

Efficient State Spaces and Policy Transfer for Robot Navigation

Kagan Tumer
Oregon State University, USA

Matt Knudson
Carnegie Mellon University, USA

Abstract

Autonomous mobile robots are critical in many real world applications, ranging from planetary exploration to search and rescue. Indeed, the application domains using such robots are steadily increasing, and as the robots get smaller, navigation policies that operate with reduced computational and memory requirements become critical. In addition, as tasks become more complex, directly developing (in this case evolving) navigation policies for such robots becomes slow at best and impossible at worst.

In this paper, we first revisit a state/action representation that allows robots to learn good navigation policies, but also allows them to transfer the policy to new and more complex situations. In particular, we show how the evolved policies can transfer to situations with: (i) new tasks (different obstacle and target configurations and densities); (ii) new sets of sensors (different resolution); and (iii) new sensor/actuation noise levels (failures or changes in sensing and/or actuation fidelity). Our results show that in all three cases, policies evolved in simple environments and transferred to more complex situations outperform policies directly evolved in the complex situation both in terms of overall performance (up to 30%) and convergence speed (up to 90%).

1 Introduction

The list of applications to which autonomous robots apply as solutions is growing rapidly. Many applications including planetary exploration and unmanned flight (Helmick & Roumeliotis, 2006) contain complex tasks requiring increased autonomy. Exploration type domains and applications of this type require that autonomous navigation play a vital role. As a result of this and the ever growing task complexity has resulted in navigation algorithms becoming very domain specific with a high demand for resources (Thrun et al., 2005; Thrun & Montemerlo, 2006).

For autonomous navigation with mobile robots to be successful, it needs the ability to operate with partial information and in dynamic environments. These combine to produce what can be heavily stochastic observations and action outcomes. One solution is to try to produce as much domain knowledge as possible, typically at the cost of high resolution sensing and other resources (Thrun & Montemerlo, 2006). Producing such a high level of domain knowledge is useful to producing a large amount of detailed past experience, ideal in such model-based approaches. A more general approach, that requires much simpler modeling, is to provide or develop a direct mapping between sensory inputs to actions, avoiding the need for detailed domain knowledge (Cummins & Newman, 2007). Model-free methods have proven successful in applications where past experience is limited (Knudson & Tumer, 2011).

One of the most popular ways to mitigate the complex nature of robotic tasks is to focus on simple tasks first and then transfer that knowledge into more complex tasks. Such an approach is termed *transfer learning* and has proven successful in many applications (Taylor & Stone, 2009; Cao et al., 2010; Buffet et al., 2007). The primary reason for the success of transfer learning is in the decomposition of a task into either stages (Taylor et al., 2011) or into sub-tasks the knowledge of which is combined to accomplish the more complex task (Taylor et al., 2008). Typically both the source and target are modeled using Markov Decision Processes (MDPs), and therefore the development of how states and actions transfer between models is the primary challenge (Taylor & Stone, 2009). However, it has also been shown that the representation of the task itself can also be a powerful tool, especially if that representation is the same across tasks (Verbancsics & Stanley, 2010; Knudson & Tumer, 2011).

In this work, we explore a neuro-evolutionary approach where policies are incrementally evolved through tasks with increasing degrees of difficulty (Fernandez-Leon et al., 2008; Mouret & Doncieux, 2008). Neuro-evolutionary approaches fall in the *policy search* category where the aim is to search directly across policies. This search, interspersed with policy improvements through selection, allows for the discovery of new and robust navigation strategies. Neuro-evolutionary approaches have been successfully applied both to benchmark problems (Chandra et al., 2010) and to real world control problems (Agogino & Tumer, 2008; Grasmann et al., 2008). Often the policy (e.g. an artificial neural network) is simple in construction and therefore is inexpensive to modify and evaluate in practice, providing resource cost benefits as well (Knudson & Tumer, 2011).

In this paper we present state and action spaces for use with mobile robots that do not maintain any information regarding the environment surrounding them (Knudson & Tumer, 2011). Instead, a policy is incrementally evolved that can instantaneously examine the visible world surrounding the robot and make a single decision on what to do for the next time step. We examine various situations affecting the reactionary nature of the state and action spaces via incremental evolution to assert that such a representation facilitates a broad range of capabilities.

The key contribution of this paper is to couple a state/space representation with a reactive algorithm and use transfer learning to extend simple behavior into increasingly complex domains. Section 2 provides a short background on related work. Section 3 describes the robot navigation problem, the state-space representation and navigation algorithms as well as the simulated capabilities of the robot used in this work. Section 4 presents the three transfer learning scenarios, as well as the results and analysis of the robot performance. Finally, Section 5 provides a discussion of the work, implementation issues, and directions for further research.

2 Related Work

Devising successful robot navigation policies for continuous and complex environments is an important challenge, particularly when the robot has limited observational capabilities (Desouza & Kak, 2002; Santos et al., 2010). One solution to this problem is to explicitly model the robot and the environment in which it operates to determine the limits of functionality and interaction (Thrun et al., 2005). In general, planning techniques utilize such models as well, and have also proven successful for navigation in unknown environments (Koenig et al., 2003; Santos et al., 2010). Heuristic search based planning techniques include the successful algorithm Dynamic A* (Mudgal et al., 2004), which allows for replanning in the face of new situations.

Often it is found that creating detailed, complete, or even passable models of an environment can be very difficult, if not impossible. Typically as the environment becomes more dynamic, producing more

stochasticity, it becomes more difficult to provide model accuracy to any significant degree. Navigation policies that can react to inaccurate, incomplete, or missing environment models then provide several advantages (Thrun et al., 2005). Approaches that use learning allow the navigation policy to adjust internal representations of the models, or learn them from scratch, have proven quite successful as well (Ross et al., 2008; Wu et al., 2008). As with the work in this paper, evolutionary algorithms have also proven successful for robot navigation where environments are unknown and stochastic (Hasircioglu et al., 2008).

For more complex robot behavior, it is often difficult for the robot to learn the task in one step, particularly for neural network policies (Seipone & Bullinaria, 2005; Liu & Lampinen, 2005). Incrementally improving a simple policy, or transferring the knowledge from simple to complex domains, is therefore, a critical tool (Defaweux et al., 2005). Incremental evolution has been successful in many domains, including robot gait development (Garder & Høvin, 2006) and in multiagent situations with simple ant-like agents (Cazangi et al., 2005).

3 Robot Navigation

Robot navigation is a critical first step in many key mobile robot applications (Salichs & Moreno, 2000). Most current robot navigation algorithms are computation intensive (Thrun & Montemerlo, 2006). In cases where robots have limited resources, it is critical to focus on a simple mapping between incoming information and a navigation action. Regardless of resources, the robot must have the ability to choose safe and efficient paths through an environment to reach a specific destination. This includes the ability to avoid obstacles and maximize robot speed, while maintaining a level of robustness to inaccuracies and noise in sensor and actuator signals.

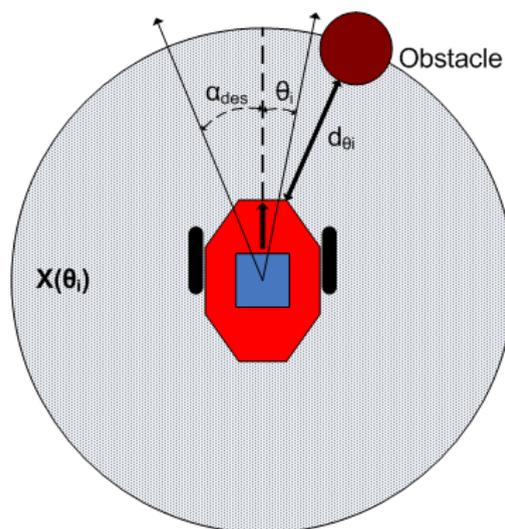


Figure 1: Graphical depiction of state and action spaces (Knudson & Tumer, 2011).

3.1 State and Action Spaces

In order to remain as close to a real platform as possible, the robot simulated in this work has limited sensing capabilities (discussed further in Section 3.4) as well as intentionally non-deterministic outcomes of actions taken. For example, at the next time-step, the robot may not be pointing exactly in the direction chosen by its navigation policy. This is because physical robots in general provide non-deterministic outcomes of actions requested. By and large this is due to the unpredictable environment in which the robot operates, but also to hardware limitations. In the mobile robot case, the interface between actuating devices (i.e., motors) and articulation devices (i.e., gears and wheels) as well as the interface between these devices and the environment (i.e., terrain) can be unpredictable via friction energy loss and slipping.

Figure 1 shows the state space representation that encodes as much information as possible from the sensors, as simply as possible, using two state variables. The first state variable represents object distance by simulating ultra-sonic type sensors (Knudson & Tumer, 2011; Hans Jacob S. Feder, 1999), such that for each vehicle relative potential path heading angle θ_i , a distance to the nearest object d_{θ} is provided. This represents a potential obstacle, such as a wall or rock. The second state variable represents the *destination heading*, where the difference between the potential path heading θ_i and the vehicle relative destination heading α_{des} is provided. This indicates how significant of a correction is required for the robot to point directly toward the destination. The lone action variable is the probability of success for each of the 360 possible paths, described further in the following section.

Such abstractions can be applied to the action space as well, to reduce the impact of non-determinism while still allowing the policy to operate on the robot task. In our case, we introduced the concept of *path quality*. So, rather than picking the robot direction and speed, the “action” that the policy takes is to assign a quality $X(\theta_i)$ to all potential paths θ_i . How this assignment is made varies based on the algorithm, but how the quality is used to choose the next robot direction and speed is the same.

3.2 Probabilistic Navigation

The first robot navigation algorithm we employ is a probabilistic algorithm that selects paths based on the two states discussed above. Given a “desired location” (e.g., through a goal selection algorithm or waypoint list) the path selector needs to compute the speed and heading corrections to reach that location. In this approach, each path is assigned a probability of leading to the desired location based on the current heading of the robot, provided by inertial sensing. Then, each path is assigned a probability of safety, which is based on the presence of observable obstacles along a given path, provided by ultrasonic sensing. The product of these two probabilities provides a prediction of success for a given path and is labeled the *path quality*. The path with greatest quality (greatest “probability of success”) is then chosen as the, potentially new, desired robot heading.

Figure 2 shows the path qualities for a simple domain. In the absence of prior information, this is a likelihood based approach, where each path is evaluated solely based on collected data. However, including prior information and updating the path quality posterior probabilities would provide a true Bayesian navigation method. The heading with the highest path quality is selected, and the speed is linearly scaled based on that quality (e.g., 90% or greater path quality equates to maximum speed, whereas a 50% path quality results in half speed).

The distributions that define the probability of success, or path quality, are exponentials tuned by hand through observations and known robot capabilities. One benefit to the abstract nature of the state and action spaces is that few iterations are needed to reach stable behavior with realistic performance. Another benefit specific to the action space selection is that in assigning a path quality to each path at every time step, a smoothing effect occurs in path selection. This is particularly useful when destinations suddenly change, or

when obstacles come into or go out of the sensor field of view. Algorithms that generate complex multi-step paths to follow often suffer from the need to completely recalculate in such situations.

3.3 Neuro-evolution

The state/action structure of the probabilistic navigation control presented in Section 3.2 is well suited to path selection. It is simple, easy to predict and implementable with a mobile robot with limited sensing capabilities. To capture those benefits while injecting the benefits of adaptability into navigation control, we use the same representation in the neuro-evolutionary navigation algorithm. Specifically, one state variable represents the angular distance of a potential path from the desired heading, and one represents the distance to the nearest impassable object in that path. The action space then contains a single variable representing the quality of the path given that information.

In this work, the baseline network structure created is a two layer, sigmoid activated, artificial neural network. This network is run at each time step, for each potential path, generating a path quality function in a similar fashion to that presented in Figure 2. The difference is in the replacement of static predefined probability distributions by an adaptive artificial neural network. Indeed, Figure 3 shows a sample policy. Note the more aggressive (higher speed) selections in close distances to the destination. The network is trained using an evolutionary search algorithm (Agogino & Tumer, 2008; Moriarty & Miikkulainen, 2002; Tumer & Agogino, 2005). The specifics of the neuro-evolutionary navigation algorithm is shown in Figure 4 which displays the ranking and mutation steps. The definitions for the variables and functions located in the algorithm are as follows:

T : Indexes episodes

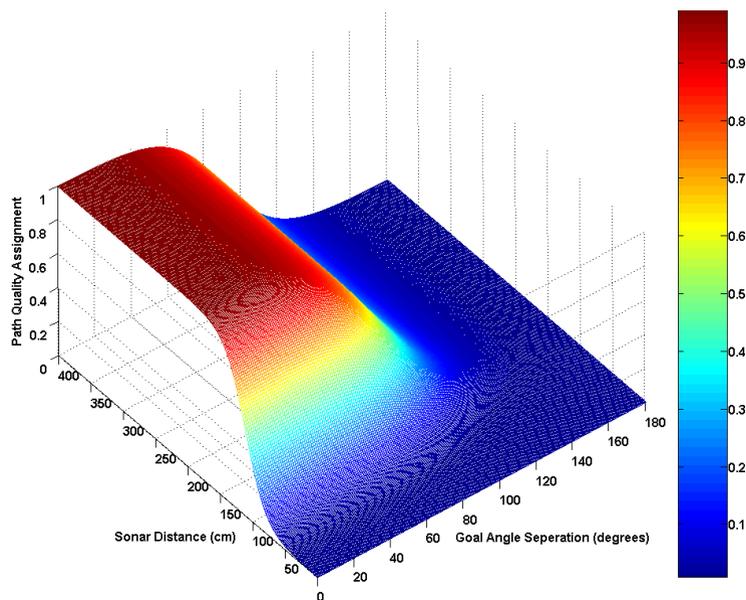


Figure 2: Path quality assignments for the entire range of potential state inputs to the probabilistic algorithm (Knudson & Tumer, 2011). Highest path quality is on paths that point to the destination and have largest distance to nearest obstacle (upper left area).

t : Indexes time-steps within each episode

θ_i : Angle of potential path i

N : Indexes networks with appropriate subscripts

N' : Mutated network for use in control of current episode

$X(\theta_i)$: Path quality assigned to each potential path

α_u : Chosen vehicle relative robot angle for next time-step

$F(X(\alpha_u))$: Linear function mapping path quality of path chosen to robot speed

V_u : Chosen robot speed

In this domain, mutation (Step 2) involves adding a randomly generated number to every weight within the network, which in this work is done by sampling from a random Cauchy distribution (Agogino & Tumer, 2008; Knudson & Tumer, 2011) where the samples are limited to the continuous range [-10.0,10.0]. The evaluation function for the currently operating network N' at the end of an episode is given by:

$$R = \eta_1 f_1 + \eta_2 f_2 - \tau_c \quad (1)$$

where τ_c is the number of time ticks the robot spends recovering from a collision with an obstacle or boundary wall and η are tuning constants, set to 1.0 and 10.0 respectively. The two feature functions f_1 and f_2 are given by:

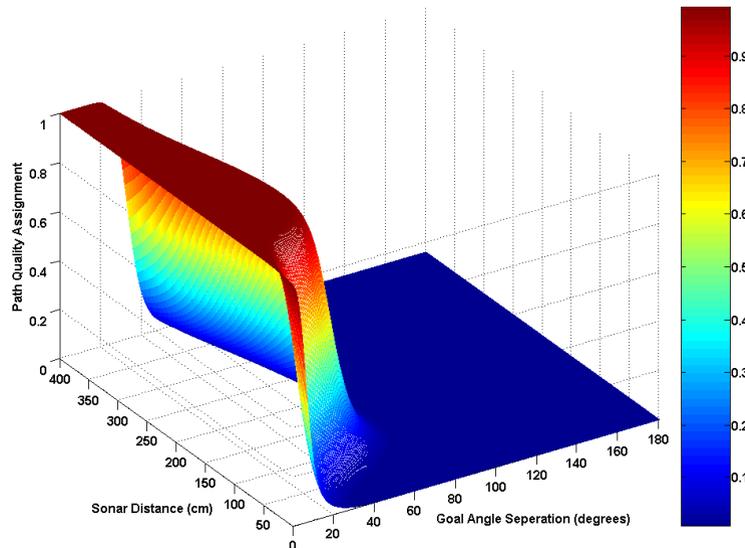


Figure 3: Network outputs for entire range of inputs in the neuro-evolutionary algorithm (Knudson & Tumer, 2011). The figure is produced after the learning process has converged within a difficult navigation situation (Section 4).

```

Initialize  $N$  networks at  $T = 0$ 
For  $T < T_{max}$  Loop:
  1. Pick a random network  $N_i$  from population
      With probability  $\epsilon$ :  $N_{current} \leftarrow N_i$ 
      With probability  $1 - \epsilon$ :  $N_{current} \leftarrow N_{best}$ 
  2. Mutate  $N_{current}$  to produce  $N'$ 
  3. Control robot with  $N'$  for next episode
      For  $t < t_{epi}$  Loop:
        3.1 For  $\theta_i \leq 360$  Loop:
          Run  $N'$  to produce  $X(\theta_i)$ 
        3.2  $\alpha_u \leftarrow \operatorname{argmax} X(\theta_i)$ 
        3.3  $V_u \leftarrow F(X(\alpha_u))$ 
  4. Rank  $N'$  based on performance
      (objective function)
  5. Replace  $N_{worst}$  with  $N'$ 

```

Figure 4: Evolutionary Algorithm (Agogino & Tumer, 2008; Knudson & Tumer, 2011): An ϵ -greedy evolutionary algorithm to determine the weights of the neural networks. See text body for definitions.

$$f_1 = \begin{cases} \frac{l}{\beta_l} & \text{if } l \leq \beta_l \\ \frac{\beta_l}{l} & \text{otherwise} \end{cases} \quad (2)$$

$$f_2 = \begin{cases} \frac{\tau_l}{\beta_\tau} & \text{if } \tau_l \leq \beta_\tau \\ \frac{\beta_\tau}{\tau_l} & \text{otherwise} \end{cases} \quad (3)$$

where l and β_l are the actual total path length of the robot during the episode, and β_l is the best possible (straight line from start to destination)¹. Similarly, τ_l is the *time* spent to execute that path, where β_τ is the shortest possible time.

This evaluation function captures the important features of how the control policy performed during an episode. Namely, it requires that the policy navigate the robot to the destination with the shortest possible path, as quickly as possible (promoting high path quality assignments to maximize robot speed), and of course without hitting anything. The inverse relationship of the feature functions was chosen because of the inherent balance between path length and time consumed. This structure captures all the inherent interactions in the policy and minimizes the number of special cases needed in the calculation.

¹If time expires and the robot has not driven anywhere, l is less than β_l .



Figure 5: Robot platforms on which the physics of the simulator are based. The robot on the left is much smaller and has much lower sensor resolution.

3.4 Robot Capabilities

The overall goal of this research is additionally pinned down by the desire to produce successful navigation behavior with as little resources as possible. We’ve discussed how this is partially achieved through the state and action space selection in Section 3.1. These spaces require only that general information about the robot motion and its environment are available, and are applicable to all mobile robot platforms. To keep with that theme, robot capabilities were selected to be as common as possible.

The physics-based simulations are modeled on two different types of robots (Oregon State University’s TekBot, and Mobile Robot’s Pioneer 3) each of which has different sensing and actuation capabilities, described in detail in (Smit et al., 2004; Knudson & Tumer, 2011) and shown in Figures 5. With sensing, the robot must be able to determine its orientation and some details about the surrounding environment. To provide this information, our platforms have inertial sensors that integrate the robot position and heading based on accelerations. To examine the environment, we provide ultrasonic sensors that provide a 360 degree view of the environment in varying increments as the experiments required. For actuation, the robot platforms have two motorized wheels and a caster. As a result the robot can move in any direction, at any turning radius. Possible accelerations and maximum speeds are based on the physical platform.

The real robot platforms also have several layers of non-determinism, both in sensing and in actuation. For example, the smaller platform is less susceptible to slipping either during acceleration or turning, so action outcomes tend to be more deterministic. The inertial and ultrasonic sensing suffers however because of large accelerations and small packaging respectively. The larger platform has lower accelerations and more accurate ultrasonics, but is much more susceptible to slipping, especially during turning. Such sensor and actuator “noise” is also modeled in our simulations, and modified as experiments required.

4 Experimental Results

In this section, we present simulation results analyzing the selected state and action representations and their impact on navigation policy selection. Incremental neuro-evolution is notably different from the traditional concept of transfer learning, because it is entirely a search through policies rather than reinforced single

policy development. As a consequence, the mechanism of the “transfer” is different: Rather than requiring an explicit mapping between tasks as may be required to transfer a Q-table (Taylor & Stone, 2009), policies can be transferred by simply seeding the population of the policy search by specific solutions. To that end, we specifically investigate the transfer of learned behavior from a simple task to three more complex situations:

1. **New Tasks:** The initial population is evolved on two simple tasks; a) pure navigation to a destination, and b) avoiding a single obstacle. The best network is then used to seed the initial population in an environment dense with many obstacles, a situation that confounds direct learning as the robot simply cannot find policies that can navigate the difficult environment (Figure 7).
2. **New Sensors:** The initial population is evolved with limited obstacle sensing resolution, and then the best individual is used to seed a population for an environment with much greater resolution (Figure 8). Conversely, we explore the situation where the initial population is evolved with near perfect resolution, and then the best individual is used to seed a population with lower resolution (Figure 9).
3. **New Levels of Sensor/Actuator Noise:** The initial population evolves when there is no sensor or actuator noise, and the best individual is selected to seed a population where there is significant noise in both sensors and actuators (Figure 10). Conversely, the initial population is evolved in a noisy environment and the best policy is used to seed a population for a low noise environment (Figure 11).

There are several ways to perform incremental evolution. For our case, when the population fitness has converged, the best network is chosen. The initial population for the new situation is then “seeded” with this network by creating mutants with the same procedure discussed in Section 3.3. The network remains in the population, but a random network is chosen for the first episode. The algorithm then progresses as usual, and ranking of the networks proceeds with the same equation, shown in Equation 1.

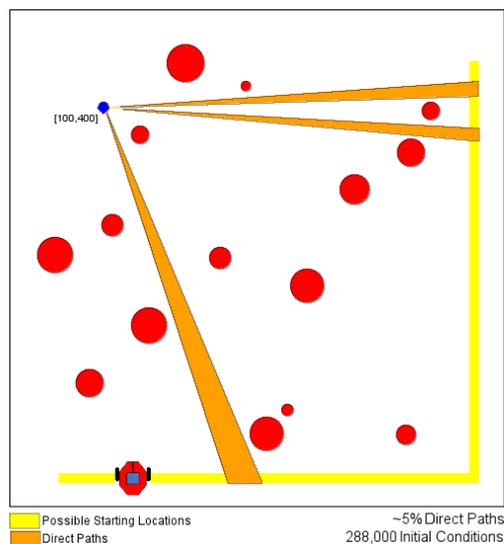


Figure 6: A sample environment dense with obstacles. The majority of experiments were performed in such a situation. The yellow areas are the possible starting locations of the robot at each episode, with a random heading. The orange “rays” shows the limited possibilities for direct paths without the need to avoid obstacles.

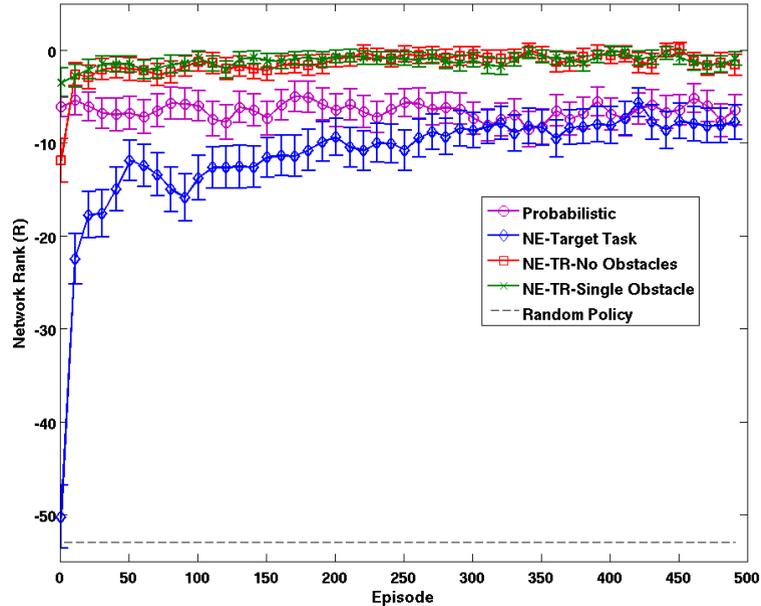


Figure 7: The results of evolution are shown for two source tasks; a) an open environment, and b) an environment with a single obstacle. The target task is an environment dense with obstacles. The probabilistic navigation and direct policy evolution on the target task are shown for comparison.

4.1 Transfer of Policies to New Tasks

In the first set of the experiments we explore the transfer of policies from simple to complex tasks. We take the best network from a population evolved in an open environment, where the robot is tasked simply with navigating to some destination. This policy is then used to evolve a new population in an environment dense with obstacles (Figure 6 shows a sample case of an environment dense with obstacles).

Figure 7 shows the results of transferring policies from two situations (initial task with no obstacles and initial task with one obstacle), along with results from the probabilistic navigation and a policy evolved directly for the dense obstacle environment. This process tests whether the agent can learn one portion of the state space first (path to destination) then learn to deal with obstacles. While seeding the population in an open environment (no obstacles) starts below -10 , below the probabilistic algorithm, it still begins well above the random policy and climbs quickly to significantly exceed both, converging near zero (meaning the robot spent less than $1.1s$ recovering from collisions in the entire episode). This is a significant result, indicating that the transfer of policies in this domain not only improves learning speed, but significantly improves overall performance. The population seeded from an environment with a single obstacle converges to a statistically similar performance, but does begin at higher performance, indicating that the policy did gain the ability to avoid obstacles in a specific situation.

4.2 Transfer of Policies to New Sensors

In the second set of the experiments we explore the transfer of policies in two situations, each investigating different aspects of the state/action representation. First, we transfer policies from a configuration where the agent has limited sensing (8 sectors) to a configuration where the agent has significantly better sensing

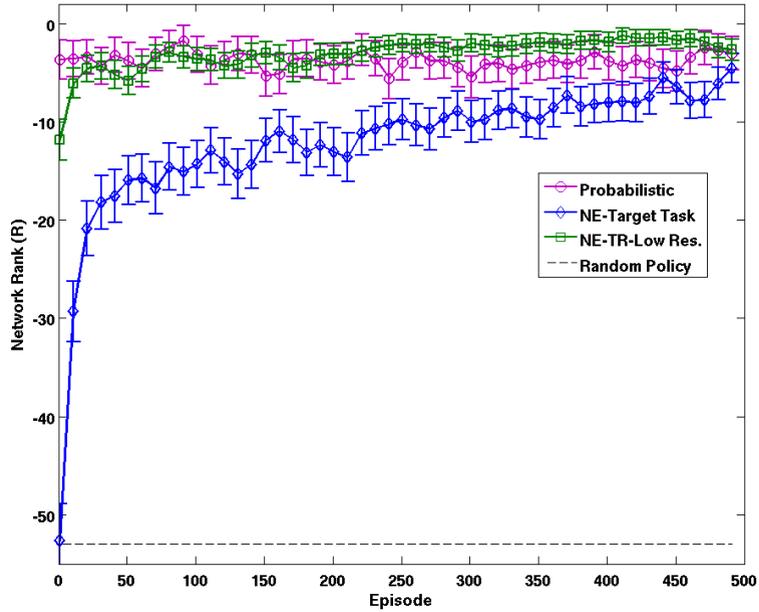


Figure 8: The results of evolution are shown for starting with 8 sectors and moving to 32. The probabilistic navigation and random initial population algorithms are shown for comparison.

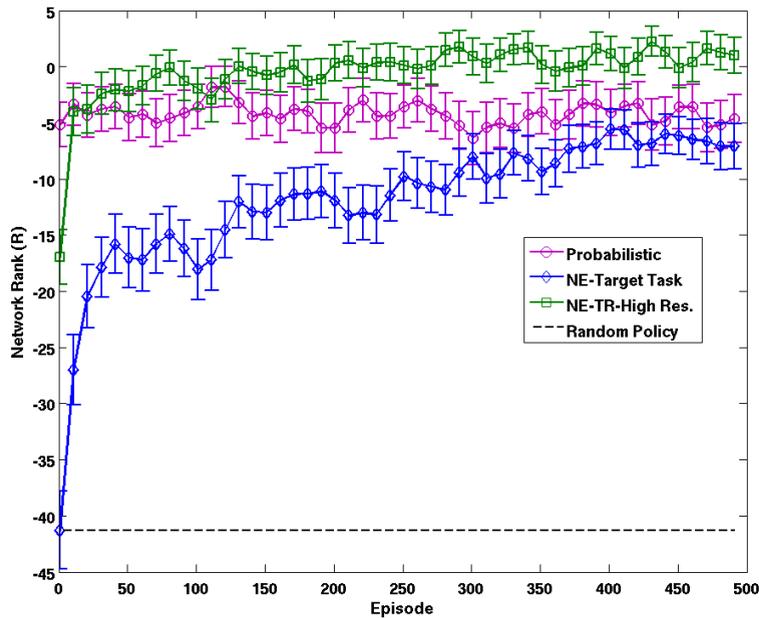


Figure 9: The results of evolution are shown for starting with 180 sectors and moving to 16. The probabilistic navigation and random initial population algorithms are shown for comparison.

(32 sectors). Then we transfer policies from a configuration where the agents have near perfect observational capability (180 sectors) to a configuration with significantly reduced resolution (16 sectors). These experiments serve to show how changes to the robot capabilities can be quickly incorporated into navigation policies, and are performed in an environment dense with obstacles, similar to the environment in the previous experiments.

Also, it is important to note (from Section 3.3) that the change in sensing resolution does not change the network structure. The network is still run for every potential path, and as the number of sectors increases or decreases, the higher (or lower) the resolution becomes. For example, increasing from 8 to 32 sectors gives the network more specific information, meaning the obstacle state variable is richer with specific information.

Figures 8 and 9 show the results of transferring policies when the robot sensing capabilities are changed. We again can see that learning speed is dramatically improved for both situations and performance convergence above evolving the policy from scratch. Though it also outperforms the probabilistic algorithm, that difference is small (though statistically significant). This reduced performance is due primarily to the lack of state space exposure in the 8 sector learning. The probabilistic algorithm will automatically exploit any additional information, naturally including it in the decision process. The neuro-evolutionary algorithm however has to discover specific relationships between states and actions, and therefore struggles to embrace the additional information, as it originally had a very coarse view of the environment.

Conversely, in the second experiment, the sensing capabilities change more dramatically, dropping from 180 to 16 sectors. This experiment not only shows the potential for policy transfer, but also the robustness of the state/space representation. Indeed, in most cases, navigation algorithms are designed to exploit all possible sensing information, and suffer when that information is suddenly lost. These results show that the policy transfer outperforms both the probabilistic algorithm and policies evolved from scratch. This is a strong robustness result, showing the effectiveness of the state representation. While higher levels of information are beneficial and exploited, a dramatic change in the amount of that information does not significantly impact the algorithms performance, and it can easily adjust to the new situation.

4.3 Transfer of Policies to New Noise Levels

In the final set of the experiments we explore the transfer of policies in the presence and absence of sensor/actuation noise. First, the agent learns when no sensor/actuator signal noise is present, then is transferred to a configuration with noisy sensors/actuators (5%). Second, the agent learns in a very noisy configuration (10%) and then continues in a noiseless setting. The first simulates sensor failures in the field, the second simulates whether noise can be used as a training (exploration) tool. These experiments show whether the evolved navigation policy is robust to sudden increases in noise (and can further adapt to handle it) and whether learning in a noisy environment damages the policy, causing erratic behavior in normal conditions.

Figures 10 and 11 show that transferring policies is beneficial both in learning speed and converged performance. Interestingly, Figure 10 shows a significantly lower starting point (around -30) of the transferred policy when transferred from a noiseless situation. This points to the potentially significant impact noise can have on policies developed in simulation and transferred to real-world situations. However, clearly knowledge is still retained about the navigation task from the noiseless situation, because the algorithm quickly recovers and easily outperforms the probabilistic navigation algorithm and learning the target task directly.

There is also a significant drop at the start for the policy that moves from a very noisy situation to a noiseless one, shown in Figure 11. However, the population again quickly recovers. Both situations significantly outperform the probabilistic algorithm, as well as the algorithm starting from a random population, converging near zero. These results show that the policy is indeed robust to changes in sensor and actuator

quality, and that good behavior is maintained even if the policy is learned in a poor environment.

5 Conclusions and Future Work

Autonomous navigation for mobile robots is particularly challenging when information about the operating environment is not maintained. In these situations, decisions on safe and direct paths to a destination must be made quickly on only information immediately available. As such, the information gathered about the environment must be represented in a way that navigation policies can operate on it simply to take effective actions. The work in this paper presents such a state / action representation that is specific enough to be effective, but abstract enough to be applicable to other situations, and adaptable to changes in robot capabilities. Furthermore, this representation allows for the effective transfer of policies from simple to complex tasks as well as from one sensor configuration to another.

The results of the policy transfer show that the robots can develop robust navigation policies for a variety of environmental conditions. The policies also have the benefit of being purely reactive, simply consisting of a mapping from sensor inputs to robot actions. This simplicity is key to the results, in that the robot performance and successful policy transfer happen because of the simple representation and mapping, not in spite of it. Additionally, these results show that the policy can be developed on robots with differing sensing capabilities, and still be applicable to another where sensing either provides more or less information. Finally, the experimental results show that the policy is robust to sensor and actuator signal noise, whether it is a result of failures, or of simulation environment errors. These conclusions are a key step in proving that not only can neural networks inherently capture key aspects of the mobile robot navigation task, but also that, given a proper representation, navigation performance can be improved by training on simple tasks

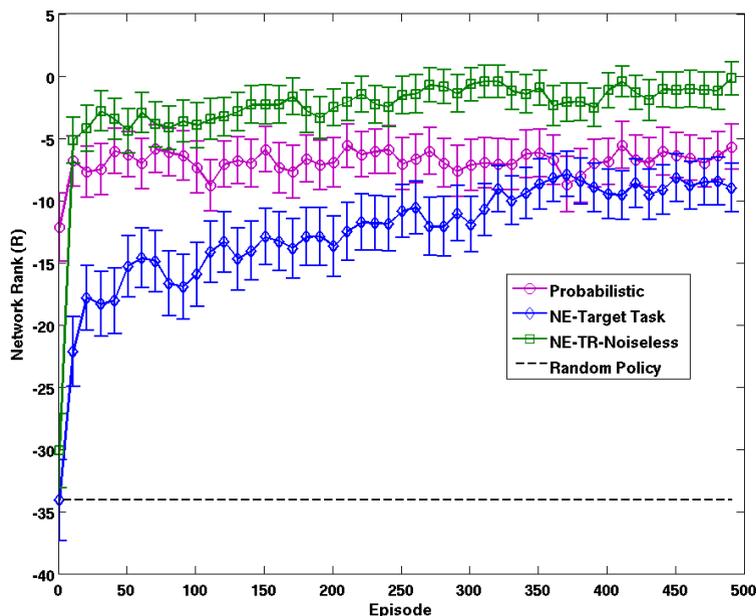


Figure 10: The results of evolution are shown for evolving with no noise and injecting 5%. The probabilistic navigation and random initial population algorithms are shown for comparison.

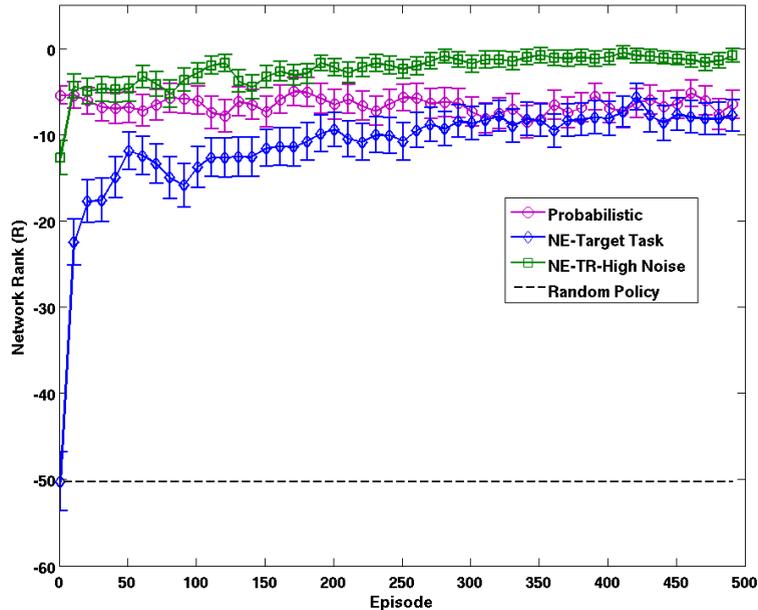


Figure 11: The results of evolution are shown for evolving with 10% and then removing all noise. The probabilistic navigation and random initial population algorithms are shown for comparison.

first.

We are currently investigating several avenues for further research. While the state and action representations as well as the neuro-evolutionary algorithm have been shown to be successful in hardware for simple tasks (Knudson & Tumer, 2011), our current work is to expand this work to more complex navigation tasks in real-world experiments. Indeed, the transfer of policies from one sensing/actuation situation to another or from one noise level to another is an ideal tool to have when transferring policies from simulation to hardware. In addition, we are investigating modification to the policy to include information concerning earlier navigation decisions (for example adding a third input to the artificial neural network) that may help select better paths. Due to the simple state/action representation and policy structure, this does not burden the algorithm much computationally, but is providing significant improvements in navigation in our preliminary results.

References

- Agogino, A. K. & Tumer, K. (2008). Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2), 257–288.
- Buffet, O., Dutech, A., & Charpillet, F. (2007). Shaping multi-agent systems with gradient reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 15, 197–220.
- Cao, B., Pan, S. J., Zhang, Y., Yeung, D.-Y., & Yang, Q. (2010). Adaptive transfer learning. In *AAAI Conference on Artificial Intelligence*.

- Cazangi, R. R., Von Zuben, F. J., & Figueiredo, M. F. (2005). Autonomous navigation system applied to collective robotics with ant-inspired communication. In *Proceedings of the 2005 conference on Genetic and evolutionary computation* (pp. 121–128).
- Chandra, R., Freat, M., & Zhang, M. (2010). An encoding scheme for cooperative coevolutionary neural networks. In *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science* (pp. 253–262). Springer.
- Cummins, M. & Newman, P. (2007). Probabilistic appearance based navigation and loop closing. In *Robotics and Automation, IEEE International Conference on*.
- Defaweux, A., Lenaerts, T., van Hemert, J., & Parent, J. (2005). Transition models as an incremental approach for problem solving in evolutionary algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation* (pp. 599–606).
- Desouza, G. & Kak, A. (2002). Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2), 237–267.
- Fernandez-Leon, J. A., Acosta, G. G., & Mayosky, M. A. (2008). Behavioral control through evolutionary neurocontrollers for autonomous mobile robot navigation. *Robotics and Autonomous Systems*, 57, 411–419.
- Garder, L. M. & Høvin, M. E. (2006). Robot gaits evolved by combining genetic algorithms and binary hill climbing. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 1165–1170).
- Grasemann, U., Stronger, D., & Stone, P. (2008). A neural network-based approach to robot motion control. In U. Visser, F. Ribeiro, T. Ohashi, & F. Dellaert (Eds.), *RoboCup-2007: Robot Soccer World Cup XI*. Berlin: Springer Verlag.
- Hans Jacob S. Feder, John J Leonard, C. M. S. (1999). Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18, 650–668.
- Hasircioglu, I., Topcuoglu, H. R., & Ermis, M. (2008). 3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 1499–1506).
- Helmick, D. M. & Roumeliotis, S. I. (2006). Slip-compensated path following for planetary exploration rovers. *Advanced Robotics*, 20, 1257–1280.
- Knudson, M. & Tumer, K. (2011). Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, 59, 410–420.
- Koenig, S., Tovey, C., & Smirnov, Y. (2003). Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147(1-2), 253–279.
- Liu, J. & Lampinen, J. (2005). A differential evolution based incremental training method for RBF networks. In *Proceedings of the 2005 conference on Genetic and Evolutionary Computation* (pp. 881–888).
- Moriarty, D. & Miikkulainen, R. (2002). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5, 373–399.

- Mouret, J.-B. & Doncieux, S. (2008). Incremental evolution of animats' behaviors as a multi-objective optimization. In *Proceedings of the 10th international conference on Simulation of Adaptive Behavior* (pp. 210–219).
- Mudgal, A., Tovey, C., & Koenig, S. (2004). Analysis of greedy robot-navigation methods. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (AMAI)*.
- Ross, S., Chaib-draa, B., & Pineau, J. (2008). Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *Robotics and Automation (ICRA 2008)*.
- Salichs, M. A. & Moreno, L. (2000). Navigation of mobile robots: open questions. *Robotica*, 18(3), 227–234.
- Santos, A., Tarrataca, L., & Cardoso, J. (2010). The feasibility of navigation algorithms on smartphones using j2me. *Mobile Networks and Applications*, 15, 819–830. 10.1007/s11036-010-0236-8.
- Seipone, T. & Bullinaria, J. (2005). Evolving improved incremental learning schemes for neural network systems. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3 (pp. 2002–2009).
- Smit, A., Heer, D., Traylor, R., & Fiez, T. S. (2004). A custom microcontroller system used as a platform for learning™ in ece. In *ASEE Annual Conference Proceedings* (pp. 2733–2740).
- Taylor, M. E., Kuhlmann, G., & Stone, P. (2008). Autonomous transfer for reinforcement learning. In *Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 283–290).
- Taylor, M. E. & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10, 1633–1685.
- Taylor, M. E., Suay, H. B., & Chernova, S. (2011). Integrating reinforcement learning with human demonstrations of varying ability. In *Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 617–624).
- Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. Cambridge, MA: The MIT Press.
- Thrun, S. & Montemerlo, M. (2006). Stanley: The robot that won the darpa grand challenge. *Robotic Systems*, 23, 661–692.
- Tumer, K. & Agogino, A. (2005). Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *The Genetic and Evolutionary Computation Conference* Washington, DC.
- Verbancsics, P. & Stanley, K. (2010). Evolving static representations for task transfer. *Journal of Machine Learning Research*, 11, 1737–1769.
- Wu, H.-Y., Liu, S.-H., & Liu, J. (2008). A new navigation method based on reinforcement learning and rough sets. In *Machine Learning and Cybernetics*.