A Survey of Control in Mobile Autonomous Embodied Robots

Nate Gauntt

May 12, 2010

Abstract

Autonomous robots have been an object of human imagination for some time, since at least the industrial revolution. Only recently, with the advent of analog and digital computers, have we been able to design and implement systems of sufficient complexity to be considered autonomous. The earliest autonomous robot systems were built in the early 60's, were notoriously sensitive to environmental noise, and were built to accomplish tasks so specific and simple as to question their usefulness in the wider world. As computers have become exponentially more powerful, robots have attempted to harness this power to become more useful, and to adapt and learn from complex and changing environments. This survey concerns the history of mobile robots in the last 50 years, starting by describing the problem each robot was tasked to solve and methods used in robot design. The performance of these robots is qualitatively compared over time, ending with a discussion of current trends and future work in robotics.

Early Robots

BlockWorld

Inspired by pioneering work in computer image analysis done by Roberts at MIT in 1963, the research group there lead by Patrick Winston, in 1972, built a robot that could view a scene of blocks in a photo and reconstruct a similar scene out of blocks with its manipulator arm. The program was known as copy-demo, and helped justify the classical AI approach in which a complete 3D model of the environment is used to control a robot.

Shakey

From 1966 to 1972, the Artificial Intelligence Center at SRI (Stanford Research Institute) designed and programmed a mobile robot curiously nicknamed "Shakey" [**Nilson 84**]. Two versions were created, the first in 1969 (figure 1) was equipped with a TV camera, touch sensors, and radio equipment to both relay images to a larger support computer and relay commands back to the robot. The second was completed in 1971 and among other improvements, had access to a larger host computer, with 192K 36 bit words (0.8 MB) of working memory, and ran control programs totaling in size of about 1.35 MB. In 1970, Life magazine referred to Shakey as "the first electronic person", and Shakey gained much notoriety in the public imagination, in part from a short 24 minute film demonstrating Shakey's abilities [**SRI 06**].

One of the stated goals of the project was to 'develop concepts and techniques in artificial intelligence enabling an automaton to function independently in realistic environments'. Hardware complexity was intentionally rejected in favor of implementing that complexity in the computer

programs controlling Shakey. It was implicitly assumed that improvements in computer speed and miniaturization would improve performance of the algorithms and enhance the robot's independence over time. Also, it was assumed that, though Shakey's environment would start out much simplified, that the system would be able to be iteratively extended with more complex environments until Shakey could eventually deal with truly realistic environments.

Most criticisms of Shakey deal with the vastly oversimplified environment the robot was made to operate in. All of the objects in the environment were specially constructed rectangular parallelepipeds and wedges, and were specially colored with flat (non-specular) paint. Originally the objects were painted black for high contrast, but this had to be changed to red, as the rangefinder needed reflected light to operate. The overhead lighting was placed to eliminate all shadows, special baseboards were added to the walls to enable robot orienting, and the floor was specially designed to have no identifiable texture or markings and reflect only matte light [**Nilson 84**].

Brooks notes that after 30 years of research into computer vision and edge detection, no research group finding lines and surfaces in natural image processing has come close to the accuracy of Shakey's line finding in the test lab environment [**Brooks 91**]. At the time of writing, no research team operating independent robots in real office environments could match the number of features or performance of Shakey, despite several decades of research. It is surmised that the only way Shakey's rule-based inference programs could efficiently operate and interact with the world was from the vastly reduced world representation that the rules modeled and operated upon.

Stanford Cart

The Stanford Cart was an attempt to take the successes of Shakey out of the lab [**Moravec 83**]. The cart was un-tethered, but like Shakey, had on-board video cameras and a radio link that transmitted and received commands from a more powerful computer nearby. The cart used several stereo vision algorithms to deduce the nature and position of objects around it, and reconstructed representations of these objects in its internal 3D model of the world. Using this representation, the cart would plan a path through the obstacle course and move a short way through the course, then stop to adjust or recalculate the path as necessary.

Unlike Shakey, the cart did learn, in a limited sense, from its environment. Where Shakey's input environment (lighting, color, wall geometry) was altered to suit the algorithms and sensors, the cart would perform sensor calibrations when first brought into the new environment, such as finding the camera focal length and distortion. Shakey used an inverse projection matrix technique to range find objects, which is only possible if the object geometry is known in advance and the object edges are easily and noiselessly computed in experiments. The cart had no such advantages, and used convolution operators over similar (nearby) images of the scene to more robustly find objects. Using stereo cameras along with the convolution object finding allowed to calculate the range of objects from the cart.

One effect of this visual object finding method was that the cart cannot move 'too far', or the convolution fails as the images are too different. Thus, while the object finding and path planning were capable of planning multi meter path lengths, the actual distance traveled along a path was limited to 0.75 meters, as a compromise between not breaking object detection and having a reasonable speed through the obstacle course [**Moravec 83**]. Due to image computation and path planning time, the real averaged speed of the cart was 3 to 5 meters per hour. Also, Moravec notes that the vision algorithms were brittle, easily recognizing some objects such as chairs, but failing in general to recognize less detailed objects such as matte cardboard cutouts. The reasons

for this are complex and subtle, and serve as a cautionary tale for computer vision research. The clean white faces of the cardboard, as opposed to the more graduated texture of 'natural' objects, reflected so much light as to overwhelm the digitizer circuits inside the video cameras, and the corresponding picture input to the algorithm was severely muted by the gain correction automatically applied by the cameras. Further, this condition occurred mostly in bright sunlight and not in other situations, making the problem initially harder to pinpoint.

CMU Rover

The Standford Cart, while more adaptable and robust than Shakey, suffered from significant design problems, the worst of which was the computational time taken to analyze video frames. The time taken to analyze frames was so long, obstacle courses had to be shortened and simplified to get test data in a reasonable amount of time, and correspondingly, there was less time for robot improvements based on this feedback [**Moravec 83**]. The CMU Rover was the next evolution of the cart, and among other improvements, was fitted with sonar sensors, infrared sensors, and on-board processors in an attempt to push the problem of obstacle avoidance to a lower level, fast response system. Unfortunately, as of this writing, no more was published on the (yet unfinished) CMU Rover.

Robots of the 80's

Mobots

In 1987, Brooks and his research team built a series of robots designed to challenge the classical AI theory of robotics, where robots build a detailed internal world model in order to plan and accomplish tasks [**Brooks 87**]. Brooks' robot design methodology directly opposed mainstream academic robotic design. The robots were tested and run outside of a controlled lab environment, where they would have access to, and have to handle, realistic stimulus. He popularized the *subsumption* architecture, where multiple behaviors were designed independently and run in parallel on the robot, and behaviors would compete for control of the robot's actuators. Furthermore, all behaviors were derived from simple finite state machines and were required to operate in real-time, with no centralized world representation or centralized planning and control. Despite this, his team's robots (fig. 4) were able to mimic intelligent behavior and interact with a changing environment in a way that other systems of the time could not.

Ghengis

Another robot project by Brooks was started and completed in the summer of 1989, with the help of just two students, Grinnel More and Colin Angle [**Brooks 89**]. The goal of this robot was to show that complex behaviors, such as insect walking, can be implemented by the same layered architecture as before, with simple, distributed control elements and no central representation or planning. Also, the intent was to show that such a system can be evolved, where simple FSM layers can be built to provide basic behaviors, and more complex behaviors can be added on by adding FSM layers *without* removing, replacing, or even changing the earlier layers. Instead, as before, the high level layers will compete for control by strategically suppressing low level layers.

In the new robot design, the FSM's have been slightly enhanced by the addition of programmable timer events and programmable events triggered by certain combinations of FSM register (memory) states. The robot is much smaller and more self contained, measuring 35cm long with 6 rigid legs and a combined leg span of 25cm (fig. 5). The most interesting part of the robot is its unique walking algorithm. All the legs operate independently, and are initially in the 'balance' routine that move to keep the robot stationary. Periodically, one of the legs (in sequence) is sent a brief 'raise' signal. The other legs reflex-

ively respond by pitching forward, and the raised leg reflexively responds to being in a raised position by executing the 'lower' routine, which lowers the leg. All the legs then are back to the 'balance' routine until the next 'raise' timer event is sent to the next leg. Amazingly, the robot is able to mimic the well known 'alternating tripod' motion seen in insects, albeit with far less pitch and roll control. By instrumenting force feedback in the legs, the authors were able to make improvements to pitch when crossing a test path with a step obstacle, and infra-red sensors were added to create a high level 'prowl' behavior which would follow humans around the lab.

Toto

One of the criticisms of Brooks' work, as well as of reactive robot design in general, is the lack of goal directed behaviors. This is in part due to reactive robots not having internal world representations, which more easily allow for human defined goals on these representations. In 1990, Maja Mataric of the MIT AI Lab built a reactive robot called Toto (fig 6) designed to address these flaws [**Mataric 90**].

Centralized, detailed world representations suffer from some significant flaws, including sensor noise and corresponding drift in the accuracy of the internal world model. The solution in Toto was to build a distributed, noise resistant internal world model. Instead of mapping areas in detail and trying to correct for sensor drift, Toto identified landmarks, such as corners and obstacles, as consensus calculations from the sonar, compass, and wheel sensors. Integrated distance and compass measurements are used to relatively position landmarks, and new landmarks are created as the robot explores the environment. All landmarks are compared in parallel with the sensor measurements to position the robot in the landmark graph, and *spreading activation* is used to make topologically close nodes to the current landmark more likely candidates for recognition, improving robot orientation accuracy.

Toto can navigate to a specific landmark by following a repeating call sent from the 'goal' landmark. The call attenuates as it traverses the landmark graph, so the strongest signal will come from the closest neighboring landmark to the current landmark. The robot then proceeds in the direction of the neighboring landmark closest to the goal landmark, subject to the layered behaviors that avoid obstacles and wander out of blind corners. In this way, Toto achieved real time path planning to a previously learned location by using an internal world representation, albeit a fuzzy and decentralized one.

Work on Toto led to research on improvements to the spreading activation technique for robot control. In [**Kortenkamp 93**], The authors propose a number of simple improvements for encoding additional semantic information in the landmark graph, such as one way paths, popular paths, landmark length measurement, and additional landmark types like gateways and corners. Such semantic information may be important in animal navigation, and allows for performance improvements and scalability in the general technique. In [**Kuipers 91**], the authors suggest extending Kortenkamp style semantic graphs by creating instead a hierarchy of graphs that store different semantic information at different graph resolution, and this hypothesis is tested on the NX robot simulator.

Robots of the 90's

Dervish

Dervish is a mobile robot developed at Stanford, and is noted for winning the office delivery event of the 13th National Conference on Artificial Intelligence [**Nourbakhsh 95**]. Dervish is unique in that the traditional ring configuration of sonars is replaced by mostly forward sonar configuration, and the angle of sensors is varied to detect obstacles not present in a single z plane relative to the

robot's geometry. There are 2 sonar sensors on the sides of the robot (perpendicular to the forward sensor array), and the robot is equipped with 2 macintosh powerbook laptop computers that control planning and movement commands.

Dervish's functionality is determined by the (very constrained) contest problem of navigating in an office building from a start room to a goal room using a map designed to show connectivity of regions without distance information. Because of this, Dervish has task-specific feature recognition for open doors, closed doors, hallways and the like. It uses information about doors encoded in the map and read from sensors to orient in the environment, and the position generated is a set of states to represent uncertainty in positioning. Planning and movement are interleaved, as sensor information affects the most likely position state, and paths are replanned if this state changes.

Xavier

Xavier is a mobile robot developed by CMU with the goal of being able to function and deliver packages in an operating office environment. User requests for the robot's attention are generated via the web, and the robot attempts to service these requests with optimal efficiency [**Simmons 97**]. The sensors on Xavier include bump panels, wheel (distance) encoders, a sonar ring for obstacle ranging, a movable 30 degree FOV laser rangefinder, and a single color camera (fig 7). Unlike the Stanford cart, color vision is used only for fine positioning and verifying landmarks, whereas the laser rangefinder, sonar, and odometers control the primary navigation and object detection. The choice to minimize the importance of machine vision makes sense for Xavier, as it is limited to the on-board processing power of three 486 CPUs, and machine vision algorithms are both computationally intensive and can reduce the ability of the robot to respond in real-time.

The navigation is done in Xavier using partially observable Markov decision process (POMDP) models. Any robot using an internal representation of topological maps needs to be able to estimate its orientation relative to the map (ie. the compass North), and then its position on the map. Xavier uses integration over time to determine average heading, and integration over sensors to reduce noise and determine a kind of average surrounding geometry. Also, averaging sensors allows information from groups of sensors to be treated as independent samples from the 'distribution' of surrounding geometry data, which feeds into the probabilistic POMDP model that estimates robot position and controls orienting movements.

Task planning involves maintaining a simple world model describing the package delivery process of the robot. It has states such as 'has-item', 'acquire-item', 'in-room-123', 'goto-deliver-room-123', and so fourth. The task planning algorithm deals with maintaining many such process trees, where similar trees have parent / sibling relationships and correspond to nearby tasks. The planner optimizes which trees to traverse next on throughput constraints and on information, such as 'in-room-123', taken from the POMDP navigation module. Since task planning is done in parallel with obstacle avoidance and position estimation, plans can change quickly in response to a changing environment or in response to detecting sensor errors.

Xavier combines an expert decision system in task planning, a POMDP machine learning system in location mapping, a heuristic A^* search in path planning, and a purely reactive system in obstacle avoidance and default wandering. The low level behaviors dealing with robot safety (avoid obstacle) take precedence over high level behaviors (deliver package). The behaviors operate in parallel and independently; for example, while waiting for a path to be planned, the robot reverts to the low level behavior of wandering and avoiding obstacles. In this way Xavier is a hybrid of behavior centric and model centric approaches.

Rhino

The RHINO system is capable of exploring and navigating previously unknown indoor office environments at a speed of approximately 3 ft/sec [**Thrun 98**]. It uses both onboard and offboard computers, and utilizes decentralized, asynchronous communication between software processes and between computing devices to accomplish tasks.

RHINO constructs representations of the environment as it moves, in the form of two dimensional maps. The maps are formed as a combination of grid-based and topological techniques. Grid-bases maps are constructed on a regular grid and are better at orienting the robot and sensors, but are large and computationally expensive to compute with. Conversely, topological maps are sparsely sampled only at necessary points and can be used in real-time computations, but are difficult to initially create. RHINO uses these maps as an internal world model during path planning to avoid obstacles, and updates the maps with information from its sensors.

Sensors on the robot (fig 8) collect range information with a ring of ultrasound sonar sensors surrounding the robot as well as with a top mounted stereo camera. The ultrasound sensors directly give ranges of obstacles, but have low resolution and fail to detect objects that absorb sound. Correspondingly, the visual sensors have high resolution and can detect most objects with visible edges, but have more limited range finding capabilities. The authors note that obstacle maps constructed using only feedback from actuator sensors and 'dead reckoning' from a known position are near useless, so visual and audio sensors play a critical role in generating accurate maps.

Modern Robots

Stanley

The 2005 Darpa Grand Challenge was to design and implement an autonomous vehicle capable of navigating and driving unaided, at high speeds, through an unrehearsed course in rough desert terrain. Stanley was the car robot built by engineers at Stanford, Volkswagen, and Intel that won the competition. Since the course description was given to the teams before the race as a list of longitude and latitude waypoints, no path planning was involved in the challenge. Other than integrating sensors, actuators, and computers into a Volkswagen Taureg, the robot is fundamentally similar to other embodied autonomous robots. Most of the sensors on Stanley were attached to the roof rack (fig 9), including GPS for finding waypoints, five laser range finders for obstacle detection to 25m, a color camera for road following (where possible), two 24GHz radar sensors for obstacle detection to 200m, and a GPS compass. In the trunk was an array of 6 Pentium M computers, with a total power draw of 500W provided by the car alternator.

Stanley's architecture is closely related to the three-layer architecture, where a reactive system operates in closed loop to perform critical (obstacle avoidance, safety) functions, and in parallel, a higher level system computes objectives while the arbitration system shares control of the hardware between the other two systems [**Gat 98**]. Information flows one way in the system, from the sensors to software modules that keep the data, then to other modules that 'subscribe' to this data store. Guarantees are kept that no software modules sees the same sensor data more than once. Special modules track the state of only other software modules, rebooting or power cycling the components as necessary when a failure state is detected. In total, the race software consisted of 30 such software modules executed in parallel.

The heart of Stanley's navigation uses an unscented Kalman filter (UKF), based on technology used by NASA in the Apollo program to create estimates of physical state based on sensor networks and the relevant physics models describing the dynamic motion of an object [**Wiki 10**].

The UKF is able to take noisy sensor data from the robot, combine it with known vehicle physics, and estimate the new vehicle position much better than related techniques, such as dead reckoning. The calculated vehicle coordinates, orientation, and velocity are fed to other modules as error-corrected inputs. Correctness of this positioning data is critical for navigation during frequent GPS outages over rough terrain.

Errors in positioning strongly affect the performance of the robot. Since Stanley uses long range lasers to construct a height map of the area in front of it for obstacle avoidance, the authors noted that very small errors ($< 0.5^{\text{deg}}$) in the vehicle's assumed orientation with respect to surrounding terrain were magnified by the distance of the laser rangefinders, such that obstacle classification became useless. Modeling the drift error in orientation with respect to time helped define the confidence of obstacle detection, and pruned out spurious obstacles using a statistical test. Calculating the parameters of this test is not simple, however, and reinforcement learning with multivariate gradient ascent on the parameters, based off human driving trials, is used to learn good parameter values.

Given that the laser rangefinders have an effective range of only 25m, a color camera and vision algorithms are used to find extended height maps (and so drivable surfaces). This is necessary to navigate at speeds high enough to successfully complete the challenge specification. Road classification using color vision is generally difficult, and this is enhanced by the varied types of roads in the course. Stanley uses an clever method to make this problem both more tractable and adaptive to changing road types and conditions. The robust drivable terrain detection in the limited range laser scanners is superimposed onto the lower portion of the color camera's image space. Then, the image algorithm has a partial example of what drivable and non-drivable terrain looks like. Using this dynamically generated reinforcement learning from the laser finders, the image algorithm is able to find drivable regions in the rest of the image. Special attention is paid to how often and by how much the visual classifier is updated on subsequent image frames to optimally adapt to changing lighting conditions and changing road types.

Robot Soccer

There is a trend in the last ten years to explore teams of autonomous robots, instead of just one operating independently. One popular venue for such robot teams is robot soccer competitions. In [**Lenser 01**], the soccer match is played by robots provided by Sony to the research team, similar to the commercial AIBO robot dog but with programmable capability (fig 10). Each team consists of three uniformed robot dogs. The playing field is marked around the border by 6 markers and 2 goal areas, each specially colored to help robot localization.

The hardware on Sony AIBO clones is limited, necessitating efficient algorithms using simplifying and task-dependent assumptions. Lenser and his team have published a freely available library that performs color correction, color segmentation, and object detection from the segmentation and task-specific queues. Object detection is limited by having a single color camera as the only remote sensor, and all the problems of occlusion and range estimation apply here, such that information about objects tends to be sparse and noisy. Good recognition currently needs a supervised calibration to correct for differing lighting conditions. After recognition, objects are translated into robot-relative coordinates, accounting for camera tilt, pan, roll, and robot body position.

Because of computational constraints, the localization of a robot on the field does not depend on a long history of past location updated with actuator readings. Instead, they use a novel technique called Sensor Resetting Localization, described in [**Lenser 00**]. The basic technique in-

volves sampling the visual field and calculating the probability of being in an area given the distribution of the sample points. This probability changes over time, and can be invalidated by, for example, a human referee placing a robot in a different location. When invalidation of the location probability occurs, it is recognized by real sensor samples not matching expected location, and the location probability is reset to the uniform distribution over the field. Also, sensors can cause non-uniform generation of sample points, for example, to cluster around a potential landmark should a previous uniform sample happen to coincide with one.

Most of the behaviors for the soccer robots are reactive to the inputs, with the exception of prediction of the soccer ball given occluding robots. This is modeled as a virtual sensor predicting ball location, whose behavior 'chase ball' is reactive with respect to this 'memory' sensor. Behaviors are organized both hierarchically and sequentially to compose more complex behaviors from simple behaviors. Behaviors sets are chosen based on sensor input and the current goal being satisfied, and compete for control of actuators in a pseudo winner-take-all strategy that minimizes actuator conflict.

In order to translate behaviors into movement, high level behavior requests are sent through a movement layer that models physical properties such as the joints, center of mass, and approximated overall stability. This layer composes low level movements, such as individual leg movements, into compatible movement sequences that minimize instantaneous velocity changes and smooth overall movement. In general, the robot performed well, and scored third place in RoboCup 2000 international competition, losing only one game.

Animal Inspired Robots

AmphiBot II

In human walking, humans can dynamically adapt the normal walking gait to changing terrain. This is accomplished through pulsing collections of neurons called central pattern generators (CPG) in the spine [**Grillner 03**]. Many non-wheeled robots, including the snake-like AmphiBot II (fig 11), use CPG's to control and adjust movement, in this case to control the speed and direction of the snake robot [**Crespi 06**]. This robot is unique in that it can locomote on both land and in water, using the CPG to tune the phase and amplitude of the traveling wave. Also, small processors and high energy density Lithium-Ion batteries make this system self contained. Other than the waveform generated by the CPG oscillators, however, this system has no other behaviors, making it minimally autonomous from the point of view of previous robots discussed.

BigDog

BigDog is a quadruped walking robot, somewhat resembling a dog (fig 12). The BigDog robot and its predecessors from the MIT LegLab are all controlled by regulating three activities: supporting the body with a vertical bounding motion, controlling the tilt of the body by pushing off with different legs, and putting the feet in the right place for the next step [**Raibert 08**]. Though BigDog shows good stability walking forward over very rough terrain, it has no other autonomous behaviors and requires a human operator to steer and pathfind via a computer radio link.

In [**Iida 04**], a different approach to quadruped movement is presented where, instead of compensating for an unstable load using sophisticated active dynamics, the walker's passive dynamics is engineered to be stable. Generating a bounding gait in these machines requires no feedback, unlike BigDog, which has both gyroscopic and force feedback on the legs and cart. Furthermore, since the design is passively stable, the

control system can be reduced to a sine wave generator (much like a CPG), where simply changing the phase parameter is shown to control the speed of the gait, despite the many degrees of freedom involved in the robot. Iida's quadruped has not been tested on varied terrain, however, and possess no high-level behaviors other than forward movement. Discussion and simulation of passively stable biped walking is found in [**Vaughan 04**].

Analysis

Early robots such as the BlockWorld manipulator arm and Shakey made a lot of assumptions about the operating environment. When a preconceived algorithm had trouble with noise, the experimenters would change the environment, effectively changing the problem statement, to suit the algorithm. For example, to perform geometry analysis using the inverse projection matrix technique, one needs clear, accurate line finding on relatively simple objects. Therefore, the Shakey experimenters limited the color, shape, and number of objects that the robot had to detect, and also constrained the lighting and texture of other parts of the scene, to get the algorithm to perform acceptably. With the CMU cart, attempting to take the same general method out of this constrained environment led to a steep degradation of performance, to the point of not even being able to test the robot in a reasonable amount of time on the much more complicated, 'real-world' test environment. Much of this criticism can also be applied to early AI work in general, as hard problems tended to be reduced to very simple subdomains until the point at which they became tractable to the computing power and methods of the time. Game playing algorithms received much attention, for example, and early successes were shown for machines playing various classes of games. However, the AI work invested in games did not necessarily transfer at all to other problem domains.

The counter-revolution to classical AI techniques in robotics came from Rodney Brooks and his work on Mobots at the MIT AI lab. While the Stanford cart was taking tens of minutes to compute the next move in it's environment, Brooks's machines were zooming around the AI lab, interacting with their environment in real-time. While avoiding moving objects were written out of the Stanford problem statement, the Mobots avoided unforeseen objects and environments with ease while performing their tasks. Complex control activities such as coordinating the leg movement of insect-like robots could be accomplished with Brooks's subsumption architecture, where simple state machines suppress one another for control of robot hardware. All of this was done without recreating any explicit 3D model of the world, with no symbol model and no abstract algebra for logic or inference, and with very little central coordination in general.

Despite these successes, Brooks's methodology, termed *reactive* control, had detractors with legitimate concerns. It was difficult to tell reactive robots to achieve specific goals, as they possessed no representation of goal states or how to achieve them. Also, reactive robots generally don't learn from their environment, and aren't able to perform functions not explicitly programmed in them. Finally, each robot was a piece of art in essence, with carefully tuned finite state machines and often with custom hardware, so there were few general algorithms or techniques that could be applied to other problems or robots.

Purely reactive control was not a popular design methodology outside of the MIT lab, and was not practiced in all but the simplest robot designs past the late 1980's. However, many lessons of the reactive robots became standard practice, such as designing and testing a robot in the real world, limiting computation to run on the hardware inside the embodied robot, and having hierarchies of behaviors that operate in parallel. Dervish and Xavier are examples of this philosophy. Both robots have low level collision avoidance that operates largely independently of the

higher-level goal seeking and orienting behaviors. This way, the robot keeps itself safe in real time and can make progress on movement while the goal seeking and path planning slowly compute their results. Both robots were hybrids, as they had reactive systems for critical functions and also simple world models that predicated their actions. One consequence of having world models was that the robots had to deal with sensor error and drift in the internal world model from reality. Dervish did this using probability to construct the world model and having world state be a set of states associated with various uncertainty. Xavier did this in a more structured, and ultimately more popular way by viewing position and orientation as a probability distribution that is updated and corrected with sensor measurements.

Despite good performance on office tasks and navigation, this class of robots has serious shortcomings. Feature recognition is carefully tuned to the problem statement, as both Xavier and Dervish have custom routines to detect doors, hallways, and walls, and can not navigate well outside of (or possibly even in different) office environments. Thus they suffer the same problem of over fitting to a specific environment that Shakey and the Stanford cart had two decades previously. With the inclusion of even such simple world state lies the temptation to make limiting assumptions that simplify the problem at the cost of flexibility.

Toto, another reactive robot built by MIT, is more judicious in this regard. Instead of memorizing a map and counting doors (or probability of doors) to navigate, the maps that Toto makes are distributed in nature and learned from the environment. Landmarks are identified as unusual or unique sensor ensemble readings and are generated as the robot's reactive behavior blindly explores the environment. Positioning data from a compass and wheel sensors connect landmarks into a rough topological map, and a distributed, parallel algorithm is used to navigate piecewise to a defined goal landmark.

Rhino builds on the successes and failures of these office robots, keeping a reactive system for safety and the notion of parallel, distributed modules responsible for different behaviors. Like Toto and unlike previous office bots, Rhino builds maps as it explores. Rhino has neural networks interposed at the sonar sensors in order to calibrate for different wall types or test environments without changing behavior code. Like Xavier Rhino uses vision to help detect features not seen or ambiguous to sonar, using vision to enhance the sonar sense as opposed to relying on vision for main navigation.

Seven years later, the creator of Rhino would take many of the lessons learned there and put them into Stanley, the autonomous car that wins the 2005 Darpa Grand Challenge race across the Mojave desert. Stanley, like Rhino, is a modified three-layer architecture with a reactive safety system interacting with higher level path planning and control. Stanley, like Rhino, uses vision as an adjunct to improve the laser range finding sense. The internal world model of Stanley is more complex, and requires sophisticated mathematical techniques to keep drift with reality within bounds, but is largely successful in this. Stanley is an amazing feat of engineering, and manages to perform a complex task with a high degree of adaptability that just two years ago, in the 2003 challenge, it seemed no robotics team was very close to.

Conclusion

Robots have made sweeping advances in mobility and capability over the last 50 years. We now have a car that drives itself safely over rough desert roads, something unimaginable to the BlockWorld creators in the late 60's, or maybe only unimaginable that such a thing would happen in so short a time. In one sense, robotics has exceeded all expectations. Another way of looking at the success of robotics, however, is that we have grossly underestimated what it means to be intelligent.

People were amazed when the robot was able to recreate block stacks from a picture. People were amazed when robots could mimic insect walking. People were amazed when robots could deliver packages in an office building. People were amazed when a robot could drive a car. In all of the above examples of interesting robot behavior, not one of the robots learned a skill. The skill has been designed into the robot by a team of engineers studying the problem, and the flexibility of even advanced robots like the 2005 Stanley have fairly primitive and limited ability to adapt to novel circumstances. In the case of Stanley, despite the perceived performance in the driving task, Stanley is unable to drive without a human supplied map of waypoints, and probably would fail utterly if asked, even with a map, to drive on off-road terrain. Robotic success has been defined, then, in exceeding human expectations of what tasks robots in general can do, rather than what kinds of intelligent behavior an individual robot displays.

The intellectual descendants of reactive robots have perhaps perceived this for a long time, and suggest that progress in intelligence is achieved by more accurately modeling animal movement. Their thesis is that the hard part of intelligence is sophisticated control, that evolution has worked millions of years on the control problem and that intelligence is a recent side effect, or perhaps consequence, of good control. I would argue that they are wrong too, by construction. Robots like Amphibot and BigDog mimic very closely snake and dog movement, respectively. Other animalistic robots not mentioned here are able to fly by flapping tiny wings, and swim by swishing fish-like tails, often in sophisticated and energy efficient ways. These machines are also not intelligent in the sense that we commonly associate with insects and fish.

Some have accepted embodiment as a necessary grounding for AI algorithms that failed to transfer their intelligence outside of very limited domains. Some have accepted design of robots using real world environmental inputs to aid in robust sensing and reflex behavior. I think few have accepted that making robots perform a certain complicated task will not lead to a robot that is generally intelligent, despite a long history of this approach not working in classical AI research. Perhaps, in robot soccer, there is a seed of a correct methodology here, that we have to make robots embodied in the real world, solving difficult physical control problems, and most importantly, competing with other robots in a way that mimics the evolutionary process that has evolved intelligent behavior up until now.

References

[**Roberts 63**] "Machine Perception of Three-Dimensional Solids"; Roberts, Larry; *MIT Lincoln Laboratory, TR No. 315*; 1963

[**Winston 74**] "New Progress in Artificial Intelligence"; Winston, Patrick; *MIT AI Laboratory Technical Report*; AI-TR-310, 1974

[**Nilson 84**] "Shakey the Robot"; Nilson, N; *Artificial Intelligence Center, Technical Note* 323; 1984

[**Moravec 83**] "The Stanford Cart and the CMU Rover"; Moravec, Hans; *Proceedings of the IEEE*; Vol 71, Issue 7, 1983

[**Brooks 87**] "Intelligence Without Representation"; Brooks, R; *Artificial Intelligence*; Vol 47, pp.139-159, 1991

[**Brooks 89**] "A Robot that Walks: Emergent Behavior from a Carefully Evolved Network"; Brooks, R; *Neural Computation*; Vol 1-2, 1989

[**Mataric 90**] "Environment Learning Using a Distributed Representation"; Mataric, M; *Robots and Automation*; Vol 1, pp.402-406, 1990

[**Brooks 91**] "Intelligence Without Reason"; Brooks, R; *Computers and Thought, IJCAI-91*; 1991

[**Kuipers 91**] "A Robot Exploration and Mapping Strategy Based on Semantic Hierarchy of Spatial Representations"; Kuipers, B; Byun, Y; *Journal of Robotics and Autonomous Systems*; Vol. 8, pp.47-63, 1991

[**Kortenkamp 93**] "A Directional Spreading Activation Network for Mobile Robot Navigation"; Kortenkamp, D; Chown, E; *Proceedings of the 2nd Intl. Conf. on Simulation of Adaptive Behavior*; MIT Press, 1993

[**Nourbakhsh 95**] "DERVISH: An Office-Navigating Robot"; Nourbakhsh, I; Powers, R; Birchfield, S; *AI Magazine*; Vol. 16, No. 2, 1995

[**Dieter 97**] "The Dynamic Window Approach to Collision Avoidance"; Fox, D; Burgard, W; Thrun, S; *IEEE Robotics and Automation Magazine*; 4(1):22-33, 1997

[**Simmons 97**] "Xavier: Experience with a Layered Robot Architecture"; Simmons, R; Goodwin, R; *ACM SIGART Intelligence Magazine*; 1997

[**Thrun 98**] "Map Learning and High Speed Navigation in Rhino"; Thrun, S; Bucken, A; et al.; *AI-based Mobile Robots: Case Studies of Successful Robot Systems*; MIT Press, 1998

[**Gat 98**] "On three-layer architectures"; Gat, E; Bonnasso, P; Murphy, R; et. al.; *Artificial Intelligence and Mobile Robots (book)*; pp.195-210, 1998

[**Lenser 00**] "Sensor resetting localization for poorly modeled mobile robots"; Lenser, S; Veloso, M; *Proceedings of ICRA-2000*; 2000

[**Lenser 01**] "CMPack: a complete software system for autonomous legged soccer robots"; Lenser,S; Bruce,J; Veloso,M; *Proceedings of the 5th Intl. Conf. on Autonomous Agents*; pp204-211, 2001

[**Miller 02**] "Snake Robots for Search and Rescue"; Miller, G; *Neurotechnology for Biomimetic Robots*; MIT Press, 2002

[**Grillner 03**] "The motor infrastructure: From ion channels to neuronal networks"; Grillner, S; *Nature Reviews Neuroscience*; 4(7):pp573-586

[**Vaughan 04**] "The Evolution of Control and Adaptation in a 3D Powered Passive Dynamic Walker"; Vaughan, E; Di Paolo, E; Harvey, I; *Proceedings of 9th Intl. Conf. on the Simulation and Synthesis of Living Systems*; pp139-145, 2004

[**Iida 04**] "Cheap rapid locomotion of a quadruped robot: Self-stabilization of bounding gait"; Iida, F; Pfeifer, R; *Intelligent Autonomous Systems 8*; 642-649, 2004

[**SRI 06**] "AIC Home > Achievements > Shakey"; Internet; *http://www.ai.sri.com/shakey/*; accessed 2010

[**Crespi 06**] "AmphiBot II: An Amphibious Snake Robot that Craws and Swims Using a Central Pattern Generator"; Crespi, A; Ijspeert, A; *Proceedings of the 9th Intl. Conf. on Climbing and Walking Robots*; Brussels, Belgium 2006

[**Thrun 06**] "Stanley: The Robot that won the Darpa Grand Challenge"; Thrun, S; Montemerlo, M; Dahlkamp, H; et. al.; *Journal of Field Robotics*; 23(9),pp661-692, 2006

[**Raibert 08**] "BigDog, the Rough-Terrain Quadruped Robot"; Raibert, M; Blankespoor, K; Nelson, G; Playter, R; *Boston Dynamics Website, http://www.bostondynamics.com/img/BigDog_IFAC_Apr-8-2008.pdf*; 2008, accessed 2010

[**Wiki 10**] "Kalman Filter"; Internet; *http://en.wikipedia.org/wiki/Kalman_filter*; Accessed 2010

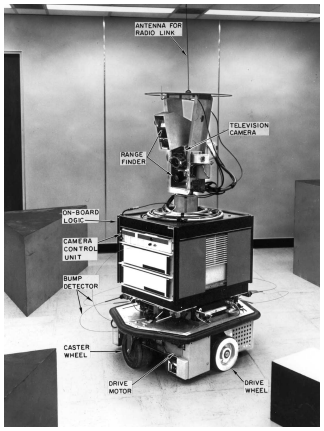

Figure 1: Shakey, a prototype mobile robot circa 1969

Figure 2: Stanford Cart, circa 1983

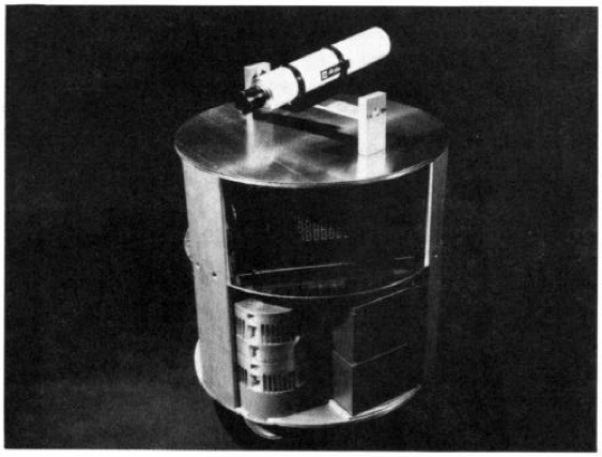

Figure 3: CMU Rover, circa 1983

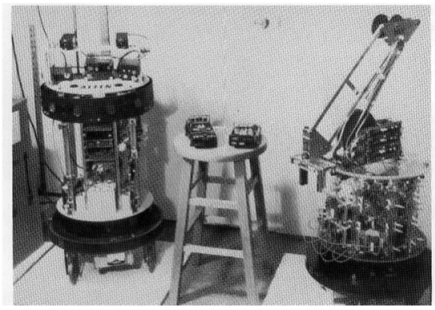

Figure 4: MIT ‘mobots’ circa 1987

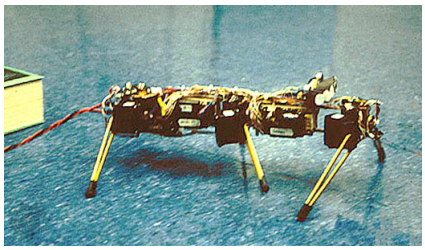

Figure 5: MIT robot ‘ghengis’, circa 1989

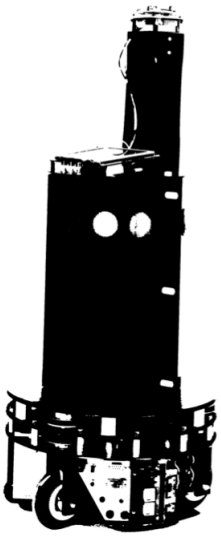

Figure 6: MIT robot ‘Toto’, circa 1993

Figure 7: XAVIER, an office delivery robot circa 1997

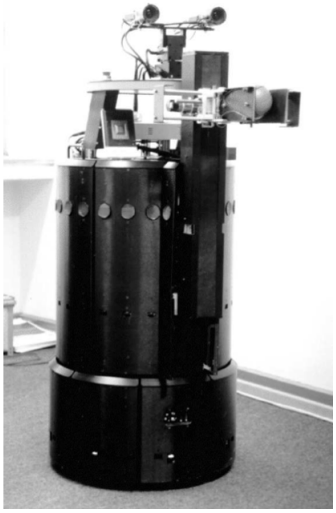

Figure 8: RHINO, a fast mobile robot circa 1998

Figure 9: Stanley, a semi-autonomous car, winner of 2005 Darpa challenge

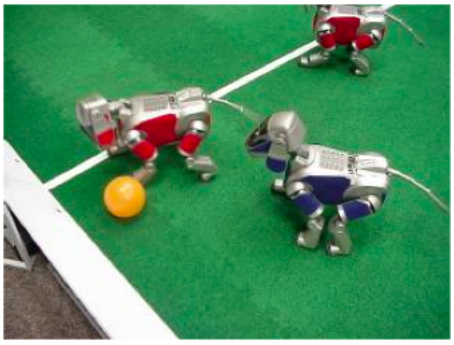

Figure 10: CMPack, Sony robot quadrupeds playing soccer, circa 2001

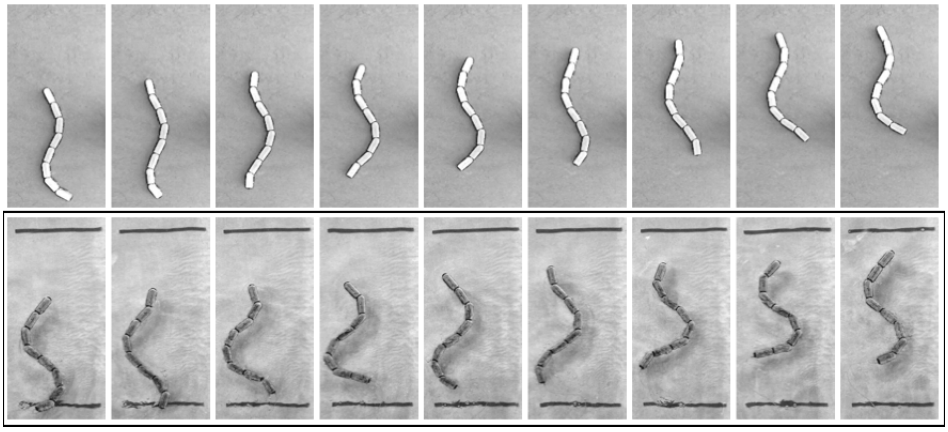

Figure 11: Amphibious snake robot on land (top) and water (bottom), circa 2006

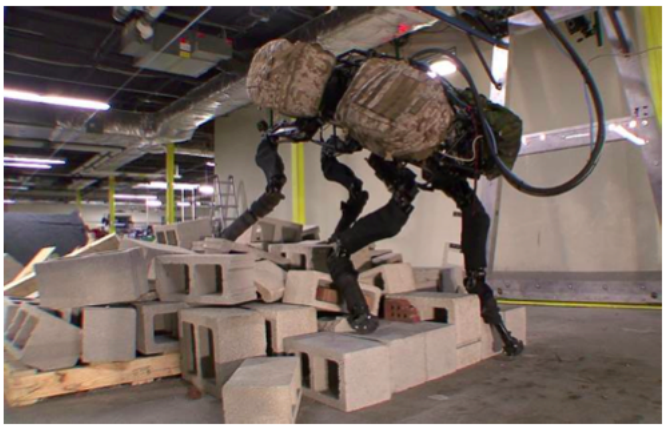

Figure 12: Large semi-autonomous quadruped robot 'BigDog', circa 2008