Use your Illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge

Richard Vaughan and Mauricio Zuluaga

Autonomy Lab, School of Computing Science,
Simon Fraser University, Burnaby, BC, Canada
vaughan,mzuluaga@sfu.ca

Abstract. We present a practical application of sensorimotor self-simulation for a mobile robot. Using its self-simulation, the robot can reason about its ability to perform tasks, despite having no model of many of its internal processes and thus no way to create an *a priori* configuration space in which to search. We suggest that this in-the-head rehearsal of tasks is particularly useful when the tasks carry a high risk of robot "death", as it provides a source of negative feedback in perfect safety. This approach is a useful complement to existing work using forward models for anticipatory behaviour. A minimal system is shown to be effective in simulation and real-world experiments. The virtues and limitations of the approach are discussed and future work suggested.

1 Introduction: Let your hypotheses die in your stead

To solve some problems, autonomous agents must plan ahead. One common problem that requires planning is finding an efficient route between a set of places in the world: the family of problems that includes the classical Traveling Salesman Problem. Finding efficient routes between places of interest can clearly be seen to be adaptive; for example a squirrel visiting nut caches or a female lion patrolling and freshening her territorial urine marking sites can save time and energy for other tasks if a good route is chosen.

Formally, planning is the process of finding continuous trajectories through the agent's configuration space between the start and goal states. Configuration space is the set of all possible states that can be achieved by the system. Conventional planning techniques construct a model of the configuration space: either a static model such as a traversability map, or a generative model such as a production system. In either type of model, all possible state transitions are known.

Now suppose we have an intelligent agent, a robot, that contains some components of unknown function. By definition, the agent can not have an *a priori* model of its state evolution, due to the unknown internal states of the mysterious components, and their contribution to the system's output. Thus it can not construct an *a priori* model of configuration space. Such a system does not know what it can do, so how can it plan its future actions?

One solution is to learn a model of the mysterious systems by running them for a while and observing their inputs and outputs. Once a sufficiently good input/output mapping model of the mystery system is constructed, the model can be used to create a configuration space. The major flaw with this approach is that it suffers from the general learning problem of being critically dependent on the training data samples: the model can only be expected to be correct when it operates in situations similar to those seen during learning. This is a serious problem because there is a particular set of situations that are very important, and can never be experienced in training: the situations that cause the robot to be destroyed. Avoiding doom is a very important part of adaptive behaviour, and it can not be learned by negative experience.

This risky-learning problem can be solved by learning in simulation instead of the real world. It is often possible to construct a good *a priori* model of the outcome of a robot's motor actions in the world in terms of its new sensor readings: a *sensorimotor simulation*. It is commonplace for real adaptive systems to be usefully tested and trained in simulation. For example, commercial pilots spend a considerable part of their training time in flight simulators. In addition to routine flying, pilots can rehearse dangerous scenarios such as engine and instrument failures in complete safety. While we should be mindful of the advice of Brooks [1] about the limitations of world models, it is a fact that many robot control programs have been developed, learned, or evolved in simulation and successfully transferred to the real world with few or no changes, e.g.[2, 3].

Thus a robot with unmodeled mystery components could employ a sensorimotor simulation, observe its simulated actions and build a model of the mystery components. The resulting model can be used to construct a configuration space in which to plan.

But with the sensorimotor simulation in place, we have a simpler alternative. Why model the mystery components at all? Instead we can just execute candidate plans in the simulation and evaluate the outcomes. The mystery components just run as they would in the real world, remaining an unmodeled mystery, but we can still observe their effects on the world. The only requirement is that the sensorimotor simulation is a usefully good approximation of the robot's interactions with the real world.

Once we have taken this step, an appealing simplification presents itself. Why have an explicit model of *any* part of the robot's control system? If we have an explicit model M of robot control code C that is intended to implement process P, there is always a possibility of discrepancies between M, C, and P. An unfortunately common situation is when the code C contains bugs which prevent it from implementing the programmer's intention P correctly. Model M derived from P will probably not contain the same bugs, but may contain different bugs of its own. Plans computed in a configuration space from M may not be executable using C. But plans observed to work in a sensorimotor simulation in which C runs directly are guaranteed to reflect the actual function of C, bugs and all. Again, this is limited by the fidelity of the simulation, but *only* by the fidelity of the simulation.

Brooks' aphorism "the world is its own best model" [1] is well known. We propose the complementary idea: *the agent is its own best model*. More strongly, we can say that an agent's control software (or a provably correct transcoding of that software) is the *only* reliable model of itself. Sometimes simulating the agent's interaction with the world is relatively easy if you choose the appropriate level of abstraction [2].

This idea is appealing from an intuitive point of view. We can consciously observe the imagined results of our actions in the world, but we often do not have conscious models of what we can do, or how we do it. Few people could write down a correct dynamical systems or neurophysiological model of themselves riding a bicycle, but most riders can imagine themselves cycling down the street - even a street they have never seen. Similarly, most people can imagine swinging a golf club to strike a ball, even if they have never held a club. This could be explained by the existence of a generalized model of our motor interactions with the world, which can be used to rehearse novel situations. Intriguingly, there is evidence that athletes can improve their performance at motor tasks by performing such in-the-head rehearsal. The effect is more pronounced among individuals who already have expert skills, suggesting that the fidelity of the in-the-head model may be important in the successful transfer of imagined performance to the real world [4, 5].

In this way we relate sensorimotor self-simulation to the folk psychology notion of *imagination* as a mechanism to consider the outcomes of our behaviour without having to fully understand it, and without having to try everything out for real. The relationship between imagination and simulation is concisely expressed by Dawkins:

> "We all know, from the inside, what it is like to run a simulation of the world in our heads. We call it imagination and we use it all the time to steer our decisions in wise and prudent directions" [6].

This informal idea is consistent with *the emulation theory of representation*, developed by Rick Grush[7]. In this philosophical framework, phenomena external to an agent are represented internally by *processes* rather than the symbol systems of conventional cognitive science. This idea can be seen to underly this paper and much of the related work.

We can also consider an executable plan as a statement of truth about what the robot can do, and an untested plan as a hypothesis. In this sense we can view the robot as a Popperian scientist seeking truth by eliminating bad hypotheses through in-the-head experimentation:

> "The scientist can annihilate his theory by his critique, without perishing along with it. In science, we let our hypotheses die in our stead."[8]

Interpreting this famous statement rather more literally than originally intended, a robot can observe itself dying a thousand times in simulation as a result of bad plans, and thus eliminate those plans without risking its neck in

the real world. The robot need have no model of itself beyond the immediate sensory outcomes of its motor behaviour.

In practical terms, the proposed model offers a method for allowing high-level strategic or "cognitive" function to reason about the actions of other behaviour-producing systems without understanding how they work. This could be a useful engineering strategy for adding strategic layers on to existing behaviour-based systems. It also may hint at how evolutionarily recent cognitive systems could come to usefully exploit the functions of more ancient control systems in the brains of animals. This approach may also be a useful principle for robustness: if all system components are treated as if they are unknown, then the results of internal failures will be immediately apparent in the simulation without needing to update any internal model.

1.1 Related work

In the 1960s, Jewett placed lesions in the brains of cats which eliminated the inhibition of action commands during the REM sleep stages of dreaming, and (in Jewett's interpretation) allowed the cats to act out their dreams. The cats displayed such behaviours as fighting, grooming, exploring, running away and showing rage. Jewett concluded that dreams are rehearsal of vital survival activities that are likely to occur in real life [9]. In [10] rats were trained in a maze while awake, and could be observed rehearsing the maze experiments while sleeping.

Several authors have described systems in which sensorimotor forward models are able to predict how sensory information changes through sequences of motor commands [11–14]. In contrast, this paper shows how the outcomes of plans consisting of sequences of relatively high-level operations can be predicted. Our "motor commands" are `goto()` operations that abstract away a powerful and complex navigation system that is part of the agent, but completely unmodeled.

A framework that incorporates simulation to speed up learning in an evolutionary experiment is presented in [15]. Their proposed method combines into a single framework learning from reality and learning from simulation.

In this paper we present a practical application of sensorimotor self-simulation for a mobile robot. Using self-simulation, the robot can reason about its ability to perform tasks, despite having no model of many of its internal processes and thus no way to create an *a priori* configuration space. This approach is a useful complement to existing work using forward models for anticipatory behaviour. A minimal system is shown to be effective in simulation and real-world experiments. The virtues and limitations of the approach are discussed and future work suggested.

2 The application of imagination

2.1 Task

A robot lives in an office-like environment. At some moment it is asked to visit a set of places in the world. There is no preferred order of visits: the only requirement is that all of them are visited in the shortest possible time. This task

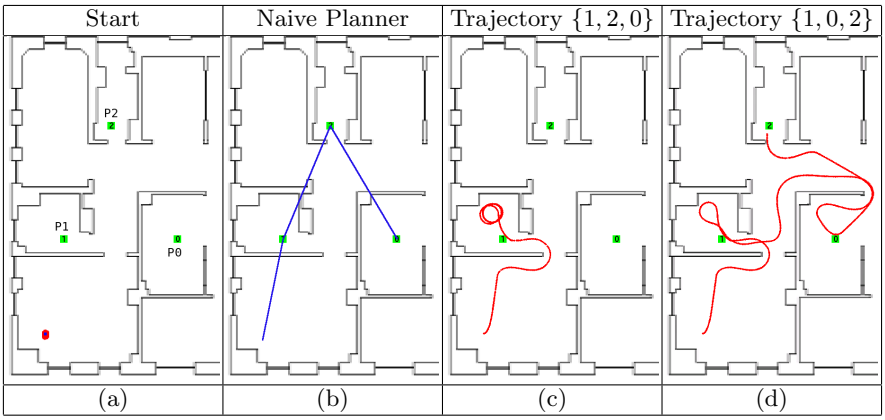


Fig. 1: (a) Starting state; (b) Naive shortest path, and two actual trajectories (c,d) of robots guided by VFH.

is a variation of the Travelling Salesman Problem where the traversal cost on the arcs connecting nodes is initially unknown.

Figure 1 shows such a scenario. Image (a) shows a map containing the robot start position and 3 goal locations. Image (b) shows the shortest path that visits all goal points. Image (c) shows the path taken by the robot using VFH navigation (described below) as it attempts to reach the locations in the order suggested by the shortest path: $\{1, 2, 0\}$. Due to the dynamics of VFH, the robot is fatally trapped. If the goal locations are submitted to the robot in the order $\{1, 0, 2\}$, the robot easily completes its task, reaching all locations by the path shown in Image (d). It is very difficult to characterise the configuration space created by VFH in a traversibility map. The system described below solves this problem.

2.2 Procedure

A typical lab robot R_0, operates in world W_0. At any time R_0 can run a sensorimotor simulator that models the interactions between a model robot R_i running controller C in model world W_i over time $S(R_i, W_i, t) \Rightarrow S(R_i, W_i, t + 1)|i > 0$. The real and simulated robots use the identical controller C. To decide which order to visit the goal locations in reality, R_0 uses its simulator to internally rehearse all possible visit orders. If there are n possible plans, R_0 runs n simulations $S(R_m, W_m, t)|m = 1, ..., n$.

The simulation results are evaluated and the plan that caused the fastest traversal in any simulation is selected as the plan for real-world execution.

A naive implementation will not scale well to large values of n, but the difficulty of scaling is not unique to our approach as the Traveling Salesman Problem is known to be NP complete, i.e. no scalable solution is known. A

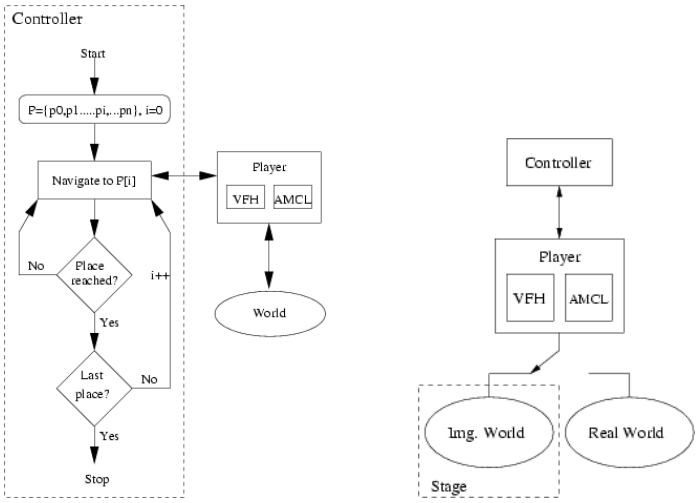


Fig. 2: Control Architecture (left) and Imagination Engine (right)

sophisticated implementation could prune the space of simulations to improve performance, and this method is ridiculously parallel (i.e. it parallelizes perfectly to n processors).

One useful optimization is immediately apparent. If the simulations are real-time, or a constant multiple of real time, we can evaluate plans trivially: run all the simulations in parallel, the first to finish must be the route that takes least time to traverse, and all the others can be aborted without loss.

2.3 Control Architecture

Our control system makes extensive use of Player, a well-known Free Software system for robot control over a network interface [16]. To greatly simplify the robot controller, we use the *VFH* obstacle-avoidance algorithm [17] and the Adaptive Monte Carlo Localization map-based localization algorithm [18] provided with Player. The Player server and its VFH and AMCL modules are treated as "black boxes" of mysterious internal construction.

The robot controller C receives as input an ordered vector of places to visit $P = \{p_1, \ldots, p_n\}$ and an initialized index $i = 0$ marking the current goal, the ith member of P. $p_i = (x_i, y_i, r)$, where where x_i and y_i are Cartesian coordinates in the plane. If the Cartesian distance from the robot to place (x_i, y_i) is less than r then the robot is considered to have visited (x_i, y_i). The controller takes the initial goal location p_0 and submits it to Player's VFH implementation as a goal location. Player then attempts to drive the robot to that position while avoiding obstacles. All locations are specified in the robot's localization coordinate system, as determined by the AMCL implementation. Player reports the current robot

pose back to the controller. When the robot visits its goal location, the location index is incremented $i = i+1$ and the new goal location p_i is submitted to VFH. When the last goal location is reached, the task is complete and the robot stops. A schematic of the controller is given in Figure 2.

The output from the "known", i.e. non-blackbox part of the controller is a sequence of commands of the form `goto(`x, y`)`. The blackbox parts of the controller attempt to achieve the current goal without crashing into obstacles by reading from the range sensors and sending a stream of motor commands of the form `move(`v, ω`)`, where v is the desired forward velocity and ω is the desired angular velocity. Though we as designers can state the purpose of these modules, there is no way for the system to know what they do, and how they would effect the robot's configuration space.

In fact VFH has some dynamic properties that are crucial to a planning system. It is an excellent local planner and obstacle-avoider, but as such it suffers from local minima problems that cause it to make bad decisions under certain conditions. Consider the situation presented in Figure 3: the robot is blocked from reaching its goal by walls to the front and sides. If the side walls are long enough, VFH is unable to escape from the "trap" and will instead make small loops indefinitely, eventually exhausting the robot's energy supply. Depending on the task and the availability of human assistance, this may be a fatal error that the robot should never experience for real.

We desire our robot system to be able to take into account the complex dynamic properties of VFH when choosing the best route to take, with no *a priori* model of VFH. The same argument applies to the dynamics of the AMCL localization system, and the Player TCP server, the details of which we omit for lack of space.

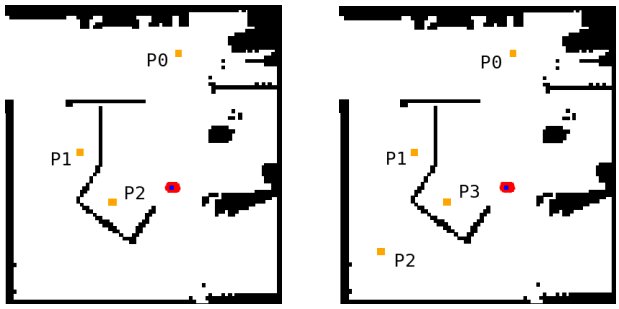
2.4 Sensorimotor simulation implementation

In the real robot, the `move(`v, ω`)` commands are converted by the robot's embedded computer into pulse-width modulated signals that drive amplifiers that power the wheel motors. The physical motion of the wheels is reflected in subsequent measurements taken by the robot's physical sensors.

We can replace the physical part of the system with the well-known Stage robot simulation engine [19]. Player using simulated Stage devices is known as *Player/Stage*, and has been used in published experiments by many authors. For convenience, we introduce the term *Player/real* to indicate a Player server connected to real robot hardware. If reasonably careful in the assumptions made in the implementation of a robot controller, one can expect grossly similar robot behaviour in Player/Stage and in Player/real. We have anecdotal evidence from the community that Stage models real robots reasonably well. We present experimental evidence below that backs this up (Section2.6).

(a) (b)

Fig. 3: Pioneer robot in the Real World (a) and VFH trap: The robot's goal is to reach the small square outside the room. (b)



(a) (b)

Fig. 4: Simulated world maps showing robot start location and goal locations for (a) Exp.1: with 3 locations, and (b) Exp.2 with 4 locations.

2.5 Experiment 1: Simulation proof of concept

In this proof of concept we work entirely in simulation, i.e. R_0 and W_0 are simulated and model the real robot and world, butR_0 is still unique in that only it can spawn child simulations. R_0 must choose the best order in which to visit n locations, by observing the behaviour of $R_1 \ldots R_{n!}$ simulated robots, one for each possible route.

Figure 4(a) shows the scenario of the first experiment. There are $n = 3$ places to visit, so $P = \{p_0, p_1, p_2\}$. To visit a place, the robot must come within $r = 0.5m$ of the place. The world map is an approximation of our real robot arena, and was automatically created with the pmap mapping utility [1].

R_0 spawns $m = n! = 6$ threads, each containing a complete simulation and robot controller. The first thread in which a robot visits all locations is the

[1] pmap was written by Andrew Howard and available from http://playerstage.sourceforge.net.

winner R_{win}. The other threads are stopped and R_0 executes the route taken by R_{win}.

With small values of m, we can run all the simulation threads in real time on a modest workstation. A more scalable solution would allow threads to be spawned on multiple computers. On a machine with a single CPU there is little advantage to be gained from multiple threads and instead we could explicitly run each simulation in turn for a short time in a single thread, thus avoiding the thread switching overhead.

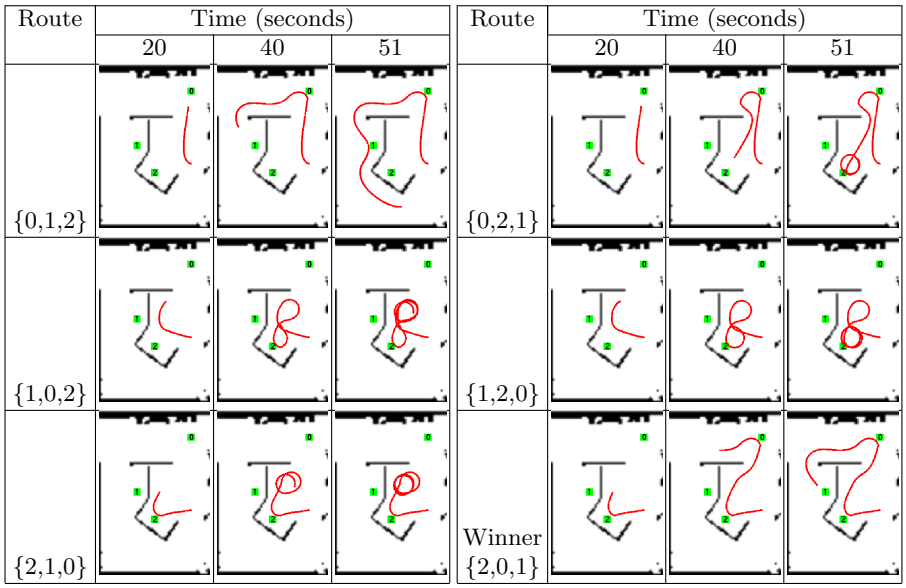


Fig. 5: Multi-threaded example: 6 threads are generated, one for each possible route. Route {2,0,1} is completed in 51 seconds, when this happens all the other threads are stopped.

Results Figure 5 shows the progress in time of all 6 threads that execute all possible routes. Plots of the route travelled by each simulated robot at 20-second intervals. At 50 seconds the robot executing route $\{2, 0, 1\}$ finishes and all the other simulations are stopped. All routes except the winner and Route $\{0, 1, 2\}$ were in looping states (the fatal situations we want the real robot to avoid) and were not chosen for execution. Route $\{0, 1, 2\}$ appears to be on its way to accomplishing the task, but it is taking a longer path than the winning route. R_0 now executes the winning route, and its path is shown in Figure 6.

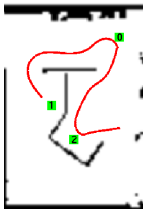


Fig. 6: The path taken by R_0, having selected the winning route $\{2, 0, 1\}$.

2.6 Experiment 2: R_0 in the real world

In this experiment we use the real-world Pioneer 3DX robot shown in Figure 3 for R_0 and increase the number of places to visit $n = 4$, located as shown in Figure 4(b). There are now $m = n! = 24$ possible routes, and the robot's onboard computer could not run 24 simulations in real time, so we implemented a single-threaded imagination. Figure 7 (top-row) shows the simulated robot path in 5 of the 24 possible routes.

Results Route $\{3, 0, 1, 2\}$ 7 (top-row, left-most plot) was the first to finish and was selected for execution by the real robot. The path taken by the real-world robot (as estimated by the AMCL localization system) is shown directly below, in Figure 7(bottom-row, left-most plot). The real robot path is qualitatively similar to the simulation path, and completes the task successfully.

In order to examine how closely the behvaiour of the Stage-simulated robots predicts the real-world behaviour, we run four other possible routes on the real-world R_0. Compare the paths taken by the simulated robots in the top row of Figure 7 with their real-world executions in the bottom row.

Route $\{3, 0, 1, 2\}$ and Route $\{3, 0, 2, 1\}$ are qualitatively similar in simulation and reality. Unfortunately there is no standard metric for quantifying the similarity of robot trajectories. Route $\{3, 1, 2, 0\}$ did not finish but shows the same behaviour in simulation and reality. Routes $\{0, 1, 2, 3\}$ and $\{3, 2, 1, 0\}$ finished in reality but stayed in a cycle during simulation. Though the eventual outcome in simulation and reality was different for these routes, they showed similar dynamics in that they spent most of their time stuck in a VFH trap. Note that because neither of these routes was the winner, the divergence of simulated and real behaviour did not effect the performance of the real-world robot.

3 Future Work

In the short term we plan to create more complex experiments that explore the possibilites of this approach. Some ideas include:

– **Optimized simulations:** We can explore the use of heuristics and the reuse of previous simulatons to speed up exploration of imagined worlds.

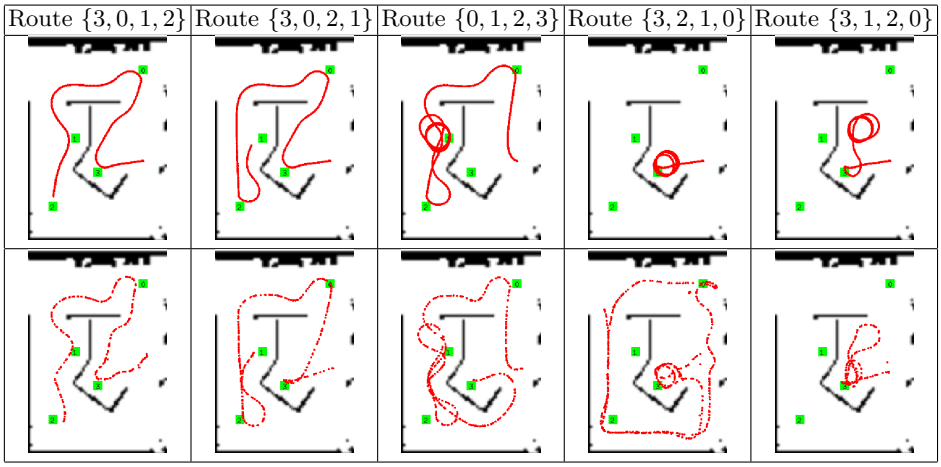


Fig. 7: Simulation (top row) vs. Reality (bottom row)

- **On-line world model aquisition:** Requiring a complete world model for simulation is a serious constraint. We seek to aquire world models on-line and simulate using the latest models.
- **Dynamic worlds:** The worlds in this paper have been static apart from the motion of the single robot. We seek to include models of other agents that can effect the environment and behaviour of our robot.
- **Concurrent imagination:** While acting in the real world, a robot can still imagine alternative scenarios, to anticipate its reaction to unexpected events.
- **Recursive imagination:** Currently only R_0 can spawn simulations. There may be utility in allowing simulated robots to spawn their own simulations, creating a tree of imaginary robots, rooted at R_0.
- **Multiple objective optimization:** In this paper we chose a simple task with a single objective function. A natural progression of this problem is to allow for multiple goals.

4 Conclusion

We have described a novel framework loosely analagous to imagination, in which agents can use sensorimotor self-simulation to reason about their ability to perform tasks, despite having no model of most of their internal processes and thus no way to create an *a priori* state-evolution model with which to search. This in-the-head rehearsal of tasks is particularly useful when the tasks carry a high risk of agent robot "death", as it provides a source of negative feedback in perfect safety. This approach is a useful complement to existing work using forward models for anticipatory behaviour. A simple but useful implementation was shown to be effective in simulation and real-world experiments, and future directions outlined.

References

1. Brooks, R.A.: Intelligence without representation. Artificial Intelligence Journal **47** (1991) 139–159
2. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: The use of simulation in evolutionary robotics. Lecture Notes in Computer Science **929** (1995)
3. Zuluaga, M., Vaughan, R.T.: Reducing spatial interference in robot teams by local-investment aggression. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS2005 (2005)
4. Damasio, A.R.: The Feeling of What Happens: Body and Emotion in the Making of Consciousness. Harcourt (1999)
5. Ramachandran, V.: Phantoms in the Brain. HarperCollins (1998)
6. Dawkins, R.: The evolved imagination: Animals as models of their world. Natural History magazine **104** (1995) 8–11,22–23
7. Grush, R.: The emulation theory of representation: Motor control, imagery, and perception. Behavioral and Brian Sciences (27) (2004) 377–442
8. Popper, K.: Alles Leben ist Problemlsen. ber Erkenntnis, Geschichte und Politik. Piper (1996)
9. Holley, J.: First investigations of dream-like cognitive processing using the anticipatory classifier system. Technical Report UWELCSG04-002, Clares MHE Ltd, Wells, Somerset UK (2004)
10. Wilson, M.A., Louie, K.: Temporally structured replay of awake hippocampal ensemble activity during rapid eye movement sleep. In: Neuron. Volume 29. (2001) 145–156
11. Hoffmann, H., Moller, R.: Action selection and mental transformation based on a chain of forward models. In: Proceedings of the Eight International Conference on Simulation of Adaptive Behaviour, SAB 2004. (2004)
12. Möller, R.: Perception through anticipation. a behavior-based approach to visual perception. In Riegler, A., Peschl, M., von Stein, A., eds.: Understanding Representation in the Cognitive Sciences. Plenum Academic / Kluwer Publishers. (1999) 169–176
13. Ziemke, T.: Cybernetics and embodied cognition: On the construction of realities in organisms and robots. Kybernetes **1/2**(34) (2005) 118–128
14. Jirenhed, D.A., Hesslow, G., Ziemke, T.: Exploring internal simulation of perception in mobile robots. Neurocomputing **68** (2005) 85–104
15. Zagal, J.C., del Solar, J.R., Vallejos, P.: Back to reality: Crossing the reality gap in evolutionary robotics. In: IAV 2004 the 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal. (2004)
16. Gerkey, B.P., Vaughan, R.T., Stoy, K., Howard, A., Sukhatme, G.S., Matarić, M.J.: Most valuable player: A robot device server for distributed control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (2001) 1226–1231
17. Ulrich, I., Borenstein, J.: Vfh+: Local obstacle avoidance with look-ahead verification. In: IEEE International Conference on Robotics and Automation (ICRA). (2000)
18. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: Efficient position estimation for mobile robots. In: National Conference on Artificial Intelligence (AAAI). (1999)
19. Vaughan, R.T., Gerkey, B.P., Howard, A.: On device abstractions for portable, reusable robot code. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, U.S.A (2003) 2121–2427 (Also Technical Report CRES-03-009).