A Robust Satisficing Feedback Strategy for Autonomous Navigation

Dai Feng and Bruce H. Krogh
Department of Electrical and Computer Engineering
and the Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents a robust algorithm for dynamic navigation of autonomous mobile robots (AMRs) using local feedback information. The structure of a previously proposed algorithm is reviewed which uses the concept of a satisficing strategy to make steering decisions while the AMR is moving. The original feedback algorithm is modified to guarantee collision avoidance in the presence of limited sensor range, sensor and servo errors, and computational delays for processing on-line sensor data to update the internal map of the local visible environment. Details of the approach are presented for an omnidirectional AMR (OAMR) and a conventionally steered AMR (CAMR). Simulation examples demonstrate the robustness of the feedback algorithm for realistic sensor and servo errors and computational delays.

1 Introduction

This paper concerns the problem of guiding an autonomous mobile robot (AMR) in an unknown environment, taking into account realistic system characteristics such as sensor noise and servo errors in the drive system. The approach is based on a previously proposed satisficing feedback algorithm¹ in which local feedback information is used to choose subgoals and make steering decisions while the robot is moving [1], [2]. This algorithm is executed by the AMR navigator in the hierarchical control architecture illustrated in figure 1.

With respect to autonomous navigation, two major deficiencies of most research on robot path planning are (1) the assumption that the environment is known, and (2) the absence of kinematic and dynamic constraints on the robot motion [3], [4], [5]. Lumelsky addressed the first deficiency by considering the navigation problem under the assumption that only local information about the environment is available from tactile or vision sensors [6], [7]. Kinematic and dynamic constraints are not taken into account by Lumelsky's algorithms. Our navigation strategy is similar in spirit to Lumelsky's approach in that we use only information about the locally visible environment. However, our method takes into account the dynamic constraints and steering capabilities of the system.

As illustrated in figure 1, the navigator uses two sources of feedback information: local maps of environment obtained from on

board sensors and estimates of the AMR state (including its position and velocity) obtained from wheel encoders and other system sensors. The local map represents a "snap shot" of the environment, taken by the on board environment sensors at discrete points in time. The sensor information is processed by the map generator to form the local map which indicates the obstacle locations in the local environment. The basic idea of our satisficing feedback algorithm is to map constraints on the future dynamic state of the AMR, due to the obstacles, into the current decision space (changes in the propulsion and steering commands) of the navigator. The navigator makes the best decision within these constraints to guide the AMR to the goal while assuring it will be in a safe state at the next decision point.

We have developed and demonstrated the satisficing feedback algorithm by simulation for two classes of AMRs: an Omnidirectional AMR (OAMR) currently operated in our laboratory [8], and a conventionally-steered AMR (CAMR) with front wheel steering and rear wheel drive [9]. In this paper we introduce modifications to the satisficing feedback algorithm to make it robust in the presence of sensing limitations and uncertainties encountered in actual mobile robots. In particular, we address three sources of uncertainty in real systems: time delays in the generation of local maps from on board sensor data; inaccuracies in position estimates due to noisy sensor data and servoing errors in the low-level control of the AMR; and lack of knowledge about the distant environment due to limited sensor range. The resulting feedback algorithm can be implemented with current sensing hardware and map-build algorithms [10], [11].

2 Satisficing Feedback Strategies

In this section, we summarize the basic structure of our solution to the local navigation problem which is described in detail in two previous papers [1], [2]. The navigation algorithm is decomposed into two parts: *subgoal selection algorithm* (SSA) which generates temporary subgoals to be pursued when the final goal is not in sight; and the *steering decision algorithm* (SDA), which chooses steering vectors to drive the AMR toward the current subgoal. The internal structure of the navigator is shown in figure 2 in which index k denotes the k^{th} execution of the SSA at time t_k and index n denotes the n^{th} execution of the SDA at time t_n . The navigator drives the AMR to the final goal p_g , supplied by the planner, using feedback information in the form of a local map $O(t_k)$ and the AMR state $x(t)$. The local map is a 2-D polygonal representation of the visible obstacle faces. In order to compensate for the AMR's finite size, an obstacle growing algorithm is used to expand the obstacles in the local map.

¹to *satisfice*: to decide on and pursue a course of action that will satisfy the minimum requirements necessary to achieve a particular goal. (*Supplement to the Oxford English Dictionary*, 1982)

At each time t_k when a new local map of the obstacles is available, the SSA attempts to update the subgoal. This new subgoal must be validated to ensure its safeness. A subgoal is *safe* if the SDA can generate a reference trajectory bringing the AMR to rest at the subgoal without colliding with any obstacles in the local map. Once a new subgoal is validated as being safe, it becomes the current subgoal, denoted by g , which is used by the SDA along with map $O(t_k)$ from which the subgoal is generated. The SDA generates a sequence of *steering vectors* $u(t_n)$, which are applied to the reference model to produce the desired reference trajectory $r(i)$. In general, the SDA is executed much more frequently than the SSA, that is, $t_{n+1} - t_n \ll t_{k+1} - t_k$. The subgoal can be updated by the SSA while the AMR is moving. Thus, the AMR does not necessarily stop at subgoals before reaching the final goal.

Assuming perfect feedback information and servoing, the basic approach of the SSA is as follows. If an obstacle lies between the AMR and the final goal, subgoals are generated at near the obstacle edges, guiding the AMR around the obstacle until it does not block the line-of-sight to the final goal. Subgoals are chosen a specified distance away from the vertex of an obstacle so as to maintain a safe distance between the AMR and obstacles and to allow a better field of vision for the AMR as it goes around the obstacle. Because of this displacement of the subgoal from the obstacle vertices, it is possible that the generated subgoals are behind other obstacles. When this occurs, additional subgoals are generated recursively and pushed on a stack until a visible subgoal is obtained. Subgoals on the stack are pursued in sequence as they become visible and safe (see reference [1] for details).

The overall structure of the SDA is illustrated in figure 3. The steering vector $u(t_n)$ is selected from a constraint set which is the intersection of the *system constraint set* $U_s(t_n)$ and the *environment constraint set* $U_e(t_n)$. The system constraint set $U_s(t_n)$ represents the kinematic and dynamic operating limits of the AMR. When the steering vector is within this set it is guaranteed the resulting reference trajectory can be followed by the servoed AMR within a known tolerance. The environmental constraint set $U_e(t_n)$ results from mapping the obstacle constraints into constraints on the steering vector using a representation of the locally visible free-space, $C(t_n)$. When the steering vector $u(t_n)$ is within this set, it is guaranteed that the state of the AMR will be safe at the next decision time, t_{n+1} . As indicated in figure 3, if the constraint set $U_s \cap U_e$ is empty, the SDA generates a *default maneuver*, a predetermined reference trajectory which brings the AMR to a stop before running into obstacles. When $U_s \cap U_e$ is nonempty, the steering vector $u(t_n)$ is chosen from $U_s \cap U_e$ to approximate a desired steering vector $d^*(t_n)$ called the *objective vector*. (Details of the SDA for the OAMR and CAMR are given in [9] and [2].)

3 Compensation for Uncertainty

There are several limitations and constraints in real systems which must be accommodated by a navigation algorithm to guarantee no collisions will occur. In this section we describe three specific sources of uncertainty against which the navigation algorithm must be robust. In each case we propose a general model of the uncertainty and enhancements to the satisficing feedback algorithm which provide robustness (i.e., collision avoidance) when the uncertainty is present.

3.1 Time Delay in Local Feedback Information

Due to the large amount of computation required to process the sensor data, it is expected that there will be a delay in obtaining the local map during the AMR navigation. Because the AMR is in motion while the sensor data are processed to generate the local map, this time delay can cause errors in the computed positions of the AMR and obstacles if it is not accounted for. In particular, let τ denote the time delay involved in processing the sensor data. Then the local map available at time t is based on the sensor data taken at time $t-\tau$. This situation is illustrated in figure 4. Assuming the AMR is in motion during the time interval τ , the AMR positions at times t and $t-\tau$, $p(t)$ and $p(t-\tau)$, are different. Therefore, if the time delay is not considered, a position error of $p(t)-p(t-\tau)$ in the obstacle locations relative to the AMR position is introduced.

A second problem introduced by the time delay is "perspective distortion" in the local map. Again we refer to figure 4 where the final goal p_g is completely blocked from the view of the AMR at $p(t-\tau)$. Because of the relative locations of the obstacles and the AMR, only a portion of the face on obstacle A is shown in the local map. However, from the position $p(t)$ the local map can be erroneously interpreted as showing the final goal being visible because the line of sight from that position to the final goal is not blocked by obstacles that are represented in the local map. Thus, the SSA may choose the final goal as a visible subgoal. Furthermore, the SDA may generate a free-space that actually overlaps the obstacle A.

The problems introduced by the processing time delay can be avoided by generating the subgoals from the perspective of $p(t-\tau)$ in the SSA. The determination of the safe state and safe subgoal as well as the generation of the free-space must still be done using the state at the current time t . It is possible that a subgoal generated from the perspective of $p(t-\tau)$ is not visible from the perspective of $p(t)$. For such cases the subgoal is not safe, and the subgoal generated from previous iterations of the SSA would not be updated. Thus, in order to assure the robustness of the algorithms in the presence of the local map delay, the following two modifications are needed in the SSA:

- Use the position information of the AMR at time $t-\tau$ to offset the position errors introduced by the time delay;
- Generate subgoals from the perspective of the position $p(t-\tau)$ where the sensor data for the local map are taken in order to avoid the erroneous interpretations of the local map.

3.2 Sensor and Servo Errors

Inaccuracies in the local map can be caused by sensor and servo errors. These errors can hinder the performance of navigation algorithms in two ways: (1) the navigator will not be able to drive the AMR to the precise goal location; and (2) obstacle avoidance cannot be guaranteed. To solve the first problem, the navigator would need to determine the true position of AMR. This would require some type of global sensing, which we do not assume to be available in our algorithms. Thus, the final goal location will be reached only within the accuracies of the sensing and servo systems.

In this section we consider the second problem, that is, how

sensor and servo errors effect our algorithm in avoiding the obstacles and what measures we can take to make the algorithm robust in the presence of these errors. First, we define sensor error as the difference between the actual location of a point on an obstacle face relative to the AMR position, and the estimated location of the point in the local map. We assume that the sensor error for a particular point is proportional to the distance from the sensor to the point and that the error is bounded. The work of Miller and Wagner on the optical range finder [12] provides good support for this assumption. We bound the maximum error for a point represented in the local map as:

$$e_{sen}(s) \leq \zeta_{sen}s$$

where s is the Euclid distance between the point and the AMR location. The sensor error constant ζ_{sen} should be determined experimentally.

We define the servo error as the difference in the commanded trajectory and the actual trajectory generated by the servo controller. Again, we consider a worst-case model in which the the servo error is bounded and proportional to the distance along the reference trajectory. This assumption is based on our observations with the Ilonator AMR [8] and the reported performance data for the AMR Blanche at AT&T Bell Laboratories [13]. The maximum servo error for a given point on the reference trajectory is modeled as:

$$e_{ser}(s) \leq \zeta_{ser}s$$

where s is the curvi-linear length of the reference trajectory and the servo error constant ζ_{ser} is determined by the capabilities of the AMR servo system.

In order to ensure the robustness of the algorithms in the presence of the sensor and servo errors, we incorporate two changes in our algorithms. First, the obstacle growing factor $\sigma(e)$, which determines the amount of expansion of the obstacles in the local map at each vertex e is computed using the following formula:

$$\sigma(e) = \rho + (\zeta_{sen} + \zeta_{ser})s$$

where s is the distance from the location of the sensor to the vertex e ; and ρ is the radius of the circle that encompass the AMR. This modification guarantees that the free-space does not intersect the obstacles whose locations are uncertain due to the sensor errors. Furthermore, the grown obstacles reduce the free-space in size so that even if the actual trajectory does not completely lie within the free-space due to the servo error, it does not enter the obstacle regions.

The second modification deals with the fact that the sensor and servo errors are random in nature. This means that the sensed location of a given obstacle vertex can shift in local maps generated at different times. Since our SSA generates subgoals that are associated with the vertices of the obstacles, this poses a problem. We note that as the AMR approaches an obstacle, the sensor error decreases. The subgoals that are generated at a far distance are based on information of lower accuracy. Thus, to take advantage of the more accurate data, we modify the SSA so that subgoals in the stack are updated each time a new local map is available. This

modification allows us to take advantage of the sensor error characteristics: the subgoal closest to the AMR is generated based on the most accurate information available at that time.

3.3 Limited Sensor Range

All sensor systems have a limit on the range within which they can provide accurate information. It is therefore necessary for us to consider the effects of the limited sensor range on our algorithm and to modify the algorithm in order to compensate for these effects.

When an obstacle that blocks the view to the final goal is out of the sensor range, it will not be represented in the local maps generated using data collected from the sensor. The unmodified SSA operates under the assumption that the sensors have infinite range. Thus, the SSA would determine that the final goal is visible and present it as the subgoal to the SDA. The SDA would then use the local map to construct the free-space which overlaps an obstacle region. In this situation, the SDA would generate a sequence of steering vectors that would accelerate the AMR toward the obstacle until it became visible, at which point there may not be sufficient breaking capability to avoid a collision.

To circumvent this problem, we make the following modifications to the SSA and the SDA. Whenever there is no obstacle in the sensor range that blocks the final goal p_g from the AMR position $p(t_k)$, we generate an additional subgoal at a distance of $(1 - \zeta_{sen} - \zeta_{ser})\rho$ from the point $p(t_k)$ on the line between p_p and p_g . This ensures that the subgoals are at least a distance of $(\zeta_{sen} + \zeta_{ser})\rho$ away from any obstacles that are out of the sensor range. The SDA will then drive the AMR at a velocity which is safe with respect to any obstacles which are currently out of the sensor range but will become visible in the future.

4 Simulation Results

In this section, we present results of simulation implementation of the SSA and SDA for both the OAMR and the CAMR. In the simulation experiments, the environment is scanned to generate a local map before each execution of the SSA. The simulated local map contains line segments representing the obstacle faces which are visible from the current location of the AMR. In the figures illustrating the simulation results, the obstacle regions are shown by shaded areas; the local map for the current AMR position is shown by the solid lines on the boundary of the obstacles; and the grown map by dotted lines.

For the OAMR, we used the double integrator model for the reference trajectory:

$$\begin{pmatrix} \ddot{p} \\ \ddot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ u \end{pmatrix},$$

where p , v and u are the two-dimensional position, velocity and acceleration, respectively, of the AMR. Based on experimental data from the Ilonator platform [8], we assume the following system constraints:

$$\begin{aligned} \|u\| &\leq a_{max} = 1.0 \text{ m/s}^2; \\ \|v\| &\leq v_{max} = 8.0 \text{ m/s}. \end{aligned}$$

For the CAMR, we use the reference model:

$$\begin{aligned}\dot{v}_r &= u_1 \\ \dot{\delta} &= u_2 \\ \dot{p}_x &= v_r (\cos\theta - \sin\theta \tan\delta) \\ \dot{p}_y &= v_r (\sin\theta + \cos\theta \tan\delta) \\ \dot{\theta} &= \frac{1}{l_f} (p_y \cos\theta - p_x \sin\theta)\end{aligned}$$

where v_r is the rear wheel speed, δ is the front wheel steering angle, $p=[p_x, p_y]^T$ is the AMR position and θ is the AMR orientation in the global coordinate frame. The following system constraints are applied in the simulation examples:

$$\begin{aligned}|\dot{v}_r| &\leq \dot{v}_{max} = 1.0 \text{ m/s}^2; \\ |v_r| &\leq v_{max} = 8.0 \text{ m/s}; \\ |\dot{\delta}| &\leq \dot{\delta}_{max} = 0.8 \text{ rad/s}; \\ |\delta| &\leq \delta_{max} = 0.785 \text{ rad}.\end{aligned}$$

Figure 5 shows a trajectory generated by the feedback algorithm for the OAMR from the initial position indicated by the cross, to the final goal indicated by the small circle. In this example, no sensor or servo error, data processing time delay, or sensor range limit is simulated. To illustrate the effect of the time delay introduced to process the sensor data and generate a local map, figure 6 is the trajectory resulting when the local map is delay by 1.0 second. The modifications proposed in section 3.1 were implemented in this simulation. Comparing figure 6 with figure 5, we see little change in the path generated by the feedback algorithm. There is a difference in the trajectories, however, in that the OAMR travels more slowly around obstacle corners when there is a time delay. This is what we would expect because the delay in the availability of the local map delays the updating of subgoals by the SSA.

We incorporated the modifications described in section 3.2 in the obstacle growing algorithm and the SSA in our simulation program. The sensor error is simulated by adding random noise of bounded magnitude to the obstacle vertex locations in the local map at each iteration of the SSA. The servo error is simulated by adding random noise of bounded magnitude to the AMR positions at each iteration of the SDA. In figure 7, the example shows trajectory generated by the SSA and SDA in the presence of sensor and servo errors. The error constants we used are $\epsilon_{ssm}=0.01$ and $\epsilon_{sps}=0.02$. Figure 7 shows trajectory generated for the CAMR using exactly the same obstacles and initial positions as in the example shown in figure 5. In this case, instead of going below obstacle 1, the trajectory goes above it. This is because in figure 5 the two edges of obstacle 1 are at equal distance from the initial position of the CAMR where the SSA selects a subgoal next to the lower edge of obstacle 1 by default. In figure 7, because of the simulated sensor error, the upper edge of obstacle 1 appears closer to the initial position. Therefore the SSA selects a subgoal which leads the trajectory toward the top of the obstacle.

In figures 8 through 13 we show the simulation results of the algorithm in the presence of sensor range limits with then proposed modification from section 3.3. In these examples, we simulate a sensor range of 30 meters and with same sensor and servo error

constants used in previous examples. In figure 8, the first segment of the OAMR trajectory is shown. Notice the position of the subgoal generated by the SSA which keeps the OAMR in a state so that it can be stopped when obstacle 1 comes into the range of the sensor. Figure 9 shows the trajectory after 45 seconds. Notice the subgoal generated is next to the lower edge on the obstacle face in the local map because it is closer to the final goal, even though the obstacle extended much further down. Because of the sensor range limit, a "bad" navigation decision like this is expected. Finally, figure 10 shows the complete trajectory.

Figure 11 shows the CAMR trajectory in the same obstacle environment. The CAMR starts from a initial orientation of -90 degrees. This results a trajectory at maximum curvature turning toward the final goal. When obstacle 1 comes into the sensor range, the local map shows the upper edge of obstacle face as being closer to the final goal because of the position at which the map is generated and the sensor range. This led the SSA to generate a subgoal next to the upper edge of the obstacle face. Figure 12 shows the trajectory after 40 seconds. Because obstacle 2 is out of sensor range at this point, the SSA generates a subgoal leading the CAMR directly toward the final goal. The complete trajectory is shown in figure 13. Again we note the slower speed of the trajectory in order to anticipate obstacles that may suddenly come into the sensor range.

5 Conclusion

In this paper, we present a feedback approach to the problem of AMR navigation in unknown environments. The proposed satisficing feedback strategies generate feasible trajectories to the goal position using only local information. We illustrate how constraints on the AMR due the obstacles and the operating limits can be accommodated for two classes of AMRs, namely, conventionally-steered and omnidirectional AMRs. Implementations of the navigation algorithms based on this approach for both types of the AMRs show the effectiveness of the approach under ideal conditions where the sensor, servo, and state estimation errors are neglected.

Currently, the second author is implementing the algorithms described in this paper on the NavLab in the Robotics Institute at Carnegie Mellon University [14]. Local maps are generated using a laser range scanner which scans at a rate of 2 cycles/sec. In preliminary experiments the satisficing feedback strategy has been applied to drive the NavLab around the corner of a building and through a narrow passage. Details of the NavLab implementation and experiments will be reported in a forthcoming paper.

Acknowledgments

This research was supported in part by the National Science Foundation under Grants ECS-8404607 and DMC-8451493, and by the Robotics Institute, Carnegie Mellon University.

References

1. B.H. Krogh and D. Feng, "Dynamics Generation of Subgoals for Autonomous Mobile Robots Using Local Feedback Information", *IEEE Trans. on Automatic Control*, May 1989.
2. D. Feng and B.H. Krogh, "Satisficing Feedback Strategies for Local Navigation of Autonomous Mobile Robots", *IEEE Transaction on Systems, Man and Cybernetics, Special Issue on Unmanned Systems and Vehicles*, 1989, submitted for review

3. T. Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements", *IEEE Trans. Systems, Man and Cybernetics*, Vol. SMC-11, No. (10), 1981, pp. 681-689.
4. T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach", *IEEE Trans. Computers*, Vol. C-32, No. 2, 1983.
5. Schwartz, Jacob and Yap, *Algorithmic and Geometric Aspects of Robotics*, Lawrence Erlbaum Associates, Hillsdale, N.J., Advance in Robotics, Vol. Vol. 1, 1987.
6. V.J. Lumelsky and A.A. Stepanov, "Path Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape", *Algorithmica*, 1987.
7. V. Lumelsky and T. Skewis, "A Paradigm for Incorporating Vision in the Robot Navigation Function", *IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April 1988, pp. 734-739.
8. D. Feng, et al, "The Servo-Control System for an Omnidirectional Mobile Robot", *IEEE International Conf on Robotics and Automation*, 1989.
9. D. Feng, *Satisficing Feedback Strategy for Local Navigation of Autonomous Mobile Robot*, PhD dissertation, Carnegie Mellon University, May 1989.
10. I. J. Cox, "Blanche: An Autonomous Robot Vehicle for Structured Environments", Tech. report 311401-2599, AT&T, September 1987.
11. A. Elfes, "A Sonar-Based Mapping and Navigation System", *IEEE Internat. Conf. Robotics and Automat.*, San Francisco, Apr. 1986, pp. 1151-1156.
12. G.L. Miller and E.R. Wagner, "An Optical Rnagefinder for Autonomous Robot", Tech Memo TM 11228-870330-02, AT&T Bell Laboratories, March 1987.
13. W.L. Nelson and I.J. Cox, "Local Path Control for an Autonomous Vehicle", Report, AT&T Bell Laboratories, 1988.
14. R. Wallace, et al, "Progress in robot road-following", *IEEE Internat. Conf. Robotics and Automat.*, San Francisco, Apr. 1986, pp. 1615-1621.

Figure 1: The hierarchical control system for an AMR.

Figure 2: The internal structure of the navigator

Figure 3: Structure of the steering decision algorithm (SDA).

Figure 4: Effects of time delay in processing sensor data for local map.

Figure 5: Trajectory of the OAMR. Total trajectory time is 97 seconds; OAMR displayed at 2 seconds intervals.

Figure 6: Trajectory of the OAMR generated with local map time delay of 1.0 second. Total trajectory time is 102 seconds; OAMR displayed at 2 seconds intervals.

Figure 7: CAMR trajectory generated in the presence of sensor and servo errors.

Figure 8: OAMR trajectory generated using local map generated with sensor of limited range. Trajectory show OAMR positions during the first 4.5 seconds.

Figure 9: OAMR trajectory generated using local map generated with sensor of limited range. Trajectory show OAMR positions during the first 45 seconds.

Figure 10: OAMR trajectory generated using local map generated with sensor of limited range. The complete trajectory after 87.5 seconds.

Figure 11: CAMR trajectory generated using local map generated with sensor of limited range. Trajectory show CAMR positions during the first 8 seconds. The CAMR positions are shown at 0.8 second intervals.

Figure 12: CAMR trajectory generated using local map generated with sensor of limited range. Trajectory show CAMR positions during the first 40 seconds. The CAMR positions are shown at 2 second intervals.

Figure 13: CAMR trajectory generated using local map generated with sensor of limited range. The complete trajectory after 97.2 seconds. The CAMR positions are shown at 2 second intervals.