On Improving Mobile Robot Motion Control

Michail G. Lagoudakis

Intelligent Systems Laboratory*
Department of Electronic and Computer Engineering,
Technical University of Crete,
Kounoupidiana, 73100 Chania, Crete, Hellas (Greece)
lagoudakis@intelligence.tuc.gr

Abstract. This paper describes two simple techniques that can greatly improve navigation and motion control of nonholonomic robots based on range sensor data. The first technique enhances sensory information by re-using recent sensor data through coordinate transformation, whereas the second compensates for errors due to long control cycle times by forward projection through the kinematic model of the robot. Both techniques have been succesfully tested on a Nomad 200 mobile robot.

1 Introduction

The quest for artificial intelligence is best expressed in the field of robotics, where intelligent behavior and interaction are embodied and express themselves in the real world. The age of robot intelligence comparable to human intelligence may be still quite far away, however simpler, yet crucial, subproblems in this context are currently within the reach of the present technology. For example, a first step towards higher-level intelligence is the ability of a mobile robot to reason about its mobility and navigate safely in the environment.

Many mobile robots, from small laboratory robots to larger autonomous automobiles, rely on wheeled mobility and range data from sonar, laser, or infrared sensors to navigate. These robots are commonly nonholonomic [1], that is, their motion is constrained by steering. Despite significant differences at the higher level (path planning), most robot navigation algorithms at the lower level (motion control) rely on reactive controllers which take into account instantaneous range data in the robot's egocentric framework. This work indentifies two common problems in this context and proposes techniques to address them. The proposed techniques have been tested successfully on a Nomad 200 mobile robot in coordination with a neural map local path planner and a multi-objective motion controller in both simulated and real environments [2].

2 Sensor Data Transformation

Instantaneous sensor data are noisy and potentially inaccurate. To reduce instantaneous sensor uncertainty we maintain a short-term memory of the most

* Research conducted while the author was with the University of Louisiana, Lafayette.

G. Antoniou et al. (Eds.): SETN 2006, LNAI 3955, pp. 551–554, 2006.

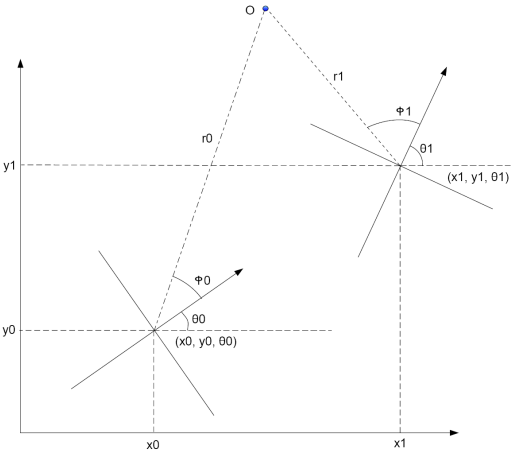

Fig. 1. Sensor range reading transformation

recent sensory information. Thus, the last few sensor readings are reused in the next few steps before they are discarded. This trick compensates for potential inaccuracies, providing at the same time a richer sense of the surrounding environment. A key parameter is the size of the memory window. A short window offers almost no advantage, whereas a large one leads to "inertia" in sensing changes. An interval of 2–3 seconds of real time is sufficient in most cases. Given an internal estimate of the cycle time through the robot's clock, the system automatically determines the required memory size.

The readings in the short term memory cannot be reused in a straightforward way; appropriate transformation is necessary. Due to robot motion, a previous reading may be reused at a configuration different than the one it was obtained from. Following Figure 1, consider a sensor range reading (r_0, ϕ_0) (with respect to the robot's local coordinate frame) corresponding to some obstacle O obtained from configuration (x_0, y_0, θ_0) (with respect to a global coordinate frame). Given a new configuration (x_1, y_1, θ_1) of the robot, what would the range reading (r_1, ϕ_1) corresponding to the same obstacle O from that configuration be? The transformation proceeds from the local coordinate frame at the first configuration to the global coordinate system and back to the local coordinate frame at the second configuration[1]:

$$r_1 = \sqrt{\big(x_0 + r_0 \times \cos(\phi_0 + \theta_0) - x_1\big)^2 + \big(y_0 + r_0 \times \sin(\phi_0 + \theta_0) - y_1\big)^2}$$

$$\phi_1 = \texttt{arctan2}\big(y_0 + r_0 \times \sin(\phi_0 + \theta_0) - y_1, x_0 + r_0 \times \cos(\phi_0 + \theta_0) - x_1\big) - \theta_1$$

To compute the transformation we use the odometry of the robot, which is quite accurate for short time frames. In summary, a cyclical buffer stores the last k range scans, obtained during the last k control cycles, along with the configuration

[1] $\texttt{arctan2}(y, x)$ calculates $\texttt{arctan}(y/x)$ and returns an angle in the correct quadrant.

Fig. 2. The hybrid nature of discrete control on continuous robot motion

of the robot at each scan. Through the above transformation, the stored readings can be made to appear as if they were all obtained from the current (or any other) configuration. Through this technique it is possible for the robot to maintain sensory information even outside its current sensory scope.

3 Configuration Prediction

Mobile robots are typically controlled by computers either on-board or off-board. Thus, control commands are issued only at discrete points in time and not continuously because of the required computation time in each cycle. This is also true for sensing, since the world is perceived only at discrete points in time. Figure 2 shows the repeated control cycle: a control command is issued (action), the world is perceived (perception), and in time Δt_1 the next action is computed. In fact, the cycle time Δt_k between two consecutive control commands can vary due to contingent events (communication delays, running processes, etc.). In any case, there is a finite time interval Δt during which there is no control over the motion of the robot. Moreover, the control action issued at a particular time step corresponds to the world as it was perceived at the previous time step. The impact of this problem becomes apparent in high speed motion, where a fast moving robot might be in a completely different configuration by the time the control command is issued.

Our proposed solution is to estimate the time Δt between control steps and predict the configuration of the robot at the beginning of the next cycle. Then, the next action is computed with respect to the predicted configuration as if the robot was there. In particular, given the current configuration (x_0, y_0, θ_0) and the current rotational and translational velocities (u_0, v_0) of the robot, we seek the configuration (x_1, y_1, θ_1) after time Δt. A nonholonomic robot moving with constant speeds follows a trajectory given by the unicycle model equations [3]:

$$x(t) = \frac{u_0}{v_0}\sin(v_0 t) \qquad y(t) = \frac{u_0}{v_0}\big(1 - \cos(v_0 t)\big) \qquad \theta(t) = v_0 t$$

Given that speeds are held almost constant during the interval Δt, we can estimate the resulting configuration as follows:

$$x_1 \approx x_0 + x(\Delta t) \qquad y_1 \approx y_0 + y(\Delta t) \qquad \theta_1 \approx \theta_0 + \theta(\Delta t)$$

To obtain an estimate of the cycle time Δt, the controller measures the elapsed real time using the robot's clock. For averaging, we use a simple convex combination of the last two measured cycle times with a higher weight on the most

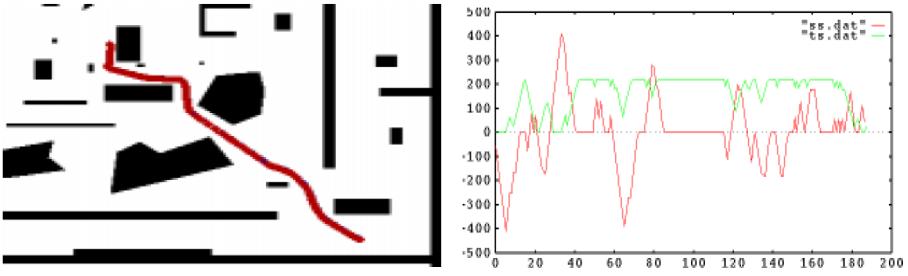

Fig. 3. A simple navigation task: robot path (left) and control commands (right)

recent one. Thus, the controller quickly adapts to the performance and load of the underlying platform, with some "inertia" to avoid contingent oscillations.

4 Improved Navigation and Consclusion

The proposed techniques have been integrated in a motion controller which issues incremental velocity commands guided by a holonomic path planner. The controller finds the best rotational and translational velocities by discrete optimization of a multi-criteria objective function defined over the *dynamic velocity window* [4] as determined by the current velocities and the maximum acceleration and deceleration ability of the robot. The objective function combines the proximity of the resulting configuration to the current goal with the potential of unforeseen, perhaps unavoidable, collisions and issues commands that eliminate or minimize collision impact. Such collisions may become possible due to dynamic obstacles, but also due to the holonomic nature of the path planner. The complete system achieves accurate, smooth, and safe navigation (an example is shown in Figure 3). The proposed techniques are quite general. Any robot navigation algorithm can project its knowledge (recent sensor data) in the near future (predicted configuration) and compute proactively and adaptively the next control command to improve navigation accuracy and performance.

References

1. Lafferriere, G., Sussmann, H.: Motion planning for controllable systems without drift. In Tarn, T., ed.: Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California, IEEE Press (1991) 1148–1153
2. Lagoudakis, M.G., Maida, A.S.: Neural maps for mobile robot navigation. In: IEEE International Joint Conference on Neural Networks (IJCNN'99). Volume III., Washington, DC, IEEE Press (1999) 2011–2016
3. Luca, A.D., Oriolo, G.: Local incremental planning for nonholonomic mobile robots. In Straub, E., Sipple, R.S., eds.: Proceedings of the 1994 International Conference on Robotics and Automation, Los Alamitos, California, IEEE Press (1994) 104–110
4. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics and Automation Magazine **RA–4** (1997) 23–33