Comparative experiments on the emergence of safe behaviours

Yuri Gavshin, Maarja Kruusmaa

***Abstract*—This paper explores the idea that robots can learn safe behaviours by learning to reverse actions. Previously we have demonstrated that obstacle avoidance behaviour emerges when a robot learns to suppress irreversible actions and we have also demonstrated emergence of territorial behaviour in case of more complicated scenarios.**

In this paper we represent comparative experiments with two different robots to investigate if a code based on this abstract principle is applicable on different robots with different shape, size and polarity of proximity sensors in different environments. Furthermore, we compare the performance of the algorithm based on the reversibly of actions to a dedicated Q-learning obstacle avoidance algorithm. The experimental results show that the performance of the algorithm is the same on both platforms and is 10% lower than of Q-Learning algorithm. We interpret this as the evidence confirming the hypothesis. We conclude that the reversibility based algorithm can be used on different robotic platforms with minor modifications to sensory-motor interface.

I. Introduction

This paper is concerned with safety of robot behaviour by applying an abstract principle of reversibility on real robots. In [1] we demonstrated that the principle "Don't do things you can't undo" generates a concrete safe behaviour of obstacle avoidance. We speculated further that this abstract principle can be applied to different robots in different environments. Furthermore we speculated that this principle could generate variety of safe behaviours. In [2] we demonstrated that a more complex territorial behaviour can emerge as a result of avoiding irreversible sequences of actions.

We speculate that a robot governed by such an abstract principle will behave safely in a wide variety of environments, since many undesirable actions such as damage of the robot/environment or getting stuck is characterized by irreversibility. Although not all irreversible actions are undesirable, it is safe to say that all reversible actions are safe.

Reversibility, or absence of irreversibility, is an extension of stability in the way that reversibility can be task-specific: positive changes after "good" actions will be identified as non-stable, but reversible.

The idea of using abstract principles to govern robot behaviour has already been studied before. Kaplan and Oudeyer in [3] showed that a robot can develop visual competences from scratch driven only by internal motivations independent of any particular task: predictability, familiarity and stability.

The main benefit of using the abstract principle, instead of specific routines such as avoiding obstacles, falls, traps, risky regions or routes or staying near some known landmark, is its generality. It explains "why" a robot should behave that way and if a new problematic action/situation occurs, a robot avoiding irreversible actions will avoid these new dangers after some learning period.

The main contribution of this article is a comparative test to confirm/reject the hypothesis that the code based on this abstract principle can be run without major changes on different robots with different shape, size and polarity of proximity sensors in different environments.

In the following section we present our ideas in a more formal way. In section 3 we describe the experimental setup, the algorithms used, explain the differences between the two robots used in experiments, their test environments and specific implementation details. In section 4 we present the experimental results and discuss them together with applicability of the concept of reversibility. In the last section we make conclusions and speculate about some possible directions of future work.

II. Theoretical Framework

This section briefly describes the general theoretical framework used to ground the reversibility based algorithm and to test the robots. Emergence of obstacle avoidance behaviour is also explained in the end of this section. The reader is referred to [2] for more details.

A. Definitions

A robot's **world** is a labelled transition system $(S, \Lambda, \rightarrow)$, where S is a set of experienced states, Λ is a set of labels (a label contains an action or a sequence of actions), and $\rightarrow$ is a set of labelled transitions between the states. When the result of an action a in state s is not wholly determined by the robot, multiple transitions from s are labelled with the same action a and it is the world that determines which transition actually happens.

A ***reversibility*** for world W is a quintuple of three states and two actions: $(s_0, a_0, s_1, a_1, s_2)$. Generally speaking, a

Manuscript received May 15, 2008.

Yuri Gavshin, corresponding author, is with the Center for Biorobotics, Tallinn University of Technology, Tallinn, Estonia; (e-mail: yury@biorobotics.ttu.ee).

Maarja Kruusmaa, is with the Center for Biorobotics, Tallinn University of Technology, Tallinn, Estonia; (e-mail maarja@biorobotics.ttu.ee).

composite action a_0a_1 produces a transition from s_0 to s_2 through s_1 in W. Also, the action sequence a_0a_1 is expected to work for any states x and y with $d_{orig}(x,s_0) \le \varepsilon_{orig}$ and $d_{dest}(y,s_1) \le \varepsilon_{dest}$, where d_{orig}, d_{dest} are metrics on states and ε_{orig}, ε_{dest} are their thresholds.

The ***reversibility*** $(s_0, a_0, s_1, a_1, s_2)$ ***holds*** in W if there exists a transition path from s_0 to s_2 through s_1 consisting of two transitions labelled accordingly a_0 and a_1, and $d_{rev}(s_0, s_2) \le \varepsilon_{rev}$, where d_{rev} is a prametric ($d_{rev}(x,y) \ge 0$ and $d_{rev}(x,x) = 0$) on states and ε_{rev} is a threshold; **fails** otherwise.

An ***action*** a_0 in an arbitrary state s ***is expected to be reversible*** (by action a_1), if the reversibility $(s_0, a_0, s_1, a_1, s_2)$ holds and $d_{orig}(s, s_0) \le \varepsilon_{orig}$.

A **reversibility model** of the robot is a set of reversibilities that are expected to hold.

B.Explanations

A reversibility model can be given to the robot in advance, transferred from another robot, extracted by a human from the knowledge about the world or learned by the robot. Using this model a robot can predict whether the action from the state is reversible by iterating through its experience and using obtained reversibilities to ground the predictions.

The actions used are symbolic actions and it is irrelevant whether they are atomic or complex actions. These actions can also be interpreted as discrete choices if used by a high level symbolic decision maker. The only requirement is that every action must have a reverse action, i.e. the action that undoes (reverses) it.

States are also discrete but with metrics d_{orig} and d_{dest} defined on the set of the states. These metrics are used to search for the reversibilities to ground the predictions. Metric d_{orig} together with its threshold value ε_{orig} are used to filter reversibilities by calculating the distance between its initial state and the current state. The smaller the distance, the higher is the probability that the actual outcome of making the same action from the current state will generate a similar reversibility.

A prametric d_{rev} is used to calculate how strongly the reversibility holds. A prametric is used instead of a metric to reward transitions from "worse" states to "better" ones (in case of goal-oriented learning); if d_{rev} is a metric, then the calculated number measures stability.

C.Emergence of obstacle avoidance behaviour

Let us explain how and why the obstacle avoidance behaviour emerges as a result of avoiding irreversible actions. To simplify the example we will use a robot with a proximity sensor and two actions - "move 10 steps forward" and "move 10 steps backward". Without loss of generality we can assume that "steps" and values of proximity sensors are given in comparable units.

The robot tests these actions in different situations and checks whether the obtained reversibilities hold. The ones that fail usually correspond to collisions of some sort or other negative outcomes. Consider the following 4 cases, where the robot makes 10 steps forward and then 10 steps back:

1)If the robot is at least 10 units away from the obstacle, say, 12 then it doesn't touch the obstacle and we obtain the reversibility which holds:

$$((12),(+10),(2),(-10),(12))$$

2)If the robot is less than 10 units away from the wall, say, 8 then it touches the wall and its motor stall, we obtain the reversibility which doesn't hold:

$$((8),(+10),(0),(-10),(10))$$

3)If the robot touches the wall and its wheels slide on the surface then we obtain the same reversibility as in case 2.

4)If the robot touches the obstacle, but the obstacle is light enough to be moved, then the obtained reversibility will also be identical to case 2 from the robot's point of view.

This way the robot discovers that running into or pushing an obstacle is "bad" without even knowing what the "obstacle" or "pushing" is. A reversibility model with such reversibilities will allow a robot to distinguish those state-action pairs in which "bad things happen" from those in which they do not.

III.EXPERIMENTAL SETUP

The purpose of the experiments is to verify how abstract the implementation of the principle is. For this purpose we compare the performance of the reversibility based algorithm on two different robots and compare these results to another well-known algorithm for obstacle avoidance (Q-Learning).

A.Comparative experiments

The experiments consist of two test runs (5200 steps each) on two different robots. Each test run is divided into two phases – Phase 1 (data collection phase) and Phase 2 (simulation phase)**.**

During Phase 1 the robot makes pseudo-random moves and the input data (sensors data, actions made and outcomes of the actions) is collected and saved into log files. The predictions are made on-line during Phase 2 using data collected in the test runs. The performance is measured by sampling algorithms' predictions of whether the next action will succeed and calculating the success rate of those predictions.

B.The robots

Comparative experiments are conducted on two common research robot platforms, Khepera II by K-Team and Scitos G5 by MetraLabs. The experiments on Khepera II are reported in our previous work [1]. In this paper these experiments are repeated on Scitos G5 robot in comparable environmental conditions. The size of the environment was increased proportionally to the size of the robot.

The relevant technical aspects of these robots are presented in table I for comparison. The most important differences between these robots are their shape and polarity of their sensors.

TABLE I
COMPARISON OF KHEPERA II AND SCITOS G5 ROBOTS

Property	Khepera II	Scitos G5
Width	70 mm	617 mm
Length	70 mm	737 mm
Height	30 mm	582 mm
Weight	0.080 kg	60 kg
Payload	0.250 kg	50 kg
Number and type of sensors	8 Infra-red proximity and ambient light sensors with up to 100mm range	24 ultrasonic range finders with up to 3000 mm range
Sensors polarity	Counter-proportional to distance	Proportional to distance

Both Khepera II and Scitos G5 are differential drive robots but with different size and slightly different geometry. Khepera II has the circular shape and the rotation axis is exactly at the centre of the circle. Therefore it can rotate freely in very close proximity (1-2 mm) to the obstacle without touching it.

Scitos G5 also has a circular shape but with an additional compartment at the back side for the passive third wheel, which considerably changes the way it can rotate its own body: a 360° turn can be completed without touching the obstacle only if the distance to the obstacle is larger than approximately 200 mm (the size of the passive wheel compartment).

The sensors of Khepera II and Scitos G5 differ considerably. Khepera's sensors are counter proportional to the measured distance with non-linear characteristics, while Scitos' sensors have linear characteristics and are proportional to the distance measured.

C.Environments

The environment for Khepera II is a right-angled triangle with side lengths 196mm, 125mm and 233mm. It was set up by using an additional wall in a smaller rectangular box. The material for the smaller box, as well as the additional wall, is a package box for electronic devices, its surface is flat and robot's wheels do not slip on it. We set up a very small test environment to make negative outcomes (collisions) appear more frequently. The program was run on a PC, it communicated with the robot through a serial interface to read sensors' data and give commands to the motors. The cable provided both serial link to computer and power for Khepera II.

The environment for Scitos G5 is a rectangular box of size 970mm x 1500mm. Floor is linoleum and walls are made from corrugated cardboard.

Fig. 1. Khepera II physical setup: the experiment was conducted using the setup with the wall as shown in the bottom picture. Upper pictures show how the whole setup looked like.

Fig. 2. Scitos G5 physical setup: the robot inside the box, walls are covered with corrugated cardboard for better sensor readings.

D.Robot Movements

The robot is given a set of actions with corresponding reverse actions: movements forward-backward and turning left-right are pair wise reverse-actions to each other.

Actions are defined in terms of wheel commands. An action $a = (m_1, m_2)$ consists of a pair of motor

displacement commands, for left and right wheels, expressed in native wheel encoder units for Khepera II. A discrete set of actions is used in the experiments:

$a_0 = (-200,200)$ rotate counter-clockwise,

$a_1 = (200,200)$ make a step forward,

$a_2 = (-200,-200)$ make a step backward,

$a_3 = (200,-200)$ rotate clockwise.

, where

$a_0 = -a_3, a_3 = -a_0, a_2 = -a_1, a_1 = -a_2$

In other words, going forward undoes going backward and turning right undoes turning left, and vice-versa.

The wheel commands in Khepera II internal units were translated to the speed commands of Scitos G5 as following: 200 units correspond to approximately 150 mm forward/backward movements and approximately 42 degrees rotation angles. For Khepera II these values are approximately 16mm and 30 degrees.

In the world W the state vector is $s = (d_0, d_1, d_2, d_3)$ where d_i are sensor values for front, back, left and right sensors, accordingly. The robot moves using the algorithm described in Fig. 3.

1. Record current state $s_i = (d_0,...,d_3)$
2. Execute a random action as a_i.
3. Record the state $s_{i+1} = (d_0,...,d_3)$.
4. Execute the reverse action for a_i as a_{i+1}.
5. Record the resulting state as s_{i+2}.
6. Execute a random action as a_{i+2}.
7. Add 3 to i and repeat.

Fig. 3. Movement algorithm (Phase 1)

In other words, the robot makes a random move followed by its reverse action, then makes another random action, but without a reverse action, and then repeats the pattern. The purpose of the first two actions is to generate at least one pair of actions to test if the reversibility holds. The purpose of the next (random) action without a matching reverse action is to make the robot to explore the environment.

E.Software design

The code consists of the following units (see Fig. 4):

- an independent agent that generates the sequence of actions to move the robot during the first phase
- Q-Learning and reversibility based algorithms running in parallel.
- a "switch" to route data between the agent and the algorithms, or to simulate the test run in the second phase

In Phase 1 real-world data is gathered from the test run and saved into a log file. The file contains sensor readings data, actions made and the outcomes of the actions.

Phase 2 is a simulation and can be executed without a robot. In the beginning the log file from Phase 1 is loaded into memory, parsed as sensor readings and actions and then this history is fed to the algorithms, getting predictions of actions' successfulness simultaneously (see Fig. 5).

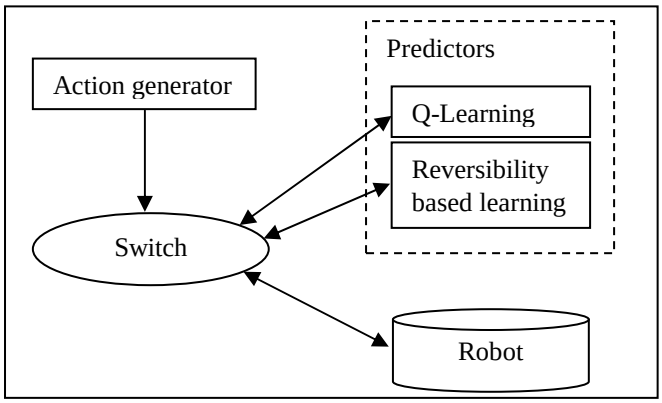

Fig. 4. Software design diagram

1. Read current state as $s_i = (d_0,...,d_3)$ from log.
2. Register action s_i at each algorithm.
2. Read the next action as a_i from log.
3. Get predictions from the algorithms and compare them to the real outcome.
4. Register action a_i at each algorithm.
5. Calculate reward signal for the last action based on the info from log, register it at Q-Learning algorithm.
6. Add 1 to i and repeat.

Fig. 5. Prediction data collection algorithm (Phase 2)

F.Reversibility based algorithm

The aim of the reversibility based algorithm is to predict if a certain action from a certain state is reversible or not.

The algorithm is described in Fig. 6. It takes a sequence of states and actions as an input: $s_0, a_0, s_1, a_1, s_2, a_2, s_3, ...$

At every $i > 1$, if $a_{i-1} = -a_{i-2}$ then the reversibility $(s_{i-2}, a_{i-2}, s_{i-1}, a_{i-1}, s_i)$ is added to robot's experience, which is a vector of reversibilities.

To predict the outcome of making action a_t from state s_t, an expected irreversibility value v_{rev} is calculated using a set of reversibilities selected from the experience vector (in the experiments we select reversibilities with the same forward action and $d_{orig}(s_0, s_t) < \varepsilon_{orig}$, where s_0 is the initial state of the reversibility under consideration).

The value of v_{rev} is a weighted average of $d_{rev}(s_{i-2}, s_i)$ values of selected reversibilities. Reversibilities are sorted by $d_{orig}(s_0, s_t)$ in an ascending order and their weights are $1/i^2$ (1, 1/4, 1/9, 1/16, etc), i.e. reversibilities with a "closer"

initial state have a stronger influence.

In the experiments we use the Euclidean metric to calculate d_{orig} and d_{rev}; the values ε_{orig} and ε_{rev} are finite and selected manually. The metric d_{dest} was not used in the experiments, i.e. $\varepsilon_{dest} = \infty$.

1. Read current state $s_i = (d_0, ..., d_3)$ and the next action a_i from log.
2. Choose a number of reversibilities from the set of experienced ones with a_i forward action, based on d_{orig} between s_i and experienced reversibility's initial state.
3. Calculate the expected irreversibility value v_{rev} using d_{rev} of experienced reversibilities' initial and final states.
5. If no reversibilities are selected, make no prediction.
6. If v_{rev} is greater than ε_{rev}, then predict negative outcome, predict positive outcome otherwise.
7. If $i < 2$, add 1 to i and repeat.
8. Read the last action as a_{i-1} and the previous action a_{i-2} from log.
9. If a_{i-1} is not a reverse-action of a_{i-2}, add 1 to i and repeat.
10. Add the new obtained reversibility as $(s_{i-2}, a_{i-2}, s_{i-1}, a_{i-1}, s_i)$ to the set of experienced reversibilities.
11. Add 1 to i and repeat.

Fig. 6. Reversibility based algorithm

G.Reinforcement learning algorithm

Reinforcement learning is a commonly used learning method to learn obstacle avoidance by trial and error ([5], [6], [7]). Therefore we have chosen a Q-Learning algorithm to compare the performance of the reversibility based learning to a standard method.

The main difference between reinforcement learning algorithms and the reversibility based algorithm is that a reinforcement learning algorithm receives an external reward signal indicating the success of an action. Reversibility based algorithm, on the other hand, uses only sensor data to determine the success of an action.

In the Q-Learning algorithm the expected reward of a state-action pair is updated using the following expression:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \\ + \alpha_t(s_t, a_t)[r_t + \gamma Q(s_{t+1}, a_t) - Q(s_t, a_t)] \quad (1)$$

Our experiment consists of random movements. Therefore the long-term reward is irrelevant and only short-term reward should be used, for this reason we take $\gamma = 0$.

The prediction value is calculated as $sign(Q(s_t, a_t))$, i.e. negative Q means a negative prediction, positive Q means a positive prediction. Initially, Q values are set to 0 and if Q still has the initial value, no grounded prediction can be made.

H.Other implementation details

Khepera's infra-red sensors are very sensitive to indoor ambient light, therefore its test environment was placed into a box to reduce sensor noise. The corrugated cardboard for Scitos G5 environment was chosen because it reflects ultrasound much more uniformly than other non-corrugated materials.

Scitos G5 default configuration file was altered to change the way sensor readings were made (low noise mode is on, reading interval is 50ms, 4 sensors per measurement) and rotational PID controller's Kp was set to 0.2. Sensor values for Scitos are in metres, therefore they are multiplied by 1000 to be of equal scale to the ones of Khepera. This doesn't affect the reversibility based algorithm, but makes saving and loading the log file simpler.

During the experiments $\alpha_t(s_t, a_t)$ for Q-Learning update expression was set to 0.01. Threshold values ε_{orig} and ε_{rev} were constant throughout the experiments, but the experiments for different robots used different values. In experiments with Khepera II the settings were: $\varepsilon_{orig} = 6300$ and $\varepsilon_{rev} = 5000$. In experiments with Scitos G5 the settings were: $\varepsilon_{orig} = 35000$ and $\varepsilon_{rev} = 48000$.

IV. RESULTS

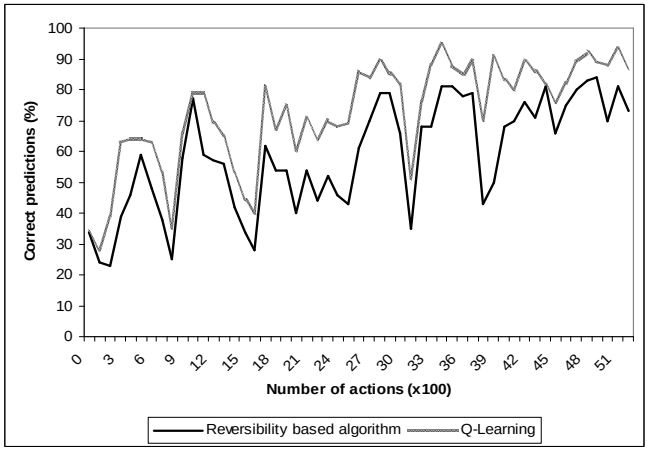

Fig. 7. Test run results for Khepera II

We run two algorithms on both robots and both learning methods are predicting the possibility of collisions with

obstacles. Fig. 7 represents the test results for Khepera II, it compares prediction success rate for the Q-Learning and reversibility based algorithms. These results are also reported in [1]. Fig. 8 represents the comparative experimental results for Scitos G5.

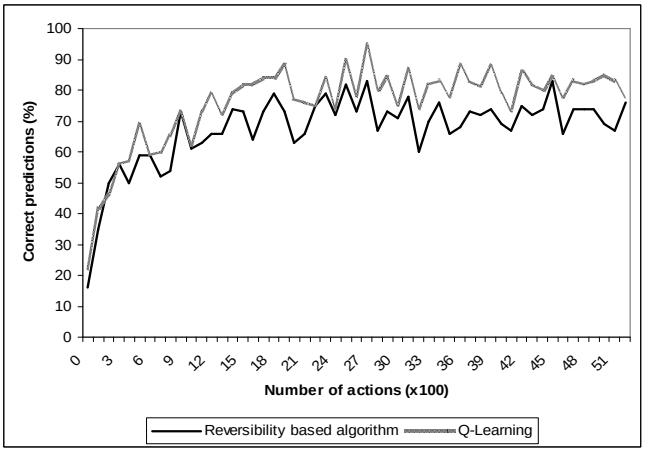

Fig. 8. Test run results for Scitos G5

A.Q-Learning vs. Reversibility based learning

It appears that on both robots Q-Learning converges to a 10% higher prediction success rate than the reversibility based learning.

Let us remind the reader that while the Q-Learning algorithm is explicitly designed to avoid obstacles (at every collision the robot gets a negative reward signal proportional to the size of unfinished movement), the robot learning a reversibility model has no concept of an obstacle or collision.

The reversibility based algorithm, at the same time, does not use the reward signal and only tries to predict whether the action will be reversible or not. If the robot suppressed the irreversible actions it would emerge to obstacle avoidance behaviour very similar to the one achieved by a dedicated obstacle avoidance Q-Learning algorithm.

The 10% higher performance of the Q-Learning algorithm can be easily explained. The method of measuring the success of predictions always works in advantage of the Q-Learning algorithm. The Q-Learning algorithm predicts future rewards based on the experienced rewards, while the reversibility based algorithm predicts future rewards based on sensor data alone.

B.Khepera II vs. Scitos G5

It appears also that reversibility based algorithm performs equally well on both robots: it converges to about 70% success rate in predicting collisions after about 3000 steps.

The Khepera's graph has several drops in prediction success rate around regions of 800, 1700, 3100 and 3900 steps. The Khepera II robot was stuck occasionally during those periods of time, which decreased the learning and prediction success rates.

The aim of these experiments was to confirm/reject the hypothesis that the code based on an abstract principle of avoiding irreversible actions can be run without major changes on different hardware in different environments.

We interpret the results as positive, since, indeed, a concrete robot behaviour of obstacle avoidance is observed on two different robots to emerge from the abstract principle "Don't do things you can't undo". However, there are problems with this straightforward plain-sensor approach: it is influenced by many factors like sensor precision, sensor noise, actions' precision, etc. However, this problem belongs more to the realm of the state identification: Q-Learning algorithm severely suffers from the same problems.

It is difficult to distinguish sensors by their importance for the particular action. For example, sensors on the back side of the robot are useless when predicting whether moving forward will succeed or not. Different kind of sensors can also be a problem, since Euclidean distance takes all numbers equally into account. Thus, a sensor returning current time stamp or a sensor returning distance in millimetres and others in metres will be a huge problem in this case and will render both algorithms almost useless without additional tuning.

Q-Learning uses discrete states, thus, space state tiling is a problem, also the source of the reward signal must be chosen carefully to reward only collision-free movements and to penalize only collisions.

The reversibility principle based algorithm has a similar problem of state identification, sensors' linearity must be as strong as possible, and the scale of sensor values must be the same or proportional to the importance of the sensor for state identification. Another problem is to choose threshold values ε_{orig}, ε_{dest} and ε_{rev}. We chose those values manually using statistical information of the particular test run data.

V.CONCLUSIONS AND FUTURE WORK

The goal of this paper was to verify whether a concrete behaviour of obstacle avoidance can emerge from an abstract principle of avoiding irreversible actions. Also we wanted to compare the performance of the strategy on different robotic platforms.

We conclude that both robots involved in the experiments demonstrated similar performance compared to each other and to Q-Learning algorithm.

We see the future of this research as a cooperative work of environment-model-aware algorithms in conjunction with abstract principles to guide them on a higher level of control with the higher level of abstraction. Another direction is to use the principle of reversibility to make other learning algorithms learn faster or safer, or both.

REFERENCES

[1] M. Kruusmaa, Y. Gavshin and A. Eppendahl, “Don’t Do Things You Can’t Undo: Reversibility Models for Generating Safe Behaviours,” in Proc, of the IEEE Conference on Robotics and Automation, Rome, 2007, pp. 1134–1139.

[2] J. Gavšin, “Using The Concept of Reversibility To Develop Safe Behaviours in Robotics”, MSc thesis, University of Tartu, Tartu, 2007, http://hdl.handle.net/10062/1990 .

[3] F. Kaplan And P.Y. Oudeyer, “Motivational principles for visual know-how development,” in Proc. Of the Third International Workshop on Epigenetic Robotics, Lund University Cognitive Studies ,2003.

[4] A. Eppendahl and M. Kruusmaa, “Obstacle Avoidance as a Consequence of Suppressing Irreversible Actions,” in Proc, of the Sixth International Workshop on Epigenetic Robotics, Lund University Cognitive Studies, 2006, vol. 128.

[5] M. Lin, J. Zhu and Z. Sun, "Learning Obstacle Avoidance Behavior Using Multi-agent Learning with Fuzzy States", Lecture Notes in Computer Science, Springer Berlin/Heidelberg, pp 389-398, 2004.

[6] D. A. Gutnisky and B. S. Zanutto, "Learning Obstacle Avoidance with an Operant Behavior Model", Arteficial Life, Winter 2004, Vol.10, No.1, pp 65-81, 2004.

[7] K. Maček, I. Petrović, N. Perić, "A Reinforcement Learning Approach to Obstacle Avoidance of Mobile Robots," in Proc. of the 7th IEEE International Workshop on Advanced Motion Control - ACM 2002, Maribor, Slovenia, pp.462-466, 2002.