states ACCEL, MOVE, and DECCEL the control process servos the first temporal derivative of a parameter.

HOLD state is particularly useful for maintaining a desired heading when starting a forward displacement, or for holding a position while turning. For example, accelerating at the beginning of a forward displacement provokes an error in orientation. This error is detected, and corrected by the command process for orientation.

The DECCEL state decreases the velocity of the vehicle so that it comes to a halt at the distance or orientation specified in the move or turn command.

The movement and orientation control loops require measured values for both instantaneous speed and for accumulated displacement. These are provided by the position estimation process.

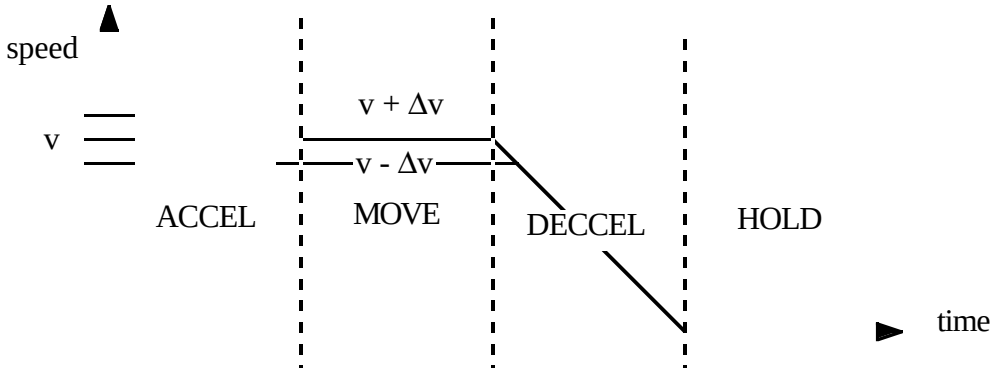

Figure 4.1 States of the command process. This process is operates independently for both the forward movement and rotation.

5 Estimating Pose and Uncertainty.

This section describes the calculation of the estimated pose and its uncertainty. It presents a general model for the error in position of a vehicle. This model is used to estimate the uncertainties in position and orientation.

5.1 Estimating Position and Orientation

The odometric position estimation process is activated at a regular interval by a real time scheduler. The process requests the translator to provide the change in the translation, ΔS, and rotation, $\Delta \alpha$, since the last call. The process also reads the system real-time clock to determine the change in time, ΔT, since the last call. The accumulated translation and rotation are calculated as:

$$S := S + \Delta S$$
$$\alpha := \alpha + \Delta\alpha$$

The derivatives of the translation and rotation are calculated as:

$$S' := \frac{\Delta S}{\Delta T}$$

$$\alpha' := \frac{\Delta\alpha}{\Delta T}$$

Accelerations are used to estimate the uncertainty in vehicle position and orientation. At each cycle, the position estimation procedure calculates the accelerations in translation and rotation as the difference in the instantaneous velocities.

$$S'' = \frac{S'_t - S'_{t-1}}{\Delta T}$$

$$\alpha'' = \frac{\alpha'_t - \alpha'_{t-1}}{\Delta T}$$

In order to update the pose, we assume that the rotational speed of the vehicle is constant within a cycle of the estimation process. Thus the average orientation during a cycle is given by the previous estimate plus half the incremental change.

$$\bar{\theta} = \hat{\theta} + \frac{\Delta\alpha}{2}$$

This gives a change in position of

$$\Delta P = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \Delta S \, \mathrm{Cos}(\hat{\theta} + \frac{\Delta\alpha}{2}) \\ \Delta S \, \mathrm{Sin}(\hat{\theta} + \frac{\Delta\alpha}{2}) \\ \Delta\alpha \end{bmatrix}$$

Vehicle pose is then updated as

$$\hat{P} := \hat{P} + \Delta P$$

5.2 Estimating the Uncertainty in Position and Orientation

Variances are used as a mathematical tool for manipulating uncertainty intervals [Crowley-Ramparny 87]. The estimated position is accompanied by an estimate of its uncertainty, represented by a covariance matrix, $\mathbf{C}_p$. The covariance is used as a mathematical tool for estimating the precision of the estimation; its use does not require or imply that the actual errors be distributed as a Normal Distribution.

Position uncertainty is updated by :

$$\hat{C}_p := \hat{C}_p + \varphi_\alpha^T \hat{C}_p \varphi_\alpha + \varphi_S^T \hat{C}_p \varphi_S + C_w$$

The matrix φ_α and φ_S represent the change in estimated position as a function in change in translation and rotation. These matrices are a discrete model of the odometric process. They tell the sensitivity of the covariance matrix to uncertainty in the terms ΔS and $\Delta\alpha$. The update matrices may be defined using a Jacobian:

$$\varphi_S = \frac{\partial \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}}{\partial \Delta S} = \begin{bmatrix} \mathrm{Cos}(\theta + \frac{\Delta\alpha}{2}) \\ \mathrm{Sin}(\theta + \frac{\Delta\alpha}{2}) \\ 0 \end{bmatrix}$$

and

$$\varphi_\alpha = \frac{\partial \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}}{\partial \Delta\alpha} = \begin{bmatrix} -\Delta S \mathrm{Sin}(\theta + \frac{\Delta\alpha}{2}) \\ \Delta S \mathrm{Cos}(\theta + \frac{\Delta\alpha}{2}) \\ 1 \end{bmatrix}$$

For pose estimation, it is generally the case that an uncertainty in orientation strongly contributes to an uncertainty in Cartesian position. The term φ_α dominates the odometric error model.

The term C_w models uncertainty due to translation and rotation and their derivatives. A major source of odometric error is an imbalance in the wheel radii, due to uneven loading or unequal tire pressure. This effect shows up as a drift in the vehicle heading as the vehicle translates. We model this effect by multiplying a term K_{drift} by the distance travelled. This term is added to the orientation.

On our mobile platform (A Robosoft Robuter) we have measured the drift to be about 1.0 $\frac{\mathrm{deg}^2}{\mathrm{m}^2}$. For vehicles with inflatable tires or unequal loading, K_{drift}, can be substantially larger.

Errors in the center of rotation can also contribute to the error in orientation. The contribution of a rotation can be modelled by the change in angle squared times a constant K_{rot}. On our vehicle, we have measured K_{rot} to be approximately 5° of error for each 360° of rotation. This translates to a standard deviation of 5/360 = 0.01, or a variance of Krot = 0.0001.

$$\mathbf{C}_W = \Delta S^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & K_{drift} \end{bmatrix} + \Delta\alpha^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & K_{rot} \end{bmatrix}$$

It is also possible to add additional terms to C_w to account for the contributions to velocity and accelerations in translation and rotation to the uncertainty of the pose. We have found this to be unnecessary for our vehicle.

6 The Virtual Vehicle

Given a control protocol which independently specifies speeds for translation and rotation, it is natural to build a control cycle which operates in terms of these parameters. It is relatively easy to translate most vehicle geometries into a "virtual vehicle" which operates in terms of rotation and translation.

In this section we describe the translator for a robot which is driven by two independently controlled power wheels. We first show how to translate the change in encoder counts into a change in position and orientation. We then show how to translate a commanded translation velocity and orientation velocity into independent wheel movements.

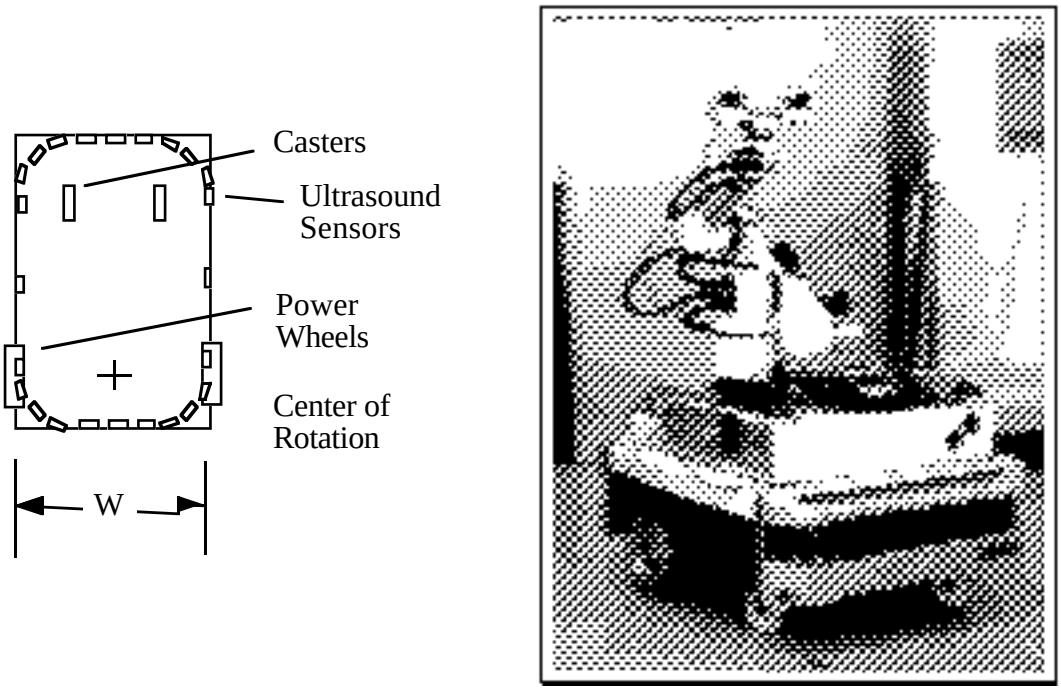

Figure 6.1 On the left is the geometry of our differentially steered vehicle. The right shows a photograph of the vehicle. (Ed: A high quality rendering of photo is available)

The physical vehicle on which the current vehicle controller has been developed is illustrated in figure 6.1. The vehicle has a rectangular form with dimensions of 80 cm wide by 120 cm long and

40 cm high. A commercial robot arm is mounted towards the rear of the vehicle. Translation and rotation are determined by the movement of a pair of independent "power" wheels located on an axis aligned with the base of the arm. The origin of the vehicle's coordinate system is located mid-way between the power wheels, directly under the arm. All vehicle position measurements are specified with respect to this reference point. Two large "casters" are mounted in the front of the vehicle to maintain balance. The vehicle is ringed with 24 ultrasonic range sensors. The robot arm acts as a neck for a binocular head.

6.1 Estimating Forward Displacement and Orientation.

If we assume a constant rotational velocity, the change in position and orientation of the vehicle is given exactly by the sum and difference of the left and right wheel displacements. This relation may be seen from figure 6.2. Consider a pair of wheels which travel an arc of radius, R, with change in angle $\Delta\alpha$. For a wheel base w, the left wheel travels an arc of radius R – w/2 with a length of ΔS_{left} while right wheel travels an arc of radius R + w/2 with a length of ΔS_{right}.

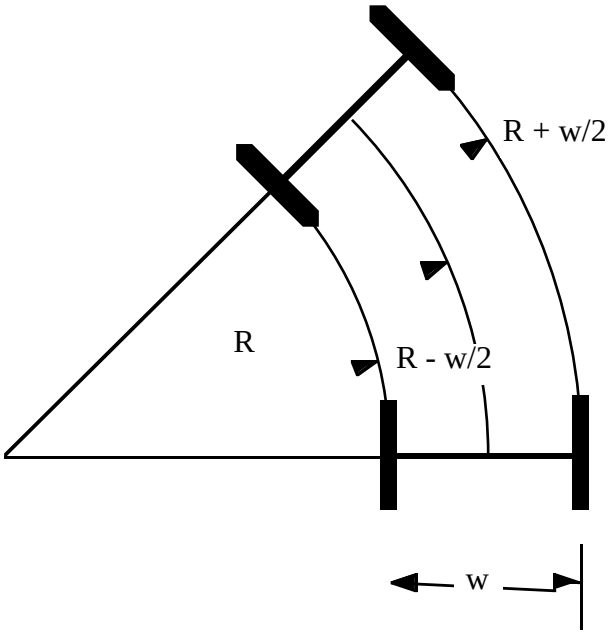

Figure 6.2 Arc of Constant Curvature for Wheels.

The length of an arc, ΔS, is related the angle, $\Delta\alpha$, and the radius, R, by the formula $\Delta S = \Delta\alpha\ R$. Thus

$$\Delta S_{left} = \Delta\alpha\ (R - \frac{w}{2})$$

$$\Delta S_{right} = \Delta\alpha\ (R + \frac{w}{2})$$

Subtracting these relations gives

$$\Delta S_{right} - \Delta S_{left} = \Delta\alpha\,(R + \frac{w}{2}) - \Delta\alpha\,(R - \frac{w}{2})$$
$$= \Delta\alpha\, w$$

Thus

$$\Delta\alpha = \frac{\Delta S_{right} - \Delta S_{left}}{w}$$

The arc length of the mid-point between the wheels is given by the sum of these relations as seen by

$$\Delta S_{right} + \Delta S_{left} = \Delta\alpha\,(R + \frac{w}{2}) - \Delta\alpha\,(R - \frac{w}{2})$$
$$= 2\,\Delta\alpha\, R$$
$$= 2\,\Delta S$$

Thus the forward displacement is given by

$$\Delta S = \frac{\Delta S_{right} + \Delta S_{left}}{2}$$

Given, ΔC_{left} and ΔC_{right} as the change in value of the right and left encoders, we can define virtual encoders for displacement and orientation as:

$$\Delta C_S = \frac{\Delta C_{right} + \Delta C_{left}}{2}$$

$$\Delta C_\alpha = \frac{\Delta C_{right} - \Delta C_{left}}{w}$$

Translation from encoder counts to meters is determined by a coefficient, K. This coefficient can be calculated from the resolution of the encoders, N (counts/revolution), the wheel radius, R (meters/revolution), and the reduction ratio of the gears r.

$$K = \frac{2\,\pi\,R\,r}{N}\ \frac{\text{meters}}{\text{count.}}$$

In practice the coefficient K is more often determined by calibration: Simply drive the vehicle forward for one meter and measure the difference in encoder counts. For a precise measure of the distance traveled, we attach a marking pen to the vehicle so that a trace of the trajectory is left on the floor. It is better to avoid indelible marking pens.

Increments in forward translation are given by

$$\Delta S = K \frac{\Delta C_{right} + \Delta C_{left}}{2}$$

and increments in orientation are given by

$$\Delta\alpha = \frac{360}{2\pi} K \frac{\Delta C_{right} - \Delta C_{left}}{W}$$

6.2 Commanding Displacement of Differential Wheels

The relation between commands for translation and orientation for the virtual vehicle, and displacement of the left and right wheels follows the geometric relationship described above. Given an order to travel with forward speed v_s in meters/sec, and rotational speed v_α, expressed in radians/sec, the motor controllers are given an order to travel at velocities v_{left} and v_{right}.

$$V_{left} = V_S - V\alpha \frac{w}{2}$$
$$V_{right} = V_S + V\alpha \frac{w}{2}$$

Although the motor controllers on our vehicle permit velocity control, the precisions of velocity commands which can be specified are too coarse to be useful. Thus, on our vehicle, the translator generates increments to position, by dividing wheel velocity by the expected controller cycle time. In position mode, the motor controllers control the acceleration and deceleration ramps of each wheel.

7 Following a Trajectory

The standard vehicle controller is in everyday use on our laboratory mobile robot and forms the basis for experiments in navigation techniques. One such navigation technique is to follow a trajectory specified in terms of a sequence of postures [Kanayama 85], where a posture is a position-orientation triple (x, y, θ) specified in external coordinates.

The set of trajectories which the vehicle can follow are the family of chlotoide curves. The chlotoides are a family of curves specified by the formula:

$$C(S) = k S + C_0$$

where s is the forward displacements and c(s) is the curvature as a function of displacement, and co is the initial curvature. Curvature is defined as the derivative of orientation with respect to forward displacement, has units of degrees/meter, and is equivalent to the inverse of the turning radius, R.

$$C = \frac{1}{R}.$$

The parameter k is the "sharpness" of a curve, which translates directly to the first derivative of curvature with respect to forward displacement, and has units of degrees/meter2.

When a vehicle turns, its orientation undergoes an acceleration, then a constant change, then a deceleration. Such a motion translates directly to a composition of move and turn commands. The three phases of the turn, ACCEL, MOVE, DECCEL, correspond directly to the states for orientation control in the vehicle controller.

For a given forward displacement speed, v_s, the parameter k of the chlotoide is determined directly from the rotational acceleration, a_α.

$$k = \frac{a_\alpha}{v_s^2}$$

A number of interesting navigational techniques can be realized using a trajectory following function. An important class of such procedure are "following" actions [Wallace et. al. 85], [Crowley 87], in which the vehicle chases a moving posture determined by a perceptual operation. This class of techniques includes following structures such as roads, walls, halls, etc, as well as chasing a moving target. Such procedures are constructed as a cycle which determines a target posture and then commands that the system replace its current target with this newest target. Heading to the target translate directly to a turn command.

Acknowledgements:

Much of the code for the first version of the vehicle controller was written by Regis Lallement and Francois Chabanel. Regis and Francois also performed the calibration experiments on which the

uncertainty estimates are based. The uncertainty model for odomoetry was developed in discussions with Joe Hummel of Denning Mobile Robots. Fredrick Chenavier has validiated and extended this odometric error model. This work has been supported by Transitions Research Corporation, Société AERO, and project EUREKA EU 110: MITHRA.

Bibliography

[Chenavier et al 92] F. Chenavier and J. L. Crowley, “Position Estimation for a Mobile Robot Using Vision and Odometry”, 1992 IEEE Conference on Robotics and Automation, Nice, May , 1992.

[Crowley 85] J. L. Crowley, "Navigation for an Intelligent Mobile Robot", IEEE Journal of Robotics and Automation, Vol 1 (1) March 1985.

[Crowley 87] J. L. Crowley, "Coordination of Action and Perception in a Surveillance Robot", Proc. of the 1987 IJCAI, Milan, Italy, Kaufmann Press, August, 1987, (also appeared in IEEE Expert, Nov. 1987).

[Crowley-Ramparany 87] J. L. Crowley and F. Ramparany, "Mathematical Tools for Representing Uncertainty in Perception", AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion, October, 1987.

[Crowley 89] J. L. Crowley, "World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging", 1989 IEEE Conference on Robotics and Automation, Scottsdale, Az, 1989.

[Crowley et. al. 92] J. L. Crowley, P. Stelmaszyk, T. Skordas and P. Puget, "Measurement and Integration of 3-D Structures By Tracking Edge Lines", International Journal of Computer Vision, July 1992.

[Kanayama 85] Y. Kanayama, "Trajectory Generation for Mobile Robots.", Third Int. Symp. on Robotics Research, ISRR-3, Paris, Oct. 1985.

[Kanayama 88] Y. Kanayama and S. Yuta, "Vehicle Path Specification by a Sequence of Straight Lines", IEEE Journal of Robotics and Automation, Vol. 4 (3) June 1988.

[Moravec 83] H. P. Moravec , "The Stanford Cart and the CMU Rover", Proc. of the IEEE, Vol 71, July 1983.

[Nilsson 69] N. J. Nilsson, "A Mobile Automaton: An Application of Artificial Intelligence Techniques", Proceedings of the First IJCAI, 1969.

[Nilsson 80] N. J. Nilsson, Principles of Artificial Intelligence, Tioga Press, 1980.

[Thorpe et. al . 87] C. Thorpe, M. Hebert, T. Kanade and S. Shafer, "Vision and Navigation for the Carnegie-Mellon University NavLab", IEEE Journal of Robotics and Automation, Vol 3 (2) March 1987.

[Wallace et. al 85] R. W. Wallace, T. Stentz, C. Thorpe and T. Kanade, "First Results in Road Following", IJCAI 85, Los Angeles, August 1985.