In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*
pages 3467 - 3472, Washington, DC, May 11 - 15, 2002.

Exploiting physical dynamics for concurrent control of a mobile robot

Brian P. Gerkey   Maja J Matarić   Gaurav S. Sukhatme

Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, USA
{bgerkey|mataric|gaurav}@cs.usc.edu

Abstract

Conventionally, mobile robots are controlled through an action selection mechanism (ASM) that chooses among multiple proposed actions. This choice can be made in a variety of ways, and ASMs have been developed that demonstrate many of them, from strict priority schemes to voting systems. We take a different approach, which we call concurrent control. Abandoning explicit action selection, we rely instead on the physical dynamics of the robot's actuators to achieve robust control. We claim that with many noisy controllers and no arbitration among commands we can elicit stable predictable behavior from a mobile robot. Specifically, we concern ourselves with the problem of driving a single planar mobile robot with multiple controlling agents that are concurrently sending commands directly to the robot's wheel motors. We state analytically conditions necessary for our concurrent control approach to be viable, and verify empirically its general effectiveness and robustness to error through experiments with a physical robot.

1 Introduction

Some of the greatest advances in control systems of the last century were made just prior to and during World War II [2]. As is often the case, war stimulated science. A particular problem with which the U.S. army was concerned was that of aiming anti-aircraft guns at enemy airplanes. In a startling demonstration of the far-reaching applicability of control theory and practice, the most successful automated anti-aircraft director, the M-9, was developed by communications engineers at Bell Telephone Laboratories [8].

The M-9 was made possible by a peculiar invention of one of the lead scientists, D. B. Parkinson. Parkinson found that, by appropriately shaping a potentiometer and attaching it to a servo, mathematical functions could be solved electromechanically, simply because of the tendency of the servo to minimize the error. Parkinson's superior, C. A. Lovell, even noted that (quoted in [8]):

> "servomechanisms may be used directly in making transformation from one coordinate system to another..."

In this paper, we take inspiration from Parkinson's work in examining a modern problem: mobile robot control. We are interested in exploiting the physical dynamics of our system in order to achieve robust control. Specifically, we concern ourselves with the problem of driving a single planar mobile robot with multiple controlling agents. These agents can each be different in a number of ways, from the data on which they operate to the control law which they apply to that data. Avoiding altogether classical approaches to action selection [10], we opt instead to let the motors of our robot "solve" the problem for us, much as Parkinson's potentiometer did for him.

We claim that, with many noisy controllers and no command arbitration, one can elicit stable predictable behavior from a mobile robot by virtue of the simple fact that robot's motors tend to temporally average their inputs (we elaborate on necessary conditions in Section 3). Thus many agents, possibly dispersed over a network, can concurrently control a single robot without explicit communication. We support this claim through experiments with a physical robot and show that such a control system has several desirable properties, including fault-tolerance, distributivity, and scalability.

2 Related Work

Closely related to our work is recent research in the area of collaborative control. In [4], Fong et al. present a system for the collaborative teleoperation of a robot team. The robots are essentially autonomous; the human operator is treated as a "limited source of planning and information." In contrast, a wholly synthetic collaborative control system is presented in [6]. In that paper the authors consider the case of an "ensemble" of agents controlling a single simulated robot and explore the behavior of the system when failures occur and when malicious agents are introduced.

However, by using an extremely simple discrete-event simulator instead of a physical robot, they missed the key point of concurrent control, which is that the behavior of the system is the result of complex dynamics and asynchronous communication, neither of which was used in their simulation. In part, we seek in this paper to refute the counter-intuitive conclusions drawn in [6].

The general problem of action selection for robots has been studied at great length; for an excellent taxonomy and overview of recent developments in mechanisms for action selection, consult [10]. To use the terminology of that survey, our approach to low-level motor control is an instance of *superposition-based command fusion*. Another instance of superposition, which is related to our work, is the use of artificial potential fields. Potential fields have been successfully applied to the control of both manipulators [7] and mobile robots [1]. The work we present here differs from these other superposition approaches in that we perform the superposition "calculation" physically in the motors, instead of symbolically in a digital computer.

Analogous to command fusion is sensor fusion, which has also been applied to robot control (e.g., [9]). However, because of the inherent differences in timescale, units, and magnitude of data derived from different sensors, systems that fuse such information generally must perform substantial computation in order to produce a control command. We advocate the opposite approach, in which each controller may take in data from different sensors, but they all speak the common "language" of simple motor commands.

Peripherally related to our work are control schemes that employ voting among multiple participants. For example, the DAMN architecture [11], based on a weighted sum voting system, has been used to control mobile robots, both simulated and real, indoor and outdoor. For a theoretical discussion of the most popular voting systems, see [3]. Voting has also been applied to pattern recognition tasks, the *Pandemonium* architecture [12] being an early example. Pandemonium is a layered classifier system in which each layer consists of "demons" that match certain features and "shriek" their outputs up to the next layer; the final output of the system depends upon whose shriek was the loudest.

3 Method

As stated earlier, our goal is to let the motors do the work of combining multiple control signals. Our hypothesis is that, by their nature, the motors temporally average their inputs. That is, we expect each motor to "compute" the function

$$\Omega = \frac{\sum_{t=0}^{n} \omega_t}{n}$$

where the ω_t are the requested wheel speeds over time, Ω is the final wheel speed and n is small.

Focusing on the standard Pioneer robot (see Section 4) used in our experiments, we note that the robot has considerable inertia and thus takes some time to accelerate and decelerate. This time is quite large when compared to the time between successive motor requests[1]. Upon receiving a wheel speed request, the motor will servo toward the designated velocity target. However, a new request could be received before the old target is reached, causing the motor to servo toward a new target. Thus a sequence of different velocity requests over time will cause the motor to servo to a velocity that is approximately the mean of the requests.

Whereas robot control systems usually ignore or, if possible, avoid these low-level dynamics, we seek to exploit them in constructing distributed control systems. Specifically, we propose that a population of non-communicating controllers drive the robot by interleaving commands to the motors. The resultant robot motion will be the average of the commands over time. From another point of view, the robot's velocity vector will be the normalized superposition of the input velocity vectors. The next section details how we use the Player device server [5] to easily achieve the required interleaving of commands.

We can generalize from the particular problem of mobile robot control and loosely state and analyze some conditions of control system and task that are necessary for our concurrent approach to be viable:

1. The plant must have significant inertia.
2. The control signals must arrive with high frequency.
3. Signal blending must be meaningful for the task.
4. There must be a net bias in the input signals that is "correct".

Condition 1 is satisfied by many physical systems, but by few simulated systems. Condition 2 can almost always be satisfied, except when computation and/or bandwidth is lacking. Condition 3 is the most restrictive in that it constrains the space of achievable tasks to those in which the desired behavior of the system is a form of servo. This limitation is quite natural, and still permits the execution of common low-level tasks such as trajectory-following and target-following. In fact, almost any high-level task could, in principle, be broken down to servo components, however inconvenient the transformation might be. The final condition is actually a constraint on the controllers involved; since the system will average the control inputs, the controllers must produce an average value that will achieve the goal of that task. Thus, given the the classic example of half of the controllers requesting to turn left and half requesting to turn right, our method would behave similarly to potential field approaches: the robot would move directly forward.

[1]The frequency of the microcontroller that drives the motors is 10Hz, meaning that new wheel speeds can be requested every 100ms.

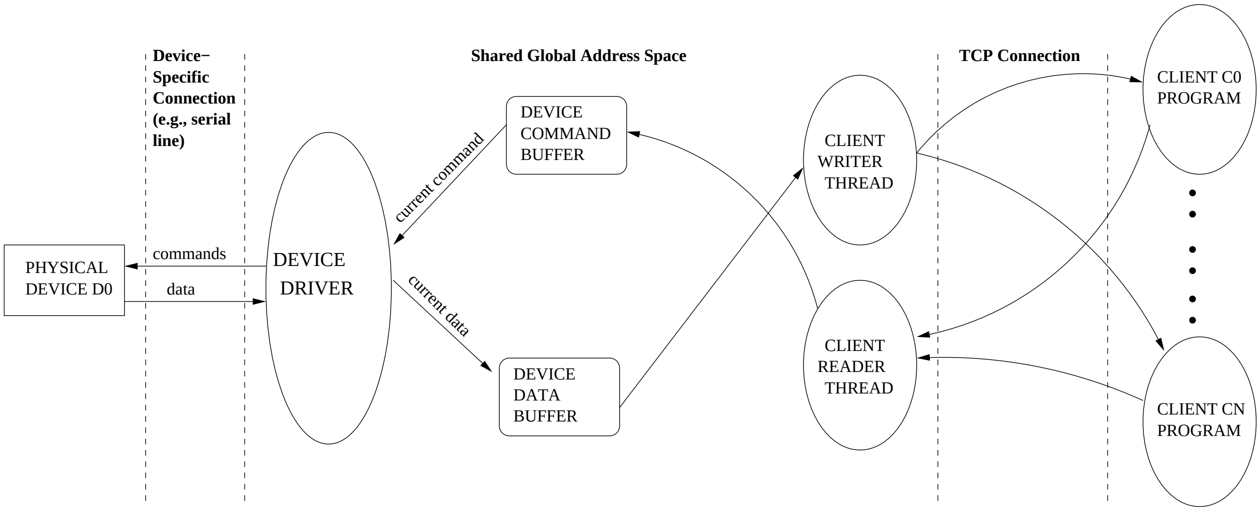

Figure 1: *Partial system architecture of Player*

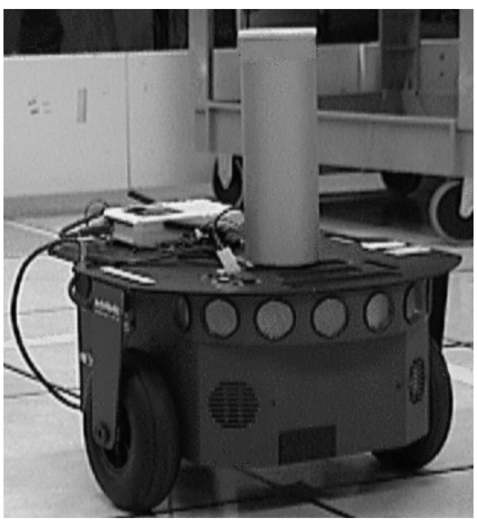

Figure 2: *The Pioneer 2-DX used in these experiments. The cylinder mounted on the top of the robot is a laser beacon, used to gather ground-truth trajectory information for performance evaluation.*

4 Hardware / Software Details

We used as our experimental test-bed an ActivMedia Pioneer 2-DX mobile robot (see Figure 2). The Pioneer 2-DX is a 44cm x 38cm x 22cm non-holonomic two-wheeled base that is differentially steered (a passive caster provides balance). This research robot can be configured with many different peripherals, including front and rear sonar arrays, compasses, pan-tilt-zoom cameras, and laser range-finders.

Each robot houses a Pentium-based computer running Linux. Low-level sensor and actuator control is handled by the TCP socket-based device server Player[2], which executes on-board the robot. For details of Player's internal architecture, consult [5].

Of particular importance to the topic at hand is the way in which Player handles multiple clients. As shown in Figure 1, Player mediates the flow of commands and data between clients and devices through shared buffers. Whenever a client sends a new command to a device, Player puts that command in the device's command buffer. The appropriate device driver, at its leisure, retrieves the command from the buffer and passes it on to the physical device. In order to allow maximal flexibility in the design of distributed control systems, Player does not implement device locking; when multiple clients are connected to a Player server, they can all write into a single device's command buffer. There is no queuing of commands and each new command will overwrite the old one; the driver for the device will only send to the device itself whatever command it finds each time it reads its command buffer.

For the experiments detailed in this paper, the controllers were implemented as Player clients and executed on a separate desktop machine. Data and commands were passed between clients and server over a wireless Ethernet (IEEE 802.11) network that has an effective shared bandwidth of approximately 1.9Mbps. In all cases the device being controlled was the pair of DC motors that drive the robot's wheels. The clients receive as data the robot's current position and heading. The robot's translational velocity is fixed at a constant positive (forward) value, leaving the clients to command a single parameter: the robot's angular velocity. In order to ensure that the clients have equal chance to actually command the motors, we added a small

[2]Player was developed at USC and is freely available from: `http://robotics.usc.edu/player`.

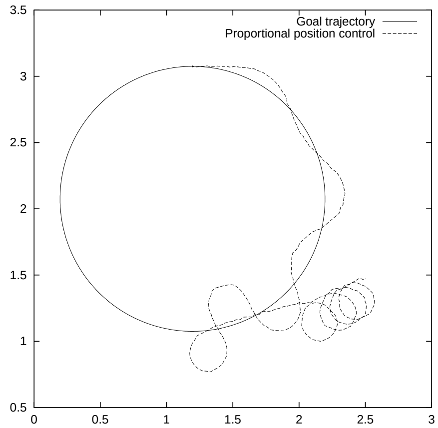

Figure 3: *Example trial of purely proportional position control (units are meters).*

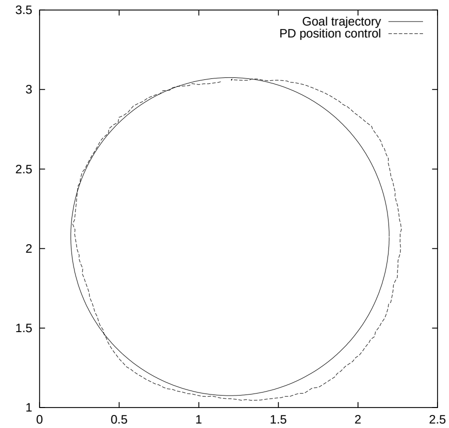

Figure 4: *Example trial of proportional and derivative controllers in parallel (units are meters). The performance for this trial was 88.54%.*

random delay to each client.

To evaluate the performance of our various controller configurations, we required objective ground-truth as to the robot's position. We obtained this information from an external metrology system that uses a laser range-finder that is accurate to approximately 2cm. The robot was outfitted with a special "beacon" (Figure 2) that is covered with a paper retroreflective in the infrared spectrum in which the laser operates. Experimental trajectory information is derived from this metrology system.

5 Experimental Results

Following the lead of [6], we validated our approach on a simple task[3]: follow a circle of radius 1 meter. We also apply their performance metric, in which the error for a trial is defined as the total difference in area between the actual trajectory and the target trajectory (i.e., the 1 meter circle). For each segment in the robot's trajectory[4], the error is computed by subtracting the area of the triangle formed by the end points of the segment and the center of the circle from the area of the sector described by the radius of the circle and the change in angle corresponding to the segment. Thus this error is measured in square meters. We normalize the error to the area of the ideal circle (πm^2), subtract the result from 1 and multiply by 100, yielding a performance figure that varies from 100% (perfect trajectory execution) to 0% (divergent paths).

For our first experiment we wrote a proportional controller that attempts to hold the robot's (x, y) position on the boundary of the circle. This controller executes the following rule: if the robot is inside the circle then turn away from the center of the circle; if the robot is outside the circle then turn toward the center. Figure 3 shows an example trial of this experiment, with the ideal 1 meter circle for reference. Clearly, this controller suffers from lag-induced overshoot which makes it unstable.

Control theory tells us that we can correct for overshoot by adding to the command signal a term that is the derivative of the error signal. Thus we wrote a second controller that is purely derivative. For our second experiment we ran this derivative controller in parallel with our proportional controller. Figure 4 shows a sample result. As can be seen, the overshoot is corrected and the robot tracks the target trajectory quite well. Thus, by executing P and D controllers separately, with no command arbitration, we achieve PD control through the physics of the motors. For comparison, we also implemented PD control in a single agent by computing and adding the proportional and derivative terms. We ran 10 trials of each kind of PD control. Using an unpaired t-test, the resultant errors are *not* statistically different, from which we can conclude that our parallel control approach performs as well as the conventional method.

In the first experiment, the overshoot is caused by a delay between the command signal, which adjusts the robot's heading, and the error signal, which depends on the robot's position. Thus another way to combat overshoot is to add a component that servos directly on heading. We wrote a new proportional controller that assumes the robot is on the boundary of the circle, calculates the local tangent, and uses it as a target heading. Figure 5 shows an example trial of this heading controller, alongside the original position controller. Though it keeps an approximately correct heading, this new controller suffers from positional drift. For our third experiment we ran the position controller in parallel with the heading controller. As Figure 6 shows, we

[3]We acknowledge that this task can also be achieved open-loop by holding constant the speeds of the two wheels, with a fixed difference between them.

[4]Although the robot moves continuously, our metrology system gives the trajectory as a sequence of discrete points.

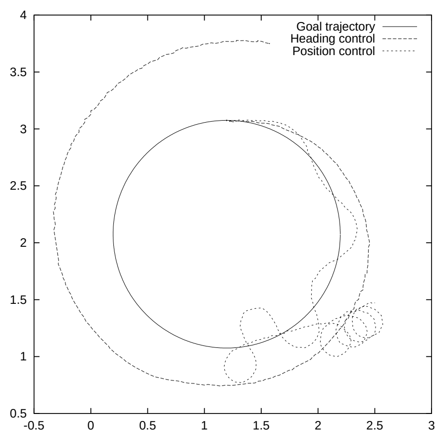

Figure 5: *Example trials of heading and position controllers (units are meters).*

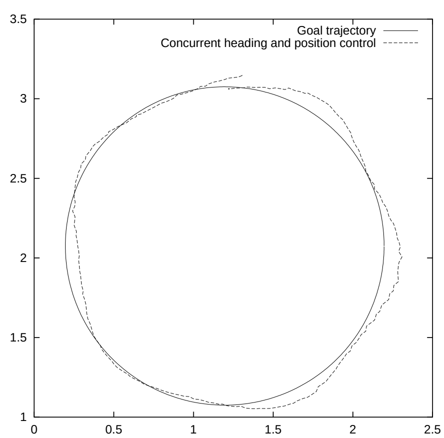

Figure 6: *Example trial of heading and position controllers together (units are meters). The performance for this trial was 91.09%.*

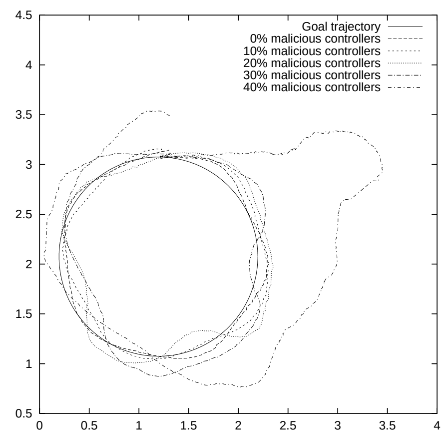

Figure 7: *Examples trials with various proportions of malicious controllers (units are meters).*

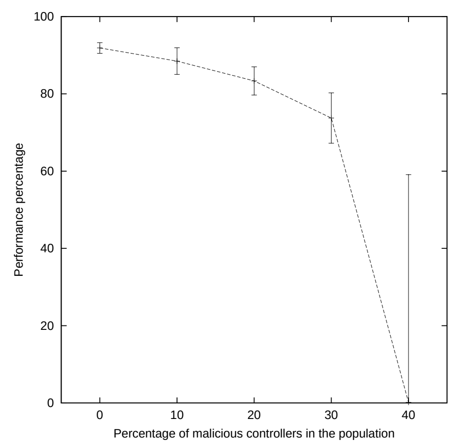

Figure 8: *Performance on the circle following task as the proportion of malicious controllers increases.*

have found another set of controllers that, though independently poor, perform well when combined in the motors.

Having established that our method of concurrent control is viable when all controllers are doing the "right" thing, we wanted to know how the system responds to erroneous input; i.e., how fault-tolerant the system is. For our fourth experiment we replicated the third experiment, except that instead of one position controller and one heading controller we ran 25 of each. In this population of 50, we designated some controllers to be "malicious." A malicious controller computes the same command signal as the corresponding "good" controller but then sends the additive inverse of the signal, effectively turning the robot the wrong way. We varied the percentage of malicious controllers in increments of 10% from 0% to 40%, running 10 trials at each level. Figure 7 shows a representative trial for each. The system performs well for small proportions of malicious controllers (the error at 10% is not statistically different from that at 0%) and degrades sublinearly until it breaks down completely at 40%. This catastrophic failure is clearly visible in Figure 8, which plots the mean error against the percentage of malicious controllers. We note that these results directly contradict [6], in which the authors conclude through experiments with a robot simulator that error in a concurrent control system tends to zero as the proportion of malicious controllers approaches 50%.

As a final experiment we changed the task from following a 1 meter circle to following a figure-8 composed of two 0.75 meter circles. We ran, with slight modifications, one each of the position and heading controllers. Figure 9 shows the overlaid paths from ten trials for this new task. These results suggest that our control method could be used to drive a robot along any sufficiently smooth trajectory.

One subtle, but important, point is worth making. Before running the experiments, we instrumented Player to

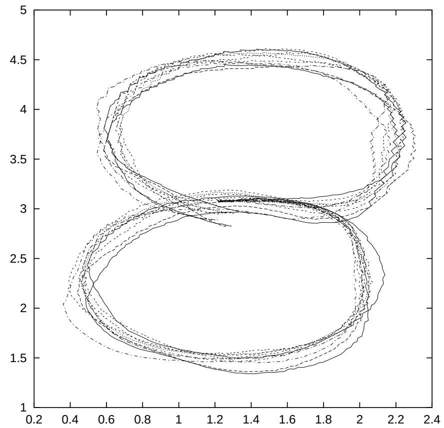

Figure 9: *Overlaid tracks from 10 trials of the figure-8 task (units are meters).*

cause it to record "whose" command actually got through to the robot's motors in each cycle. We were then able to verify that the proportion of a type of controller (e.g., good heading, malicious position) in the population does indeed correspond directly with the proportion of time during which the motors are actually commanded by that type of controller.

6 Conclusion

In this paper we proposed a method for the control of a mobile robot by a large population of independent controllers. These controllers do not communicate with each other and perform no action selection. Instead, they command the robot concurrently and the macroscopic behavior of the system is determined by a mechanical "calculation" that is performed by the motors.

We stated conditions necessary for this approach to be viable and successfully applied it to planar mobile robot control. Further, we showed through experiment that a concurrent control system can be remarkably robust to systematic error in the input signals. In addition to fault-tolerance, concurrent control has the advantage of being easily distributible over a network. As a consequence, concurrent control systems are inherently scalable, up to the bandwidth and computation available in the network.

We do not claim that this approach to control is advisable in all situations, even when the necessary conditions laid out in Section 3 are met. Rather we offer it as a novel and insightful example of how system dynamics can be beneficially exploited in order to elicit goal-directed behavior.

One possibility for further exploration is to measure the effectiveness of our approach when the controllers are operating on more disparate sensor information (e.g., stereo vision and echolocation). Another direction is to tune system performance by giving more or less weight to one kind of controller; this adjustment can be made easily by adding or removing controllers, thereby changing the relative proportion of controllers in the population.

Acknowledgments

The research reported here was conducted at the Interaction Lab, part of the Robotics Research Lab at the University of Southern California Computer Science Department. The work is supported in part by DARPA Grant DABT63-99-1-0015, and Office of Naval Research Grants DURIP N00014-00-1-0638 and N00014-01-1-0354. We thank Andrew Howard and Richard T. Vaughan for their helpful comments.

References

[1] Ronald C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 264–271, Raleigh, NC, May 1987.

[2] Stuart Bennett. A Brief History of Automatic Control. *IEEE Control Systems Magazine*, 16(3):17–25, June 1996.

[3] Douglas M. Blough and Gregory F. Sullivan. A Comparison of Voting Stategies for Fault-Tolerant Distributed Systems. In *Proc. of IEEE Symposium on Reliable Distributed Systems*, pages 136–145, 1990.

[4] Terrence Fong, Sébastian Grange, Charles Thorpe, and Charles Baur. Multi-robot remote driving with collaborative control. In *IEEE Intl. Workshop on Robot-Human Interactive Communication*, Bordeaux and Paris, France, September 2001.

[5] Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, et al. Most Valuable Player: A Robot Device Server for Distributed Control. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1226–1231, Wailea, Hawaii, October 2001.

[6] Ken Goldberg and Billy Chen. Collaborative control of robot motion: Robustness to error. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 655–660, Wailea, Hawaii, October 2001.

[7] Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The Intl. J. of Robotics Research*, 5(1):90–98, 1986.

[8] David A. Mindell. Automation's Finest Hour: Bell Labs and Automatic Control in World War II. *IEEE Control Systems Magazine*, 15(6):72–80, December 1995.

[9] Robin R. Murphy and Ronald C. Arkin. SFX: An Architecture for Action-Oriented Sensor Fusion. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1079–1086, Raleigh, NC, July 1992.

[10] Paolo Pirjanian. Behavior Coordination Mechanisms - State-of-the-art. Technical Report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, October 1999.

[11] Julio K. Rosenblatt. The Distributed Architecture for Mobile Navigation. *Journal of Experimental and Theoretical Artifical Intelligence*, 9(2–3):339–360, April–September 1997.

[12] O. G. Selfridge. Pandemonium: A paradigm for learning. In *Mechanisation of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory*, London: HMSO, November 1958.