Sensor-based Control of Non-Holonomic Car-like Robots

Amey Parundekar
Georgia Institute of Technology
Metz, France
amey@gatech.edu

***Abstract*—This paper reviews various sensor-based control mechanisms for non-holonomic car-like robots. We cover sensor-based control approach for two different tasks. Firstly, we look at a go-to-goal task in which we maneuver a car-like robot to a given goal location. In the second task, we try to maneuver the robot to a goal location and approach a given orientation.**

***Index Terms*—Sensor-based control, Non-holonomic robots, Systems and Control**

I. Introduction

Sensor-based control for a robot involves performing certain tasks based on feedback from sensors attached to the robot. These tasks depend on what the robot is supposed to do. For example, we can have a robotic arm with a camera attached to its end or an autonomous car with several sensors attached to it. In case of a robotic arm, the tasks would be to move the arm based on several degrees of freedom (DoF) of the arm. In case of a car, the task would be to drive the car based on feedback from several sensors on the car.

In this paper we will focus on autonomously driving a car, based on sensor feedback, irrespective of what type of sensor is used. A sensor-type agnostic model will help us design the control approach with any sensor, given it is able to provide us with certain information required about the environment of the car.

The paper is arranged as follows. In Section II we will look at the kinematics of a car-like robot. In Section III we look at sensor-feedback based error-correcting control of the robot. Lastly, in Section IV we apply the knowledge from Section II & IV to two different control problems for a car-like robot.

II. Kinematics of a car-like robot

Consider the kinematics of a car as shown in figure 1. For simplicity, we will consider a two-wheel control with the assumption that the 4-wheeled car had only two wheels located at the center of the axle. The front wheel can be steered while the car is driven forward by linear velocity provided by the rear wheel. Hence such a car can be represented by 4-D generalised coordinates $q = (x, y, \theta, \phi)$. Here, x, y are the Cartesian coordinates of the vehicle whilst θ is the orientation of the principal axis of the vehicle (normal to the axle) with respect to the $x - axis$ and ϕ is the steering angle of the front wheels.

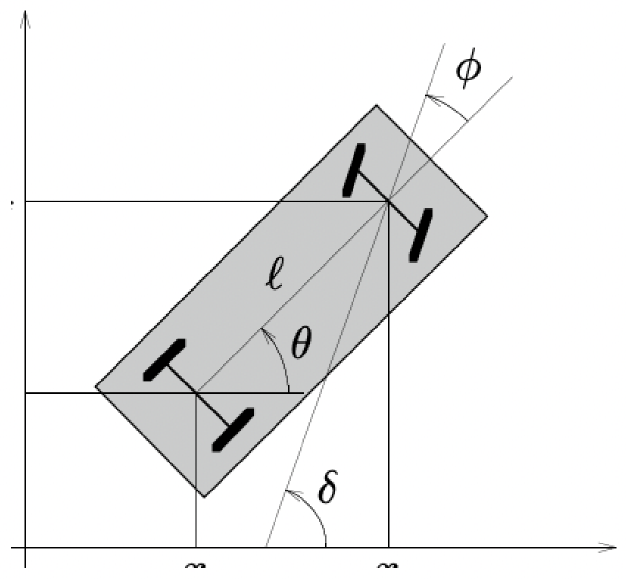


Fig. 1. Kinematics of a car-like robot [1]

A. Non-holonomic constraints

A car's system is constrained to non-holonomic constraints. This means that the number of inputs available to the car are less than the degrees of freedom available to the car. While the car can control it's linear velocity in 1 dimension and angular velocity which mean it has two control signals; it has 3 degrees of freedom.

B. Control with non-holonomic constraints

Consider the velocity of the rear wheel in the direction of the principal axis of the car. The rear wheels make an angle of θ with the $x - axis$. If v is the velocity vector of the rear wheel along the principal axis of the vehicle, we can break it into two components one along each Cartesian coordinate axis. Hence the velocity along the x-axis is $vcos(\theta)$ and the velocity along the y-axis is $vsin(\theta)$.

$$\begin{aligned}\dot{x} &= vcos(\theta)\\ \dot{y} &= vsin(\theta)\end{aligned} \tag{1}$$

Hence comparing above equations, we get

$$\dot{x}sin(\theta) - \dot{y}cos(\theta) = 0 \tag{2}$$

Similarly for the front wheel, we have

$$\dot{x_f}sin(\theta + \phi) - \dot{y_f}cos(\theta + \phi) = 0 \tag{3}$$

where, x_f, y_f are coordinates of the front wheel. We can write equation 3 using 4.

$$\begin{aligned}x_f &= x + lcos(\theta)\\ y_f &= x + lsin(\theta)\end{aligned} \tag{4}$$

This gives us,

$$\dot{x}sin(\theta + \phi) - \dot{y}cos(\theta + \phi) - \dot{\theta}lcos(\phi) = 0 \tag{5}$$

Thus the constraint matrix using equation 2 & 5 is

$$C(q) = \begin{bmatrix} sin(\theta + \phi) & -cos(\theta + \phi) & -lcos(\phi) & 0 \\ sin(\theta) & -cos(\theta) & 0 & 0 \end{bmatrix} \tag{6}$$

For rear wheel drive, the kinematics model is derived as follows where $v1$ is the linear velocity and $v2$ is steering velocity.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ tan(\phi/l) \\ 0 \end{bmatrix} v1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v2 \tag{7}$$

III. Sensor-based Servo Control

This part of the paper is derived from the works of Chaumette and Hutchinson (C&H) [2].

The goal of a sensor based control mechanism is to minimize an error $e(t)$ which is typically defined by

$$e(t) = s_{observed} - s^* \tag{8}$$

Here $s_{observed}$ is the sensor measurement and s^* is the desired value of the sensor measurement.

Once we fix a sensor, we want to represent the change in sensor value as a function of the input to our system. In case of a car, the sensor can be a laser pointer, a camera, LIDAR, ultrasonic sensor etc. Since error is measured in sensor domain, the type of reading is irrelevant in the formation of our control scheme. As seen in Section II, the car has two inputs i.e. the linear velocity and steering velocity. Hence we wish to represent the change in the sensor readings in relation to the velocity vector of the car. Doing this would help us understand how the sensor values change as the input to the car changes. Conversely based on the sensor reading error, we can use the inverse of the above relationship to figure out the inputs to be provided to the car to reduce the error.

Consider velocity vector of the car v_c as

$$v_c = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{9}$$

Let the relationship between $\dot{s}$ and v_c be given by

$$\dot{s} = L_s v_c \tag{10}$$

Differentiating equation 8 with respect to time tells us that

$$\dot{e} = \dot{s} \tag{11}$$

because s^* is constant.

hence 11 in 10 gives,

$$\dot{s} = L_s v_c \tag{12}$$

Which means,

$$v_c = L^+ \dot{e} \tag{13}$$

Here L^+ is the Moore-Penrose inverse of L and is used because L might not be invertible. The matrix L is referred to as the interaction matrix.

If we want to try to ensure an exponential decay of error, then

$$\dot{e} = -\lambda e \tag{14}$$

Hence 14 in 13 gives,

$$v_c = -\lambda L^+ e \tag{15}$$

Hence equation 15 gives us a way to calculate the velocity to be provided to the car for an exponential decay of the error in the sensor value to achieve the desired goal.

IV. Control Tasks

Using equation 15 we can start defining various control tasks for a car. The sensor features s are chosen such that they give us an idea of the pose of the car at any given time. Assuming the car movement on a plane surface, we choose s as the x, y Cartesian coordinates of the car on the movement plane. We will approach the control law in a way that we are always in the sensor frame of reference. Hence we define in the sensor reference frame, the x-axis is always aligned to the direction that the car is heading and the y-axis is normal to this direction. The origin is taken at the center of the rear axle of the car.

The inputs to the system are as follows. The car has a linear velocity v_x in the x direction but no velocity components $y\&z$ direction. Similarly the car has a angular velocity w_z along the z axis but no angular velocity along $x\&y$ axes respectively.

As the car moves, from the perspective of the sensor frame of reference, the Cartesian axis (x,y,z) move. Hence for example, in a go-to-goal location maneuver our aim is to bring the goal to the car and not take the car to the goal.

A. Go-to-goal task

In the go-to-goal task, our aim is to take the car to a goal location x, y, z. Since the car is restricted to a plane, we can ignore the z value. Consider the coordinates of the goal location in sensor frame of reference (FoV) as G(x_g, y_g). In the sensor FoV, the location of the car will always be S$(0, 0)$.

Considering the car-goal couple as a rigid body, we can write the velocity of the goal location with respect to the car velocity.

$$\begin{aligned} \dot{G} &= -v_c - \omega_s \times G \\ &= -\begin{bmatrix} v_{cx} \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ w_{sz} \end{bmatrix} \times \begin{bmatrix} x_g \\ y_g \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -1 & y_g \\ 0 & -x_g \end{bmatrix} \begin{bmatrix} v_{cx} \\ w_{cz} \end{bmatrix} \end{aligned} \tag{16}$$

Thus using equation 16, the interaction matrix is given as

$$L = \begin{bmatrix} -1 & y_g \\ 0 & -x_g \end{bmatrix} \tag{17}$$

As we want to drive the car to the goal location, thus in the sensor frame, the error between the current sensor value for the goal location is the difference between Cartesian coordinates of the goal location (x_g, y_g) and the desired sensor value is the Cartesian coordinates of where to goal needs to be and hence $(0, 0)$. Using this fact and the interaction matrix in equation 17 and substituting in equation 15 we get,

$$v_c = \begin{bmatrix} v_{cx} \\ w_{cz} \end{bmatrix} = -\lambda \frac{1}{x_g} \begin{bmatrix} -x_g & -y_g \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_g - 0 \\ y_g - 0 \end{bmatrix} \tag{18}$$

Notice here we were able to directly use the inverse of L and needn't calculate the Moore-Penrose inverse. From equation 18, we can deduce that as the car comes closer to the goal the x_g, y_g error reduces and as the car approaches the goal location, the error becomes 0 and hence ideally v_c tends to 0.

B. Go-to-goal with orientation task

In this task, we consider convergence to two goal locations. In sensor frame, the first goal location is the location which we want to bring to the origin. Let (x_g, y_g) be the current goal location. The second goal location then is situated at $(x_g cos(\theta), y_g sin(\theta)$ where $tan(\theta)$ is the slope of the line passing through the two points in the sensor frame at t = 0 (when the robot has not started moving yet).

Thus we can stack the interaction matrices for the two goal locations and get a combined interaction matrix as follows

$$L = \begin{bmatrix} -1 & y_g \\ 0 & -x_g \\ -1 & y_g sin(\theta) \\ 0 & -x_g cos(\theta) \end{bmatrix} \tag{19}$$

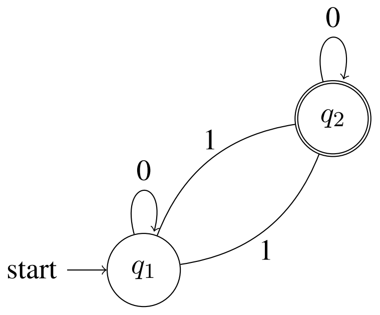
The error can be written as

$$e(t) = \begin{bmatrix} x_g - 0 \\ y_g - 0 \\ x_g cos(\theta) - dist \\ y_g sin(\theta) \end{bmatrix} \tag{20}$$

where $dist$ is the distance between the two goal locations at t = 0.

Hence using this interaction matrix and the error, we can calculate the control signals required to drive the car at a given location with orientation of θ with the x-axis. As it turns out, it is observed that for various situations the car does not converge to the required goals and seems to be stuck in a local minima. To avoid such a situation, a hybrid solution can be created where in if the car gets stuck in a local minima, it switches to a go-to-goal model (2-D error) until it is within the threshold distance of the goal and then switches back to a 4-D goal error mode.

C. Hybrid goal switching control



$$\begin{tikzcd} & q_2 \arrow[loop above, "0"] \arrow[dl, bend left, "1"] \\ \text{start} \arrow[r] & q_1 \arrow[loop above, "0"] \arrow[u, bend right, "1"'] \end{tikzcd}$$

Fig. 2. State Machine for Hybrid Control

Consider a hybrid state machine as given in figure 2 where the control follows a 4-D interaction matrix with 4-D error function as given in equation 19 & 20 when the state machine is on execution of $q1$ else it follows a 2D interaction matrix and error as given by equation 18 when it is on execution $q2$. When the state is 0 the machine loops through the process on the same node whilst when the state changes to 1 the machine switches node. The state transition works as follows:

1) On node $q1$:
    a) Initial state = 0, drive the car using 4-D L, e
    b) If car velocity approaches a low velocity threshold but the car is far away from the goal then switch to state = 1. This is because when the car velocity is very low but the goal is far away, the car must have been stuck in a local minima and hence we must switch to the standard 2-D error based go-to-goal model so that the car reaches the goal.
2) On node $q2$:
    a) Initial state = 0, drive the car using 2-D L,e
    b) If the car reaches the goal switch to state = 1, which means as the car reaches the goal we have successfully removed the car from a local minima and hence we can switch back to the 4-D L,e which tries to correct the orientation of the car as location correction is already done at this stage.

We can include an end state to this state machine which is reached when the car reaches the required goal pose within a given threshold.

Such a hybrid switching model was used in multi-sensor based control of car for parking problem by in

[3] for multi-sensor based autonomous parking of a car.

D. Converge to line

To make the car converge to a line, we need to represent the line in a way that we can represent the line motion with respect to the sensor in terms of the car twist. We make use of the Euclidean Plücker coordinates of the line to achieve this. We do this as described by Rives and Espiau [4].

1) Euclidean Plücker coordinates of a line: A line can be re-presented in its parametric form as follows:

$$X = X_0 + \xi u \tag{21}$$

Here X_0 is a point on the line and u is a vector that represents the direction of the line. Conveniently we use the vector u in it's normalised form.

We then use another vector h orthogonal to the plane of u and X_0 defined as follows:

$$h = X_0 \times u \tag{22}$$

As u is a unit vector, the magnitude of h depends on the distance of the point X_0 from the origin. As described in [5] we can hence represent a line with 3 vectors as follows:

$$L = (\underline{h}^T, \underline{u}^T, h)^T \tag{23}$$

where $\underline{h} = \frac{h}{||h||}$ and $\underline{u} = \frac{u}{||u||}$.

2) Motion of line: Using above representation of a line and as described in [4], we can represent $\dot{u}$ and $\dot{h}$ in terms v and ω the twist of the car as follows.

$$\begin{aligned} \dot{u} &= \Omega \times u \\ \dot{h} &= V \times u - h \times \Omega \end{aligned} \tag{24}$$

Assuming that a car is constrained to the ground plane (flat surface assumption) we can argue that both h and Ω are orthogonal to the ground plane and hence parallel to each other. This means the equations given in 24 term reduces as follows

$$\begin{aligned} \dot{u} &= \Omega \times u \\ \dot{h} &= V \times u \end{aligned} \tag{25}$$

Hence $\dot{u}$ captures the angular twist Ω and $\dot{h}$ captures the linear twist v of the car.

Using equations given in 25 we can produce the interaction matrix as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_z \\ h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} 0 & -y \\ 0 & x \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ y & 0 \end{bmatrix} \begin{bmatrix} v \\ \Omega \end{bmatrix} \tag{26}$$

Here,

$$X_0 = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{27}$$

As we can clearly see, we can reduce the 6-D vector given in equation 26 to a 3-D equation by removing redundant dimensions as follows:

$$Lv = \begin{bmatrix} \dot{u_x} \\ \dot{u_y} \\ \dot{h_z} \end{bmatrix} = \begin{bmatrix} 0 & -y \\ 0 & x \\ y & 0 \end{bmatrix} \begin{bmatrix} v \\ \Omega \end{bmatrix} \tag{28}$$

The error matrix can be written as

$$e = \begin{bmatrix} u - x - axis \\ h \end{bmatrix} \tag{29}$$

Hence using equation 28 and 29 we can setup a control law to converge the car's trajectory to the line.

V. Simulations

Simulations were performed to test the approach described in this paper on turtlesim in Robot Operating System (ROS). Consider in 2D Cartesian world at t = 0. Let the robot be at $S(0,0)$ with forward orientation in the positive x direction and the goal be at $G_1(2,2)$. Since at the beginning of time the world frame and sensor frame coincide; G_1 in sensor frame is also at $(2,2)$. The orientation goal in sensor frame is defined by the point $x_g cos(\theta), y_g sin(\theta))$. Let θ be -2 radians. Hence, we want to bring the goal $G_1(2,2)$ at $(0,0)$ in sensor frame while orienting the robot in a way that it roughly points in the 3rd quadrant since rotation by -2 radians would mean clockwise rotation by roughly 114.5 degrees. This motion is shown in the figure 3.

As can be seen by the velocity profiles and the state transition profile in figure 4, we switch from a 4-D error to a 2-D error as the velocity approaches to zero midway while the robot is away from the goal. As the robot switches to state 1 which represents node $q2$, the linear velocity increase again. As the robot approaches the goal position, we see another state switch back to the original 4-D error and at this point the linear velocity remains around 0 but the angular velocity rotates the robot to point towards -2 radians of rotation.

Now consider convergence to a line. With the logic described in the previous section on turtlesim we setup an experiment to converge to the line passing through points $(2,0)$ and $(0,2)$. We see the motion of turtlesim for this line in figure 5.

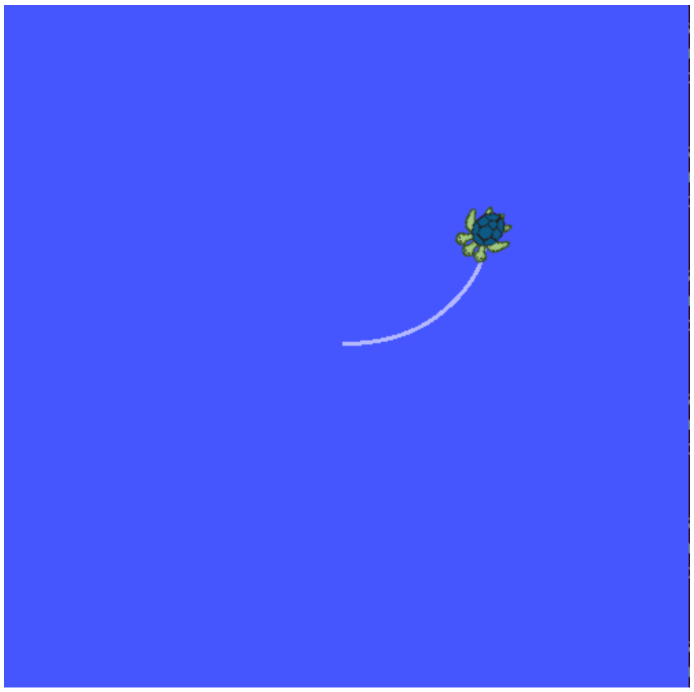


Fig. 3. Turtlesim trajectory for $G_1(2,2)$ and -2 radian orientation

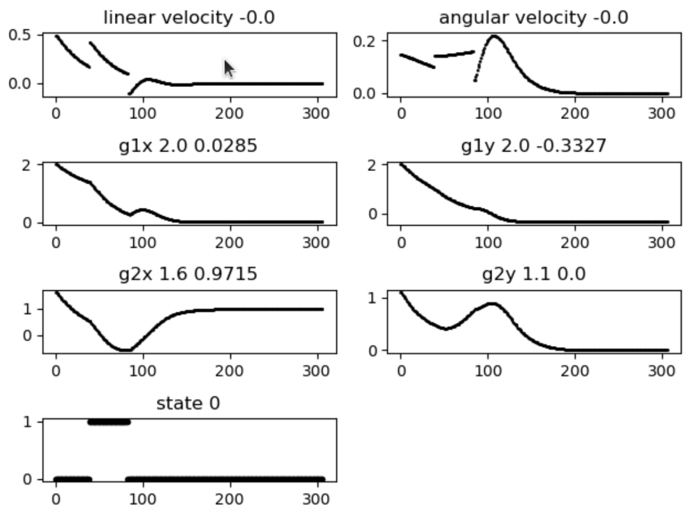


Fig. 4. Velocity and goal profiles with hybrid go-to-goal approach

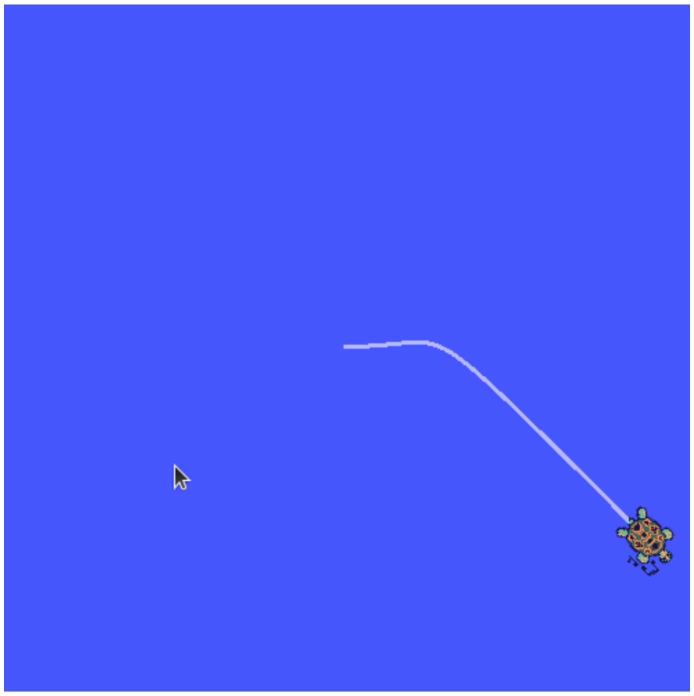


Fig. 5. Turtlesim trajectory for line passing through (2,0) and (0,2)

VI. Conclusion

In this paper we summarised various techniques used in sensor-based control of non-holonomic car-like robots.

1) Sensor-based predictive control is useful for performing autonomous tasks.
2) Interaction matrices along with exponentially decaying errors can be used to model a system to drive vehicles autonomously.
3) We saw how to do a sensor-based control for a go-to-goal task and a go-to-goal with orientation task.
4) Simple state machines can be incorporated into the control to avoid getting stuck in local minimas while approaching the goal.

References

[1] De Luca A., Oriolo G., Samson C. (1998) Feedback control of a nonholonomic car-like robot. In: Laumond J.P. (eds) Robot Motion Planning and Control. Lecture Notes in Control and Information Sciences, vol 229. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0036073

[2] F. Chaumette and S. Hutchinson, ”Visual servo control. I. Basic approaches,” in IEEE Robotics & Automation Magazine, vol. 13, no. 4, pp. 82-90, Dec. 2006, doi: 10.1109/MRA.2006.250573.

[3] David Pérez-Morales, Olivier Kermorgant, Salvador Domínguez-Quijada, Philippe Martinet. Multi Sensor-Based Predictive Control For Autonomous Parking. IEEE Transactions on Robotics, IEEE, inPress. ffhal-03286432

[4] Patrick Rives, Bernard Espiau. Estimation récursive de primitives 3D au moyen d’une caméra mobile. [Rapport de recherche] RR-0652, INRIA. 1987

[5] Nicolas Andreff, Bernard Espiau, Radu Horaud. Visual Servoing from Lines. IEEE International Conference on Robotics and Automation (ICRA ’00), Apr 2000, San Francisco, United States. pp.2070–2075