THESIS ON INFORMATICS AND SYSTEM ENGINEERING C68

# Intrinsic Robot Safety through Reversibility of Actions

JURI GAVŠIN

TUT PRESS

TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Centre for Biorobotics

**This dissertation was accepted for the defence of the degree of Doctor of Philosophy in Engineering on November 15, 2011**

| | |
|---|---|
| **Supervisor**: | Prof. Maarja Kruusmaa<br>Centre for Biorobotics<br>Faculty of Information Technology<br>Tallinn University of Technology, Estonia |
| **Opponents**: | Prof. Bruno Siciliano<br>PRISMA Lab<br>Department of Computer and Systems Engineering<br>University of Naples Federico II, Italy |
| | Prof. Tanel Tammet<br>Department of Computer Science<br>Faculty of Information Technology<br>Tallinn University of Technology, Estonia |
| **Defence**: | December 16, 2011 |

**Declaration**: I hereby declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any other academic degree.

*/Juri Gavšin/*

European Union
European Social Fund

Investing in your future

Copyright: Juri Gavšin, 2011

ISSN 1406-4731

ISBN 978-9949-23-219-2 (publication)

ISBN 978-9949-23-220-8 (PDF)

INFORMAATIKA JA SÜSTEEMITEHNIKA C68

# Iseeneslik ohutus robootikas tegevuste pööratavuse põhjal

JURI GAVŠIN

TTÜ
KIRJASTUS

# Abstract

This thesis presents a novel safety architecture that enables combination of conventional safety procedures with reversibility-based safety assessment as a backup for unpredicted situations. Within this architecture the thesis presents two approaches to assess safety using the principle of reversibility – "Don't Do Things You Can't Undo".

The research is motivated by the fact that robots are entering our life and safety is now one of the most important aspects of their usage. Robots' autonomy increases rapidly and it is just a matter of time before robots will be making decisions on their own. The goal of this study is to develop a framework to enable future robots to behave safely in a wide variety of situations by combining extrinsic and intrinsic safety procedures.

Safety in robotics is usually viewed in the context of mechanical safety of robot's body and manipulators as well as their conventional control using situation-specific routines. Alternatively, safety is viewed in the context of ethics, legal rights and as a responsibility of robots' designer. The research on autonomous robot actions, based on intrinsic motivations and abstract principles, is not concerned about safety of resulting behaviors.

In the approach described in this thesis, robot designer supplies safety rules for known situations, while the reversibility-based safety assessment is applied to truly autonomous decisions – when no situation-specific logic is applicable. This way the proposed safety architecture can be integrated into a conventional robot control system, serving as a backup for unpredicted situations. In such situations only reversible actions are intrinsically safe and irreversible actions must be suppressed to make the robot behave safely.

This thesis proposes two approaches to assess safety within the proposed hybrid safety architecture. The world-model-free approach is using distance measures between the states to find relevant data and to measure action reversibility. The world-model-based approach proposes to assess action reversibility through calculation of the predicted cost of undoing the action; robot actions in the environment are simulated internally to calculate the cost of returning back to the initial state.

The experiments conducted in simulated environments with two different robots demonstrate that the robot, governed by the world-model-free reversibility assessment, exhibits a safe behavior of obstacle avoidance. The simulated experiments with actions of different types and lengths demonstrate that the world-model-free approach can also be used to identify the pairs of actions that undo each other. In the experiments conducted in simulated and real environments with a movable obstacle, the world-model-based approach to assess action reversibility permits the robot, governed by the hybrid safety architecture, to increase area coverage using reversible object manipulation.

# List of Contributing Publications

This thesis is based on the work reported in the following papers:

A M. Kruusmaa, Y. Gavshin, and A. Eppendahl, “Don’t Do Things You Can’t Undo: Reversibility Models for Generating Safe Behaviours,” in *2007 IEEE International Conference on Robotics and Automation*, pp. 1134–1139, IEEE, Apr. 2007.

B Y. Gavshin and M. Kruusmaa, “Emergence of Safe Behaviours with an Intrinsic Reward,” in *Adaptive and Intelligent Systems, LNAI* (A. Bouchachia, ed.), vol. 6943 of *Lecture Notes in Computer Science*, pp. 180–191, Springer Berlin / Heidelberg, 2011.

C Y. Gavshin and M. Kruusmaa, “Identification of Reverse-Action Pairs using Reversibility of Actions,” in *2011 IEEE International Conference on Systems, Man, and Cybernetics, Cybernetics Track*, pp. 2555–2560, IEEE, 2011.

D Y. Gavshin and M. Kruusmaa, “Assessing Safety of Object Pushing Using the Principle of Reversibility,” in *Hybrid Artificial Intelligent Systems, LNAI* (E. Corchado, M. Kurzynski, and M. Wozniak, eds.), vol. 6678 of *Lecture Notes in Computer Science*, pp. 313–320, Springer Berlin / Heidelberg, 2011.

E Y. Gavshin and M. Kruusmaa, “Improving Area Coverage by Reversible Object Pushing,” in *15th International Conference on Advanced Robotics*, pp. 415–420, IEEE, 2011.

# Contents

# 1 Introduction

This thesis is addressing the problem of safe autonomous robot decisions. The goal of this work is to develop a framework that allows the robot to behave safely by assessing intrinsic safety of autonomous robot decisions using the principle of reversibility.

## 1.1 Motivation

Robotic presence in human households grows every year. Such a trend is especially noticeable in highly-developed Asian countries like Japan and South Korea. However, simple household robots like robotic vacuum cleaners and floor washers have been on the market world-wide for several years already.

Computers' energy efficiency increases every year and newer robots can be equipped with more computing power than ever before, with the same power consumption profile. Such an increase in available computer power enables engineers to implement more sophisticated algorithms allowing more autonomy for a robot.

A multi-purpose robot-assistant is one of the visions of the future. Such a robot will have to function very close to human beings in their homes and it must be very flexible and autonomous. This makes solving the issue of robot safety an important priority in robotics research.

The research, described in this thesis, is concerned with how to make robot decisions safe, especially when no specific pre-programmed behavior/response can be selected. It is impossible to prepare robot control system for all possible scenarios it can encounter, but robot's behavior must be safe and sound even under such unforeseen situations. Also, robot tasks are becoming so complex that it can be hard to define for professional roboticists with absolute precision. It is even harder to do for advanced users or non-professional technicians.

So far, very few studies deal with high-level safety of robot actions or ground robot safety rules on abstract principles. The vast majority of robot safety research deals with protecting human beings from direct injuries as a result of collisions with robots. To protect humans in case of a collision, robots and their parts can be covered by soft compliant materials, e.g. [1]. Additionally, situation-specific collision-avoiding, slowing and stopping functions can be employed to make it impossible for a robot to collide with a human, e.g. [2, 3, 4, 5, 6].

The vast majority of intrinsic motivation research in robotics deals with how to motivate a robot to learn new things [7, 8, 9], while disregarding safety concerns completely. An exception is the work of Eppendahl and Kruusmaa [10], proposing the use of reversibility principle to ground safe behaviors. Their work was extended and formalized in [11] and [12], but the approach was focused solely on application of the reversibility principle, not considering robot control system as a whole.

The approach in this thesis aims at integrating the procedure of safety assessment through action reversibility into a general safety module to achieve safety-aware goal-oriented behavior.

## 1.2 Contributions of the Thesis

The general problem this thesis aims at solving is how to enable a robot to make an autonomous safe decision when no specific pre-programmed behavior/response can be selected in an unpredicted situation. Contributions of this thesis are:

- formalization of the safety architecture that combines extrinsic safety procedures with intrinsic safety assessment (as a backup for situations not covered by the extrinsic safety procedures)
- formalization of the world-model-free and the world-model-based intrinsic safety assessment through the analysis of action reversibility
- implementation of the world-model-free safe control system and experiments validating the world-model-free safety assessment
- implementation of the world-model-based safe control system and experiments validating the proposed safety architecture with the world-model-based safety assessment

## 1.3 Outline of the thesis

This thesis is organized as follows. Chapter 2 describes the related work in robot safety, roboethics, machine ethics, intrinsic motivation and reversibility. In Chapter 3 we explain the principle of reversibility, present an architecture of the proposed hybrid safety module and formalize two approaches to assess safety through action reversibility. Chapter 4 describes the implementation of the architecture to test the world-model-free safety assessment and the proposed safety module with the world-model-based safety assessment. Chapter 5 contains summaries of the contributing publications. Conclusions and directions of future work are presented in the last chapter.

# 2 Related Work

This section defines the context of the thesis by describing the current state-of-the-art in related research areas.

## 2.1 Robot Safety

Safety is a very important aspect of robotic research and its primary focus is on safety of humans surrounding the robot. The recent overview of robot ethics [13] identified *safety & errors* and *law & ethics* as the most urgent categories of safety-related issues for current and future robotics. *Safety & errors* category identifies current problems with software and hardware design and implementation of robots made by human beings, who are prone to errors. *Law & ethics* category identifies future legislative problems of introducing robots to general public and their internal ethics to distinguish right from wrong and keep humans safe.

Atlas of physical human-robot interaction [14] identifies *mechanics and actuation* together with *control techniques* as the most important aspects of human safety in the close proximity to robots. In robotic surgery, safety is mostly viewed in the context of additional risks [15] or lowering the risks [16, 17] of robotic assistance, compared with the conventional surgery and preoperative inspection.

## 2.2 Safety Standards

Industrial robotics is currently the only kind of robotics with established safety standards. The latest 2011 revision of the ISO 10218 “Robots and robotic devices – Safety requirements for industrial robots” standard specifies requirements and guidelines for the inherent safe design, protective measures and information for use of industrial robots; it describes basic hazards associated with robots and provides requirements to eliminate, or adequately reduce, the risks associated with these hazards [18, 19].

The ISO 10218 standard does not apply directly to non-industrial robots, but its latest editions now account for collaborative operation with humans, employing stopping functions, speed, position, power and force controls for such collaborative modes. The ISO/DIS 13482 “Robots and robotic devices – Safety requirements for non-industrial robots – Non-medical personal care robot” standard is currently under formulation [20] and a draft standard (possibly a Part 2 of the ISO/DIS 13482) is currently in development for robots in medical care [21].

## 2.3 Roboethics and Mechanical Safety

Roboethics is mostly concerned with ethics of the robots' designers, manufacturers and users as well as legal aspects of using robots [22]. Traditionally, the safety in robotics is viewed as a responsibility of the designer [23] – it should not be possible to harm a human being by design or the injury must be as light as possible. For example, by the first rule of IEEE code of ethics, an engineer commits "to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment" [24].

In industrial robotics workspaces of humans and robots are clearly divided to safeguard humans from injuries – robots are either completely enclosed or equipped with limited barriers, light curtains or floor mats around them [25]. Similarly, in military robotics applications the armed autonomous robot guards only the demilitarized zone – anyone who appears inside it is considered an enemy and nobody is shot outside of the zone [26].

In contrast to industrial robotics, in personal and service robotics the physical barriers are highly inappropriate – on the contrary, interaction, physical contact and joint work with humans are the primary tasks of such robots. The ISO/DIS 13482 standard identifies a personal care robot as "a service robot with the purpose of either aiding actions or performing actions that contribute towards improvement of the quality of life of individuals" and identifies three types of applications – mobile servant, people carrier and physical assistant [20]. As a measure to reduce risks, the standard defines a three-step procedure for designing personal care robots, which encourages developers to reduce risks first by means of inherently safe design of the natural properties of the robot, and then by making use of protective measures if the former is not possible [21].

The biggest threat for a human is a collision with the robot's body, manipulator or an item attached to it. The existing methods and tools from automotive industry, like crash-tests using anthropomorphic robot-dummies cannot be used directly to assess and enhance the safety performance of robot systems. The assessment scheme for robotics has to be specifically designed for the characteristics of stiffness and impact intervals arising in robot-human crash [27]. In [28] a standardized crash-testing in robotics, together with suitable models of crash test dummies, is proposed for blunt collisions of the manipulator with different parts of a human body.

In the recent overview of possible classes of human injuries from physical contact [29], blunt head or chest impacts without clamping at typical robot speeds are identified as not life-threatening. Other quite serious head injuries, such as fractures of facial and cranial bones, are very likely to occur already at moderate velocities; however, the appropriate injury indicator for this class of injuries is related not to head acceleration but to impact forces. Authors showed that even low-inertia robots can become very dangerous in near-singular manipulator configurations if clamping occurs.

The need for intrinsically safe mechanical design of robotic arms for domestic robotics has been discussed in [30]; the general framework was set up to work towards a safety measure, quantifying skin damage injuries as a function of robotic design parameters.

A mobile manipulator that is designed to be safe and capable in human environments is presented in [1]. Mechanical design safety includes minimizing inertia, making appropriate materials selection, providing back-drivability, eliminating pinch points and carefully managing kinetic and potential energies as well as force output. Software design safety includes mechanical energy limits and layering of code for sensor and actuator fault tolerance.

A hybrid actuation approach for human-friendly robot design is proposed in [31], better performance and higher safety are achieved by replacement of the heavy electrical actuators with pneumatic artificial muscles. Due to the nature of pneumatic actuation, it generates high force for its size yet achieves low output impedance and small on-joint electrical motor compensates for the low dynamics of the pneumatic muscle.

## 2.4 Control Safety Modules

In addition to mechanical safeguards and intrinsically safe design of robot body and manipulators, a robot control system must include a safety-specific module to guarantee safe behavior. For example, a safety module can stop/slow down a manipulator when a person enters its operating zone or to stop/slow down a robot if it gets too close or touches a human. The safety module can be independent of the entire system by limiting its functionality externally, or can act as a part of the system altering its behavior from within.

One of the earliest of the reported safety sub-systems for industrial robotics is a “safety computer” [32]. Authors investigated how to reliably detect safety violations, based on information from ultrasonic, micro-wave, infra-red and capacitance sensors connected to a single micro-processor. In case a violation is detected, authors proposed the following sequence: to activate an alarm when the intruder first enters the outer perimeter, then slow the robot if he enters the outer workspace, and completely halt the robot if the intruder is in danger of imminent collision.

A slightly more recent approach is a failure-to-safety system of an industrial robotic arm [2], which employs the AND gate to gather binary signals from different subsystems into one. Emergency stop procedure is executed if any of the signals become 0, indicating a failure in a sub-system.

A similar data fusion approach to a safety module is reported in [3], using fuzzy logic for sensor data fusion. Risk level of human-robot collision for the system is calculated based on robot-human distance, speed and acceleration. The risk level is considerably increased if different measurement systems don't agree, causing robot to slow down or to stop.

Another example of robotic safety module is the safety sub-system of the Lisa mobile platform [4]. It consists of 6 laser range-finders for both 2D and 3D obstacle avoidance, bumpers, robotic arm-mounted stereo-vision cameras and an infrared camera for human interaction detection. The robot and its manipulator stop immediately when a person enters robot's workspace; robotic arm joints include electronic torque measurement and contouring error control as additional safety functions.

A methodology of ensuring safety during human-robot interaction through multi-level planning and control, based on explicit quantification of the level of danger to a human, is proposed in [6]. Authors use a hierarchy of three safety modules: the safe planning module to generate safe plans, the trajectory scaling module to decelerate to a safe approach speed along the planned path (if needed) and the reactive control module to evaluate and correct the selected plan at each control step. The reactive part is a safety module, responsible for moving the robot/manipulator to a safe location and stopping it afterward to wait for a new plan; module logic is based on danger index, which depends on the distance between the robot and the object, their relative velocity, and the effective inertia [33].

Another example of safety modules in robotic manipulator control is the "impact potential control" scheme for an arbitrary joint torque control algorithm [5]. The scheme ensures that the impact force at any surface point does not exceed the preset limit and the robot cannot leave the safe region in the state space by autonomous actions. Such safety module passes the torque vector generated by the motion control algorithm unchanged to the robot as long as it complies with the safety constraints. If the desired torque vector does not satisfy these safety constraints, a clipping function is applied to the torque vector.

## 2.5 Machine Ethics

The notion of safety module fits well into the more general framework of machine ethics, which is concerned with ethical behavior of machines (robots and other artificial intelligent beings) towards humans and other machines. Its purpose is to find a way to teach robots to behave morally and ethically also from human point of view, which includes safety and well-being of humans surrounding the machine.

It is important to distinguish between the "implicit ethical agents" and the "explicit ethical agents" [34]. An implicit ethical agent is the one that has been programmed to behave ethically or avoid unethical behavior, without an explicit description of ethical principles. Its behavior is programmed by its designer who is following ethical principles. Such ethical agents belong to the realm of roboethics, described in Section 2.3.

A machine that is an explicit ethical agent must be able to calculate the best action using ethical principles. It must "represent ethics explicitly and then operate effectively on the basis of this knowledge" [34].

The most famous effort of formalizing such ethical rules is the Three Laws of Robotics by Isaac Asimov [35]. These laws, however, are inapplicable in real robotics, since they are too vague, as it is shown by many of Asimov's novels.

Some researchers (e.g. [36]) discuss the possibilities of better formalizing Asimov's First Law, which states that "a robot may not injure a human being, or, through inaction, allow a human being to come to harm". Others (e.g. [37]) introduce alternatives that are in the realm of roboethics, which applies to the human creator of a robot.

An example of machine ethics implementation is the combined application of the teleological (where the rightness of actions is determined entirely by the consequences of the actions) and the deontological (where the rightness of actions depends on something other than the consequences) approaches to solve ethical dilemmas in medical ethics [38].

Another notable example of machine ethics implementation is an effort of embedding ethical behavior into a military robot in the form of ethical rules [39]. The authors developed an ethical architecture, where the "ethical governor" [40] ensures that system decisions are within predefined ethical bounds by limiting actions externally, while the "ethical adaptor" [41] alters ethical constraints and guides system's behavior from within. The constraints are specific to Laws of War and are especially concerned with ethical and legal aspects of armed machine actions. The ethical governor is similar to the safety modules described in Section 2.4, since it also limits or prohibits the actions that don't satisfy the ethical constraints.

## 2.6 Robot Behavior Programming

For the machine ethics rules to be usable on robots, such rules cannot be represented by a vague description of designer's will and vision of what is right, wrong, ethical or unethical. Control systems of robots are currently based on conventional binary computers, which require creating machine code to be executed in order to make the robot to do something. Robot programming systems can be divided into manual and automatic classes of programming [42].

The procedural and behavioral types of manual programming systems, with both graphical and textual input, represent the class of systems, where the robot program is created by a human designer. A procedural system uses the traditional programming language approach and is one of the most common methods, particularly in industrial environments. In behavior-based programming a human specifies how the robot should react to different conditions instead of providing a procedural description [43].

Automatic programming systems provide very limited or no direct control over the program code the robot will execute. In contrast to manual programming approach, robot code is generated or the behavior is derived from information entered into the system or received by it autonomously. Automatic programming by demonstration is used in industrial [44] and personal [45] robotics in conjunction with procedural programming. In contrast to industrial robotics, where programming by demonstration serves the purpose of fairly simple and precise recording of movements, in personal robotics demonstration is often used as an input to machine learning techniques [46].

General learning techniques, like reinforcement learning [47] or artificial neural networks [48] can be used for both supervised and unsupervised learning, with different sources of error/reward signals. The source of such error/reward signal can be interpreted as robot's motivation, which can be extrinsic – provided by sensor values or a supervisor, or intrinsic – provided by some internal drive to e.g. play, explore, manipulate, learn, etc.[49]

## 2.7 Intrinsic Motivation

The ethical or safety rules must be sufficiently general in order to be usable by a robotic system in as many situations as possible. One of the ways to implement such rules is to express them in general terms, without reference to specific sensor values, interfaces or situations. In contrast to conventional robot programming, where responses are pre-programmed on per-situation basis, intrinsic motivation research proposes to ground robot behavior on top of abstract principles [49]. The notion of intrinsic motivation comes from psychology: it refers to doing something because it is inherently interesting or enjoyable, while extrinsic motivation refers to doing something because it leads to a specific outcome [50].

The vast majority of intrinsic motivation research is focused on learning new skills. For example, initial implementations of "artificial curiosity" [51, 7] used learning machines on abstract problems.

In the field of developmental robotics, a number of basic visual behaviors (tracking of the moving light with exploration for other light sources) are shown to emerge on a real robot from abstract motivational principles – stability, predictability and familiarity [52]. Later, authors presented the mechanism of Intelligent Adaptive Curiosity, an intrinsic motivation system that pushes a robot towards situations in which it maximizes its learning progress [8].

In the field of reinforcement learning, intrinsic motivation is based on the concepts of "novelty" and "surprise" to autonomously learn hierarchical collection of skills [9, 53]. Such simple skills together with intrinsic motivation are used to learn more complex skills and to enhance goal-oriented learning rate; authors support their claim with grid-world scenarios.

As opposed to previously described intrinsic motivation principles, in [10] it was proposed that the principle “don’t do what you can’t undo” is one of the basic abstract principles that can be used to ground robot’s behavior. The proposal is focused on safety of robot actions towards itself and its environment, which contrasts with the previously described intrinsic motivation approaches focused on learning of new skills while disregarding safety.

## 2.8 Reversibility of Robot Actions

The notion of reversibility has been studied in engineering (Tesla principle [54]), thermochemistry & thermodynamics (Thomsen–Berthelot principle of maximum work [55]) and developmental psychology (a stage of child development in Jean Piaget’s theory of cognitive development [56]). In mathematics, and later in computer science, reversibility became a well-studied topic of research. Time reversibility, for example, is an attribute of stochastic and deterministic processes; generally, such processes can be divided into sub-processes, which undo the effects of each other.

The ability to undo last action or a sequence of actions is widely used in user interfaces of computer software to edit texts, pictures, etc. It allows users to undo undesired actions and get back to the “good” state of their work. A similar approach to undo the last robot action is also used in robotic user interfaces based on laser or touch-screen systems [57, 58].

The ability to reverse a plan, when something goes wrong during its execution is studied in [59], based on the approach of identification of “good” points in a plan with ways to get back to them if needed. In the work of Thomaz and Breazeal, hand-crafted UNDO functions are employed to enable undoing of simple actions [60, 61]; authors report improved reinforcement learning performance in a grid-world example task.

In [10], authors proposed that obstacle avoidance is a natural consequence of the principle of reversibility (“don’t do what you can’t undo”) and conducted a one-dimensional test (the robot moved back and forth between two walls) to back up their hypothesis.

## 2.9 Novelty of the Proposed Approach

The main priority of the approach described in this thesis is robot safety, which is viewed in the context of autonomous decision making in situations, when no situation-specific logic/rules can be applied. This contrasts with the work described in Sections 2.1, 2.2 and 2.3, where safety is viewed in the context of mechanical safety, barriers and situation-specific stopping/slowing/avoiding functions. The approach, however, does not discard the need for mechanical safety, but studies the problem of safety in robotics from another point of view. To be safe, the robot, whose behavior is bounded by the proposed safety system, must also conform to all safety standards relevant to mechanical safety.

The safety module proposed in this thesis is based on the assessment of reversibility of actions. This contrasts with the modules described in Section 2.4, which use situation-specific pre-programmed stopping, slowing or avoiding routines to prevent collisions of robots with any part of a human body. The proposed approach is focused on higher-level safety concerns, such as indirect dangers when a human or a robot is stuck and suffers from hunger or battery depletion. The abstract nature of reversibility also accounts for other negative consequences, including the situations where a person is injured or a robot is damaged/destroyed.

The approach also belongs to the research area of machine ethics, described in Section 2.5. The approach is not trying to implement Asimov's Laws of Robotics; it is rather more similar to the ethical governor that ensures that system decisions are within predefined ethical bounds [40]. Instead of ethical bounds specific to Laws of War, the proposed safety module acts as a governor, which ensures that system decisions are within predefined safety bounds.

The approach presented in this thesis is based on the ideas presented in [10], where the reversibility principle "don't do what you can't undo" is proposed to be the basic abstract principle to make robot's behavior safe. This principle is used as a basis of intrinsic motivation to behave safely by grounding the safety rules for the proposed safety architecture.

# 3 Hybrid Safety Assessment Using Action Reversibility

This chapter describes the principle of reversibility in general, its application through the hybrid safety module for a robot control architecture and the two proposed approaches to assess action reversibility.

## 3.1 Reversibility of Actions

Traditionally, safety in robotics is concerned mostly with mechanical safety of robot's movements. The usual safety measures are separation of robot's workplace and implementation of the situation-specific safety procedures, often based on thresholds expressed in sensor data values. Our idea is to use the principle of reversibility "Don't do things you can't undo" to make the robot behave safely; it was proposed in [10] and developed further in [11] and [12].

Basis of the approach described in this thesis is the observation that *negative consequences of robot's actions are also irreversible*, which naturally leads to the idea that *reversible actions are intrinsically safe*. We argue that a robot, making reversible actions, will behave inherently safely as it will avoid actions with irreversible consequences. Instead of specifying routines such as avoiding obstacles, falls, traps, risky regions/routes or staying near to some known landmark, it is rather told not to do things it cannot undo. It explains *why* a robot should behave that way and if a new problematic situation occurs, a robot avoiding irreversible actions will avoid these new dangers by identifying them as irreversible.

For example, damaging a robot is bad, if there is no way to fix it. Locking a door with no knowledge of how to unlock it is also undesirable, as the robot gets stuck inside a room and would, eventually, cease functioning when the battery is depleted. Going too far away from the charging base without the knowledge of how to get back has the same negative and irreversible consequence as a situation, where the robot is stuck inside a room.

In contrast to irreversible actions, reversible actions are safe, because it is possible to return to the initial situation and make another action or repeat the action again. For example, if a desk with some items on it is being cleaned, it is safe to lift things up, if the agent has the knowledge how to put them back on the same place. In another example, when a robot is exploring the area or trying to find a path form one point to another, it is intrinsically safe to try the paths leading to positions from where the initial point is reachable.

It is worth mentioning that not all irreversible actions are unsafe or unwanted. On the contrary, many useful actions a human (or a robot on his/her behalf) needs to do are irreversible. There can be irreversible actions that the robot is allowed or expected to do (e.g. a vacuum cleaning robot cleans a floor irreversibly), but then the robot is designed for doing those actions, and it is an informed decision of the user to use this robot for these purposes.

In practice, household and service robots have a goal and they function according to the logic specified by designers to fulfill the goal. Thus, the principle of reversibility alone is not sufficient to control a robot, since the robot guided *only* by reversibility does not have a useful purpose. Such a reversibility-only-based robot would do nothing in most environments, as it is usually the safest way to exist.

The best way to use principle of reversibility is to apply it in unpredicted situations or in situations with many equally good or bad options. For example, if there are several unexplored ways from one point to the other of the same length, it is safer to use the ones, along which it is always possible to come back. In an unpredicted situation all options are equal and the most reversible one is the safest.

In the following sections we present the safety architecture that combines extrinsic safety procedures with the intrinsic safety assessment; the latter is used as a backup for situations not covered by the former.

## 3.2 Hybrid Extrinsic+Intrinsic Safety Module

Our approach to practical safety in robotics is to combine the reversibility-based intrinsic safety assessment with designer-based safety procedures and goal-oriented intrinsically unsafe overrides. We propose to encapsulate safety-related rules into a *safety module* for a robot control architecture. Such a safety module ensures that system's decisions are safe and within predefined bounds – similar to the *ethical governor* in [40]. Before taking an action, a system checks with the safety module whether the action is allowed from the safety perspective (see Fig. 3.1 for the activity diagram). The query to the safety module consists of the state-action pair $(s, a)$ to be analyzed. The safety module returns a tuple: the boolean value, whether the action $a$ is allowed in the state $s$, and the *post-action* – an optional sequence of actions to be done after the initial one to ensure safety.

The first stage of safety assessment inside the module is application of the extrinsic safety rules to permit or prohibit actions. If the specified state-action pair matches any of the extrinsic rules, the answer is generated based on this extrinsic knowledge. If no extrinsic rule can be applied, then the reversibility module is queried to assess safety of making action $a$ in state $s$. When the action is prohibited by the safety module, either extrinsically or by reversibility sub-

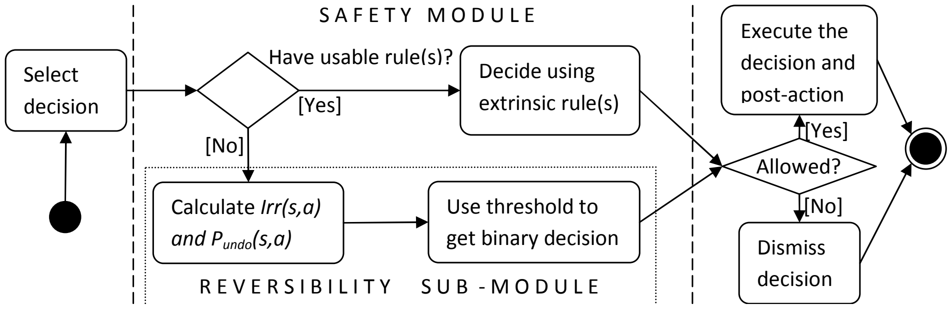
SAFETY MODULE
Select decision
Have usable rule(s)?
[Yes]
[No]
Decide using extrinsic rule(s)
Calculate $Irr(s,a)$ and $P_{undo}(s,a)$
Use threshold to get binary decision
REVERSIBILITY SUB-MODULE
Execute the decision and post-action
[Yes]
Allowed?
[No]
Dismiss decision

Figure 3.1: Activity diagram for the proposed architecture with the safety module and its reversibility sub-module

module, no post-actions are required, since the initial one was not executed.

The context- and task-specific overrides should be added to the set of extrinsic safety rules. This set represents the designer-based knowledge, mostly domain-specific or situation-specific. For example, an extrinsic rule can prohibit pushing any objects – "don't drive ahead, if there is an obstacle in front". A rule can also allow or prohibit all the actions in all states. Also, a pre-programmed post-action can be associated with a rule, for example "inform the user, if any object was pushed".

### 3.2.1 Reversibility Sub-module

The purpose of the reversibility module is to assess the intrinsic safety of decisions through the study of their reversibility. When no extrinsic rule can be applied, and the robot is going to make an autonomous unauthorized decision, the reversibility module is used to assess intrinsic safety of making the desired action from the state in question.

Assessment returns a tuple $(Irr(s,a) >= 0, P_{undo}(s,a) = (a_1, ..., a_n))$, where $Irr(s,a)$ is the predicted value of how *irreversible* is to execute the path $(a, a_1, ..., a_n)$ from state $s$, $P_{undo}(s,a)$ is the undo-path – a sequence of actions expected to undo the action $a$ in the state $s$. The $Irr(s,a)$ value can also be interpreted as a cost of the path $P_{undo}(s,a)$ alone, or preceded with the action $a$.

The action is considered safe if it is reversible – $Irr(s,a)$ is less than a predefined threshold. Reversible actions are intrinsically safe and therefore are allowed; irreversible actions are prohibited.

### 3.2.2 Examples

Consider the situation when a vacuum-cleaning robot is cleaning a floor and the robot plans to move the chair in front of itself to clean beneath the chair. The safety module is queried whether it is allowed to push the chair. If the extrinsic rules prohibit the robot to push chairs, then the action is not allowed. The action is allowed with optional post-action if the rules allow pushing chairs. Alternatively, when no extrinsic rule can be applied, the reversibility of pushing the chair is analyzed. If the robot has the knowledge and the ability to go around the chair and push it back to its initial position, then the action is reversible and it is allowed with a post-action of "go around the chair and push it back". However, the cycle of making the action followed by the undo-path can be prone to errors, taking very long time-wise or requiring too much energy, etc. Such cases are characterized by a large value of $Irr(s, a)$, therefore the action is deemed irreversible and is prohibited if $Irr(s, a)$ value exceeds a predefined reversibility threshold.

Consider another simple example, where a robot is moving around without any goal, model of the environment or extrinsic rules, but it "knows" that action $a$ (1 meter forward) is undone by action $b$ (1 meter backward). The safety module is queried whether it is allowed to take action $a$. If the robot has already tried the same action $a$ in a similar situation before, followed by $b$, and the $a$ was undone by $b$, then $a$ should also be reversible in the current situation. This way the experienced reversible movement allows predicting reversible and intrinsically safe outcome of the same action in the current situation. The experienced irreversible movement, on the contrary, tells the robot, that taking the same action in a similar situation is irreversible; thus, it is intrinsically unsafe and must be disallowed.

## 3.3 Reversibility Assessment

To formalize the process of action reversibility assessment inside the reversibility module, we use the concept of *reversibility model* that tells the robot which actions in which states are reversible and how to reverse them if they are. The reversibility module encapsulates the reversibility model to be used as a part of the safety module and, consequently, some robot control architecture.

A reversibility model aims at answering the following question:

How reversible (or irreversible) is action $a$ in state $s$?

The answer to this question is continuous, since in practice no action is absolutely reversible. The continuous value of reversibility assessment can be made discrete (e.g. "yes"/"no"/"unknown" to be used as an answer of the reversibility module) by mapping continuous values to desired classes of answers using, for example, thresholds or intervals.

The assessment of action reversibility is viewed best in the context of how costly it is to reverse the effects of the action. This way the absolutely irreversible action would have infinite cost and the cost for absolutely reversible one would be zero.

Generally, there are no strict rules how a reversibility model must be implemented. However, it must answer the question of how reversible (or irreversible) is action $a$ in state $s$ in a form of a tuple $(Irr(s, a), P_{undo}(s, a))$, as described in Section 3.2.1.

In this thesis we present two reversibility model approaches that are aimed at two different levels of behaviors and state-action abstractions. The world-model-free framework is meant to work directly on sensor data without any knowledge about the environment and the embodiment, by using distance measures between the states. This approach has very few prerequisites, but its application is limited by simple scenarios. The world-model-based framework calculates the predicted cost of undoing the action by simulating the world and calculating the cost of returning back to the initial state. It has many prerequisites, including planning, environment simulation and state/action identification, but its application is limited mostly by the quality of the external sub-routines.

### 3.3.1 World-model-free Approach

This section describes the reversibility model approach to assess how reversible (or irreversible) is action $a$ in state $s$ without an explicit model of the environment.

Actions are symbolic and the only requirement is that every action must have a reverse-action – the action that undoes it. An action can be atomic or complex; it can also be interpreted as a discrete choice, if used by a high-level symbolic decision maker.

The states are distinct and can represent pure sensor values or symbolic states. Several distance measures are defined to calculate how similar two states are or how far one state is from another. It is possible to use straight-forward metrics like Euclidean or Manhattan distance as well as more complex distance measures.

The reversibility model is created through the analysis of the state-action transitions during execution of an action and then its reverse-action. Ideally, the reverse-action must undo the forward-action and return to the same state. In practice, the low-level state is never the same and the reversibility-distance measure $d_{rev}$ is used to calculate the distance from the final state of the transition back to the initial one. The reversibility model consists of a set of such objects of analysis, called “reversibilities”.

To predict how reversible is action $a$ in state $s$, the set of experienced reversibilities of the reversibility model is filtered using another distance measure. The similarity-distance measure $d_{orig}$ is used to calculate how similar is the current state $s$ to the initial state of the reversibility object. A number of reversibility assessment objects, most similar to the current state $s$, are selected and the result of the assessment is based on how reversible they are.

In the rest of this subsection we present the formalization the world-model-free approach for assessing reversibility, some explanations and an example.

## Formal Framework

A robot's world is a labelled transition system $(S, \Lambda, \rightarrow)$, where $S$ is a set of experienced states, $\Lambda$ is a set of labels (a label contains an action or a sequence of actions), and $\rightarrow$ is a set of labeled 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 labeled with the same action $a$ and it is the world that determines which transition actually happens.

A reversibility for a world $W$ is a quintuple of three states and two actions: $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$. Generally speaking, a composite action $a_{forward}a_{reverse}$ produces a transition from $s_{init}$ to $s_{final}$ through $s_{interim}$ in $W$.

Also, the action sequence $a_{forward}a_{reverse}$ is expected to work for any state $x$, if $d_{orig}(x, s_{init}) \leq \varepsilon_{orig}$, where $d_{orig}$ is a hemimetric ($d_{rev}(x, y) \geq 0$; $d_{rev}(x, x) = 0$; $d_{rev}(x, y) \leq d_{rev}(x, z) + d_{rev}(z, y)$) on states and $\varepsilon_{orig}$ is a pre-defined threshold.

A reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ *holds* in $W$, if $d_{rev}(s_{final}, s_{init}) \leq \varepsilon_{rev}$, where $d_{rev}$ is a hemimetric on states and $\varepsilon_{rev}$ is a threshold; *fails* otherwise.

An action $a_{forward}$ in an arbitrary state $s$ is expected to be reversible (by action $a_{reverse}$), if the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ holds and $d_{orig}(s, s_{init}) \leq \varepsilon_{orig}$.

A *reversibility model* of the robot is a set of experienced reversibilities.

Using the reversibility 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 $d_{orig}$ hemimetric is used to search for the reversibilities to ground the predictions. The $d_{orig}$ hemimetric together with its threshold value $\varepsilon_{orig}$ 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.

The value $v_{rev}$ of reversibility assessment result is calculated based on how irreversible the filtered reversibilities are. It can be calculated using different ways of combining several values into one, like (weighted) average, median, minimum, maximum, etc. The $d_{rev}$ hemimetric is used to calculate how irreversible is the reversibility in question – the higher the value, the more irreversible it is.

Usage of hemimetrics instead of metrics relaxes the identity and symmetry conditions to make the approach more general by allowing usage of less strict measures. For example, with hemimetric it is possible to reward transitions from "worse" to "better" states in case of a complex biased distance measure.

The value of $v_{rev}$ can also be used as a source for the intrinsic reward signal. If such signal is counter-proportional to the value of $v_{rev}$, then it will motivate the robot to choose the actions that are reversible. The intrinsic reward signal can be generated, when a sequence of an action with its reverse-action is observed. In this case, the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ is observed and the value $v_{rev} = d_{rev}(s_{final}, s_{init})$ is calculated. The intrinsic reward can then be calculated, for example, using the following expression:

$$r = \varepsilon_{rev} - v_{rev} \ .$$

The value of the reversibility sub-module's safety assessment can be calculated as follows. The irreversibility value is measured by $Irr(s, a) = v_{rev}$ and can be transformed into the binary assessment result for the reversibility sub-module by using the $\varepsilon_{rev}$ threshold. The post-action is always the undo-action – $P_{undo}(s, a) = (-a)$, since the undo action $(-a)$ is expected to always undo the forward action $(a)$.

**Example**

It should also be explained how and why a safe behavior would emerge as a result of avoiding actions, identified as irreversible by the world-model-free approach. As an example, let's consider a robot with a proximity sensor in front and the two actions – "move 10 steps forward" and "move 10 steps backward". Without loss of generality, it can be assumed that "steps" and values of proximity sensors are given in the same units. For simplicity, let's take $\varepsilon_{rev} = \varepsilon_{orig} = 0.5$ and use the Manhattan metric for $d_{rev}$ and $d_{orig}$.

The robot tests 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 (see Fig. 3.2):

1. If the robot is at least 10 units away from the obstacle, 12 for example, then it does not touch the obstacle and we obtain the reversibility

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

   that holds, since $d_{rev}(12, 12) = 0 \le \varepsilon_{rev}$.

2. If the robot is less than 10 units away from the wall, 8 for example, then it touches the wall and its motor stall; the obtained reversibility

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

   does not hold, since $d_{rev}(10, 8) = 2 > \varepsilon_{rev}$.

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

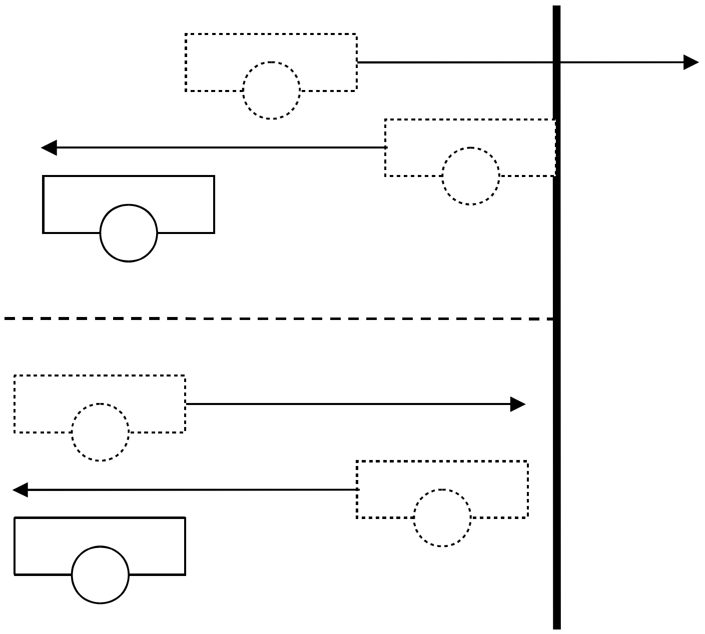

Figure 3.2: Obstacle avoidance as a consequence of suppressing irreversible actions. In the upper example the reversibility does not hold, in the lower – holds

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 an obstacle (or pushing it) 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. As a result, by avoiding irreversible state-action pairs, the robot will avoid pushing obstacles or hitting walls – the obstacle-avoidance behavior will emerge.

If the reversibility of the action $[10]$ in the state $(11.5)$ needs to be predicted, the reversibility $((12), [+10], (2), [-10], (12))$ will be used to ground the prediction, since $d_{orig}(11.5, 12) = 0.5 \leq \varepsilon_{orig}$. The predicted irreversibility value $v_{rev}$ of the state-action pair is $d_{rev}(12, 12) = 0$. The predicted intrinsic reward signal for such state-action pair can be calculated as follows:

$$r(s, a) = \varepsilon_{rev} - v_{rev} = 0.5 - 0 = 0.5 \ .$$

### 3.3.2 World-model-based Approach

This section describes the reversibility model approach to assess how reversible (or irreversible) is action $a$ in state $s$ using a model of the environment. In this approach actions and states are symbolic; they are identified by external identification routines.

To predict how reversible is action $a$ in state $s$, an external planning module is asked to provide a set of "candidate" paths (sequence of actions) to undo the action $a$. The paths in the set are tested, selecting the ones that return robot back to the state $s$. The predicted cost is calculated based on the costs of actions in the selected paths; the lower the cost, the more reversible is the action $a$ in the state $s$.

Such approach to assess reversibility is motivated by the principle of reversibility itself. It encapsulates what means to be able to undo an action – to return back to the initial state before making the action. External modules are used instead of an internal logic to make the world-model-based approach more general and applicable to many real-world problems.

In the rest of this subsection we present the definitions to formalize the world-model-based approach to assess reversibility. Some explanations and an example follow.

**Formal Framework**

*Reversibility MDP-Model (RMM)* is a finite Markov Decision Process with a reversibility function $C(s,a) \leq 0$.

MDP is a 4-tuple $(S, A., P.(.,.), R.(.,.))$, where $S$ is a finite set of states, $A_s$ is a finite set of actions available in state $s$, $P_a(s,s')$ is the probability that action $a$ in state $s$ will lead directly to state $s'$, $R_a(s,s')$ is the (expected) immediate cost of making action $a$ in state $s$, followed by a transition to state $s'$ with probability $P_a(s,s')$.

The value of $C(s,a)$ is the total expected cost of $s \rightarrow s' \rightarrow s$ transition, i.e. reversing the action $a$ made in the state $s$. $C(s,a) = -\infty$ for absolutely irreversible actions and $C(s,a) = 0$ for perfectly reversible ones.

To calculate the cost, a path $p = (a_0, a_1, .., a_n)$ must be found to make the $s \rightarrow s' \rightarrow s$ transition. This is done by iterating through the pre-selected set $P_s$ of candidate paths that have $a$ as a first action. For every candidate path $p = (a_0, a_1, .., a_n)$, the $C_p(s,a)$ value is calculated as follows:

$$C_p(s,a) = \min(\sum_0^n R_{a_i}(s_i, s'_i) \cdot P_{a_i}(s_i, s'_i) : s'_n = s) .$$

If none of the possible $s'_n$ is equal to $s$, then $C_p(s,a) = -\infty$.

The $C(s,a)$ cost is the maximum of the analyzed $C_p(s,a)$ values:

$$C(s,a) = max(C_p(s,a)) .$$

If there are no candidate paths found, then $C(s,a) = -\infty$.

Action $a$ in state $s$ is *reversible* if $C(s,a) \geq C_{rev}$. If $C(s,a) \geq C_{min} \geq C_{rev}$, action $a$ in state $s$ is called *super-reversible*.

The rationale behind introduction of the $C_{min}$ threshold is to optimize $C(s,a)$ calculation – search for the maximum can be stopped, if a path with $C_p(s,a) \geq C_{min}$ is found:

$$C(s,a) = C_{p_i}(s,a),\ if\ C_{p_i}(s,a) \geq C_{min}\ .$$

Simply put, a cost of making a sequence of actions $p = (a_0, a_1, .., a_n)$ is predicted with $C_p(s,a)$, taking in consideration only the possible outcomes where the final state is $s$. A set of paths is analyzed and the $C_p(s,a)$ value is calculated for each path. A path with the maximal $C_p(s,a)$ or the first path with $C_p(s,a) \geq C_{min}$ is the selected way to take action $a$ in state $s$ and then return back to the initial state $s$. The cost of such cycle is the value of $C(s,a)$ and the sub-sequence of actions $p_{undo} = (a_1, .., a_n)$ is called an *undo-path*.

In practice, the first candidate path to be tested is the action $a$ itself: if $C_{(a)}(s,a) \geq C_{min}$, then $C(s,a) = C_{p=(a)}(s,a)$. In this case the state $s$ was not left, thus the action $a$ in the state $s$ is reversible and the undo-path is empty. Such situation occurs if the action $a$ does not affect the environment or state identification module decides that changes in the environment are irrelevant for the context in question.

The value of the reversibility sub-module's safety assessment can be calculated as follows. The irreversibility value is measured by $Irr(s,a) = -C(s,a)$ and can be transformed into the binary assessment result for the reversibility sub-module by using the $-C_{rev}$ threshold. The post-action is always the undo-path of the best path $p$: $P_{undo}(s,a) = p_{undo}$, since the undo-path $p_{undo}$ is supposed to undo the action $a$.

### Example

It should also be explained how and why a safe behavior would emerge as a result of avoiding actions, identified as irreversible by the world-model-based approach. For example, let's consider a robot, moving in an environment with walls and movable objects, using actions of "move forward/backward" and "rotate left/right". The walls, the free space and the movable objects are identified by the external state identification module. Let's assume that the external simulation module can correctly simulate effects of the actions and that the external planning module is able to provide a set of correct action sequences to undo the given action in the given state.

Consider the robot to be in a state $s$ and willing to make an action $a$. For example, the robot is in the center of a room and environment state in this context is identified by position of movable objects in the room. If the environment simulation module predicts that none of the objects will be moved, then the initial state is not left and the action is identified as reversible.

If object movement is predicted, then "candidate" paths to undo action $a$, provided by the planning module, are simulated to find the ones that can actually undo the action. If there exists a path $p$ with high probability of success and acceptably small cost ($C_p(s,a) \geq C_{min}$) or cost of the "easiest" path is within reversibility limits ($C_p(s,a) \geq C_{rev}$), then the action $a$ in the state $s$ is identified as reversible.

### 3.3.3 Relation Between the Approaches

Despite the fact that the model-free and the model-based approaches assess safety of the action in different ways, both approaches' purpose is the same – to "implement" the reversibility sub-module of the safety module. The world-model-based approach, together with its prerequisites can be treated as an "implementation" of the world-model-free approach.

To remind the reader, the world-model-free approach is based on reverse-action pairs with state similarity measures and the following prerequisites must be selected/defined:

- Pairs of actions that reverse each other.
- Hemimetrics $d_{orig}$ and $d_{rev}$ to filter experienced reversibility objects and measure their reversibility, respectively.
- Threshold values $\varepsilon_{orig}$ and $\varepsilon_{rev}$ to classify the values calculated by the hemimetrics (to select appropriate reversibilities to ground the prediction and to distinguish between reversible/irreversible actions, respectively).
- The *revesibility model* – a set of the experienced reversibilities.

The rest of the subsection describes how these prerequisites can be implemented using the world-model-based approach and its external modules.

#### Pairs of reverse-actions

The reverse-action pairs can be given by designer in advance. Alternatively, they can be inferred from the model of the environment or state/action identification logic.

#### Hemimetrics

There are multiple ways to calculate the required hemimetrics $d_{orig}(x,y)$ and $d_{rev}(x,y)$, which include, but are not limited to:

- discrete metric on states (by state identification logic):

$$d(x,y) = \begin{cases} 0 \text{ if } x = y \\ 1 \text{ if } x \neq y \end{cases}$$

- by taking module of the cost of the $x \to y$ transition, calculated similarly to $C(s,a)$ calculation for $s \to s' \to s$ transition

## Thresholds

The threshold values $\varepsilon_{orig}$ and $\varepsilon_{rev}$ can be given by designer in advance or derived from values of the $C_{rev}$ and the $C_{min}$ thresholds. In case of discrete metrics, the thresholds can be set, for example, to $\varepsilon_{orig} = \varepsilon_{rev} = 0.5$.

## Set of Experienced Reversibilities

The experienced reversibilities can be obtained by internal simulation and in the actual test runs.

# 4 Experiments

This chapter describes the implementation of the safety architecture and the two approaches described in the previous chapter. It also presents the test results of assessing action reversibility using the proposed safety architecture. Section 4.1 describes two experiments to test the world-model-free approach for assessing action reversibility. The experiment with the proposed safety architecture using the world-model-based reversibility assessment is described in Section 4.2.

## 4.1 World-model-free Reversibility Assessment

### 4.1.1 Reversible Self-Movement

In this section we describe the experiment, where we create the reversibility model and test its performance. In the experiment we compare the collision prediction success rate of the world-model-free approach with performance of a model-free reinforcement learning algorithm. The experiment is also presented in [62] and the reader can refer to it for more details.

The experiment consists of two test runs to analyze the performance of the world-model-free reversibility assessment on two robots of different size with different sensor setup and number of sensors. The first test run is conducted on a Khepera II mini-robot, simulated with the Gazebo simulator from the Player/Stage project [63]. In the second test run, the same simulator is used to test the performance of a simulated Scitos G5 robot. Each test run is divided into two parts: data collection (part 1) and simulation (part 2).

In the first part, 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 in the second part using the collected data. The performance is measured by sampling algorithms' predictions of whether the next action will result in collision, followed by calculation of the success rate of those predictions.

#### Robots and Environments

Both Khepera II and Scitos G5 are differential drive robots with considerably different size and slightly different geometry. Khepera II has a circular shape and the rotation axis is exactly at the center 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 $200mm$ (the size of the passive wheel compartment).

The robots are simulated with the Gazebo simulator (version 0.8-pre3, OGRE version 1.4.9, ODE version 0.10.1) through the Player control framework (modified version 2.1.0) [63]. The environment for Scitos G5 is a rectangular box of size 970 $mm$ by 1500 $mm$, shown in Fig. 4.1 (on the right). Only 22 of 541 laser rays are simulated to optimize simulation performance. Khepera II's infrared sensors are simulated by 8 short laser rays distributed evenly around the robot with the maximum measurable distance of $100mm$. The environment for Khepera II simulated environment is a right-angled triangle box with side lengths of 196 $mm$, 125 $mm$ and 233 $mm$, shown in Fig. 4.1 (on the left).

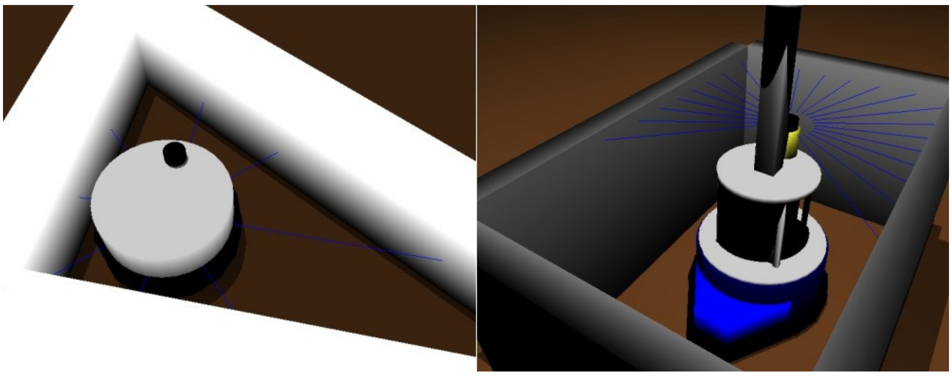
Figure 4.1: Environments for the experiments – simulated Khepera II is on the left, simulated Scitos G5 is on the right

## Robot Movements

In the experiment 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 is given a set of actions with corresponding reverse-actions: movements forward-backward and turning left-right of the same length are pair-wise reverse-actions of each other. The following discrete set of actions is used in the experiments: $F$ – make a step forward, $B$ – make a step backward, $L$ – rotate counter-clockwise, $R$ – rotate clockwise , where $F = -B$, $B = -F$, $L = -R$ and $R = -L$.

Actions are defined in terms of commands to move forward/backward or rotate left/right. An action $a = [m_{trans}, m_{rot}]$ consists of a pair of target movement deltas, $m_{trans}$ is measured in meters and $m_{rot}$ is measured in degrees. See Table 4.1 for the values selected for the actions $F$,$B$,$L$ and $R$.

| **Action** | **Khepera II movement** | **Scitos G5 movement** |
|---|---|---|
| **F** | [+0.016,0] | [+0.15,0] |
| **B** | [-0.016,0] | [-0.15,0] |
| **L** | [0,+30] | [0,+42] |
| **R** | [0,-30] | [0,-42] |

Table 4.1: Values of the actions' movement commands $[m_{trans}, m_{rot}]$

A robot moves using the algorithm described in Fig. 4.2. It makes a random action 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 measure the reversibility, which can be used as a basis to generate the intrinsic reward signal afterward. The purpose of the last random action without a matching reverse-action is to make the robot explore the environment.

1. Record the 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 of $a_i$: $a_{i+1} = -a_i$ .
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 (goto 1).

Figure 4.2: Robot movement algorithm to collect data

**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. This is done by generating an intrinsic reward signal based on the distance between the final and the initial states. The algorithm is described in Fig. 4.3. It takes a sequence of states and actions as an input: $s_0, a_0, s_1, a_1, s_2, a_2, s_3, a_3, \dots$ .

At every $i \geq 2$, 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 reversibility model, which is a set of reversibilities.

To predict the outcome of making the action $a_t$ from the state $s_t$, the intrinsic reward is calculated as an expected irreversibility value $v_{rev}$ using a set of

1. Read the current state $s_i$ and the next action $a_i$ from log.
2. Select a number of reversibilities from the set of experienced ones with $a_{forward} = a_i$, based on $d_{orig}(s_i, s_{init})$ of the experienced reversibility.
3. If no reversibilities are selected, make no prediction.
4. Calculate the expected irreversibility value $v_{rev}$ using $d_{rev}(s_{init}, s_{final})$ of the selected reversibilities.
5. If $v_{rev} > \varepsilon_{rev}$, then predict negative outcome (collision), otherwise positive outcome (no collision) is predicted.
6. If $i < 2$, add 1 to $i$ and repeat (goto 1).
7. Read the last action as $a_{i-1}$ and the previous action $a_{i-2}$ from log.
8. If $a_{i-1}$ is reverse-action of $a_{i-2}$, add the new obtained reversibility $(s_{i-2}, a_{i-2}, s_{i-1}, a_{i-1}, s_i)$ to the set of experienced reversibilities.
9. Add 1 to $i$ and repeat (goto 1).

Figure 4.3: Reversibility model creation algorithm

reversibilities, selected from the reversibility model.

The selected reversibilities are the ones with the same forward action and $d_{orig}(s_t, s_{init}) < \varepsilon_{orig}$, where $s_{init}$ is the initial state of the reversibility under consideration. In the experiment, the value of $v_{rev}$ is a weighted average of $d_{rev}(s_i, s_{i-2})$ values of the selected reversibilities. Reversibilities are sorted by $d_{orig}(s_t, s_{init})$ in an ascending order and their weights are $1/i^3$ (1, 1/8, 1/27, 1/256, etc) – reversibilities with a "closer" initial state have considerably stronger influence. The prediction decision is calculated as $sign(\varepsilon_{rev} - v_{rev})$ – the negative value means the negative prediction (collision), the positive value means the positive prediction (no collision). If no reversibilities could be selected and the $v_{rev}$ value is undefined, then no prediction is made.

## Q-learning algorithm

The main difference between the reversibility-based algorithm and the Q-learning algorithm [64] is that the latter receives an external reward signal indicating the outcome of the action. The reversibility-based algorithm, on the other hand, uses only sensor data to determine whether the irreversible (e.g. collision) will occur or not. The experimental test runs consist of random movements, therefore, the long-term reward component of the classical Q-learning update expression is discarded;

expected reward of the state-action pair is updated using the following expression:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)[r_t - Q(s_t, a_t)] \ .$$

The prediction decision is calculated as $sign(Q(s_t, a_t))$ – the negative $Q$ value means the negative prediction (collision), the positive $Q$ value means the positive prediction (no collision). Initially, $Q$ values are set to 0 and if $Q(s_t, a_t) = 0$, then no prediction is made.

### Implementation details

Sensor values for the Scitos G5 are in meters, therefore they are multiplied by $1000$ to be of the similar scale to the ones of the Khepera II. This does not affect the reversibility based algorithm, but makes saving and loading the log files simpler.

In the experiment the $\alpha_t(s_t, a_t)$ value of the Q-learning update expression is constant and is set to $0.01$. The Euclidean metric is used to calculate $d_{orig}$ and $d_{rev}$ in the experiment; the values $\varepsilon_{orig}$ and $\varepsilon_{rev}$ are finite and selected manually. Threshold values $\varepsilon_{orig}$, $\varepsilon_{rev}$ and the tile size of discrete state identification for the Q-learning algorithm are constant:

- $\varepsilon_{orig} = 11000$,
- $\varepsilon_{rev} = 10000$,
- $RLtilesize = 168$.

### Results

During the test runs the two methods are predicting collisions of simulated robots with simulated obstacles (walls). Figures 4.4 and 4.5 represent the test results for the Khepera II and the Scitos G5 respectively.

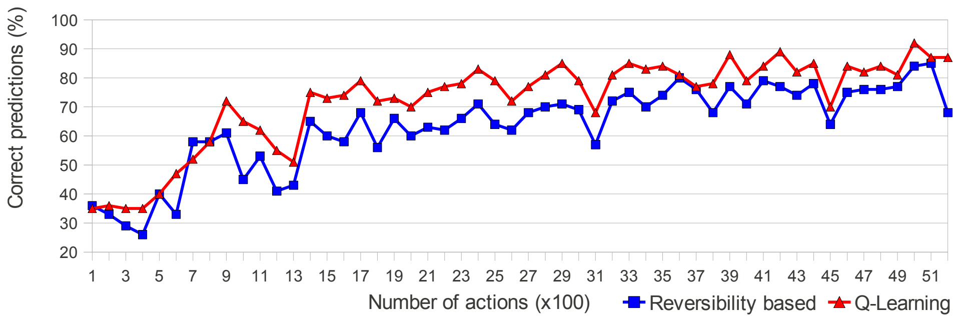

Figure 4.4: Results of the prediction performance test with the simulated Khepera II robot

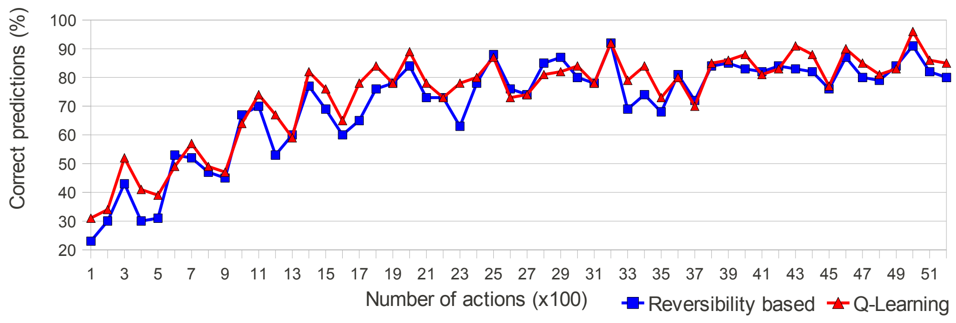

Figure 4.5: Results of the prediction performance test with the simulated Scitos G5 robot

The rate of correct predictions is calculated by sampling how many predictions out of 100 are correct. If no prediction is made, then it counts as incorrect prediction. The downward spikes in prediction rate graphs are caused by novelty of the states, since no grounded prediction can be made for such unvisited states.

The rates of correct prediction of both algorithms start at 20–40% and gradually reach the level of 70–90% after 3900 steps. In comparison to the Q-learning algorithm, the performance of the reversibility-based algorithm is about 10% lower for Khepera and about 5% lower for Scitos.

The similar performance of the reversibility-based algorithm on two robots of a different size and sensor setup shows that the motivation to take reversible actions is abstract and the reward signal is intrinsic. However, the reward signal is still based on the sensor values, requiring to select the hemimetrics and the thresholds manually, which makes algorithm's implementation somewhat dependent on the physical embodiment and the task at hand.

### 4.1.2 Identification of Reverse-Action Pairs

In this section we describe how reverse-action pairs, used in the experiment described in Section 4.1.1, can be identified using the same theoretical framework of the world-model-free reversibility assessment. The following experiment is also presented in [65] and the reader can refer to it for more details.

The method for identification of reverse-action pairs is based on analysis of reversibility of two consecutive actions. It works on a data set of states and actions, acquired during a test run with randomized selection of actions from the set of rotational (rotate left/right X radians) and translational (move X meters forward/backward) actions of different lengths. Reversibility of all consecutive pairs of actions is calculated and results are divided into different sets. These sets are then analyzed to identify specific pairs of actions to reverse each other, as well as a general rule to generate such pairs.

## Experimental Setup

In the experiment the simulated Scitos G5 robot is placed into the big room to allow collision-free movement (see Fig. 4.6). The environment is simulated using the Player platform and the Stage multi-robot simulator [63].

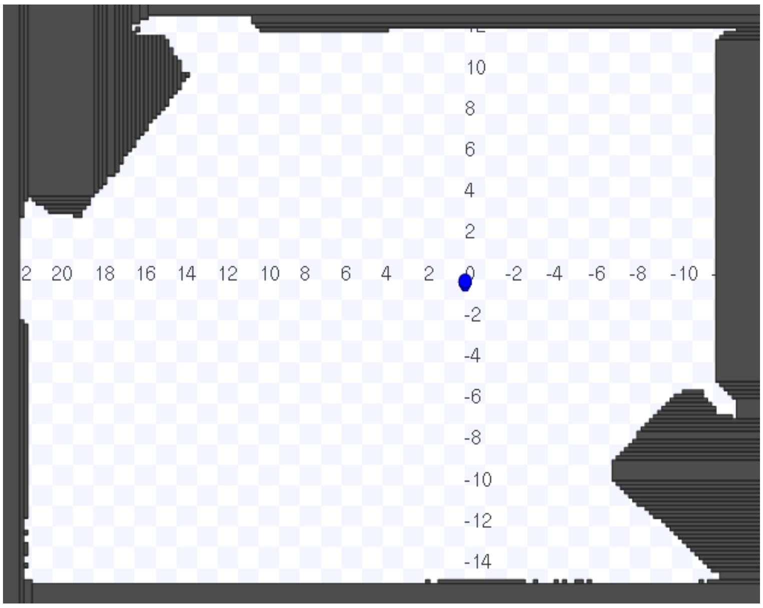

Figure 4.6: The simulated environment. Blue figure at $(0, 0)$ is the simulated Scitos G5 robot at its initial position

The laser rangefinder of the robot is configured to observe the sector of $[-105°, +105°]$ in front of the robot with $0.5°$ precision. The state is identified by the laser rangefinder data: $s = (d_0, d_1, ..., d_{420})$. The action is a two-dimensional command to move or rotate: $a = [m_{trans}, m_{rot}]$ , where $m_{trans}$ is the translational movement in meters and $m_{rot}$ is the degree of rotation in radians.

The experiment begins with the test run, where the robot moves randomly. The actions are generated in such a way, that either $m_{trans}$ or $m_{rot}$ is zero. First, it is decided, whether the robot moves forward/backward ($m_{rot} = 0$) or turns left/right ($m_{trans} = 0$) with equal probability. Then, the value of $m_{trans} \in [-1.0, +1.0]$ or $m_{rot} \in [-1.57, +1.57]$ is selected randomly from the respective interval. The laser rangefinder data before and after making every action is saved to a log-file together with the executed action data.

The acquired data is then analyzed off-line by iterating through the log-file. There are several sets of data to be analyzed. Each single action is analyzed to measure how state changes as a result of the action. For this purpose, the $d_{rev}$ of the states after and before the action is calculated for separate sets of translational and rotational actions.

Similarly, pairs of consecutive actions are analyzed as reversibility "candidates", divided into the three different sets:

- "TT" – translational+translational
- "RR" – rotational+rotational
- "RT-TR" – rotational+translational or translational+rotational

**Results**

The result of the analysis of single actions is shown in Fig. 4.7. The rotational actions are shown as black triangles, while gray diamonds represent the translational actions. The horizontal axis is the non-zero component of the action: $m_{rot}$ for the rotational and $m_{trans}$ for the translational actions. The vertical axis is the $d_{rev}$ of the states after and before taking the action.

Fig. 4.7 shows that numerical influence of robot's rotation on the change to robot's state is considerably higher than of robot's translational movement. Rotational movements generate ca 10 times bigger $d_{rev}$ than translational ones; therefore the two different thresholds are used to interpret the rest of the results. The first threshold ($\varepsilon_{revR} = 100$) is for sequences with at least one rotational action, and the second one ($\varepsilon_{revT} = 10$) is for translational-only sequences.

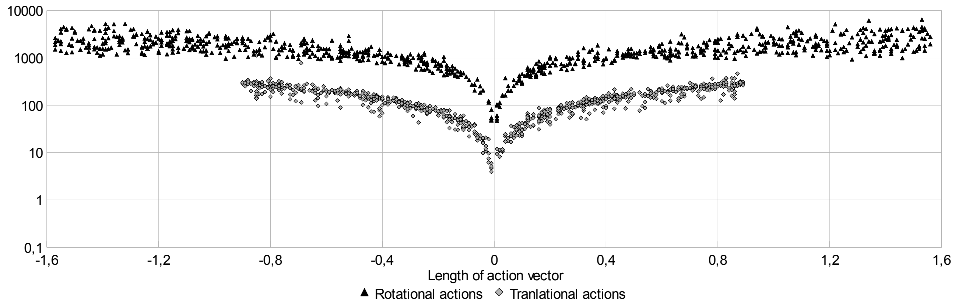

Figure 4.7: Calculated $d_{rev}$ of the states after and before making the action

Figures 4.8, 4.9 and 4.10 visualize the results of analysis of reversibility "candidates" for "TT", "RR" and "RT-TR" data sets respectively. Similarly to Fig. 4.7, $d_{rev}$ of the states after and before the action is on the vertical axis. The difference is that the action now consists of two consecutive actions $p = (p_{trans}, p_{rot})$ and $q = (q_{trans}, q_{rot})$. The horizontal axis is the "Manhattan" length of the sum of action vectors:

$$d_{act}(0, p + q) = |p_{trans} + q_{trans}| + |p_{rot} + q_{rot}| \ .$$

Since one of the components is zero in the "TT" and "RR" data sets, values on the vertical axis in figures 4.8 and 4.9 are $|p_{trans}+q_{trans}|$ and $|p_{rot}+q_{rot}|$ respectively.

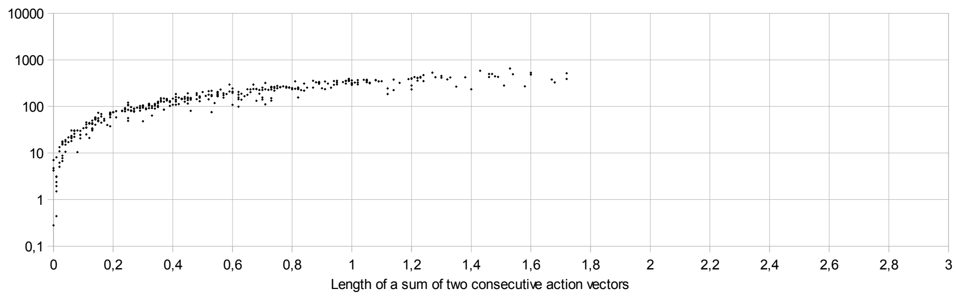

Figure 4.8: Calculated $d_{rev}$ of the final and the initial states of two consecutive *translational* actions (TT)

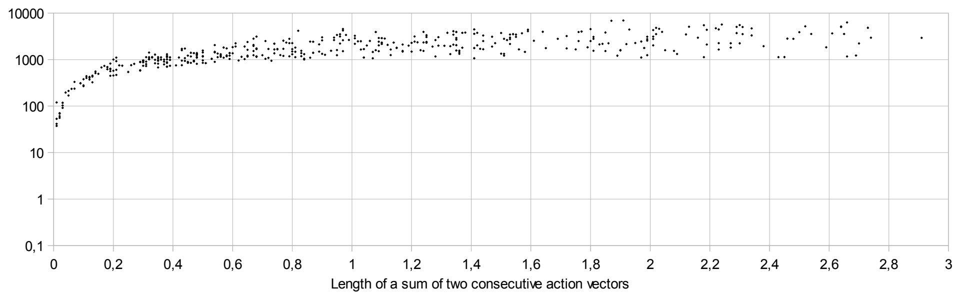

Figure 4.9: Calculated $d_{rev}$ of the final and the initial states of two consecutive *rotational* actions (RR)

Figures 4.8 and 4.9 show that the data sets “TT” and “RR” exhibit strong dependency of $d_{rev}$ from the value of $d_{act}(0, p + q)$. The closer to 0 is the sum of consecutive rotations or translations, the less is the distance between the states after and before the action $p + q$. In other words, irreversibility of the sequence of actions $p + q$ is proportional to its length. Considering the fact that one of the action’s components is zero in the “TT” and “RR” data sets, a general rule can be derived: “if moved/rotated by X, move/rotate by -X to undo”. Further, the actually experienced pairs of actions can be identified by applying the $d_{revT}$ and the $d_{revR}$ thresholds to “TT” and “RR” data sets respectively.

The “RT-TR” data sets differs from the “TT” and “RR” data sets by the fact that the two consecutive actions are of different type and modules of both rotational and translational parts are added during the $d_{act}(0, p + q)$ calculation. The analysis of the results for “RT-TR” data set predictably reveals no strong dependency of $d_{rev}$ from the value of $d_{act}(0, p+q)$, and no general rules or specific pairs of actions from this set can be identified as reverse-action pairs. Although, there are several points below the $\varepsilon_{revR}$ threshold, these points represent the actions with a very small rotational action and a bigger translational action.

The results of the experiment show that it is feasible to use the proposed

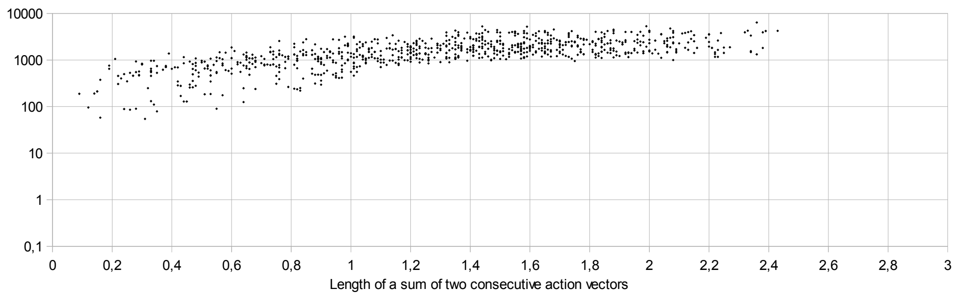

Figure 4.10: Calculated $d_{rev}$ of the final and the initial states of consecutive *rotational+translational or translational+rotational* actions (RT-TR)

world-model-free approach to identify pairs of reverse-actions, based on raw sensor data.

## 4.2 Safety Module With World-model-based Reversibility Assessment

### 4.2.1 Reversible Object Manipulation

In this section we describe the experiment, where the principle of reversibility is applied to assess intrinsic safety of actions and to adapt robot's behavior for the task of covering an area with a movable object inside. The safety module with the reversibility sub-module is acting as a governor, which allows pushing an object only if such an action is reversible. The experiment is also presented in [66] and the reader can refer to it for more details.

#### The Robot and The Environment

The MetraLabs' Scitos G5 robot is used in the real-world experiment and its model is used in the simulated environment, which is a copy of the actual room, where the real robot is operating (see Fig. 4.11). The robot control framework is connected to the Player server, which in turn controls either the actual robot or its model in the Stage simulator [63].

#### The Task

Robot's behavior mimics a vacuum cleaner. The task for the robot control algorithm is to cover the area using the "lawnmower" pattern. During the experiment the covered area is measured together with the difference between the final and the initial positions of the movable object in the global reference frame. The experiment is conducted with a single object to simplify the setup and make the object identification more robust.

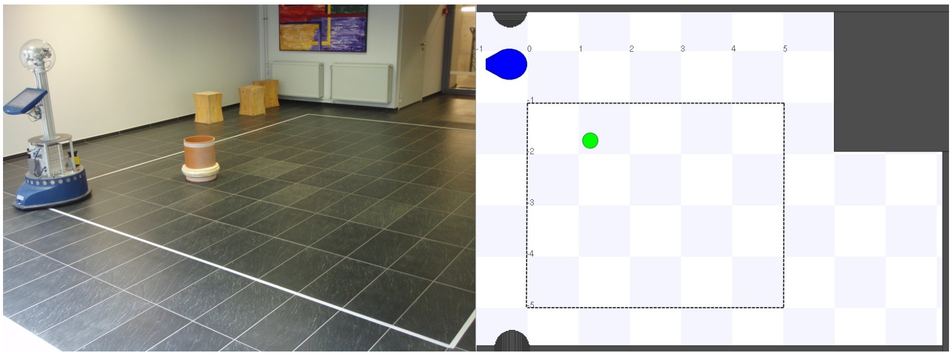

Figure 4.11: The physical experimental setup of the real room (on the left) and the simulation in Player/Stage (on the right); the dotted line and the tape mark the desired area to be covered

To benchmark the hybrid safety approach with the world-model-based reversibility assessment, the experiment consists of three sets of test runs with different modes (sets of extrinsic rules) of the safety module to compare their performance:

- “reversibility” – with the empty set of rules, the decision and the post-action are generated by the reversibility module
- “obstacle avoidance” – the extrinsic rule allows only the actions that don’t move the object
- “no safety” – the extrinsic rule allows all actions

The experiment starts with the data acquisition part to collect statistical data about the robot and the environment. It is followed by the data processing part to calculate statistics and prepare candidate cycles for $C(s, a)$ calculation. When data is collected and processed, the actual test runs are executed to measure the covered area and the distance between the initial and the final positions of the object.

### Data Acquisition

In this experiment the robot can choose between the actions described in Table 4.2. During the data acquisition part of the experiment the robot takes random actions to collect statistical data about those actions. For every action, the initial and final coordinates together with orientation of the robot are saved for both local and global odometry. The object’s movement data in the robot’s reference frame is also collected to gather statistics on how robot’s actions influence the object.

| Action | Description |
|---|---|
| **f** | move 0.15 meters forward |
| **F** | move 0.60 meters forward |
| **b** | move 0.15 meters backward |
| **B** | move 0.60 meters backward |
| **L** | rotate 60 degrees counter-clockwise |
| **R** | rotate 60 degrees clockwise |

Table 4.2: Actions used in the experiment

### Data Processing

After the data is acquired, it is processed off-line before the test runs. First, for each of the actions, robot movement statistics is calculated from the collected data – average and standard deviation of the covered distance along robot's X and Y axes together with the angle of rotation. Then, the internal model of the environment is created based on the collected data and tweaked manually to simulate resulting object movements as precisely as possible.

The tweaked model is then used to prepare the candidate cycles for on-line $C(s, a)$ calculation during the test run. It is done similarly to [67], but in this experiment hundreds of relative positions of the object in robot's reference frame are analyzed. The area around the robot is divided into cells of the size of $0.03$ meters. The centers of the cells are the analyzed points, tested for action reversibility with the object in that position. The search for cycles is conducted for those points where the object position changes as a result of the action; separate sets of such points are created for every action of the available six. On our test machine (Intel Core i7-920 CPU, NVIDIA Tesla C1060 GPU, 6GB of RAM) the calculation takes approximately one hour for the simulated and four hours for the real environment.

### Test Runs

The final part is the actual experiment, which consists of multiple test runs with different modes of the safety module. In the beginning of each test run, the object is placed randomly inside the desired area; the covered area together with the distance between the final and the initial object positions is measured in the end.

The area coverage algorithm for all three "modes" is the same, it covers the area in the "lawnmower" fashion and queries the safety module before taking an action. If the action is prohibited, an alternative action is chosen and the query is repeated. If the new action is allowed, it is executed together with the post-action, supplied by the safety module.

In the "reversibility" mode no extrinsic safety rules are applied and the reversibility module is responsible for safety assessment. The cycles, generated in the off-line data processing part of the experiment, are used during the on-line assessment of action reversibility.

The analyzed point, closest to the current position of the object, is selected from action's set of such points. Each analyzed point has a set of the previously identified cycles associated with it, these "candidate paths" are used to calculate $C(s, a)$ and select the best cycle as a basis for the post-action.

During the on-line safety assessment, the paths are simulated internally, now taking possible immovable obstacles into account.

The "obstacle avoidance" mode uses the same internal model of the environment to predict collisions. The action is prohibited if a collision is predicted, allowed otherwise; the post-action is always empty. The "no safety" mode is the most naive approach – all the actions are allowed and the post-action is always empty.

### Technical Details

The test area is $5$ by $4$ meters, it is divided into $80$ cells of the size of $0.5$ meters. The cell is considered to be covered, if the center of the robot's round compartment enters it.

Robot coordinates in the global reference frame are provided by the AMCL driver of the Player/Stage project [63]. The round object's size and its position in robot's coordinates are identified from laser rangefinder scans, filtering out the occasional "wrong" objects by the radius threshold. State space's 2D position component is divided into $0.1$ $m$ cells and orientation component is divided into $0.173$ $rad$ sectors. The movable object is round; therefore, its orientation is always 0.

Threshold values are set as follows: $C_{min} = -2$ and $C_{rev} = -20$. The search for cycles is a plain iteration over the possible paths with lengths of up to $14$ actions. During the on-line $C(s, a)$ calculation, a maximum of 8192 path candidates are tried. No run-time limits are enforced, since execution times are limited appropriately by the previously described parameters.

### Results

Fig. 4.12 shows the results of the test runs made in simulation (to the right) as well as on the real Scitos G5 robot (to the left). Each sub-figure contains the results of the 5 test runs conducted with the 3 different modes of the safety module.

The red triangles represent the results of the trials, when all the actions are allowed – the "no safety" mode. It is easy to see that this mode is the least safe of all the three – in the end the object is on average $1.5$-$1.9$ meters away from its initial position. As expected, this mode performs very well in terms of area coverage – all of the 80 cells are visited during both real and simulated test runs. In the real environment the object is moved further away from its initial position than

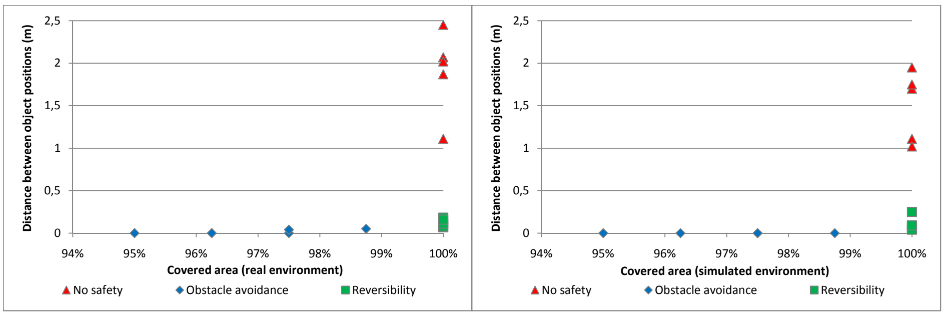

Figure 4.12: Results of the test runs in the real *(left)* and the simulated *(right)* environments

in the simulated environment. This is due to the difference in physical parameters of robot-object interaction.

The results of the "obstacle-avoidance" strategy, where only the actions that don't move the object are allowed, are shown as blue diamonds. This mode is in many ways the opposite of the "no safety": it is the safest mode with almost no object movements, but with incomplete area coverage – at least one and up to four cells remain unvisited. It is worth noting, the test runs in the real environment exhibit small movements of the object in this mode due to imperfect collision prediction and sensor noise.

Green squares represent the test runs in the "reversibility" mode – when decision about the action and the post-action is made by the reversibility sub-module. This mode performs on a par with the "no safety" mode in terms of area coverage – all the 80 cells of the area in question are covered. Also, this mode is safe – in the end the object is very close to its initial position after both the real and the simulated test runs.

With respect to the total navigation time, the fastest mode is predictably the "no safety", since the robot drives straight through the cells in the "lawnmower" fashion. The "reversibility" mode is only 3-10% slower than the "no safety" – the additional time is spent only when driving around the obstacle and pushing it back. The slowest mode is the "obstacle-avoidance" – it is at least 50% slower than the "no safety" mode. In this mode the next unvisited cell is tried many times from different positions, before it is skipped.

Based on the results of the experiment, we conclude that the proposed safety architecture, consisting of the safety module and its reversibility sub-module, can be used for practical purposes. The experiment showed that the principle of reversibility can be successfully used to ground safe object pushing behavior to increase covered area without compromising safety.

# 5 Summary of Publications

This chapter gives a short review of the publications related to this thesis. The first three papers in sections 5.1, 5.2 and 5.3 are dedicated to the world-model-free approach to assess action reversibility. Sections 5.4 and 5.5 describe the papers dedicated to the world-model-based approach.

## 5.1 Don't Do Things You Can't Undo: Reversibility Models for Generating Safe Behaviours (Paper Summary)

In this paper the abstract principle "Don't Do Things You Can't Undo" is used to govern the robot's behavior and the obstacle avoidance behavior emerges when robot suppresses irreversible actions. The idea was initially proposed in [10] by Eppendahl and Kruusmaa and this article extends the idea (by introducing the initial theoretical framework to assess action reversibility), confirms the reported results and demonstrates the efficiency of the proposed approach. The test results are evaluated in a two-dimensional environment on Khepera II mini-robot developed by K-Team. Performance of the reversibility-based algorithm to predict collisions is compared to the Q-learning algorithm as a benchmark.

It is argued in the introduction that the principle of reversibility is suitable to be one of the basic principles to ground robot behavior. It is also argued that the code based on such principles without reference to the ground meaning of sensor-motor values should function reliably in a broad range of environments and on different robots.

Instead of programming a robot with specific routines to avoid collisions, falls, etc., it is proposed to program a robot with the general principle of avoiding irreversible actions. This way the robot is told not *what* to do, but *why* something should be done. For example, locking a door without the knowledge how to unlock it is intrinsically unsafe, because the robot gets trapped and its batteries are depleted as a result.

Section 2 presents the reversibility model that tells the robot which actions are reversible and how to reverse them if they are – an initial formalization of the proposed ideas about reversibility.

Section 3 describes the experiments with the Khepera II robot moving inside two different environments – the "easy" rectangular and the "harder" triangular cardboard boxes. For each environment two sets of reverse-action pairs are used –

the one-dimensional experiments use short and long steps forward and backward, while the two-dimensional experiments also include rotational movements.

In all 4 experimental test runs the robot moves pseudo-randomly by repeatedly executing a random action together with the appropriate reverse-action first and then making another random move to explore the environment. Such action selection pattern generates at least one reversibility to be added to reversibility model of the robot. For each first action $a$ in the pattern iteration, the reversibility together with the $d_{val}$ value (the distance between the initial and the final state after making actions $a$ and $-a$) is added to the reversibility model.

This model is then used to predict whether the current action in the current state will result in collision. Rate of correct predictions for the reversibility-based algorithm is compared to the Q-learning algorithm, predicting the sign of the reward for making the action $a$ in the state $s$. The environment-based reward is calculated as follows: a positive reward is proportional to length of the collision-free movement and a negative reward is provided for hitting an obstacle.

Section 4 describes test results of the experiments. Performance of the reversibility-based algorithm is very close to the Q-learning algorithm for both experiments in the "easier" rectangular environment. In the "harder" triangular environment with higher probability of collisions, results of both 1D and 2D test runs show that performance of the reversibility-based algorithm is only 5-10$\%$ lower than of the Q-learning algorithm. The results are interpreted as positive, since their performance is comparable, but the latter algorithm uses collision signal to predict collisions, while the former one uses only sensor data to ground its predictions.

It is concluded that both methods perform more or less equally, converging to a satisfactory performance. The results suggest that the efficiency of the reversibility-based collision predictor is comparable to the reinforcement learning approach.

This paper is referenced in the thesis as [11] and is included in full as Appendix A. It was presented by the first author (Maarja Kruusmaa) during the 2007 International Conference on Robotics and Automation (ICRA 2007).

## 5.2 Emergence of Safe Behaviors with an Intrinsic Reward (Paper Summary)

In this article the theoretical framework of the approach is finalized and tested on two simulated robots as well as a real robot pre-loaded with the simulated experience.

In the introduction, it is argued that it is useful to treat intrinsic motivation of the robot to reverse actions as an intrinsic reward signal to robot learning. The rationale for such a motivational system is to teach the robot to behave safely. A robot governed by such an intrinsic motivation will behave inherently safely as it will prefer actions that don't cause irreversible damage.

Section 2 presents the theoretical framework of the world-model-free approach. The definition subsection defines a world $W$ as a labeled transition system, consisting of the experienced states, actions as labels and the experienced labeled transitions. A *reversibility* is defined as a 5-tuple consisting of the initial state $s_{init}$, the forward action $a_{forward}$, the intermediate state of the transition $s_{interim}$, the reverse-action $a_{reverse} = -a_{forward}$ and the final state $s_{final}$. The reversibility model $R$ is defined as a set of experienced reversibilities in the world $W$.

The $d_{orig}$ distance measure with the $\varepsilon_{orig}$ threshold value are used to select the reversibilities with similar initial state. The $d_{rev}$ distance measure with the $\varepsilon_{orig}$ threshold are used to assess if the reversibility holds by calculating the distance between the initial and the final state.

Section 3 describes the experimental setup with implementation details and detailed descriptions of the used algorithms. The paper contains three experiments. Tests 1 and 2 are made to analyze the performance of the reversibility-based assessment on two different robots of different size with different sensor setup and number of sensors. The purpose of the test 3 is to check the hypothesis that the real robot can perform comparably to the simulated one, if it starts the test run pre-loaded with the simulation-based experience. The test 1 is made on the Khepera II mini-robot, simulated with the Gazebo simulator from the Player/Stage project. In the test 2, the same simulator is used to test the performance of the simulated Scitos G5 robot, which is considerably larger than the Khepera and the field of view of its sensors cover only 270 degrees, while simulated Khepera has a 360-degree coverage.

Similarly to [11], in all the three tests a robot is moving inside a box using pairs of reverse-actions – move forward/backward and turn left/right. Each test run starts with the data-collection part, where a robot is making actions and data is saved to be analyzed in the data-analysis part. In the first part the robot moves pseudo-randomly, selecting a random action, executing it together with an appropriate reverse-action and then making another random move to explore the environment. As a result of such moving pattern, at least one reversibility is generated per each iteration.

In the data-analysis part of a test run, saved data is loaded into memory to assess performance of the collision prediction rates for the reversibility-based and the Q-learning algorithms. Both algorithms predict the sign of the reward for making the action $a$ in the state $s$ for every state-action pair in the saved sequence of experienced states and actions.

The main difference between the Q-learning algorithm and the reversibility-based algorithm is that the former receives an external reward signal indicating success of the action. The reversibility-based algorithm, on the other hand, uses only sensor data to determine success of the action (which may also be interpreted as an intrinsic reward arising from the similarity of the initial and final states).

Section 5 describes and discusses the test results. The results of the first two experiments show that the performance of the reversibility-based algorithm is about 10% lower for Khepera and about 5% lower for Scitos, than the Q-Learning algorithm's results. The results of the test 3 report success rate of 65-70% from the very beginning of the test run. During the experiment, the performance improves further reaching to the success rate of the simulated run (ca 80%).

It is concluded that the similar performance of the reversibility-based algorithm on two robots of different size and sensor setup shows that the goal to learn to reverse actions is abstract and the reward signal is intrinsic. The third experiment showed that the reversibility model can be learned in simulation to increase the safety of robot learning and then be corrected further within the physical robot.

This paper is referenced in the thesis as [62] and is included in full as Appendix B. It was presented by the first author (Juri Gavšin) during the 2011 International Conference on Adaptive and Intelligent Systems (ICAIS 2011).

## 5.3 Identification of Reverse-Action Pairs using Reversibility of Actions (Paper Summary)

This paper describes a method for identification of reverse-action pairs through analysis of reversibility of two consecutive actions. The method works on a data set of states and actions acquired during a test run with a randomized action selection. The experiment with a simulated Scitos G5 robot suggests that such pairs can be identified from a set of rotational (rotate left/right X radians) and translational (move X meters forward/backward) actions of different lengths. A general rule to generate such pairs can also be inferred from the experimental data.

The introduction gives motivation for the work by pointing out that the previous research, including authors', selected the reverse-actions, or undo-actions, manually.

Section 2 explains the concept of reversibility and describes the theoretical framework (as described in Section 5.2) and the approach to identify pairs of reverse-actions. It is proposed to use a new metric, which determines the difference between the two actions: $d_{act} = d_{rev} = d_{manhattan}$ , where $d_{manhattan}$ is the Manhattan metric. First, a consequence of making every single action is measured to estimate how much the action alters the environment, using the $d_{act}$ metric and to select the $\varepsilon_{rev}$ threshold. Second, each pair of consecutive actions together with surrounding states is treated as a reversibility "candidate" and the $d_{rev}(s_{final}, s_{init})$ is calculated for each "candidate". The approach can be viewed is a reverse process of checking if reversibility holds for a pair of predefined reverse-actions – "candidates" with low $d_{rev}(s_{final}, s_{init})$ are sampled to analyze how to derive $a_{reverse}$ from $a_{forward}$, so that $a_{reverse} = -a_{forward}$.

Section 3 describes the experimental setup and details of the experimental procedure. In the simulated experiment a Scitos G5 robot with a laser rangefinder is placed into a collision-free environment of a large room. The experiment begins with a test run, where the robot is moving randomly. The actions are generated in such a way, that a single action is either rotational or translational.

Several sets of data with different types of actions are analyzed during the course of the experiment. Single actions are analyzed to estimate how much the action alters the environment using the $d_{rev}$ metric. Pairs of actions are analyzed for reversibility using the $d_{rev}$ metric, separating pairs of actions with the same type from the mixed ones.

Section 4 presents the experimental results and Section 5 contains discussion. Analysis of the data sets containing single actions reveals that the numerical influence of robot rotation on the change to robot's environment is considerably higher than of robot's translational movement. Therefore, two different thresholds are used for interpretation of the results for pairs of actions.

The data sets containing pairs of actions of the same type exhibit strong dependency of reversibility from the length of sum of actions – the closer it is to 0, the less is the distance between the states before and after making a pair of actions. Considering the fact that in these data sets one of the action's components is zero, a general rule can be derived as "if moved/rotated by X, move/rotate by -X to undo".

Analysis of the data set with pairs of actions of different type reveals no strong dependency of reversibility from the length of sum of the actions. Therefore, no general rules or specific pairs of actions can be identified as reverse-actions using this set.

It is concluded that it is feasible to use the proposed approach to identify pairs of reverse-action, based on raw sensor data with no prior knowledge about the environment.

This paper is referenced in the thesis as [65] and is included in full as Appendix C. It was presented by the first author (Juri Gavšin) during the 2011 International Conference on Systems, Man, and Cybernetics (SMC 2011).

## 5.4 Assessing Safety of Object Pushing Using the Principle of Reversibility (Paper Summary)

This paper presents a theoretical framework for the world-model-based approach to reversibility assessment as well as the safety module for robot control architecture. A practical experiment is conducted to demonstrate that robot control architecture can develop complex safe behaviors. This is accomplished by application of the reversibility assessment, based on external planning, environment simulation and state/action identification modules. As the result, the robot can identify, for example, that pushing object into a corner is irreversible and thus unsafe.

The introduction gives motivation for the work by pointing out that the direct sensor-data approach of authors' previous research limited the applicability of the world-model-free approach. It is argued that the combination of the abstract principle of reversibility with human-based knowledge enables development of smarter behaviors, better applicable to complex scenarios. Abstract principles, on the other hand, can govern robot behavior in the situations, where no predefined rules can be applied.

Section 2 describes the control system and its safety module, including the reversibility-based sub-module to assess action safety. The safety module ensures that system's decisions are safe and within predefined bounds. Before making the action, the system "asks" the safety module whether the decision is allowed from the safety perspective. If the specified state-action pair matches the extrinsic safety rules, then the answer is generated based on this pre-programmed or learned knowledge. When no rule/pattern can be applied and the robot is going to make an autonomous unauthorized decision, the world-model-based reversibility assessment approach is used to predict the intrinsic safety of making the action from the state in question.

Section 3 contains the theoretical framework of the approach as well as some implementation details. The *Reversibility MDP-Model (RMM)* is defined as an MDP with a function $C(s, a) \leq 0$ to assess reversibility of making action $a$ in state $s$. $C(s, a)$ is the total expected "cost" of reversing the action $a$ made in the state $s$. $C(s, a) = -\infty$ for the absolutely irreversible actions and $C(s, a) = 0$ for the perfectly reversible ones. Action $a$ from state $s$ is classified as *reversible*, if $C(s, a) \geq C_{rev}$, where $C_{rev}$ is a threshold value.

The experimental setup with the implementation details are presented in Section 4. In the experiment the analysis is made for the effect of the action "F" (move 0.6 meters forward) on the object placed directly in front of the robot, at the distance of 0.5 meters from its center of rotation.

The test runs are made on the actual Scitos G5 robot and in the simulated environment, which is a copy of the actual "room" of size 3 by 4 meters with some furniture. In the experimental setup the movable object is the only round item in the environment and its position in robot's coordinates is identified from the laser rangefinder scans. Robot position in the global reference frame is identified using the Adaptive Monte Carlo localization algorithm.

The experiment begins with a free movement of the robot to collect statistics about robot's movements for the actions used. The second part of the experiment consists of the off-line identification of "cycles" – paths, that would undo the effect of pushing the object in free space. In the final part of the experiment the safety module is used to govern behavior of the robot on-line. The previously identified cycles are simulated internally during the world-model-based reversibility assessment, now taking also the immovable obstacles into account.

In Section 5, the results of the experiment are presented. The procedure to search for cycles has found many paths to successfully undo the pushing of the object by the robot: the identified paths go around the object and push it from the opposite side. The reversibility of object pushing was analyzed with the object placed at different positions in robot reference frame. The situations, where the object is in front of the robot, have been successfully reversed. More complicated situations, where the robot touches the object with its side and the object slides away, are harder to undo. However, some lateral positions do allow to use cycles calculated for a specific central position. The analysis of the safety module governing the "F" action through $C(s, a)$ function calculation shows that the irreversible state-action pairs are successfully identified for the situations when there is no room to maneuver around the object to push it back.

It is concluded that non-trivial and quite complex cycles of actions can be successfully identified, allowing the robot to manipulate objects in a reversible manner – pushing them from one side and undoing such action by driving around the objects and pushing them back from the opposite side. Suppression of the irreversible actions while taking immovable objects into account results in further increase of behavioral complexity – robot "understands" that pushing object into a corner is irreversible and thus unsafe.

This paper is referenced in the thesis as [67] and is included in full as Appendix D. It was presented by the first author (Juri Gavšin) during the 6th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2011).

## 5.5 Improving Area Coverage by Reversible Object Pushing (Paper Summary)

This paper applies the principle of reversibility to assess the intrinsic safety of actions and to adapt robot's behavior for the task of covering the area with a movable object inside. A governor, acting upon the principle of action reversibility, allows pushing only if such an action is reversible. A practical experiment is conducted to demonstrate the approach and to compare it with the two other governors.

In the introduction, it is argued that in the future the use of robots in our households will be limited more by social and safety aspects than by price and functionality. It is also argued that the ability to identify irreversible actions and undo reversible ones is crucial for truly autonomous decision making, when no situation-specific rules can be applied.

Sections 2 and 3 describe the theoretical framework of the world-model-based approach and the safety architecture to apply it (as described in Section 5.4). It is also noted that the prerequisites must be met for a successful implementation of the approach and the threshold values together with reversibility and run-time parameters must be set externally. The external state/action identification, environment simulation and path planning modules play very important role in performance of the system as a whole.

Sections 4 and 5 describe the implementation details and the experimental setup. The procedure of action reversibility assessment, described in Section 5.4, is extended to analyze many points in robot's reference frame, so that the robot can undo pushing of the object from different relative positions. The sets of cycles for each analyzed point are used as path candidates during the $C(s, a)$ calculation – the first action in the cycle is expected to be undone by the rest of the cycle.

The Scitos G5 robot is used for the real-world part of the experiment and its model is used in the simulated environment, which is a copy of the actual room for the real robot. The test area is 5 by 4 meters and it is divided into 80 cells of the size of 0.5 meters. Robot's behavior mimics a vacuum cleaner, which task is to cover the area by visiting all the cells in the "lawnmower" fashion.

The experiment starts with the data acquisition part to collect statistical data about the robot and the environment. It is followed by the data processing part, when statistics is calculated and candidate cycles for $C(s, a)$ calculation are prepared. When data is collected and processed, the actual test runs with the different modes of the safety module are executed to measure the covered area and the distance between the initial and the final positions of the object.

In the "reversibility" mode no explicit safety rules are applied and the reversibility module is responsible for safety assessment. The cycles, generated in the off-line data processing part of the experiment, are used during the on-line assessment of action reversibility. The "obstacle avoidance" mode uses the same internal model of the environment to predict collisions. The action is prohibited if collision is predicted, allowed otherwise; the post-action is always empty. The "no safety" mode is the most naive approach – all the actions are allowed.

Section 6 presents the experimental results. The "no safety" mode is the least safe of all the three, however, this mode performs very well in terms of area coverage – all the 80 cells are visited. The "obstacle-avoidance" mode is the safest mode with almost no object movements, but with incomplete area coverage. The "reversibility" mode performs on a par with the "no safety" mode in terms of area coverage – all the 80 cells of the area in question are covered. Also, this mode is safe – in the end, the object is very close to its initial position after the real and the simulated test runs.

It is concluded that the proposed safety architecture, consisting of the safety module and its reversibility sub-module, can be used for practical purposes. The experiment showed that the principle of reversibility can be successfully used to ground safe object pushing behavior to increase covered area without compromising safety.

This paper is referenced in the thesis as [66] and is included in full as Appendix E. It was presented by the first author (Juri Gavšin) during the 15th International Conference on Advanced Robotics (ICAR 2011).

# 6 Conclusions and Future Work

## 6.1 Conclusions

The goal of this thesis was to develop a framework to enable future robots to behave safely in a wide variety of situations by combining extrinsic and intrinsic safety procedures. The result of this study is a hybrid safety architecture, combining extrinsic safety rules with the reversibility model to assess intrinsic safety of decision as a backup for situations, where no situation-specific logic can be applied. The *world-model-free* approach employs distance measures on state-space to assess reversibility of actions, using the given pairs of reverse-actions. The *world-model-based* approach employs external modules to translate environment into symbolic representations. The external planning and environment simulation modules are used to assess reversibility of actions, using cost of returning back to the initial state as a measure of irreversibility.

Out of many studies that address safety in robotics, very few are based on intrinsic assessment of safety based on abstract principles. Most intrinsic motivation studies are not concerned about safety and the few that do are very simplistic.

The performance of the world-model-free approach to predict collisions was evaluated in simulated environments and the test results show that the success rate of predictions with this approach is similar to the Q-learning algorithm. It is worth noting that predictions of the reversibility-based algorithm were derived from the intrinsic reward for taking reversible actions, while the Q-learning algorithm used an extrinsic collision-aware reward signal.

The world-model-free approach was also tested in the experiment in a simulated environment to identify pairs of reverse-actions from a sequence of randomly generated actions of different length and type. The test results show that the approach can be used to identify the pairs of reverse-actions as well as the general rule to create such pairs.

The performance of the world-model-based approach to enlarge the area covered by the robot was evaluated in simulated and real environments with a movable object occupying part of the desired area. The test results show that the performance of this approach, acting as a part of the safety architecture, is able to increase the covered area without compromising safety by moving the object in a reversible manner.

The results of the experiments suggest that the assessment of intrinsic safety of actions is useful for robots making purely autonomous decisions when no situation-specific logic can be applied. The application of the approach, however, is limited by quality of the external state identification, planning and environment simulation modules. The personal, service and field robotics are the possible areas of application in the future, when quality of perception and environment modeling will be acceptable for real-life scenarios.

The contribution of this thesis is a hybrid strategy to combine extrinsic and intrinsic safety knowledge, as well as the two approaches, aimed at different levels of abstraction, to assess intrinsic safety through action reversibility.

## 6.2 Future Work

Our ultimate goal is a multi-purpose autonomous personal robot-assistant acting intrinsically safely. The approach described in this thesis, as well as its prerequisites, can be improved and extended in many ways.

For example, in the world-model-free approach, instead of a weighted sum applied to filtered reversibility objects, other algorithms can be used to calculate the predicted value of irreversibility. The candidate algorithms include, but are not limited to clustering, division into regions, statistical methods and neural networks.

Another promising research direction is to extend and develop further the prerequisites of the approaches to increase their applicability. The world-model-free approach would perform even better if the distance measures were accounting for noise levels and discard absolutely irrelevant components of a state. The most promising research direction is further improvement of the external state/action identification, environment simulation and planning modules, required by the world-model-based approach. However, such effort already faces many problems like symbol grounding, partial observability, sensor imprecision, conflicting knowledge (including conflicting sensor readings), etc.

# Bibliography

[1] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *2008 IEEE International Conference on Robotics and Automation*, pp. 2165–2170, IEEE, May 2008.

[2] K. Suita, Y. Yamada, N. Tsuchida, K. Imai, H. Ikeda, and N. Sugimoto, "A failure-to-safety "Kyozon" system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3089–3096, IEEE, 1995.

[3] B. Karlsson, N. Karlsson, and P. Wide, "A dynamic safety system based on sensor fusion," *Journal of Intelligent Manufacturing*, vol. 11, pp. 475–483, Oct. 2000.

[4] M. Fritzsche, E. Schulenburg, N. Elkmann, A. Girstl, S. Stiene, and C. Teutsch, "Safe Human-Robot Interaction in a Life Science Environment," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 1–6, IEEE, Sept. 2007.

[5] J. Heinzmann and A. Zelinsky, "Quantitative Safety Guarantees for Physical Human-Robot Interaction," *The International Journal of Robotics Research*, vol. 22, pp. 479–504, July 2003.

[6] D. Kulić and E. Croft, "Pre-collision safety strategies for human-robot interaction," *Autonomous Robots*, vol. 22, pp. 149–164, Oct. 2006.

[7] J. Schmidhuber, "Self-Motivated Development Through Rewards for Predictor Errors/Improvements," in *Developmental Robotics 2005 AAAI Spring Symposium* (D. Blank and L. Meeden, eds.), pp. 1994–1996, 2005.

[8] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 265–286, Apr. 2007.

[9] A. Stout, G. D. Konidaris, and A. G. Barto, "Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning," in *AAAI Spring Symposium on Developmental Robotics*, 2005.

[10] A. Eppendahl and M. Kruusmaa, "Obstacle Avoidance as a Consequence of Suppressing Irreversible Actions," in *Sixth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* (F. Kaplan, P.-Y. Oudeyer, A. Revel, P. Gaussier, J. Nadel, L. Berthouze, H. Kozima, C. G. Prince, and C. Balkenius, eds.), vol. 128 of *Lund University Cognitive Studies*, Sept. 2006.

[11] M. Kruusmaa, Y. Gavshin, and A. Eppendahl, "Don't Do Things You Can't Undo: Reversibility Models for Generating Safe Behaviours," in *2007 IEEE International Conference on Robotics and Automation*, pp. 1134–1139, IEEE, Apr. 2007.

[12] Y. Gavshin, *Using the concept of reversibility to develop safe behaviours in robotics*. Master thesis, University of Tartu, 2007.

[13] P. Lin, K. Abney, and G. Bekey, "Robot ethics: Mapping the issues for a mechanized world," *Artificial Intelligence*, vol. 175, no. 5-6, pp. 942–949, 2011.

[14] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, pp. 253–270, Mar. 2008.

[15] N. G. Hockstein, B. W. O'Malley, and G. S. Weinstein, "Assessment of intraoperative safety in transoral robotic surgery," *The Laryngoscope*, vol. 116, pp. 165–168, Feb. 2006.

[16] P. Bast, A. Popovic, T. Wu, S. Heger, M. Engelhardt, W. Lauer, K. Radermacher, and K. Schmieder, "Robot- and computer-assisted craniotomy: resection planning, implant modelling and robot safety," *The international journal of medical robotics + computer assisted surgery : MRCAS*, vol. 2, pp. 168–178, June 2006.

[17] M. Hayashibe, N. Suzuki, and Y. Nakamura, "Laser-scan endoscope system for intraoperative geometry acquisition and surgical robot safety management," *Medical image analysis*, vol. 10, pp. 509–519, Aug. 2006.

[18] "Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots." ISO Standard 10218-1, 2011.

[19] "Robots and robotic devices – Safety requirements for industrial robots – Part 2: Robot systems and integration." ISO Standard 10218-2, 2011.

[20] “Robots and robotic devices – Safety requirements for non-industrial robots – Non-medical personal care robot.” ISO/DIS Standard(draft) 13482, 2011.

[21] C. Harper and G. Virk, “Towards the Development of International Safety Standards for Human Robot Interaction,” *International Journal of Social Robotics*, vol. 2, pp. 229–234, June 2010.

[22] G. Veruggio, “The EURON Roboethics Roadmap,” in *Humanoid Robots 2006 6th IEEERAS International Conference on*, pp. 612–617, IEEE, 2006.

[23] G. Veruggio and F. Operto, “Roboethics: Social and Ethical Implications of Robotics,” in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 1499–1524, Springer Berlin Heidelberg, 2008.

[24] “IEEE Code of Ethics.” http://www.ieee.org/about/corporate/governance/p7-8.html.

[25] R. A. Hirschfeld, F. Aghazadeh, and R. C. Chapleski, “Survey of robot safety in industry,” *International Journal of Human Factors in Manufacturing*, vol. 3, pp. 369–379, Oct. 1993.

[26] J. Kumagai, “A Robotic Sentry For Korea’s Demilitarized Zone,” *IEEE Spectrum*, vol. 44, pp. 16–17, Mar. 2007.

[27] S. Oberer and R. D. Schraft, “Robot-Dummy Crash Tests for Robot Safety Assessment,” in *2007 IEEE International Conference on Robotics and Automation*, pp. 2934–2939, IEEE, Apr. 2007.

[28] S. Haddadin, A. Albu-Schäffer, M. Frommberger, J. Rossmann, and G. Hirzinger, “The "DLR crash report": Towards a standard crash-testing protocol for robot safety - Part II: Discussions,” in *2009 IEEE International Conference on Robotics and Automation*, pp. 280–287, IEEE, May 2009.

[29] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, “Safe Physical Human-Robot Interaction: Measurements, Analysis & New Insights,” *Robotics Research*, vol. 66, pp. 395–407, 2011.

[30] M. Wassink and S. Stramigioli, “Towards a novel safety norm for domestic robotics,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3354–3359, IEEE, Oct. 2007.

[31] D. Shin, I. Sardellitti, and O. Khatib, “A hybrid actuation approach for human-friendly robot design,” in *2008 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 1747–1752, Ieee, 2008.

[32] J. H. Graham, J. F. Meagher, and S. J. Derby, “A Safety and Collision Avoidance System for Industrial Robots,” *IEEE Transactions on Industry Applications*, vol. IA-22, pp. 195–203, Jan. 1986.

[33] D. Kulić and E. Croft, “Real-time safety for human-robot interaction,” *Robotics and Autonomous Systems*, vol. 54, pp. 1–12, Jan. 2006.

[34] J. Moor, “The Nature, Importance, and Difficulty of Machine Ethics,” *IEEE Intelligent Systems*, vol. 21, pp. 18–21, July 2006.

[35] I. Asimov, “Runaround,” *Astounding Science Fiction*, 1942.

[36] D. Weld and O. Etzioni, “First Law of Robotics,” in *Safety and Security in Multiagent Systems* (M. Barley, H. Mouratidis, A. Unruh, D. Spears, P. Scerri, and F. Massacci, eds.), vol. 4324 of *Lecture Notes in Computer Science*, pp. 90–100, Springer Berlin Heidelberg, 2009.

[37] R. Murphy and D. D. Woods, “Beyond Asimov: The Three Laws of Responsible Robotics,” *IEEE Intelligent Systems*, vol. 24, pp. 14–20, July 2009.

[38] M. Anderson and S. L. Anderson, “Machine Ethics: Creating an Ethical Intelligent Agent,” *AI Magazine*, vol. 28, no. 4, pp. 15–26, 2007.

[39] R. C. Arkin, “Governing Lethal Behavior: Embedding Ethics in a Hybrid Deiberative/Reactive Robot Architecture,” Tech. Rep. GIT-GVU-07-11, Georgia Tech, 2007.

[40] R. C. Arkin, P. Ulam, and B. Duncan, “An Ethical Governor for Constraining Lethal Action in an Autonomous System,” Tech. Rep. GIT-GVU-09-02, Georgia Tech, 2009.

[41] R. C. Arkin and P. Ulam, “An ethical adaptor: Behavioral modification derived from moral emotions,” in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 381–387, 2009.

[42] G. Biggs and B. MacDonald, “A Survey of Robot Programming Systems,” in *Australasian Conference on Robotics and Automation*, vol. 1, (CSIRO, Brisbane, Australia), 2003.

[43] R. C. Arkin, *Behavior-Based Robotics*, vol. 1. MIT Press, 1998.

[44] D. R. Myers, M. J. Pritchard, and M. D. J. Brown, “Automated programming of an industrial robot through teach-by showing,” in *Iternational Conference on Robotics and Automation*, pp. 4078–4083, 2001.

[45] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Survey: Robot Programming by Demonstration,” in *Handbook of Robotics*, ch. 59, MIT Press, 2008.

[46] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöllner, and M. Bordegoni, “Learning Robot Behaviour and Skills Based on Human Demonstration and Advice: The Machine Learning Paradigm,” *Control*, vol. 9, pp. 229–238, 2000.

[47] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 9 of *Adaptive Computation and Machine Learning*. MIT Press, 1998.

[48] D. W. Patterson, *Artificial Neural Networks. Theory and Applications*. Prentice Hall, 1996.

[49] P.-Y. Oudeyer and F. Kaplan, "What is Intrinsic Motivation? A Typology of Computational Approaches," *Frontiers in neurorobotics*, vol. 1, pp. 1–14, 2007.

[50] R. Ryan and E. Deci, "Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions," *Contemporary Educational Psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[51] J. Schmidhuber, "Exploring the predictable," in *Advances in evolutionary computing*, pp. 579–612, Springer-Verlag New York, Jan. 2003.

[52] F. Kaplan and P.-Y. Oudeyer, "Motivational principles for visual know-how development," in *3rd International Workshop on Epigenetic Robotics Modeling Cognitive Development in Robotic Systems* (Christopher G. Prince, Luc Berthouze, Hideki Kozima, Daniel Bullock, Georgi Stojanov, and Christian Balkenius, eds.), vol. 101 of *Lund University Cognitive Studies*, pp. 73–80, 2003.

[53] A. G. Barto, S. Singh, and N. Chentanez, "Intrinsically Motivated Learning of Hierarchical Collections of Skills," in *3rd International Conference on Development and Learning*, pp. 112–119, 2004.

[54] "Tesla's New Monarch of Machines," *New York Herald*, 1911.

[55] "Principle of maximum work - Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Principle_of_maximum_work.

[56] J. Piaget, *The essential Piaget*. New York: Basic Books, 1977.

[57] K. Ishii, S. Zhao, M. Inami, T. Igarashi, and M. Imai, "Designing Laser Gesture Interface for Robot Control," in *Human-Computer Interaction* (T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. Prates, and M. Winckler, eds.), vol. 5727 of *Lecture Notes in Computer Science*, pp. 479–492, Springer Berlin / Heidelberg, 2009.

[58] R. Rao, K. Conn, S. Jung, J. Katupitiya, T. Kientz, V. Kumar, J. Ostrowski, S. Patel, and C. Taylor, "Human robot interaction: application to smart wheelchairs," in *International Conference on Robotics and Automation*, pp. 3583–3588, IEEE, 2002.

[59] T. Eiter, E. Erdem, W. Faber, and T. U. Wien, "Plan Reversals for Recovery in Execution Monitoring," in *10th Internation Workshop on Nonmonotonic Reasoning, Action and Causality Track*, pp. 147–154, 2004.

[60] A. L. Thomaz and C. Breazeal, "Asymmetric Interpretations of Positive and Negative Human Feedback for a Social Learning Agent," in *16th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 720–725, IEEE, 2007.

[61] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, pp. 716–737, Apr. 2008.

[62] Y. Gavshin and M. Kruusmaa, "Emergence of Safe Behaviours with an Intrinsic Reward," in *Adaptive and Intelligent Systems, LNAI* (A. Bouchachia, ed.), vol. 6943 of *Lecture Notes in Computer Science*, pp. 180–191, Springer Berlin / Heidelberg, 2011.

[63] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *International Conference on Advanced Robotics*, pp. 317–323, 2003.

[64] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[65] Y. Gavshin and M. Kruusmaa, "Identification of Reverse-Action Pairs using Reversibility of Actions," in *2011 IEEE International Conference on Systems, Man, and Cybernetics, Cybernetics Track*, pp. 2555–2560, IEEE, 2011.

[66] Y. Gavshin and M. Kruusmaa, "Improving Area Coverage by Reversible Object Pushing," in *15th International Conference on Advanced Robotics*, pp. 415–420, IEEE, 2011.

[67] Y. Gavshin and M. Kruusmaa, "Assessing Safety of Object Pushing Using the Principle of Reversibility," in *Hybrid Artificial Intelligent Systems, LNAI* (E. Corchado, M. Kurzynski, and M. Wozniak, eds.), vol. 6678 of *Lecture Notes in Computer Science*, pp. 313–320, Springer Berlin / Heidelberg, 2011.

# Appendix A

M. Kruusmaa, Y. Gavshin, and A. Eppendahl, “Don’t Do Things You Can’t Undo: Reversibility Models for Generating Safe Behaviours,” in *2007 IEEE International Conference on Robotics and Automation*, pp. 1134–1139, IEEE, Apr. 2007.

2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

# Don't Do Things You Can't Undo: Reversibility Models for Generating Safe Behaviours

Maarja Kruusmaa, Yuri Gavshin, Adam Eppendahl

*Abstract*— **We argue that an ability to determine the reversibility of actions allows a robot to identify safe behaviors autonomously. We introduce a notion of reversibility model and give a definition of model refinement. We implement this on a real robot and observe that, when a reversibility model is refined by the addition of proximity sensors, obstacle avoidance emerges as a side-effect of avoiding irreversible actions. We interpret this as evidence of a deep connection between reversibility and safe behaviour. We also observe that, on the real robot, reversiblities are learned as efficiently as a dedicated reward function. We conclude that reversibility identification may provide an abstract and yet practical method of generating a variety of safe behaviours.**

## I. INTRODUCTION

This paper is concerned with a robot's ability to undo its actions. We suggest that reversibility, a necessary condition of controllability, is a fundamental concept when programming robots to behave safely and reliably. We ask if this principle can be used to govern the operation of a robot, and to generate useful behaviour on a real robot and in real time.

We speculate that the most undesirable actions in the real world, those that damage the robot or the environment, for example, are characterized by irreversibility. Thus, instead of programming the robot with specific routines that prevent collisions, prevent falls, and so on, we program the robot with a more general principle of avoiding irreversible actions. In other words, instead of telling the robot *what* should not be done, we try to tell it *why* it should not be done. For example, falling down stairs is not good because the robot does not know how to climb back or pushing the door closed is not good because it does not have knowledge of how to open it.

In this paper, we state the problem of learning a reversibility model. The reversibility model represents the robot's knowledge of state-action pairs that are reversible and the ways of reversing them. We go on to demonstrate how this reversibility model can be established and used to generate new behaviours. In [1], we showed that by suppressing irreversible actions the robot will develop obstacle avoidance behaviour. In this paper, we confirm this result and go on to demonstrate that, as a developmental system, the efficiency of our abstract approach is comparable to ordinary reinforcement learning. The reinforcement learning algorithm, however, requires a signal that identifies collisions in specific terms, while the reversibility algorithm identifies the undesirable behaviours by their abstract properties and this just happens to result in collision avoidance. Thus we see a safe, concrete behaviour emerging autonomously and efficiently from a very abstract principle.

An enormous amount of the robot literature is concerned with algorithms for avoiding collisions as this is considered an essential ability for mobile robots. In this literature, the goal of avoiding collisions is explicitly identified [2], while the solution may be coded for by hand or obtained indirectly using learning algorithms [3, 4]. Collision-free navigation can be learned, for example, by using genetic algorithms [5], adaptive fitness functions [6], neural networks [7] or Q-learning [8]. In [9], navigation behaviours are derived by classifying random sensor data. Our approach is different in that reliable navigation emerges from an abstract rule. The rule is not grounded in a specific sensor-motor semantics that explicitly identifies collisions, and so the resulting developmental system is insensitive to sensor permutations and inversions. Indeed, the code can be written without knowing the location or polarity of sensors and actuators, an odd sensation after years of reaching for a manual.

The idea of generating behaviours top-down from abstract principles is an emerging theme in parts of the autonomous robotics community. In developmental robotics, for example, relatively abstract emotional and motivational mechanisms are used to derive behaviours that facilitate social interaction [10, 11]. Kaplan and Odeyer show that a number of basic visual behaviours can emerge from abstract motivational principles based on prediction errors [12]. The general idea is to identify principles that can be expressed without reference to the ground meaning of sensor-motor values. Code based on such principles should function reliably in a broad range of environments and on different robots or on different parts of the same robot. Our maxim of avoiding irreversible actions is just one example of such a principle.

In the following section we present these ideas about reversibility in a more formal manner. In Section III, we describe an experimental set-up with a Khepera mini-robot that tests the reversibility principle. In Section IV, we present the results and, in the last section, we discuss these results, draw conclusions and envision possible directions for future work.

This work was supported in part by Estonian Science Foundation grant number 6765.

Maarja Kruusmaa, corresponding author, is with the Tartu University Institute of Technology, Nooruse 1, 50411 Tartu, Estonia. Tel. +372 5183074, fax. +372 7374900 E-mail maarja.kruusmaa@ut.ee.

Yuri Gavshin is with the Department of Computer Science, Tartu University, Tartu, Estonia. E-mail yuri.gavshin@ut.ee.

Adam Eppendahl is with the Department of Engineering Design and Manufacture, University of Malaya, Kuala Lumpur, Malaysia. E-mail a.eppendahl@mac.com.

## II. REVERSIBILITY MODELS

A reversibility model tells the robot which actions are reversible and how to reverse them if they are. In a fixed, known, exact, deterministic world, modelled by a graph $G$ of states and actions, an action from state $s$ to state $s'$ is reversible if there is an action back from $s'$ to $s$. If we admit sequences of actions, by taking $G = \mathrm{Path}G_0$, the graph of paths over $G_0$, where $G_0$ is some graph of atomic actions, then finding reversibilities in $G$ is equivalent to finding loops in $G_0$, a standard problem in graph theory.

Real robots, however, face a changing, partially known, inexact and non-deterministic world. We therefore model non-determinism using labelled transition systems, we allow inexactness with a metric on the space of states, and we define a reversibility model pragmatically to be a set of expected reversibilities that may grow or shrink as the robot gains experience.

In addition, the robot may itself be changing as it learns, reconfigures or develops. In this paper we consider one form of development, the addition of sensors, and introduce a notion of refinement that captures the relationship between the robot's world before and after this development. In the learning experiments we describe, a reversibility model for an unrefined world is adapted to a refined world (with the interesting side-effect of producing obstacle avoidance behaviour).

Suppose we have a set $S$ of states given by vectors of sensor values and a set $A$ of actions given by vectors of motor commands. If we view the states as the nodes in a graph and the actions as labels, the robot's body and environment determine a labelled transition system which we refer to as the robot's *world*. A labelled transition system is a standard structure for modelling non-determinitstic systems and consists of a directed graph with edges, called transitions, labelled by actions. 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 a world $W$ is a state-action pair $(s, a)$, together with a state-action pair $(s', \bar{a})$. A reversibility may or may not hold, in a mathematical sense or in a physical sense. Generally speaking, $\bar{a}$ is expected to produce a transition from $s'$ to $s$, assuming $a$ produces a transition from $s$ to $s'$ in $W$. Because of the non-determinism, even given a perfectly known world $W$, there are different ways to define 'holding'. A reversibility $((s, a), (s', \bar{a}))$ may *hold weakly* if there exists in $W$ a transition from $s$ to $s'$ labelled $a$ and a transition from $s'$ to $s$ labelled $a$. Or, it may *hold strongly* if there exists a transition from $s$ to $s'$ labelled $a$ and every transition from $s'$ labelled $\bar{a}$, and at least one, leads to $s$. In our implementation, we use the strong definition. In addition, the action $\bar{a}$ is expected to work for any state $x$ with $d(x, s') < \epsilon'$ and is only expected to produce a transition back to a state $y$ when $d(y, s) < \epsilon$, where $d$ is a metric on states.

A *reversibility model* for a world $W$ is a set of reversibilities for $W$ that are expected to hold. In practice, a reversibility model could be given in advance, communicated to the robot, learned empirically, deduced from knowledge about the world, or obtained in some other way. In the experiments described here, the robot is given a model for one world and uses this to learn a model for a refined world.

A *refinement (of states)* from a world $W$ to a world $W'$ is a pair of functions from the states and transitions of $W'$ back to those of $W$, that respects the graph structure and labelling and is surjective on states. In other words, every state in $W$ is the image of one or more states in $W'$, which 'refine' the state in $W$, and the action on an edge in $W'$ is given by the action on the edge it is sent to in $W$.

For any reversibility model $R$ for a world $W$ and for any refinement from $W$ to $W'$, with state function $p$, there is a refined set of reversibilites $R'$ on $W'$ defined by

$$R' = \{((s, a), (s', \bar{a})) | ((p(s), a), (p(s'), \bar{a}) \in R\}.$$

To obtain a reversiblity model for the new world $W'$ we may form $R'$ and then remove any pairs that fail in the refined world. An important aspect of this procedure is that 'it gives the robot something to do': the original model $R$ provides a specific list of actions together with the circumstances in which they should be tried.

The kind of refinement we have in mind is produced by extending a robot's sensor vector. Suppose we have a world with states given by pairs of wheel counter values $(w_1, w_2)$ and actions given by pairs of wheel displacement commands $(m_1, m_2)$. Assuming the robot is able to control its own wheels, this world is fairly deterministic, all actions are reversible and a good reversibility model R is given by taking $\bar{a} = (-m_1, -m_2)$ when $a = (m_1, m_2)$ (for any $s$ and $s'$).

Now suppose we include one proximity value (say, the front sensor) in the state vector $(w_1, w_2, d_1)$. Assuming the new sensor does not effect the robot's environment, we obtain a refinement of the original world. The state function $p$ is the projection

$$p(w_1, w_2, d_1) = (w_1, w_2).$$

When the simple model $R$ described above is refined according to this new world some of the refined reversibilities hold and some do not. In our experiments, the robot tests these refined reversibilities to discover which hold.

The interesting point here is that the ones that fail generally correspond to collisions of some sort. Consider the following four cases (in which wheel counts and proximities are given, without loss of generality, in comparable units).

1) The robot does not touch anything: we obtain, say, the successful reversiblity

$$(((0, 0, 15), (10, 10)), ((10, 10, 5), (-10, -10))),$$

where the robot approaches and retreats from an object without touching it.

2) The robot touches an object and the object slides: we obtain a failed reversibility, say

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

where the robot runs into an object, pushing it 2 units forward, and then retreats, only to find that, while its wheel encoders are back to 0 as expected, its proximity sensor now reads 10 instead of the original 8.

3) The robot runs into an object and its wheels slide: from the robots point of view, this is identical to Case 2.
4) The robot runs into an object and its motors stall: if motor commands time-out and report success, adjusting the wheel encoder counts as necessary, then this case is again identical to Case 2 (and may be thought of as a kind of internal sliding).

Not only does the robot discover that it is 'bad' to push things–without ever knowing what pushing is!–but the refined state allows the robot to distinguish those cases in which 'bad things happen' from those in which they do not. Once the robot learns a reversibility model, it may use the model to censor its actions. Because of the non-determinism, we have a growing choice of definitions. A state-action pair $(a, s)$ is *strongly reversible* in world $W$, if there is a reversibility $((s, a), (s', \bar{a}))$ that holds in $W$ for every $s'$ that can be reached from $s$ by a transition labelled $a$. Alternatively, we could ask for just one such $s'$ to get a definition of *weakly reversible*. We must also say if $((s, a), (s', \bar{a}))$ holds strongly or weakly in $W$, for a total of four definitions. In our experiments, we use, in effect, the strong-strong definition, but because we pretend the world is deterministic by ignoring $s$ (by taking $\epsilon' = \infty$), there is no real difference.

Note that it is our method of creating a reversibility model out of $R'$ by pruning that creates a pushing-is-bad model. Alternatively, when a reversibility $((s, a), (s', \bar{a}))$ in $R'$ fails, we could try replacing the action $\bar{a}$ instead of throwing out the reversibility. For example, we could construct the world $W'^{*} = \mathrm{Path}W'$. The transitions in $\mathrm{Path}W'$ are paths of transitions in $W'$ labelled by sequences of actions from $W'$. The world $W'$ embeds in $W'^{*}$, along with $R'$, but now we have sequences of actions to play with. In the object pushing example, a sequence $b$ of actions might cause the robot to go behind an object, push it back 2 units, and then return to its original place in front of the object, so that

$$(((0, 0, 8), (10, 10)), ((10, 10, 0), b)),$$

holds in $W'^{*}$. Or we could form $W'^{*}$ by adding a gripping action and simply drag the object back 2 units.

## III. EXPERIMENTS

This section describes experiments with two learning algorithms. In all the experiments, both algorithms learn from the same sequence of actions and sensor data. One learns which reversibilities hold or fail. The other one is a standard reinforcement algorithm that punishes collisions. We compare the performance of the two algorithms over four sequences of actions. These were produced by running the same action generation routine in two test environments, an easy one and a harder one, and over two sets of actions, 1D and 2D.

The experiments were conducted on a Khepera II mini-robot, which is a cylindrical robot about 7 cm in diameter (see Fig. 1) with differential drive and a ring of eight proximity sensors. In these experiments, the motor control parameters were set so that, when the robot runs into a wall, the motors stall before the wheels slip. This allows us to detect collisions by watching for stalled motors. When a collision does happen, the wheel command routine times out and reports success, up-dating the wheel counters as if the command had completed. This is equivalent to more a forceful wheel command that would cause the wheels to slip, but makes it easier to identify collisions, which is required for the reinforcement algorithm and used for evaluating both algorithms.

### A. Implementation Details

An action $a = (m_1, m_2)$ consists of a pair of motor displacement commands, for left and right wheels, expressed in native wheel decoder units. A discrete set of actions is used in the experiments:

$$\begin{aligned} a_1 &= (100, 100) \text{ short step forward,} \\ a_2 &= (300, 300) \text{ long step forward,} \\ a_3 &= (-100, -100) \text{ short step backward,} \\ a_4 &= (-300, -300) \text{ long step backward,} \\ a_5 &= (100, -100) \text{ rotate clockwise,} \\ a_6 &= (-100, 100) \text{ rotate conterclockwise.} \end{aligned}$$

In the 1D experiments, we take

$$A = \{a_1, a_2, a_3, a_4\}.$$

These actions cause the robot to move back and forth in a straight line. In the 2D experiments, we include the turning actions,

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}.$$

We provide the robot with the initial reversibility model

$$\begin{aligned} \{&((x, a_1), (x + (100, 100), a_3)), \\ &((y, a_2), (y + (300, 300), a_4)), \\ &((z, a_5), (z + (100, -100), a_6))\}, \end{aligned}$$

where $x$, $y$ and $z$ are any states $(w_1, w_2)$, consisting of a pair of wheel counter values. Because we have fixed things so that wheel commands always succeed, the reversibilities in this model always hold. We then use (in effect) a refinement function $p$, the projection from the set of states $(w_1, w_2, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, )$, which include eight proximity values, to the original set of states $(w_1, w_2)$ without the proximity values, to induce a new set of refined reversibilities from the original set. The new set contains, for example,

$$\begin{aligned} &((s, a_1), (s', a_3)) = \\ &(((w_1, w_2, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8), a_1), \\ &\ ((w_1 + 100, w_2 + 100, d_1', d_2', d_3', d_4', d_5', d_6', d_7', d_8'), a_3)) \end{aligned}$$

for any $w_i$, $d_i$ and $d'_i$. The learning algorithm then tests these to see which hold and which fail.

For our definition of 'near', we use the Manhattan metric defined by

$$d(s, s') = \sum_{i=1}^{2} |w_i - w'_i| + \sum_{i=1}^{b} |d_i - d'_i|$$

(but because our wheel commands always succeed, and the original model is correct, the wheel value part of this is always 0).

*a) Robot motion:* The Khepera runs in a real, physical environment with motions that test the pairs of the refined reversibility model. The robot moves according to the following algorithm:

1) Record current state $s_i = (w_1, w_2, d_1, \dots, d_8)$.
2) Choose an arbitrary reversibility from $R'$ and execute the forward action as $a_i$.
3) Record the state $s_{i+1} = (w'_1, w'_2, d'_1, \dots, d'_8)$.
4) Execute the reverse action 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.

So the robot performs a random forward action, then the supposed reverse action, then a random action that goes unreversed, and then another forward action, and so on.

*b) Learning the reversibility model:* As the robot moves about, it notes how well the reversibilities hold using the Manhattan metric.

For each forward action $a_i$, calculate and store $d(s_i, s_{i+2})$.

For the purposes of comparison with the reinforcement algorithm, the model is also used to predict which actions will be successfully reversed. When a failure is predicted, we note whether there is a collision during the action. So we are judging the reversibility model not by what it is meant to be learning, but by how well this happens to predict collisions.

1) Get the current state $s_i$ and the intended action $a_i$
2) From memory, choose a state-action pair $(s_k, a_i)$ that minimizes $d(s_k, s_i)$.
3) If we have $d(s_k, s_i) > \delta$, predict randomly. Otherwise, predict a collision unless $d(s_k, s_{k+2}) < \epsilon$.
4) While executing the command $a_i$ check if there is a collision. Store the predicted and the actual outcome.

*c) Reinforcement learning:* Reinforcement learning algorithms [13] are commonly used in mobile robotics. The aim here is to implement a simple version for collision avoidance so that we may compare the ungrounded reversibility method to a standard, grounded method. We have therefore implemented the reinforcement learning algorithm so that the robot is operating under similar conditions. First, the algorithm does not have a terminal state, so collision avoidance is considered to be a continuous task of reward maximization. Second, the current version of the reversibility policy is concerned only with immediate actions and reverse actions and does not work along the history of action sequences.

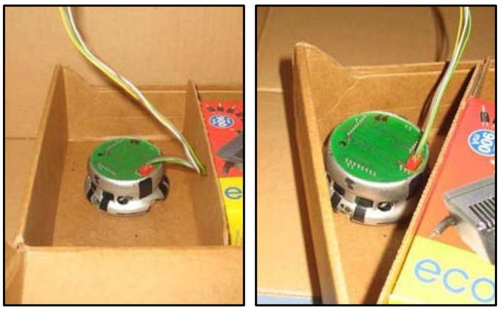
Fig. 1. The robot in Environment I and in Environment II.

Therefore we have also implemented the reinforcement algorithm to be concerned only with immediate rewards, thus with discount rate $\gamma = 0$. The initial value of the action value function is $Q(s_i, a_i) = 0$. The reward signal is defined by checking for collisions.

$$r = \begin{matrix} (|w_1| + |w_2|)/100, & \text{if there is no collision} \\ -5, & \text{if there is a collision} \end{matrix}$$

Thus a successful action is rewarded more if it moves the robot a greater distance and an unsuccessful action is strongly penalised. Note that the reinforcement learning algorithm directly checks for collisions (by watching for stalled motors) to calculate the reward, while the algorithm learning the reversibility model only aims at predicting if the robot can return to the initial state (by watching the proximity sensors). The reinforcement learning algorithm is the following.

1) Get the current state $s_i$ and the intended action $a_i$.
2) If the current value of the action value function $Q(s_i, a_i) < 0$, predict a collision. If $Q(s_i, a_i) = 0$ make a random prediction. Otherwise, predict no collision.
3) After executing $a_i$ get the reward signal $r$.
4) Update the action value function $Q(s_i, a_i) \leftarrow \alpha r + Q(s_i, a_i)$, with learning rate $\alpha = 0.1$.
5) While executing $a_i$, check for collisions. Store the predicted and the real outcome.

## *B. Test Environments*

In the experiments we compared the learning of reversibility models to the learning of a reward function that discourages collisions. To find out how sensitive the learning algorithms are to environmental conditions, the tests are conducted in two environments. These are shown in Fig. 1. Environment I is a rectangular space, whereas Environment II is a smaller, triangular space, only slightly larger than the robot, in which collisions are more probable. In both environments the algorithms are run over sequences of 1D movements and sequences of 2D movements. With 1D actions, the robot only moves forewards and backwards. With 2D actions, the robot moves in all directions. This was done

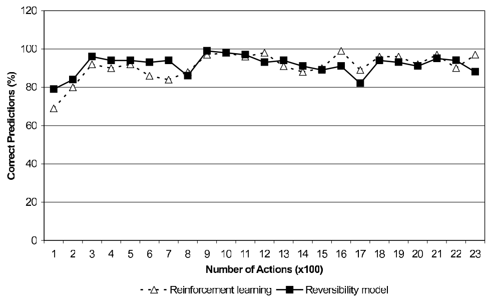

Fig. 2. Experimental results in Environment I, with 1D actions.

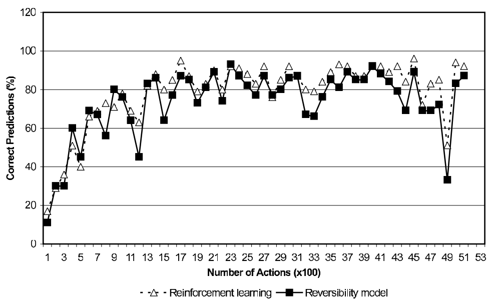

Fig. 3. Experimental results in Environment I, with 2D actions.

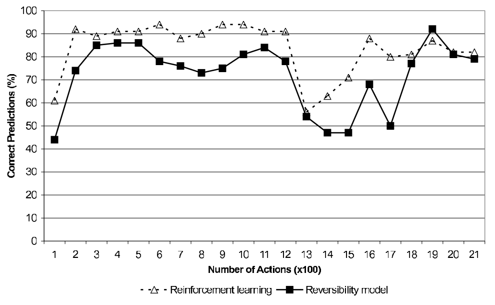

Fig. 4. Experimental results in Environment II, with 1D actions.

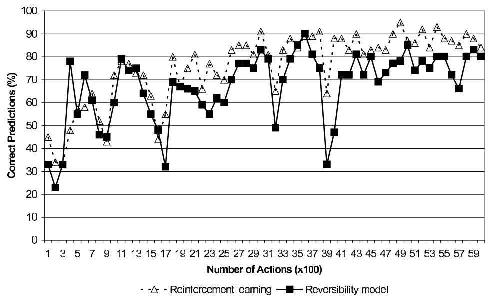

Fig. 5. Experimental results in Environment II, with 2D actions.

to get an idea of how the algorithms scale from smaller to larger, more complex action sets.

## IV. RESULTS

As described in the previous section, the robot operates by executing supposed reversals and random actions. The reversed actions are determined according to the initial reversibility model. The goal of the learning algorithms is to observe and learn to predict the outcomes of actions. These predictions are then compared to the real outcome of the action (determined by detecting collisions) and the success rate of each method is recorded.

Note that although the performance graphs for the two methods are expressed in the same terms, we are *not* comparing two techniques for solving one learning problem, but rather two learning problems whose solutions happen to result in the same behaviour. The reversibility problem is at a big disadvantage here, because we are evaluating it as if it was intended to predict collisions, which is in fact just a fortuitous emergent property.

Moreover, the motion routine, which performs reversals interleaved with random actions, allows a reversibility to be tested every third step, while the reinforcement algorithm gets a feedback signal at every step. Thus the reinforcement learning algorithm has more experiences to learn from, and yet the performance of the two algorithms is seen to be comparable.

The figures show the performance of the two algorithms over four sequences of actions and sensor values. All four graphs show the average correctness of predictions for each successive 100 actions for both prediction methods. From Fig. 2 and Fig. 4, we see that with 1D actions the robot rapidly learns to avoid collisions in both environments. The rate of successful predictions reaches 80during the first 200–300 steps, and the learning problem is equally trivial for both learning algorithms. From Fig. 3 and Fig. 5, we see that with 2D actions the learning problems are more complicated, with both algorithms converging around 1900–2100 steps.

During the runs in Environment II, the wheels occasionally got stuck on the uneven surface. These incidents can be seen on the graphs around 1400–1700 steps in 1D (Fig. 4) and 3700–4100 in 2D (Fig. 5), where there are sharp downward peaks in the prediction rates. It appears that reinforcement learning recovers better. However, this is caused more by the method we use to determine the prediction rate than a failure to relearn the reversibility model. For the robot with the blocked wheels, the reversibility of actions is perfect, since the robot certainly ends up in the same state it starts from. However, when this is used to predict no collision, which is to say no motor stall, the prediction is wrong.

In these runs, which consist of thousands of actions, it is clear that both learning problems are solved with comparable speed. The reversibility model is learned with roughly the same speed as the reward function in Environment I, whereas in Environment II reinforcement learning happens slightly faster. Likewise, both approaches scale equally well from a 1D to a 2D environment.

## V. CONCLUSIONS

This paper introduces the concept of reversibility for learning and developing robots. We show that reversibility models can be used to learn a useful new behaviour. The experiments verify the performance of the reversibility method against a well-established learning method commonly used in robotics. The results show that both of the methods converge to obstacle avoidance behaviour.

The most general conclusion drawn from the experimental results is that the efficiency of the policy of reversibility is comparable to reinforcement learning. Both methods learn more or less equally, converging to satisfactory performance. The basic difference of these methods is that the reinforcement learning algorithm uses a reward signal explicitly designed to make the robot avoid obstacles. The policy we introduce, uses a reversibility measure to learn a reversibility model, and yet the robot learns the useful behaviour of collision avoidance.

Based on these experimental results we speculate that the concept of reversibility could generate a variety of useful behaviours depending on the properties of the environment. We surmise, for example, that a robot placed initially close to an object or wall might, using reversibility models, discover behaviours like ‘do not leave the territory’ or ‘stay in the vicinity of guidelines’. Our future experiments are planned to check this hypothesis and find more evidence concerning the robustness of this principle.

Another hypothesis we are planning to test is whether learning algorithms can be accelerated by using reversibility models. Generally, learning algorithms converge to a stable behaviour by repeating actions that lead from one state to another. The problem of how the robot gets back to the state it wants to repeat, however, is not addressed. Knowing the reversibility model, it may be easier to guide the learning algorithm to faster convergence.

We also suggest that reversibility models could be used in combination with formal reasoning methods, such as task or path planning, where the plans can be checked for reversibility. For mobile robots such a reversibility check could, for example, guarantee safe homing or safe exploration. We suggest that the concepts introduced in this paper may provide handy and simple guidelines for building safe and reliable robots.

## REFERENCES

[1] A. Eppendahl and M. Kruusmaa, Obstacle avoidance as a consequence of suppressing irreversible actions, in *Proceedings of the Sixth International Workshop on Epigenetic Robotics*, Lund University Cognitive Studies, vol. 128, 2006.

[2] J. Borenstein, Y.Koren, Real-time obstacle avoidance for fast mobile robots, in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187.

[3] R. Arkin, *Behavior-based Robotics*, MIT Press, Cambridge, MA.

[4] S. Nolfi, D. Floreano, O. Miglino and F. Mondada, How to evolve autonomous robots: Different approaches in evolutionary robotics, in *Artificial Life IV*, pp. 190–197, MIT Press, 1994.

[5] D.Bajaj and M. Ang, Jr., An incremental approach in evolving robot behavior, in *Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision*, Singapore, 2000.

[6] E. Uchibe, M.Yanase and M. Asada, Behavior generation for a mobile robot based on the adaptive fitness function, in *Robotics and Autonomous Systems*, vol. 40, pp. 69–77, 2002.

[7] J. Blynel and D. Floreano, Exploring the T-Maze: evolving learning-like robot rehaviors using CTRNNs, in *Applications of Evolutionary Computing*, 2003.

[8] G.-S. Yang, E.-K. Chen and C.-W. An, Mobile robot navigation using neural Q-learning, in *Proceedings of the International Conference of Machine Learning and Cybernetics*, vol. 1, pp. 48–52, 2004.

[9] E. Simonin, J. Diard and P. Bassiere, Learning Bayesian models of sensorimotor interaction: from random exploration toward the discovery of new behaviors, in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1226–1231.

[10] C. Breazeal, *Designing Sociable Robots*, MIT Press, 2002.

[11] L. Moshkina and R. C. Arkin, On TAMEing Robots, in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3949–3959, 2003.

[12] F. Kaplan and P.Y. Oudeyer, Motivational principles for visual know-how development, in *Proceedings of the Third International Workshop on Epigenetic Robotics*, Lund University Cognitive Studies, 2003.

[13] R.S.Sutton and A.G.Barto, *Reinforcement Learning, an Introduction*, MIT Press, 1998.

# Appendix B

Y. Gavshin and M. Kruusmaa, “Emergence of Safe Behaviours with an Intrinsic Reward,” in *Adaptive and Intelligent Systems, LNAI* (A. Bouchachia, ed.), vol. 6943 of *Lecture Notes in Computer Science*, pp. 180–191, Springer Berlin / Heidelberg, 2011.

# Emergence of Safe Behaviours with an Intrinsic Reward

Yuri Gavshin and Maarja Kruusmaa

Tallinn University of Technology, Centre for Biorobotics,
Akadeemia tee 15a-111, 12618 Tallinn, Estonia
{yury,maarja}@biorobotics.ttu.ee
http://www.biorobotics.ttu.ee

**Abstract.** This paper explores the idea that robots can learn safe behaviors without prior knowledge about its environment nor the task at hand, using intrinsic motivation to reverse actions. Our general idea is that if the robot learns to reverse its actions, all the behaviors that emerge from this principle are intrinsically safe. We validate this idea with experiments to benchmark the performance of obstacle avoidance behavior. We compare our algorithm based on an abstract intrinsic reward with a Q-learning algorithm for obstacle avoidance based on external reward signal. Finally, we demonstrate that safety of learning can be increased further by first training the robot in the simulator using the intrinsic reward and then running the test with the real robot in the real environment.

The experimental results show that the performance of the proposed algorithm is on average only 5-10% lower than of the Q-Learning algorithm. A physical robot, using the knowledge obtained in simulation, in real world performs 10% worse than in simulation. However, its performance reaches the same success rate with the physically trained robot after a short learning period. We interpret this as the evidence confirming the hypothesis that our learning algorithm can be used to teach safe behaviors to a robot.

## 1 Introduction

This paper is concerned with applying an intrinsic reward signal to robot learning. In our case, the intrinsic motivation of the robot is to learn to reverse actions. The rationale for such a motivational system is to teach the robot to behave safely. We surmise that a robot governed by such an intrinsic motivation will behave inherently safely as it will avoid actions that cause irreversible damage.

Intrinsic motivation is a concept derived from psychology and in its original meaning refers to an activity done for one's inherent satisfaction rather than to achieve some specific external goal [1]. In computer science and robotics intrinsic motivation has been studied in developmental robotics and reinforcement learning. Some models are derived seeking an analogy with neural processes in the brain [2]. Schmidhuber's research ([3],[4]) introduces a system with autonomous and active exploratory behavior motivated by "artificial curiosity". Barto et al. [5] and Stout et al. [6] use advanced RL techniques in their research of robot learning motivated by the concepts of "novelty" and "surprise". These approaches are tested in a grid-world abstract agent simulation.

Kaplan and Oudeyer showed that a robot can develop visual competences from scratch driven only by internal motivations independent of any particular task: predictability, familiarity and stability [7]. They generalized their approach further and derived a mechanism of Intelligent Adaptive Curiosity, an intrinsic motivation system which pushes a robot towards situations in which it maximizes its learning progress [8]. Experiments by Kaplan and Oudeyer are made with real robots using real sensor data.

In this paper we derive an intrinsic motivation system that forces the robot to learn to reverse actions and gives the preference to reversible ones. In the opposite to [8] where the motivational system encourages robot curiosity, our system is driven towards stability and safety.

The drive to suppress irreversible actions is thus a kind of an adaptive homeostatic predictive motivation according to the classification given in the recent overview paper of computational approaches to intrinsic motivation [9]. Homeostatic systems force the robot to maintain some of their properties (e.g. the energy level). Another example of a homeostatic system is the motivation to maintain a comfortable level of social interaction [10]. In our case the homeostatic system of the robot forces it to build a connected state space where all other states can always be reached and returned back to.

Our motivation to build a learning system that learns action reversibility is to build safe autonomous learning robots. We assume that reversible actions are intrinsically safe because the robot is always able to deal with the consequences. The abstract intrinsic motivation also makes the goal of the robot independent of the environment it works in or the task it fulfills (as an external goal). Instead of specifying routines such as avoiding obstacles, falls, traps, risky regions or routes or staying near to some known landmark, it is rather told not to do things it cannot undo. 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.

Papers of Kruusmaa and Gavshin have provided an initial evidence that the principle "Don't do things you can't undo" generates a concrete safe behavior of obstacle avoidance ([11],[12]). This behavior emerges from the intrinsic goal of the robot to avoid irreversible actions as after bumping to an object/wall or wheels slippage, a simple robot cannot reverse to the previous state with the same sensor readings. In this paper we have developed their ideas further, conducted experiments on simulated Khepera II and Scitos G5 robots as well as on the real SCITOS G5 robot. With these experiments we aim at investigating:

– How well does our approach compare to some classical benchmark obstacle avoidance algorithm?

– Does it increase the safety of robot learning if we first train the robot in a simulator to avoid irreversible actions and then run the trained robot in a real environment?

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 physical and simulated robots used in experiments, their test environments and specific implementation details. In section 4 we present the results and discuss them together with general applicability of the approach. Section 5 contains conclusions and possible directions of future work.

## 2 Theoretical Framework

This section briefly describes the general theoretical framework used to ground the reversibility based algorithm and to test the robots.

### 2.1 Definitions

A robot's world is a labelled transition system $(S, \Lambda, \rightarrow)$, where $S$ is a set of experienced states, $\Lambda$ 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: $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$. Generally speaking, a composite action $a_{forward}a_{reverse}$ produces a transition from $s_{init}$ to $s_{final}$ through $s_{interim}$ in $W$.

Also, the action sequence $a_{forward}a_{reverse}$ is expected to work for any states $x$ and $y$ with $d_{orig}(x, s_{init}) \leq \varepsilon_{orig}$ and $d_{dest}(y, s_{interim}) \leq \varepsilon_{dest}$, where $d_{orig}$, $d_{dest}$ are metrics on states and $\varepsilon_{orig}$, $\varepsilon_{dest}$ are their thresholds.

The reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ holds in $W$ if there exists a transition path from $s_{init}$ to $s_{final}$ through $s_{interim}$ consisting of two transitions labelled accordingly $a_{forward}$ and $a_{reverse}$, and $d_{rev}(s_{init}, s_{final}) \leq \varepsilon_{rev}$, where $d_{rev}$ is a prametric ( $d_{rev}(x, y) \geq 0$ and $d_{rev}(x, x) = 0$ ) on states and $\varepsilon_{rev}$ is a threshold; fails otherwise.

An action $a_{forward}$ in an arbitrary state $s$ is expected to be reversible (by action $a_{reverse}$ ), if the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ holds and $d_{orig}(s, s_{init}) \leq \varepsilon_{orig}$. A reversibility model of the robot is a set of reversibilities that are expected to hold.

### 2.2 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 $\varepsilon_{orig}$ 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. In other words, $d_{orig}$ and $\varepsilon_{orig}$ are used to identify a "region" or a "cluster" of states.

A prametric $d_{rev}$ is used to calculate how strongly the reversibility holds. A prametric is used instead of a metric to make it possible 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 would measure stability.

The intrinsic reward for making an action is counter-proportional to the value of $d_{rev}$. When applied to our learning algorithm it forces the robot to give higher weight to the actions that are reversible. The intrinsic reward can be generated, when a sequence of an action with its reverse-action is observed. In this case, the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ is observed and the value of $d_{rev}(s_{init}, s_{final})$ is calculated. The intrinsic reward can then be calculated, for example, using the following expression:

$$r = \varepsilon_{rev} - d_{rev}(s_{init}, s_{final}) \ .$$

## 3 Experimental Setup

The purpose of the experiment is to validate the reversibility-based approach to safe learning proposed in this paper. The experiment consists of:

- Tests 1 and 2: two test runs of the same length with simulated Khepera II and Scitos G5 (5200 steps each).
- Test 3: the physical test run (1000 steps long) on Scitos G5 pre-loaded with simulation data (first 4000 steps from Test 2).

Each test run is divided into two phases: data collection (phase 1) and simulation (phase 2).

During the first phase the robot (physical or simulated) makes pseudo-random moves and the input data (sensors data, actions made and outcomes of the actions) are collected and saved into log files. The predictions are made during the second phase using the data collected in the first phase. The performance is measured by sampling algorithms' predictions of whether the next action will succeed, followed by calculation of the success rate of those predictions.

### 3.1 The Robots

Comparative experiments are conducted on two common research robot platforms, Khepera II by K-Team and Scitos G5 by MetraLabs. For this paper both robots are tested in the simulator and Scitos G5 robot is tested physically.

Both Khepera II and Scitos G5 are differential drive robots but with different size and slightly different geometry. Khepera II has a circular shape and the rotation axis is exactly at the center 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 $200mm$ (the size of the passive wheel compartment). The laser range finder is used in the test reported in this paper.

### 3.2 The Environments

Both Khepera II and Scitos G5 robots are simulated by Gazebo simulator (version 0.8-pre3, OGRE version 1.4.9, ODE version 0.10.1) through Player (modified version 2.1.0) interface [13].

The physical environment for Scitos G5 is a rectangular box of size $970mm$ by $1500mm$ (see Fig. 1). Absolute size for simulated Scitos G5 and its environment matches closely the real one and the laser rangefinder is located in the correct position and pose in respect to the robot's body. However, only 22 of 541 laser rays were simulated to optimize performance, since only 8 rays were used in the experiments.

Khepera II infra-red sensors are simulated by 8 short laser rays distributed evenly around the robot with the maximum measurable distance of $100mm$. The environment for Khepera II simulated environment is a right-angled triangle with side lengths $196mm$, $125mm$ and $233mm$.

### 3.3 Robot Movements

In the experiments 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 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. A discrete set of actions is used in the experiments: $F$ – make a step forward, $B$ – make a step backward, $L$ – rotate counter-clockwise, $R$ – rotate clockwise, where $F = -B$, $B = -F$, $L = -R$, $R = -L$.

Actions are defined in terms of commands to move forward/backward or rotate. An action $a = [m_{trans}, m_{rot}]$ consists of a pair of target movement deltas – $m_{trans}$ is in metres and , $m_{rot}$ is in degrees. For simulated Khepera II the values were set as follows:

$F = [+0.016, 0]$ – make a step forward,
$B = [-0.016, 0]$ – make a step backward,

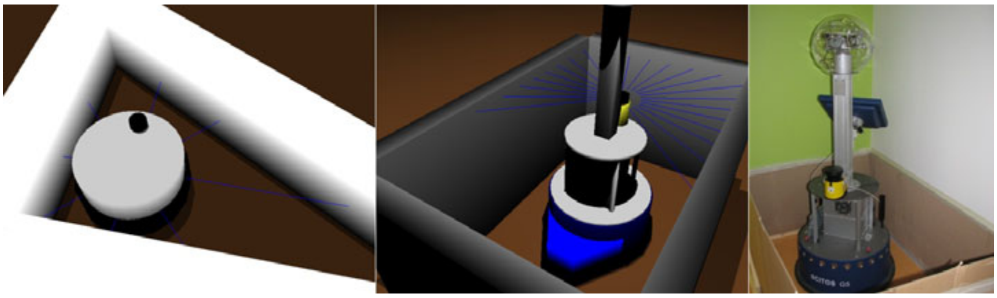

**Fig. 1.** Environments for the experiments. Simulated Khepera II is on the left. Simulated Scitos G5 is in the center. Real Scitos G5 is on the right.

```
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 (goto 1).
```

**Fig. 2.** Movement algorithm (Phase 1)

$L = [0, +30]$ – rotate counter-clockwise,
$R = [0, -30]$ – rotate clockwise.
For both simulated and real Scitos G5 the values were set as follows:

$F = [+0.15, 0]$ – make a step forward,
$B = [-0.15, 0]$ – make a step backward,
$L = [0, +42]$ – rotate counter-clockwise,
$R = [0, -42]$ – rotate clockwise.

The robot moves using the algorithm described in Fig. 2 – 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 generate intrinsic reward signal.The purpose of the next (random) action without a matching reverse action is to make the robot to explore the environment.

### 3.4 Software Design

The code consists of the following units:

- 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.

```
 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
    $a_{forward} = a_i$, based on $d_{orig}(s_i, s_{init})$ of experienced reversibility.
 3. If no reversibilities are selected, make no prediction.
 4. Calculate the expected irreversibility value (intrinsic reward) $v_{rev}$ using
    $d_{rev}(s_{init}, s_{final})$ of experienced reversibilities.
 5. If $v_{rev} > \varepsilon_{rev}$, then predict negative outcome, positive otherwise.
 6. If $i < 2$, add 1 to $i$ and repeat.
 7. Read the last action as $a_{i-1}$ and the previous action $a_{i-2}$ from log.
 8. If $a_{i-1}$ is not a reverse-action of $a_{i-2}$, add 1 to $i$ and repeat.
 9. 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.
10. Add 1 to $i$ and repeat (goto 1).
```

**Fig. 3.** Prediction data collection algorithm (Phase 2)

In the first phase real-world or simulated 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. The second phase is a virtual run using collected data to calculate predictions and can be executed without a robot or a simulator. The log file from the first phase 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.

### 3.5 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. This is done by generating the intrinsic reward signal based on the distance between the initial and final state representations. The algorithm is described in Fig. 3. 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 intrinsic reward is calculated as an expected irreversibility value $v_{rev}$ using a set of reversibilities, selected from the experience vector. In the experiments we select reversibilities with the same forward action and $d_{orig}(s_{init}, s_t) < \varepsilon_{orig}$, where $s_{init}$ 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 $d_{orig}(s_{init}, s_t)$ in an ascending order and their weights are $1/i^3$ (1, 1/8, 1/27, 1/256, etc), i.e. reversibilities with a "closer" initial state have considerably stronger influence.

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

### 3.6 Reinforcement Learning Algorithm

Reinforcement learning is a commonly used learning method to learn obstacle avoidance by trial and error ([14],[15],[16]). 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 (which may also be interpreted as an intrinsic reward rising from the similarity of the initial and final states).

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

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

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 $Q = 0$, then no grounded prediction can be made.

### 3.7 Other Implementation Details

Real Scitos G5's default configuration file was altered to set rotational PID controller's $Kp$ value to 0.2. Sensor values for Scitos G5 are in metres, therefore they are multiplied by 1000 to be of similar scale to the ones of Khepera II. This doesn't affect the reversibility based algorithm, but makes saving and loading the log files simpler.

During the experiments $\alpha_t(s_t, a_t)$ for Q-Learning update expression was set to 0.01. Threshold values $\varepsilon_{orig}$, $\varepsilon_{rev}$ and the tile size for Reinforcement learning state identification were constant: $\varepsilon_{orig} = 11000$, $\varepsilon_{rev} = 10000$, $RLtilesize = 168$.

## 4 Results and Discussion

### 4.1 Results

During the tests 1 and 2 both learning methods are predicting collisions of simulated robots with simulated obstacles (walls). Fig. 4 and 5 represent the test results for simulated Khepera II and Scitos G5 environments respectively. In the test 3, shown in Fig. 6, the reversibility model from simulated test run is used to predict collisions of the real Scitos G5 robot with walls during the physical test run.

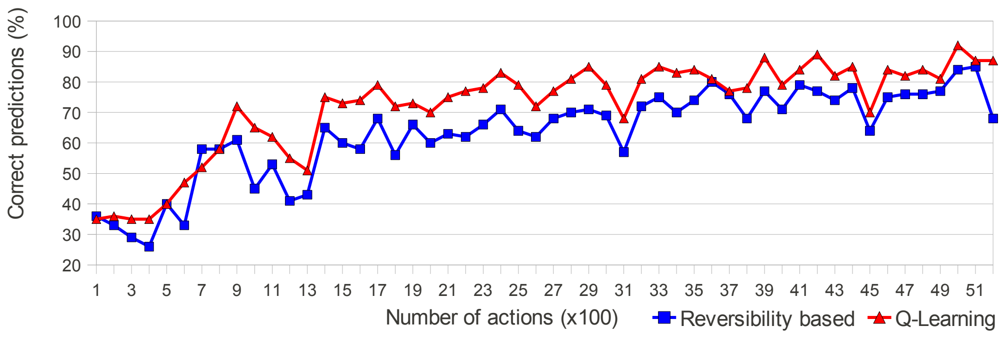

**Fig. 4.** Results of Test1 – simulated Khepera II

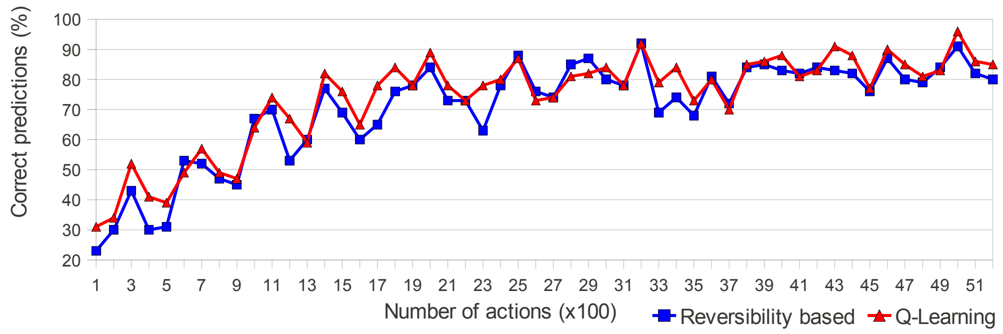

**Fig. 5.** Results of Test2 – simulated Scitos G5

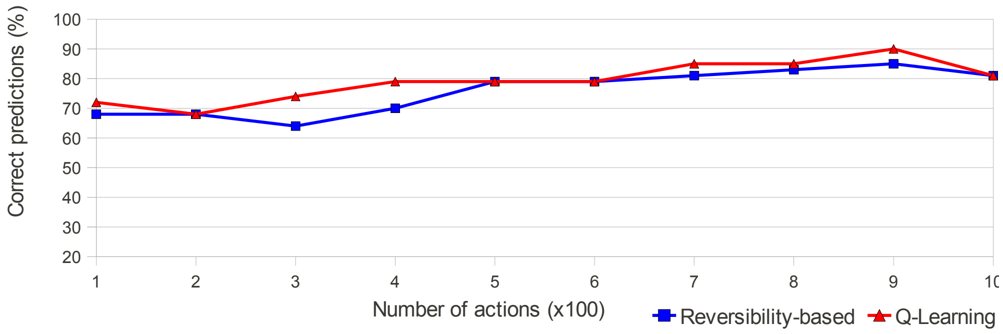

**Fig. 6.** Results of Test 3 – real Scitos G5 using simulated experience

The rate of correct prediction is calculated by sampling how many predictions out of a 100 were correct. If no prediction is made, then it counts as incorrect prediction. Downward spikes in prediction rate graphs are caused by novelty of the states, since no grounded prediction can be made for such unvisited states.

The rates of correct prediction of both algorithms in simulated environments start at 20–40% and gradually reach the level of 70–90% after 3900 steps. The robot with preloaded experience, obtained in simulation, performs quite well from the beginning. The rates of correct prediction in the real environment start at 70–75% and reach the level of 80–90% after 900 additional steps.

### 4.2 Q-Learning vs. Reversibility-Based Learning

On both simulated robots and the real Scitos G5 robot Q-Learning converges to a 5-10% higher prediction success rate than the reversibility-based learning. 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 motivated to be able to reverse its actions has no concept of an obstacle or collision. The reversibility-based algorithm does not use the external reward signal and merely tries to predict whether the action will be reversible or not, based on its internal representations of similarities between the states. Also, the method of measuring the rate of correct predictions works in advantage of the Q-Learning algorithm, which predicts future rewards based on the experienced rewards, while the reversibility based algorithm predicts future rewards based on sensor data alone.

### 4.3 Real-Life Tests with Preloaded Simulator Data

The aim of the test 3 is to train the robot in the simulator to behave safely and then test the performance of the real robot in the real environment. The physical test run with the reversibility model built during the simulation test run shows success rate 65-70% from the very beginning of the test run. During the test in the real environment the performance improves further reaching to the success rate of the simulated run (ca 80%). This is because when put to the real environment, the robot still first encounters states it has not been trained for in the simulator. However, it adapts to the changes fairly fast and reaches the performance of the simulated robot of the test 2. This shows that a reversibility model can be learned in simulation to increase the safety of robot learning and then be corrected further on a physical robot.

### 4.4 Generality of the Approach

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

It is difficult to distinguish sensors by their importance for the particular action. Different kinds 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 problem in this case and will render both algorithms almost useless without additional tuning. Another problem is to choose threshold values $\varepsilon_{orig}$, $\varepsilon_{dest}$ and $\varepsilon_{rev}$. We chose those values manually using statistical information of the data from a particular test run.

We therefore conclude that for the present approach it is not possible to run the same code absolutely interchangeably on different platforms. Despite that the goal to learn to reverse actions is purely abstract and the reward signal is intrinsic, it is still based on the real underlying sensor values, which makes the algorithm implementation somewhat dependent on the physical embodiment.

## 5 Conclusions

The aim of these experiments was to validate the concept of learning using an intrinsic reward signal based on the reversibility of robot's actions. We argued that by learning to reverse its actions the robot develops understanding of its own motion in the surrounding environment. We argued that in contrast with learning algorithms designed for a special purpose (e.g. obstacle avoidance) the reversibility based algorithm has an abstract intrinsic goal of being able to reverse actions. At the same time we aimed at showing that this abstract goal can lead to concrete safe behaviors, such as obstacle avoidance, when irreversible actions are suppressed. Our aim was to investigate further if this general idea works on different robots and how it performed with respect to a benchmark Q-learning algorithm. Furthermore, we aimed at showing that if such a robot is trained in simulations and then ran in real life, the performance of the robot is safer.

In general, we interpret the results as positive, since, indeed, a concrete robot behavior of obstacle avoidance is observed on two different robots to emerge from the abstract principle "Don't do things you can't undo". We encountered some problems with this straight-forward plain-sensor approach: it is influenced by many factors like sensor precision, sensor noise, actions' precision, etc. However, such state-identification problems are inherent for any state-based approach.

The experimental data analysis leads to the following conclusions:

1. The Q-learning algorithm based on an external reward signal is 5-10% more successful than the reversibility based algorithm using an intrinsic reward signal.
2. The real robot running with simulator pre-loaded data is ca 10% less successful than the robot trained in real environment. After additional learning steps it is able to quickly adjust its performance and measures up to the results achieved with the robot trained in real life. This suggests that the algorithms can mostly be learned in a simulator to increase safety of the robot and its environment.

### 5.1 Future Work

In the future we will continue testing the same principle in more complicated scenarios. We are trying to use environment-model-aware state identification, planning and internal simulation to further increase the complexity of generated behaviors. Another possible direction is to use the principle of reversibility to make other learning algorithms learn faster or safer, or both. Our ultimate goal is a multi-purpose personal robot-assistant with intrinsically safe autonomous decisions.

## References

1. Ryan, R.M., Deci, E.L.: Intrinsic and extrinsic motivations: classic definitions and new directions. Contemporary Educational Psychology 25(1), 54–67 (2000)
2. Prescott, T.J., Montes Gonzalez, F.M., Gurney, K., Humphries, M.D., Redgrav, P.: A robot model of the basal ganglia: Behavior and intrinsic processing. Neural Networks 19(1), 31–61 (2006)
3. Schmidhuber, J.: Exploring the predictable. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computing, pp. 579–612. Springer, Heidelberg (2003)
4. Schmidhuber, J.: Self-Motivated Development Through Rewards for Predictor Errors / Improvements. In: 2005 AAAI Spring Symposium on Developmental Robotics, pp. 1994–1996 (2005)
5. Barto, A.G., Singh, S., Chentanez, N.: Intrinsically Motivated Learning of Hierarchical Collections of Skills. In: ICDL 2004, pp. 112–119 (2004)
6. Stout, A., Konidaris, G.D., Barto, A.G.: Intrinsically Motivated Reinforcement Learning-A Promising Framework For Developmental Robot Learning. In: The AAAI Spring Symposium on Developmental Robotics (2005)
7. Kaplan, F., Oudeyer, P.Y.: Motivational principles for visual know-how development. In: 3rd International Workshop on Epigenetic Robotics, pp. 73–80 (2003)
8. Oudeyer, P.Y., Kaplan, F.: Intrinsic Motivation Systems for Autonomous Mental Development. IEEE Trans. Evol. Comput. 11, 265–286 (2007)
9. Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? A topology of computational approaches. In: Front. Neurorobotics, vol. 1 (2007)
10. Breazeal, C.: Designing Sociable Robots. Bradford Books/MIT Press, Cambridge (2002)
11. Kruusmaa, M., Gavshin, Y., Eppendahl, A.: Don't Do Things You Can't Undo: Reversibility Models for Generating Safe Behaviours. In: ICRA 2007, pp. 1134–1139 (2007)
12. Gavshin, Y., Kruusmaa, M.: Comparative experiments on the emergence of safe behaviours. In: TAROS 2008, pp. 65–70 (2008)
13. Gerkey, B., Vaughan, R., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: ICAR 2003, pp. 317–323 (2003)
14. Lin, M., Zhu, J., Sun, Z.: Learning Obstacle Avoidance Behavior Using Multi-agent Learning with Fuzzy States. In: Bussler, C.J., Fensel, D. (eds.) AIMSA 2004. LNCS (LNAI), vol. 3192, pp. 389–398. Springer, Heidelberg (2004)
15. Gutnisky, D.A., Zanutto, B.S.: Learning Obstacle Avoidance with an Operant Behavior Model. Artificial Life 10(1), 65–81 (2004)
16. Macek, K., Petrovic, I., Peric, N.: A Reinforcement Learning Approach to Obstacle Avoidance of Mobile Robot. In: IEEE AMC 2002, pp. 462–466 (2002)

# Appendix C

Y. Gavshin and M. Kruusmaa, “Identification of Reverse-Action Pairs using Reversibility of Actions,” in *2011 IEEE International Conference on Systems, Man, and Cybernetics, Cybernetics Track*, pp. 2555–2560, IEEE, 2011.

# Identification of Reverse-Action Pairs using Reversibility of Actions

Yuri Gavshin and Maarja Kruusmaa
Centre for Biorobotics
Tallinn University of Technology
Akadeemia tee 15A-111
Tallinn, Estonia
Email: {yury,maarja}@biorobotics.ttu.ee

***Abstract*—This paper presents a practical approach to identification of reverse-action pairs for reversibility-based learning to develop safe behaviors. The approach is to analyze reversibility of consecutive actions by introducing metrics defined on the actions and the states. The experiment is conducted in the Player/Stage simulator to test applicability of the approach. The robot is allowed to move freely in the environment by executing two-dimensional actions – moving forward/backward or rotating left/right. Reversibility of a sum of every consecutive pair of actions is then analyzed to identify the ones that undo each other. As a result, a set of action pairs are identified with a general rule "if moved/rotated by X, move/rotate by -X to undo".**

***Index Terms*—Safety, reversibility, robotics, reverse-action, undo action**

## I. Introduction

This paper is concerned with the reversibility of robot's actions to increase the inherent safety of its behavior. A robot, motivated to suppress irreversible actions, will behave inherently safely as it will avoid actions that cause irreversible damage. Such abstract intrinsic motivation makes the goal of the robot independent of its working environment or the task it fulfills. Instead of specifying routines such as avoiding obstacles, falls, traps, risky regions/routes or staying near to some known landmark, it is rather told not to do things it cannot undo. 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 idea of using an abstract principle to govern robot behavior is not new. Kaplan and Oudeyer showed that a robot can develop visual competences from scratch driven only by internal motivations independent of any particular task: predictability, familiarity and stability [1]. They generalized their approach further and derived a mechanism of Intelligent Adaptive Curiosity, an intrinsic motivation system which pushes a robot towards situations in which it maximizes its learning progress [2].

In the opposite to work of Kaplan and Oudeyer, where the motivational system encourages robot curiosity, our system is driven towards stability and safety. The drive to suppress irreversible actions is a kind of an adaptive homeostatic predictive motivation according to the classification given in the overview paper of computational approaches to intrinsic motivation [3]. In case of reversibility-based learning the homeostatic system of the robot forces it to search for a reverse-action to return back to the initial state and have the ability to repeat the action.

Our motivation to build a system that learns action reversibility is to build safe autonomous learning robots. Safety in robotics has always been an important priority for this research area. In its traditional formulation, the safety in robotics is not viewed in the context of decision-making, but rather as a responsibility of the designer [4]. Possible hazards of robot bodies and manipulators are analyzed by many researchers ([5],[6],[7]), but the safety is viewed mostly in the context of mechanical safety during collisions.

Alternatively, robot safety is considered in a wider philosophical context – the overview of roboethics in [8] identifies most urgent, evident and sensitive ethical problems related to several sub-fields of robotics. Although it reports contributions from dominant moral theories, the overview does not report any work implementing those principles in practice.

A major effort in bringing theory to implementation is exploring the idea of embedding ethical behavior into a military robot, in the form of ethical rules [9]. In contrast to such autonomous, but pre-programmed robot control, we use abstract principles as a basis for safe behavior.

Our underlying idea is that all actions that are reversible, are also intrinsically safe. Most of the harmful actions the robot can do (for example, falling down the stairs, breaking an object, etc.) are also irreversible. There can be irreversible actions that the robot is allowed or expected to do (e.g. a vacuum cleaning robot cleans a floor irreversibly) but then the robot is designed for doing those actions, and it is an informed decision of the user to use this robot for these purposes.

In work of Thomaz and Breazeal the ability to undo last action is used to improve reinforcement learning performance [10]. However, the set of undo actions is predefined by a human beforehand. In our previous work ([11],[12]) we demonstrated the emergence of obstacle avoidance behavior, based purely on internal drive to suppress irreversible actions, based on predefined set of reverse-action pairs.

In this paper we are going to present an approach to identify the action pairs that reverse each other – similar to the ones already used in our work. The pairs of actions are identified by analyzing the execution of a sequence of pseudo-continuous two-dimensional actions, consisting of a pair of commands to move or rotate.

In the following section we explain the concept of reversibility in detail and describe it in a more formal way. In section III we describe the experimental setup and specific implementation details. The experimental results are presented in section IV and discussed in section V. The last section contains conclusions with directions for future work.

## II. Concept of Reversibility

### A. Basic Definitions

A robot's world is a labelled transition system $(S, \Lambda, \rightarrow)$, where $S$ is a set of experienced states, $\Lambda$ 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: $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$. Generally speaking, a composite action $a_{forward}a_{reverse}$ produces a transition from $s_{init}$ to $s_{final}$ through $s_{interim}$ in $W$.

Also, the action sequence $a_{forward}a_{reverse}$ is expected to work for any states $x$ and $y$ with $d_{orig}(x, s_{init}) \leq \varepsilon_{orig}$ and $d_{dest}(y, s_{interim}) \leq \varepsilon_{dest}$, where $d_{orig}$, $d_{dest}$ are metrics on states and $\varepsilon_{orig}$ , $\varepsilon_{dest}$ are their thresholds.

The reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ holds in $W$ if there exists a transition path from $s_{init}$ to $s_{final}$ through $s_{interim}$ consisting of two transitions labelled accordingly $a_{forward}$ and $a_{reverse}$ , and $d_{rev}(s_{init}, s_{final}) \leq \varepsilon_{rev}$ , where $d_{rev}$ is a prametric ( $d_{rev}(x, y) \geq 0$ and $d_{rev}(x, x) = 0$ ) on states and $\varepsilon_{rev}$ is a threshold; fails otherwise.

An action $a_{forward}$ in an arbitrary state $s$ is expected to be reversible (by action $a_{reverse}$ ), if the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ holds and $d_{orig}(s, s_{init}) \leq \varepsilon_{orig}$. A reversibility model of the robot is a set of reversibilities that are expected to hold.

### B. Explanations

The focus of the current experiment is not on the prediction of how safety will the action be, but on identification of action-pairs that undo each other. Although not used directly, the following explanations are vital for understanding the purpose of the experiment, presented in this paper.

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. Selection of such reverse-actions is the aim of this paper.

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 $\varepsilon_{orig}$ 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. In other words, $d_{orig}$ and $\varepsilon_{orig}$ are used to identify a "region" or a "cluster" of states.

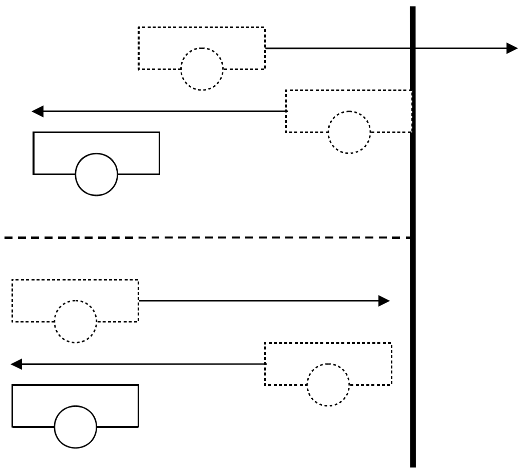
Fig. 1. Obstacle avoidance as a consequence of suppressing irreversible actions. In the upper example the reversibility doesn't hold, in the lower – holds.

A prametric $d_{rev}$ is used to calculate how strongly the reversibility holds by measuring the "distance" between two states.

### C. Emergence of safe behaviors

It should also be explained how and why the obstacle avoidance behavior emerges as a result of avoiding irreversible actions. As an example, let's consider a robot with a proximity sensor in front and two actions – "move 10 steps forward" and "move 10 steps backward". Without loss of generality it can be assumed that "steps" and values of proximity sensors are given in the same 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 (see Fig. 1):

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. This knowledge is inherently intrinsic to the system as it does not depend on the task nor the environment, but only on the robot's drive to increase its stability.

## D. Identification of discrete reverse-action pairs

Throughout the paper, all used metrics are "Manhattan" metrics:

$$d_{manhattan}(p,q) = \Sigma_{i=1}^{n} |p_i - q_i| \ ,$$

where $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$. For this experiment we define a new metric

$$d_{act} = d_{rev} = d_{manhattan}$$

, which determines the difference between the two actions.

To identify the set of discrete reverse-action pairs, a consequence of making a single action should be measured to estimate how much the reverse-action "candidate" reverses the original action. The main part of the experiment is the analysis of reversibility "candidates" – $d_{rev}(s_{init}, s_{final})$ should be calculated for all pairs of consecutive actions.

Such analysis can be viewed is a reverse process of checking if reversibility holds. Consider the reversibility $(s_{init}, a_{forward}, s_{interim}, a_{reverse}, s_{final})$ and the value of $d_{rev}(s_{init}, s_{final})$. In our previous work the $a_{reverse}$ was a reverse-action of the $a_{forward}$ by definition ([11],[12]). In the current paper, on the contrary, the reversibility "candidates" with low $d_{rev}(s_{init}, s_{final})$ are sampled to analyze how to derive $a_{reverse}$ from $a_{forward}$, so that $a_{reverse} = -a_{forward}$.

The next section explains technical details of data acquisition and analysis for the experiment.

## III. Experimental Setup

### A. The Environment

In the experiment a simulated Scitos G5 robot with a laser rangefinder is placed into a big room to allow collision-free movement (see Fig. 2). The environment is simulated using the Player robot platform and the Stage multi-robot simulator [13]. Robot's model closely resembles the real robot – its size, shape and sensor positions are the same; this allows to repeat the experiment on the real robot in the real environment without any changes.

Laser rangefinder is configured to observe the sector of $[-105°, +105°]$ ahead of the robot with $0.5°$ precision; the returned distance is in meters. The state is identified solely by laser rangefinder data: $s = (d_0, d_1, ..., d_{420})$. The action is a two-dimensional command to move or rotate: $a = [m_{trans}, m_{rot}]$ , where $m_{trans}$ is translational movement in meters and $m_{rot}$ is rotation in radians.

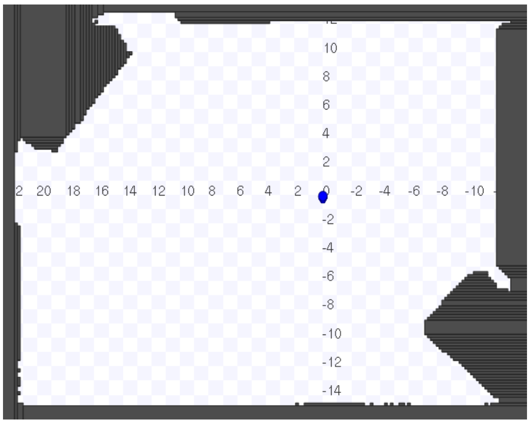

Fig. 2. Simulated environment. Blue figure at $(0, 0)$ is the simulated Scitos G5 robot in its initial position.

Execution of the action $a = [m_{trans}, m_{rot}]$ is implemented as an interactive subroutine, which uses velocity control to steer the robot and its internal odometry to calculate current speeds and identify stop conditions.

### B. The Experiment

The experiment begins with a test-run, where the robot is moving randomly. The actions are generated in such a way, that either $m_{trans}$ or $m_{rot}$ is zero. First, it is decided, whether the robot moves forward/backward ($m_{rot} = 0$) or turns left/right ($m_{trans} = 0$) with equal probability. Then, the value of $m_{trans} \in [-1.0, +1.0]$ or $m_{rot} \in [-1.57, +1.57]$ is selected randomly from the respective interval. The laser rangefinder data before and after making an action are saved to a log-file together with the executed action data.

The acquired data is then analyzed offline by iterating through the log-file. There are several sets of data to be analyzed. As described in section II-D, each single action is analyzed to measure how does the state change as a result of such an actions. For this purpose the $d_{rev}$ of the states before and after the action is calculated for separate sets of translational and rotational actions.

Similarly, pairs of consecutive actions are analyzed as reversibility "candidates", divided into three different sets:

- "TT" – translational+translational
- "RR" – rotatonal+rotatonal
- "RT-TR" – translational+rotatonal or vice-versa

## IV. Results

The result of the analysis of single actions is shown in Fig. 3. Rotational action points are shown as black triangles and gray diamonds represent translational action points. The X axis is a non-zero component of the action: $m_{rot}$ for rotational and $m_{trans}$ for translational actions. The $d_{rev}$ of the states before and after the action is on Y axis.

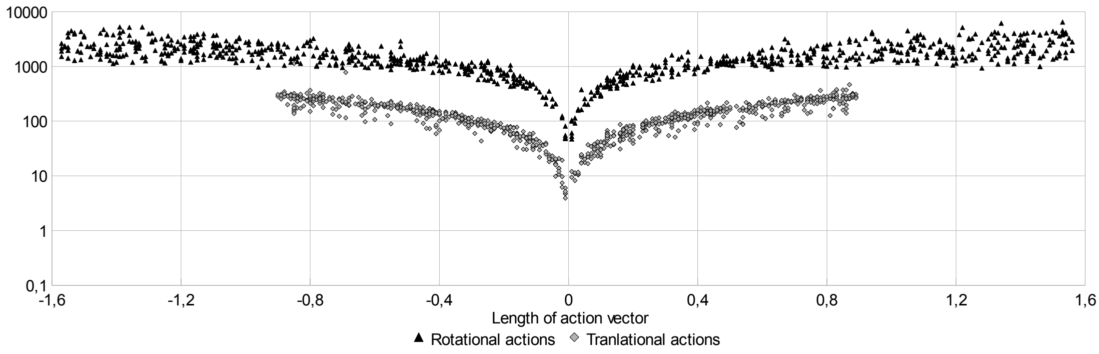

Fig. 3. Calculated $d_{rev}$ of the states before and after the action.

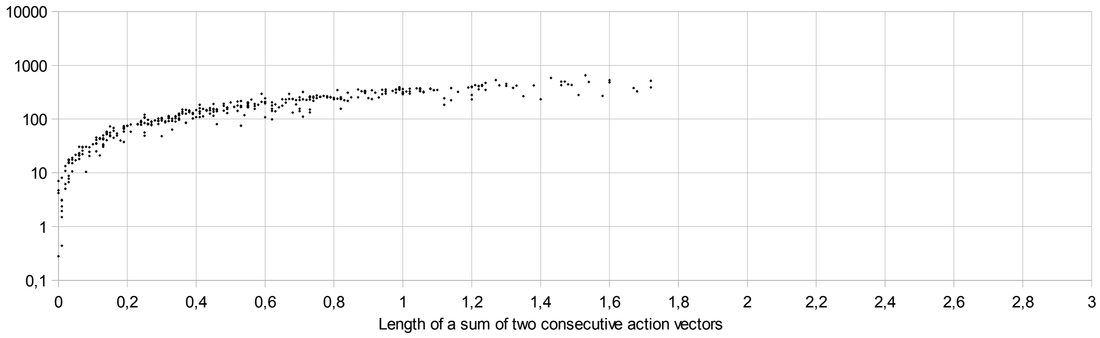

Fig. 4. Calculated $d_{rev}$ of the initial and the final states of two consecutive *translational* actions (TT).

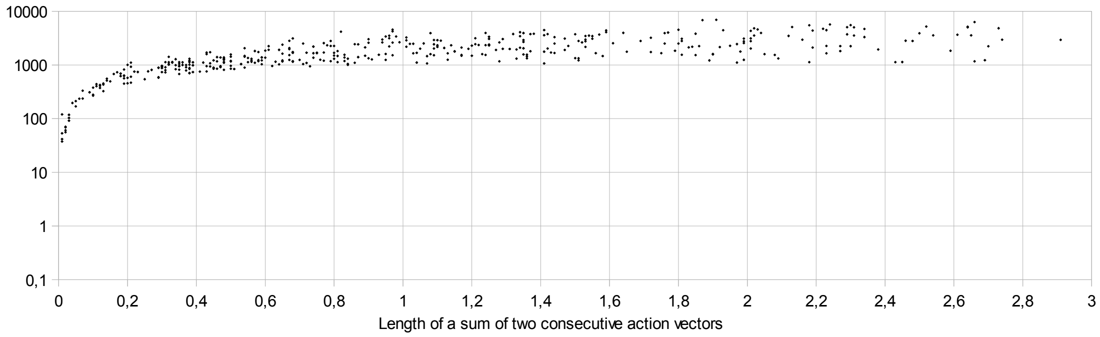

Fig. 5. Calculated $d_{rev}$ of the initial and the final states of two consecutive *rotational* actions (RR).

It is clear that numerical influence of robot rotation on the change to robot's environment is considerably higher, than of robot's translational movement. Rotational movements generate 10 times bigger $d_{rev}$ than translational ones; therefore two different thresholds are used for interpretation of the rest of the results. The first one, $\varepsilon_{revR} = 100$ is for sequences, where there is at least one rotational action, and second one, $\varepsilon_{revT} = 10$ for translational-only sequences.

Figures 4, 5 and 6 visualize the results of analysis of reversibility "candidates" for "TT", "RR" and "RT-TR" data sets respectively . Similarly to Fig. 3, $d_{rev}$ of the states before and after the action is on Y axis. The difference is that the action now consists of two consecutive actions $p = (p_{trans}, p_{rot})$ and $q = (q_{trans}, q_{rot})$. The X axis is the

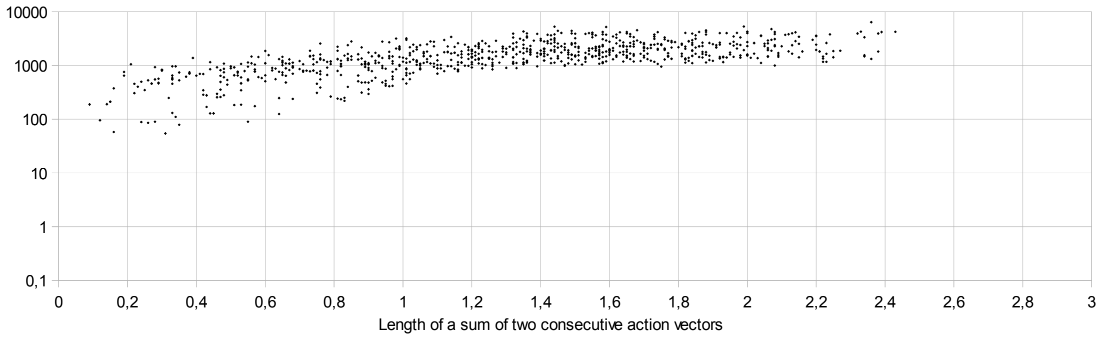

Fig. 6. Calculated $d_{rev}$ of the initial and the final states of consecutive *rotational+translational or translational+rotational* actions (RT-TR).

"Manhattan" length of the sum of action vectors:

$$d_{act}(0, p+q) = |p_{trans} + q_{trans}| + |p_{rot} + q_{rot}| \, .$$

Since in the "TT" and "RR" data sets one of the components is zero, values of X axis in figures 4 and 5 are $|p_{trans}+q_{trans}|$ and $|p_{rot} + q_{rot}|$ respectively.

Data sets "TT" and "RR" exhibit strong dependency of $d_{rev}$ from the value of $d_{act}(0, p+q)$ – the closer to 0 is the sum of consecutive rotations or translations, the less is the distance between the states before and after action $p+q$. In other words, irreversibility of the sequence of actions $p+q$ is proportional to its length. Considering the the fact that in the "TT" and "RR" data sets one of the action's components is zero, a general rule can be derived: "if moved/rotated by X, move/rotate by -X to undo". Further, the actually experienced pairs of actions can be identified by applying the $d_{revT}$ and $d_{revR}$ to "TT" and "RR" data sets respectively.

The "RT-TR" differs from mentioned "TT" and "RR" data sets by the fact that the two consecutive actions are of different types and modules of both rotational and translational parts are added during the $d_{act}(0, p+q)$ calculation. The analysis of the results for "RT-TR" data set predictably reveals no strong dependency of $d_{rev}$ from the value of $d_{act}(0, p+q)$ and no general rules or specific pairs of actions can be identified as reverse-action pairs. Although there are several points below the $\varepsilon_{revR}$ threshold, these points represent the the actions with a very small rotational action and a bigger translational action. This makes the resultant $d_{rev}$ statistically irrelevant, since the $d_{rev}$ of those single translational actions is below $\varepsilon_{revR}$ threshold, applied to mixed sequence of actions.

## V. Discussion

The results are extracted from simulated test-runs. The result of a real test-run might be different due to higher noise and other factors associated with changing the experimental setup from simulated to the real-world environment. However, due to high quality simulation and low noise of the real laser rangefinders, the correlation between the irreversibility of the sequence of actions and the length of their sum will not decrease considerably when the experiments are repeated on the real robot.

Current state of the approach is still preliminary, but the initial results of the simple scenario experiment showed that it is possible to identify reverse-actions pseudo-automatically. For a useful application of this approach, it must be extended to be fully automatic and numerical, instead of the visual analysis and manual threshold selection.

Another obstacle for real-life application of the approach is how to define metrics on the states and the actions. Metrics for the low-dimensional states and actions are easy to define – the experiment, for example, uses the very simple Manhattan metrics. However, even for such low-dimensional states and actions, not every action has a trivial undo action – generally, several actions must be executed to undo the effects of a single action.

One of the important limitations of the current approach is that only two-action sequences are analyzed. Another limitation is that it is much harder to define metrics for complex states and actions, like "door is locked", "washing machine is half-full", "unlock the door", "use washing-machine", etc. Sometimes such states and actions can be transformed into low-dimensional ones, but even then not all vector elements are equally important for state and action difference calculation. Additionally, only a few of such actions have trivial, or even simple reverse-actions. Thus, the problem of defining metrics on complex actions and states must be addressed in the future.

Despite the weak points of the approach, its results can be useful for some researchers. Identified reverse-actions can be used to make exploration policies safer by trying to undo the last action with a highly negative reward. Similar approach can also be used to identify "UNDO" actions for the study in [10], which also showed that the ability to undo the last action increases the learning rate of reinforcement learning.

A set of reverse-action pairs is also a vital component of the reversibility-based learning. In our previous work ([11],[12]) predefined reverse-action pairs were used. This paper partially solved the problem of identification of such reverse-action pairs.

## VI. Conclusions

The aim of this paper was to test the approach to identify the action pairs that reverse each other through analysis of how reversible the random pairs of actions are. We interpret the results as positive, since we were able to identify the general rule "if moved/rotated by X, move/rotate by -X to undo", based on raw sensor data with no prior knowledge about the environment. The current state of reverse-action identification research is still preliminary – simple actions/states are used in a simple environment and the analysis with threshold selection is manual. The approach must be developed further to be applicable and useful in more complex scenarios.

### *A. Future Work*

We plan to repeat the experiment on the real Scitos G5 robot in the real room and to make identification of reverse-action pairs automatic. Further, the approach must be tested in more complicated scenarios with complex actions and states. In the future we will continue to apply the principle of reversibility to develop safe behaviors. Our ultimate goal is a multi-purpose personal robot-assistant with intrinsically safe autonomous decisions.

## References

[1] F. Kaplan, P.Y. Oudeyer, "Motivational principles for visual know-how development", in *Sixth International Workshop on Epigenetic Robotics*, Lund University Cognitive Studies, 2006.

[2] P.Y. Oudeyer, F. Kaplan, "Intrinsic Motivation Systems for Autonomous Mental Development", in *IEEE Transactions on Evolutionary Computation*, Vol. 11, pp 265–286, 2007.

[3] P.Y. Oudeyer, F. Kaplan, "What is intrinsic motivation? A topology of computational approaches", in *Front. Neurorobot.*, 2007.

[4] G. Veruggio, F. Operto, "Roboethics: Social and ethical implications of robotics", in B. Siciliano, O. Khatib (eds.) *Springer Handbook of Robotics*, pp. 1499–1524, Springer, Heidelberg, 2008.

[5] M. Zinn, B. Roth, O. Khatib, and J. K. Salisbury, "A new actuation approach for human friendly robot design," *Int. J. of Robotics Research*, vol. 23, no. 4-5, pp. 379–398, 2004.

[6] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[7] S. Haddadin, A. Albu-Schaeffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *Int. J. of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[8] R. J. Sawyer, "Robot ethics", in *Science*, vol. 318, no. 5853, p. 1037, 2007.

[9] R. C. Arkin, "Governing lethal behavior: Embedding ethics in a hybrid deiberative/reactive robot architecture", in *Tech. Rep. GIT-GVU-07-11*, Georgia Tech, 2007.

[10] A. L. Thomaz, C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners", in *Artificial Intelligence*, vol. 172, no. 6–7, pp. 716–737, 2008.

[11] M. Kruusmaa, Y. Gavshin, A. Eppendahl, "Don't do things you can't undo: reversibility models for generating safe behaviours", in *ICRA 2007*, pp. 1134–1139, 2007.

[12] Y. Gavshin, M. Kruusmaa, "Comparative experiments on the emergence of safe behaviours", in *TAROS 2008*, pp. 65–70, 2008.

[13] B. Gerkey, R. Vaughan, A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems", in *ICAR 2003*, pp. 317–323, 2003.

# Appendix D

Y. Gavshin and M. Kruusmaa, “Assessing Safety of Object Pushing Using the Principle of Reversibility,” in *Hybrid Artificial Intelligent Systems, LNAI* (E. Corchado, M. Kurzynski, and M. Wozniak, eds.), vol. 6678 of *Lecture Notes in Computer Science*, pp. 313–320, Springer Berlin / Heidelberg, 2011.

# Assessing Safety of Object Pushing Using the Principle of Reversibility

Yuri Gavshin and Maarja Kruusmaa

Tallinn University of Technology, Centre for Biorobotics,
Akadeemia tee 15a-111, 12618 Tallinn, Estonia
{yury,maarja}@biorobotics.ttu.ee
http://www.biorobotics.ttu.ee

**Abstract.** This article presents an implementation of an innovative safety module for a robot control architecture. It applies the principle of reversibility to assess intrinsic safety of actions and to adapt robot's behavior. The underlying idea is that all reversible actions are intrinsically safe. A practical experiment is conducted to demonstrate that a robot control architecture can develop complex safe behaviors. This is accomplished by using the safety module in conjunction with human-based knowledge and sufficiently high level of perception. A robot is placed in a room with a movable object while the safety module analyzes movements of the robot and the object. As the result, the robot can identify, for example, that pushing object into a corner is irreversible and thus unsafe.

**Keywords:** Reversibility, safety, abstract principles, roboethics.

## 1 Introduction

The latest statistical report in [1] shows that the number of autonomous service robots for home and professional use is growing. A multi-purpose robot-assistant is the vision for the future. This makes solving the issue of robot safety an important priority for robotics research. In its traditional formulation, the safety in robotics is not viewed in the context of autonomous decision-making, but rather as a responsibility of the designer [2]. Alternatively, robot safety is considered in a wider philosophical context – the overview of roboethics in [3] identifies most urgent, evident and sensitive ethical problems related to several sub-fields of robotics. Although it reports contributions from dominant moral theories, the overview does not report any work implementing those principles in practice.

A major effort in bringing theory to implementation is exploring the idea of embedding ethical behavior into a military robot, in the form of ethical rules [4]. In contrast to such autonomous, but pre-programmed robot control, we use abstract principles as a basis for safe behavior. Our underlying idea is that all actions that are reversible, are also intrinsically safe. Most of the harmful actions the robot can do (for example, falling down the stairs, breaking an object, etc.) are also irreversible. There can be irreversible actions that the robot is allowed or expected to do (e.g. a vacuum cleaning robot cleans a floor irreversibly) but then the robot is designed for doing those actions, and it is an informed decision of the user to use this robot for these purposes.

In our previous work ([5],[6]) we used the principle of reversibility to develop safe behaviors by suppressing irreversible actions. The observed behaviors were obstacle avoidance and locality. However, further applicability of our approach was limited by trivial state identification logic – plain sensor data from the sensors was used. In this paper we are combining this abstract principle with human-based knowledge. It includes, but not limited to, environment modelling, model-aware state identification, localization and planning algorithms.

We argue that such a combination enables development of smarter behaviors, similar to other hybrid intelligent algorithms and applications [7]. One of the strongest points of the reversibility principles governing robot behavior is the abilty to behave safely without any prior knowledge. Scalability and applicability of such pure approach can be extended with human-based knowledge. Abstract principles, on the other hand, can govern robot behavior in unexpected situations, where no pre-defined rules can be applied.

In this paper we report results of the experiment where the simulated and the real robots learn to reversibly manipulate an object by pushing it back and forth. The principle of action reversibility is used to identify intrinsically-safe decisions. The robot is allowed to make such decisions autonomously, while explicit authorization is required for irreversible ones. We provide a formal framework to describe reversible actions and assess safety of system decisions, allowing a test robot to process its experience with the environment.

Next section describes the control system architecture and its safety module together with a reversibility-based sub-module to assess decision safety and alter system behavior. Section 3 contains the structure and the theoretical framework. Further, experimental setup with implementation details are presented. In sections 5 we report results of the experiment and the last section contains conclusions with plans for the future.

## 2 Safety Module for Cognitive Robot Control Architecture

As a testing ground for our research experiments to study the safety module, we created the *PAHPAM* system (Programmable Architecture for Hierarchical Perception, Actuation and Modeling). Since this system is not a primary objective of this article, we will describe only the most relevant of its aspects.

The *PAHPAM* has three standard levels of a general reactive-deliberative architecture – reactive, hybrid and deliberative. Hardware-specific functionality is implemented as a subcomponent of the reactive level, which is located on the robot, while hybrid and deliberative ones can also run on a separate more powerful machine. There are quite many assumptions and prerequisites for the whole system and the safety module to work. We have solved those prerequisites in a minimalistic fashion by hard-coding the routines we cannot generalize and formalize, since most of the problems are difficult research problems by themselves and are yet to be solved. We use Monte-Carlo localization algorithm to solve localization problem. Object identification is simplified by the round shape of the object and the IAV method [8] is used to identify such circular objects in laser rangefinder scans. State identification problem is solved by using only the relevant information with a balanced level of detail. Important part of our robot control system is a model of the environment for analysis, planning and learning. The model doesn't have to be ideal – in case of inconsistencies between the modeled and the actual movement, the model can be updated and a new plan can be created.

The safety module ensures that system's decisions are safe and within predefined bounds – similar to the *ethical governor* in [9]. Before making decision, the system checks with the safety module whether the decision is allowed from the safety perspective (see Fig. 1 for activity diagram). A set of rules/patterns to explicitly (dis)allow specific decisions in specific states is a first stage of safety assessment inside the module. If the specified state-action pair matches those patterns, the answer is generated based on this pre-programmed or learned knowledge. Context- and task-specific overrides (for example, to enable irreversible, but useful actions) should be added to this set of rules/patterns.

When no rule/pattern can be applied and the robot is going to make an autonomous unauthorized decision, a reversibility-based analysis module is used to assess intrinsic safety of making the action from the state in question. Such safety architecture allows to make a hybrid system in terms of cooperation between system's designer knowledge in form of rules with patterns to apply them and a reversibility-based logic used as a backup, when no rules can be applied.

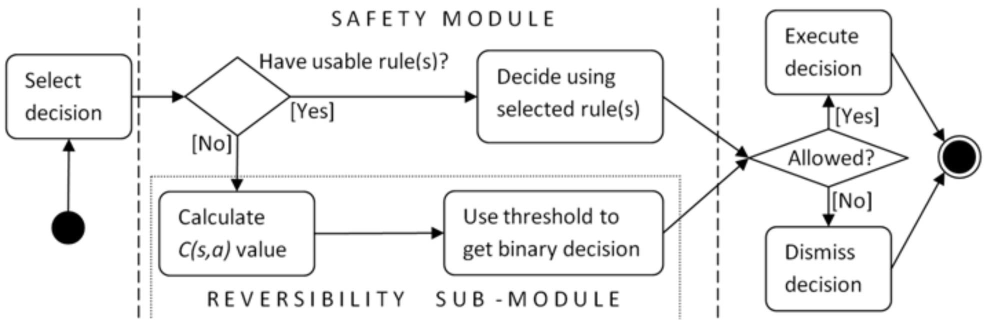

**Fig. 1.** The activity diagram for proposed safety module

## 3 Reversibility Models for Reversibility Based Sub-Module

**Reversibility MDP-Model (RMM)** is a finite Markov Decision Process with a reversibility function $C(s,a) \leq 0$. MDP is a 4-tuple $(S, A., P.(.,.), R.(.,.))$, where $S$ is a finite set of states, $A_s$ is a finite set of actions available from state

$s$, $P_a(s, s')$ is the probability that action $a$ in state $s$ will lead directly to state $s'$, $R_a(s, s')$ is the (expected) immediate cost of making action $a$ from state $s$, followed by a transition to state $s'$ with probability $P_a(s, s')$.

$C(s, a)$ is the total expected "cost" of $s \to s' \to s$ transition, i.e. reversing an action $a$ made from state $s$. $C(s, a) = -\infty$ for absolutely irreversible actions and $C(s, a) = 0$ for perfectly reversible ones. To calculate it, we search for a path $p = (a_0, a_1, .., a_n)$, where $a_0 = a$ to make the $s \to s' \to s$ transition. This is done by iterating through the set $P_s$ of possible paths; for every candidate path $p = (a_0, a_1, .., a_n)$, the candidate $C_p(s, a)$ value is calculated as follows:

$$C_p(s, a) = \min(\sum_{0}^{n} R_{a_i}(s_i, s'_i) \cdot P_{a_i}(s_i, s'_i) : s'_n = s) \ . \qquad (1)$$

If none of the possible $s'_n = s$, then $C_p(s, a) = -\infty$. If there are no suitable candidate paths found, $C(s, a) = -\infty$, otherwise the maximum (or "sufficiently high") $C_p(s, a)$ value of one of the candidate paths is returned as a result. A value is "sufficiently high", when $C_p(s, a) \geq C_{min}$. For the binary decision, action $a$ from state $s$ is reversible, if $C(s, a) \geq C_{rev}$. Both $C_{min}$ and $C_{rev}$ threshold values are set by a cognitive system or a designer, based on the knowledge about the context.

In other words, with $C_p(s, a)$ we predict a "cost" of making a sequence of actions $p = (a_0, a_1, .., a_n)$, taking in consideration only the possible outcomes, where final state is $s$. Path with the smallest $C_p(s, a)$ is the easiest way to reverse action $a$ from state $s$ and its cost is returned as $C(s, a)$ assessment value. A trivial candidate for a path is a single action $a$ itself: if $C_{p=(a)}(s, a) \geq C_{min}$, then $C_{p=(a)}(s, a)$ is returned as $C(s, a)$ value.

The main purpose of the reversibility-based module is to assess the safety of intrinsic decisions through the study of their reversibility. This sub-module, as the safety module in general, uses knowledge from the current context chosen by a cognitive system. Every context has two reversibility models: $RMMi$ for internally simulated and $RMMa$ for the actual experience. The purpose of such internal simulation is to plan robot's actions and identify (ir)reversible actions before actually making them for the first time.

$RMMi$ and $RMMa$ share the same sets $S$ and $A$., generated by state and action identification modules of a cognitive system. Probability function $P_a(s, s')$ is derived from observed actual transitions for $RMMa$ or simulated ones for $RMMi$. Similarly, reward function $R_a(s, s') \leq 0$ is derived from the simulated or actually observed cost (energy, time, damage, etc) of the action.

The on-line nature of assesment considerably limits the number of candidate paths to try. To overcome this, we use $C_{min}$ threshold to stop search process when at least one $C_p(s, a) \geq C_{min}$ value is found. Additionally, our implementation of reversibility and $C(s, a)$ calculation is based on cyclical state-action transitions. Such cycles can be given in advance by a human, deduced theoretically from an internal model of the environment or identified from statistics of actual state-action transitions.

A simple example of such a cycle is a composite action “’go 1 metre forward’, then ’go 1 metre backward”’, or vice-versa. A set of cycles can be used as candidate paths for $C(s, a)$ calculation – the first action of starting sequence of actions in the cycle is expected to be undone by the rest of the cycle.

## 4 Experimental Setup

The purpose of the experiment is to demonstrate how the forementioned action cycles can be identified and then used to assess action safety on-line through $C(s, a)$ calculation. The main purpose of the experiment is to better explain our approach and to show how the reversibility principle works in more complex scenarios than we have previously used. Our experiments in [5] and [6] showed how a robot can identify irreversible actions in the context of self-movement and demonstrated safe behaviors by avoiding such actions. In this experiment we want a robot to be able to undo the change to the environment – a round movable object. A practical example of such a behavior could be the vacuum cleaner, which cleans not only the free space, but also under movable objects placing them back after cleaning the area initially occupied by the object. As a result, after cleaning the floor, the room layout would stay the same, unless instructed otherwise.

In our experiment we use MetraLabs’ Scitos G5 robot; test-runs are made on the actual robot and in a simulated environment, which is a copy of the actual “room” (see Fig. 2). The robot control framework is connected to *Player* server [10], which in turn controls either the actual robot or its model in *Stage* simulator [10]. In our setup the “green” object is the only round item in the environment and its approximate radius is known. The round object’s size and its position in robot’s coordinates are identified from laser rangefinder scans, filtering out the occasional “wrong” objects by radius threshold; laser sensor position and settings are known. In this experiment we use the following set of

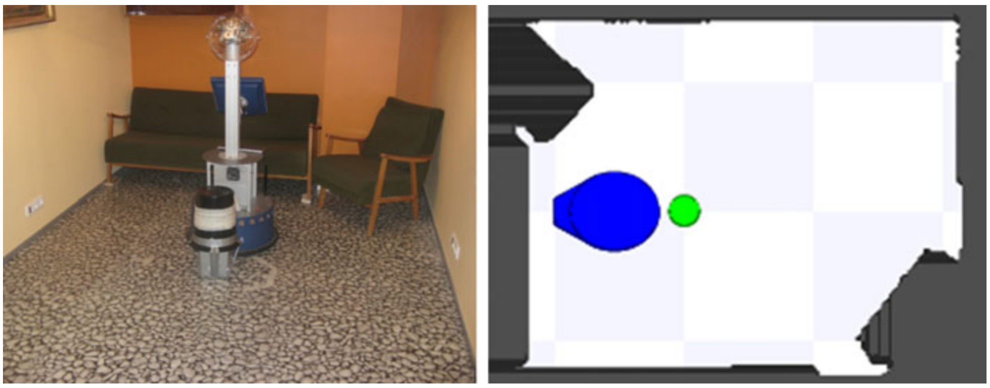

**Fig. 2.** The physical experimental setup of the room *(left)* and the simulation in Player/Stage *(right)*; the grid-map is generated from the real room

actions: "F" – move 0.6 metres forward, "B" – move 0.6 metres backward, "f" – move 0.15 metres forward, "b" – move 0.15 metres backward, "L" – rotate 60 degrees left and "R" – rotate 60 degrees right. States of the robot and the object are identified by X and Y coordinates, rounded to the nearest multiple of 0.13 metres and orientation, rounded to the nearest multiple of 0.13 radians; since the "green" object is round, its orientation is always 0.

We start our experiment with a free movement of the robot to collect statistics (average and standard deviation of covered distance and rotated angle for both local and global odometry) for the six actions used. If the robot pushes the "green" object we collect statistics about the initial and final distance between centers of the robot and the obstacle for the actions that change the state (position) of the object. For simplicity, we analyze only the "F" action effect on the object in this experiment. Additionally, we narrow analysis to specific relative position of the object in robot internal frame coordinates – when object is placed directly in front of the robot, 0.5 meters from its center of rotation.

After the data is collected we can use it to simulate different paths internally, taking only robot and object interaction into account. Next, we search for the cycles – paths that put the "green" object back to its initial state. This part can be done offline – search through the space of possible paths is a time-consuming process. Currently, on our test machine with Radeon HD5770 video, GPU-based exhaustive analysis of 11-step paths (which gives the first valid cycles) takes 10 seconds. We are searching for cycles by simple iteration over all possible paths, starting from the shortest ones. We limit the search space by setting the maximum allowed runtime (10 minutes) and length of the path (20 actions). The search is executed in a separate thread, which allows to use new path as soon as it is identified and the search can be stopped, if needed.

In the final part of our experiment the safety module is used to govern behavior of a robot on-line. The previously identified cycles that start with "F" are simulated internally, now taking immovable obstacles into account. $P_a(s, s')$ is calculated by executing internal simulation with obtained physical movement parameters. Possible next states with possibilities are received by analyzing 10 such actions with normally distributed random error, using standard deviation obtained in the beginning of the experiment. $R_a(s, s')$ is 0 minus the length of the movement in meters or rotation angle in radians; if collision is detected by simulation code, then the $R_a(s, s')$ is additionally multiplied by 100. Threshold values were set as follows: $C_{min} = 2$, $C_{rev} = 20$.

## 5 Results

The data acquisition part of the experiment revealed virtually no difference between simulated and real parameteres of physical movements in this task of a robot moving itself and a round obstacle. Therefore, most robot test-runs were performed in Player/Stage simulation, where data acquisition is fast and automated. The experiments showed that many paths can be found to successfully undo the movement of the "green" object by "F" action of the robot. The paths

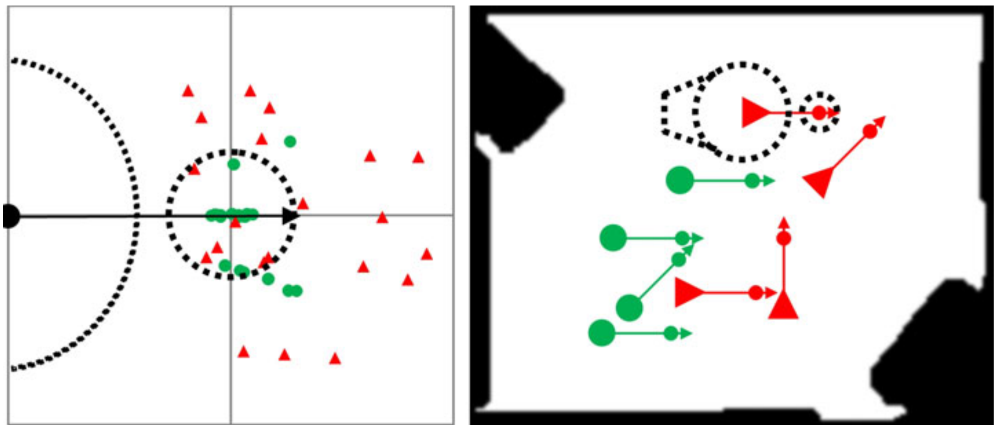

**Fig. 3.** The result of wall-free trials to reverse "F" action in a robot-centered reference frame are *on the left*, grid size is 0.5 metres. The result of the trials in the room with walls in the global reference frame are *on the right*.

go around the object and push it from the opposite side, some make "b" action before going around and some don't.

Fig. 3 on the left shows the situations learned by the end of the trial. The picture is drawn in robot-centered coordinates with the black arrow representing "F" action. The green dots are the positions of the object, from where the pushing is considered reversible. Red triangles are positions of objects from where the pushing of the objects is considered irreversible. It is easy to see that the situations where the object is in front of the robot have been successfully reversed. However, more complicated situations, where the robot touches the object with its side and the objects slides away, are harder to undo. Whereas the two additional line-like clusters of green dots are unexpected – the robot pushes the object away from X axis of initial robot's pose and then pushing it back while finishing the path around the object. It shows that the robot is even able to learn to reverse actions influenced by rather complicated physics of sliding and friction.

The result of safety module governing the "F" action in the global reference frame is shown in Fig. 3, on the right – robot's pose with object position and action length is overlaid upon the map of the room. Reversible object pushing poses, identified by the green enlarged dots with arrows are the ones made towards the object near the center of the room while the robot is closer to room's walls. The red combinations of enlarged triangles with arrows represent irreversible situations and the robot has correctly identified that there is no room to maneuver around the object to push it back.

## 6 Conclusions

Based on the results of the experiment we conclude that non-trivial and quite complex cycles of actions can be successfully identified, allowing the robot to manipulate objects in a reversible manner – pushing them from one side and undoing such action by driving around the objects and pushing them back from the opposite side. Suppression of the irreversible actions while taking immovable objects into account results in further increase of behavioral complexity – robot "understands" that pushing object into a corner is irreversible and thus unsafe.

**Future Work.** Our short-term plans are to optimize the path search and simulation algorithms. In the long-term we are going to use the same approach of using abstract principles to develop safe behaviors, but moving further towards more complex and real-life problems/scenarios.

**Acknowledgments.** This research was supported in part by scholarship from Estonian Information Technology Foundation.

## References

1. IFR Statistical Department, `http://www.worldrobotics.org/downloads/PR_2010-09-14_service_EN.pdf`
2. Veruggio, G., Operto, F.: Roboethics: Social and ethical implications of robotics. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 1499–1524. Springer, Heidelberg (2008)
3. Sawyer, R.J.: Robot ethics. Science 318(5853), 1037 (2007)
4. Arkin, R.C.: Governing lethal behavior: Embedding ethics in a hybrid deiberative/reactive robot architecture. Tech. Rep. GIT-GVU-07-11, Georgia Tech. (2007)
5. Kruusmaa, M., Gavshin, Y., Eppendahl, A.: Don't do things you can't undo: Reversibility models for generating safe behaviours. In: ICRA 2007, pp. 1134–1139 (2007)
6. Gavshin, Y., Kruusmaa, M.: Comparative experiments on the emergence of safe behaviours. In: TAROS 2008, pp. 65–71 (2008)
7. Corchado, E., Abraham, A., de Carvalho, A.C.P.L.F.: Hybrid intelligent algorithms and applications. J. Inf. Sci. 180(14), 2633–2634 (2010)
8. Xavier, J., Pacheco, M., Castro, D., Ruano, A., Nunes, U.: Fast line, arc/circle and leg detection from laser scan data in a player driver. In: ICRA 2005, pp. 3930–3935 (2005)
9. Arkin, R., Ulam, P.: An ethical adaptor: Behavioral modification derived from moral emotions. In: CIRA 2009, pp. 381–387 (2009)
10. Gerkey, B., Vaughan, R., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: ICAR 2003, pp. 317–323 (2003)

# Appendix E

Y. Gavshin and M. Kruusmaa, “Improving Area Coverage by Reversible Object Pushing,” in *15th International Conference on Advanced Robotics*, pp. 415–420, IEEE, 2011.

**The 15th International Conference on Advanced Robotics**
**Tallinn University of Technology**
**Tallinn, Estonia, June 20-23, 2011**

# Improving Area Coverage by Reversible Object Pushing

Yuri Gavshin, Maarja Kruusmaa
Centre for Biorobotics
Tallinn University of Technology
Akadeemia tee 15A-111
Tallinn, 12618, Estonia
yury,maarja@biorobotics.ttu.ee

*Abstract*— **This article presents the implementation of an innovative safety module for a robot control architecture. It applies the principle of reversibility to assess intrinsic safety of actions and to adapt robot's behavior. The underlying idea is that all reversible actions are intrinsically safe.**

**A practical experiment is conducted to demonstrate the approach. The robot's task is to cover a given area with a movable object inside. A governor, acting upon the principle of action reversibility, allows pushing only if such an action is reversible. It is compared with two other governors – one always allows all the actions and the other allows only actions that will not move the object. The covered area and changes between initial and final position of the object are measured in the end of each test run.**

**The proposed governor allows full area coverage with minimal change in object position, while other governors exhibit either incomplete area coverage or significant change in object position. This is interpreted as the ability of the proposed reversibility-based approach to generate both effective and safe robot behaviors.**

## I. INTRODUCTION

Robots are a part of our lives. It started with industrial robotics and now robots are entering our everyday lives – car driving assistant technologies, autonomous vacuum cleaners, robotic dogs and other toys for children and adults. This trend will definitely continue – the number of autonomous service robots for home and professional use is growing fast [1]. The robots are becoming increasingly autonomous – vacuum cleaners do their job without owner intervention. Also major car production companies have plans for autonomous vehicles with an "autopilot". There already exist several successfully working car prototypes with "autonomous driving" abilities, made by universities in cooperation with private companies [2]. Governments also support such trends with funding of competitions and projects.

However, our households are mostly robot-free and there may be several reasons for that. Robot hardware is expensive because production volumes are small; however advances in technology, increasing volumes of production and competition between companies will lower the prices eventually. Robot software programming is resource-consuming, but it is becoming easier to develop robot software due to increasing on-board computing power, better sensors and a growing codebase of various robotics algorithms. Also, robot's feature set is quite limited – usually a robot does one specific task, for example, vacuum-cleaning the floor, but not washing it or ironing your clothes.

We strongly believe that in the future the use of robots in our households will be limited more by social and safety aspects, than the price and functionality. The ideal household robot would be a multi-purpose personal robot-assistant you can trust, which is autonomous, yet its decisions are safe. This makes solving the issue of robot safety an important priority, especially in the context of autonomous decision making, when robot's response is not pre-programmed.

Traditionally, the system designer is responsible for safety in robotic systems. Possible hazards of robot bodies and manipulators are analyzed by many researchers ([3],[4],[5]), but the safety is viewed not in the context of autonomous decision making, but in the context of mechanical safety during collisions.

Alternatively, robot safety is considered in a wider philosophical context. The overview of roboethics in [6] identifies the most important ethical problems related to several sub-fields of robotics and reports contributions from moral theories. However, the overview does not report any work implementing those principles in practice. A major effort in implementation of roboethics is embedding ethical behavior into a military robot in the form of ethical rules [7].

In contrast to such autonomous, but pre-programmed robot control, we use abstract principles as a basis for safe behavior. Our underlying idea is that all actions that are reversible, are also intrinsically safe. The safety comes from the ability to reverse the effect of the action – return back to the initial state before making the action. Also, all the harmful actions the robot can do are irreversible, for example, falling down the stairs, breaking an object, hurting a human, etc. However, some allowed and desired robot actions are irreversible, for example, a vacuum cleaning robot that cleans floors irreversibly or a military robot that destroys targets. In these cases such actions are the purpose of the robot existence. The robot is designed for doing those actions, and it is an informed decision of the user to use these robots for such purposes.

In our work we use the principle of reversibility to develop safe robot behaviors by suppressing irreversible actions. Our experiments in [8] demonstrated emergence of safe behavior of obstacle avoidance on two simulated and two real robots, based purely on actions' reversibility. We argue that the ability to identify irreversible actions and undo reversible ones is crucial for truly autonomous decision making, when no pre-programmed rules can be applied.

In this paper we provide a formal framework to describe reversible actions and assess safety of system decisions, allowing a test robot to process its experience with the environment. We are experimenting with a safety module, which utilizes such a framework as a sub-module. As an example, we set up an experiment where the robot is given a task to cover a specific area in its environment, containing a movable object. The object decreases accessible space and makes it impossible to cover the entire area without moving it. The principle of action reversibility is used to identify intrinsically safe decisions. This way the robot learns to reversibly manipulate an object – pushing it and placing it back afterwards. We benchmark our approach by allowing the robot to make intrinsically safe decisions autonomously by traversing the area in the "lawnmower" fashion while measuring the covered area and the difference between initial and final positions of the object. A comparison is made to two simpler pre-programmed approaches: (i) all actions are allowed; (ii) only the actions that don't change the position of the object are allowed.

The next section contains the theoretical framework for the proposed approach. Sections III and IV describe application details of the proposed approach. Further, experimental setup with implementation details are presented. In section VI we report results of the experiment. The last section contains conclusions with plans for the future.

## II. THE APPROACH

This section contains theoretical basis and application details of the proposed approach to intrinsic safety of robot's decisions.

### A. Formal framework

*Reversibility MDP-Model (RMM)* is a finite Markov Decision Process with a reversibility function $C(s,a) \leq 0$.

MDP is a 4-tuple $(S, A., P.(.,.), R.(.,.))$, where $S$ is a finite set of states, $A_s$ is a finite set of actions available in state $s$, $P_a(s,s')$ is the probability that action $a$ in state $s$ will lead directly to state $s'$, $R_a(s,s')$ is the (expected) immediate cost of making action $a$ in state $s$, followed by a transition to state $s'$ with probability $P_a(s,s')$.

The value of $C(s,a)$ is the total expected cost of $s \rightarrow s' \rightarrow s$ transition, i.e. reversing an action $a$ made in state $s$. $C(s,a) = -\infty$ for absolutely irreversible actions and $C(s,a) = 0$ for perfectly reversible ones.

To calculate the cost, a path $p = (a_0, a_1, .., a_n)$ must be found to make the $s \rightarrow s' \rightarrow s$ transition. This is done by iterating through the pre-selected set $P_s$ of candidate paths that have $a$ as first action. For every candidate path $p = (a_0, a_1, .., a_n)$, the $C_p(s,a)$ value is calculated as follows:

$$C_p(s,a) = \min(\sum_0^n R_{a_i}(s_i, s'_i) \cdot P_{a_i}(s_i, s'_i) : s'_n = s) \ .$$

If none of the possible $s'_n = s$, then $C_p(s,a) = -\infty$.

The cost is the maximum of the analyzed $C_p(s,a)$ values:

$$C(s,a) = max(C_p(s,a)) \ .$$

If there are no candidate paths found, then $C(s,a) = -\infty$.

Action $a$ in state $s$ is called *reversible* if $C(s,a) \geq C_{rev}$. If $C(s,a) \geq C_{min} \geq C_{rev}$, action $a$ in state $s$ is called *super-reversible*.

The rationale behind introduction of the $C_{min}$ threshold to optimize $C(s,a)$ calculation – search for the maximum can be stopped, if a path with $C_p(s,a) \geq C_{min}$ is found:

$$C(s,a) = C_{p_i}(s,a), \ if \ C_{p_i}(s,a) \geq C_{min} \ .$$

Simply put, with $C_p(s,a)$ a cost of making a sequence of actions $p = (a_0, a_1, .., a_n)$ is predicted, taking in consideration only the possible outcomes where the final state is $s$. A set of paths is analyzed and the $C_p(s,a)$ value is calculated for each path. The first path with $C_p(s,a) \geq C_{min}$ or with the maximum $C_p(s,a)$ is the selected way to make action $a$ in state $s$ and then return back to the initial state $s$. The cost of such a cycle is the value of $C(s,a)$ and the sub-sequence of actions $p_{undo} = (a_1, .., a_n)$ is called an *undo-path*.

The obvious candidate path is the action $a$ itself: if $C_{p=(a)}(s,a) \geq C_{min}$, then $C(s,a) = C_{p=(a)}(s,a)$. In this case the action $a$ in state $s$ is reversible and the undo-path is empty.

### B. Dependencies

A number of prerequisites must be met for a successful implementation. The approach depends on the following external sub-routines and parameters:

*1) reversibility and run-time parameters:* The appropriate values must be set externally by a (cognitive) control system or a designer, based on the knowledge about the current context. $C_{rev}$ is used to get binary reversible/irreversible assessment result. $C_{min}$ and maximum numbers of (analyzable paths)/(actions per path)/(seconds to run) are used to optimize assessment time and consumed resources.

*2) state/action identification:* States and actions are discrete. Details of their identification and implementation are irrelevant for the approach. Any state identification logic can be used. Action can be a discrete value of some continuous action or a decision/sub-program to be executed.

*3) environment simulation:* It is of a great importance to simulate dynamics of the environment – execution of thousands and millions of path candidates in a real environment it simply impossible. The main requirement for simulation sub-routine is the ability to use states and actions, provided by state and action identification sub-routines.

*4) planning:* Candidate path selection for $C_p(s,a)$ calculation is crucial for the approach – bad planning logic can miss the possible path to undo the action or provide too many candidates, making the execution time unacceptably long. The planning sub-routine must also be able to use states and actions provided by state and action identification sub-routines.

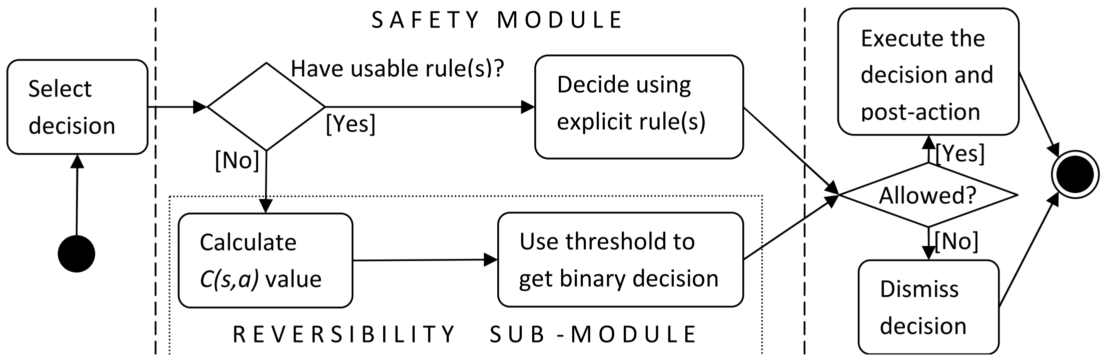

Fig. 1. Activity diagram for the proposed architecture with the safety module and its reversibility sub-module.

## III. SAFETY ARCHITECHTURE

This section describes the safety architecture applying the principle of reversibility for action safety assessment, integrated into our own robot control architecture.

### A. Safety Module

The proposed approach is implemented as the reversibility sub-module of the safety module for a robot control architecture. Such a safety module ensures that system's decisions are safe and within predefined bounds – similar to the *ethical governor* in [9]. Before making the decision, a system checks with the safety module whether the decision is allowed from the safety perspective (see Fig. 1 for the activity diagram). The query to the safety module consists of the state-action pair $(s, a)$ to be analyzed. The safety module returns a tuple: the boolean value, whether the action $a$ is allowed in state $s$, and the *post-action* – a sequence of actions to be done after the initial one to ensure safety.

The first stage of safety assessment inside the module is application of the explicit safety rules to permit or prohibit actions. If the specified state-action pair matches any of the explicit rules, the answer is generated based on this pre-programmed or learned knowledge. If no explicit rule can be applied, the reversibility module is queried to get a safety assessment. When the action is prohibited by the safety module, either explicitly or by reversibility sub-module, no post-actions are required, since the initial one is not executed.

Context- and task-specific overrides should be added to this set of explicit safety rules. For example, an explicit rule can prohibit to push any objects – "don't drive ahead, if there is an obstacle in front". A rule can also allow or prohibit all the actions in all states. Also, a pre-programmed post-action can be associated with a rule, for example "inform a user, if any object was pushed".

### B. Reversibility module

The main purpose of the reversibility module is to assess the safety of intrinsic decisions through the study of their reversibility. When no explicit rule can be applied and the robot is going to make an autonomous unauthorized decision, the reversibility module is used to assess intrinsic safety of making the action from the state in question. The action is considered safe if it is reversible – the robot knows how to undo it; unsafe actions are prohibited. The reversibility of the action $a$ in state $s$ is measured by the value of the $C(s, a)$ function, described in section II.

The reversibility module can allow an action without post-action to undo it only if the action does not change the state and there is no need to undo it. If there is a sequence of actions to undo the initial one and the initial action is reversible, the action is allowed and the undo-path $p_{undo}$ is returned as the post-action.

### C. Example

Consider the situation when a vacuum-cleaning robot is cleaning the floor and plans to move the chair in front of the robot to clean beneath the chair. The safety module is queried whether it is allowed to push the chair. If the robot is explicitly prohibited to push chairs, then the action is not allowed. The action is allowed with optional post-action if the rules explicitly allow pushing chairs. Alternatively, when no explicit rule can be applied, the reversibility of pushing the chair is analyzed. If the robot has the knowledge and the ability to go around the chair and push it back to its initial position, then the action is reversible and it is allowed with a post-action of "go around the chair and push it back". However, the cycle of making the action followed by the undo-path can be prone to errors, very long or requiring too much energy, etc.: $C(s, a) < C_{rev}$. In this case the action is deemed irreversible and is prohibited.

## IV. IMPLEMENTATION DETAILS

This section contains general implementation details of the approach, while specific technical details can be found in section V-F.

### A. Action and state identification

In the experiment reported in this paper the robot is moving itself inside a room with unmovable walls and a single movable object. Available actions are: 4 discrete translational movements forward/backward and 2 discrete rotations left/right. The specific lengths for the actions are selected by the authors by hand. A state is identified by 2D object coordinates and orientation in the global reference frame; the space state is divided into a grid of discrete states.

### B. Candidate path selection

Currently the control system doesn't have a planning module in a regular sense. Instead, our implementation of candidate path selection for $C(s, a)$ calculation is based on the cycles in simulated state-action transitions. The search for usable cycles was presented in [10]. The cycles were found to undo the action that pushes the object; a single location of the object in the robot's reference frame was analyzed.

In this experiment we extend the search procedure by analyzing many points in robot's reference frame, so that the robot can undo pushing of the object from different relative positions. As a result, a set of "cycles" is obtained for every analyzed relative position of the object. These sets are used as path candidates during $C(s, a)$ calculation – the first action in the cycle is expected to be undone by the rest of the cycle.

### C. MDP for reversibility models

Every context has a $RMMai$ reversibility model for the $C(s, a)$ calculation. Such reversibility model is a mixture of the actual ($RMMa$) and simulated ($RMMi$) experience.

$RMMai$, $RMMi$ and $RMMa$ share the same sets $S$ and $A$., generated by external state and action identification modules of a system.

Probability function $P_a(s, s')$ is derived from observed actual transitions for $RMMa$ or simulated ones for $RMMi$. Similarly, reward function $R_a(s, s') \leq 0$ is derived from the simulated or actually observed cost (energy, time, damage, etc) of the action.

In this paper the $RMMi$ is used as $RMMai$. Experienced transitions are used to update parameters of environment simulation logic: statistics about changes in robot and object positions for every action.

## V. EXPERIMENTAL SETUP

### A. Robot and Environment

The MetraLabs' Scitos G5 robot is used for the real-world part of the experiment and its model is used in the simulated environment, which is a copy of the actual room for the real robot (see Fig. 2). The robot control framework is connected to *Player* server [11], which in turn controls either the actual robot or its model in *Stage* simulator [11].

### B. The Experiment

Robot's behavior mimics a vacuum cleaner. The task for the robot control algorithm is to cover the area in a "lawnmower" pattern. For an abstract vacuum cleaner robot, covering the area means cleaning it. During the experiment the covered area is measured together with the difference between the initial and the final positions of the visible objects in the global reference frame. Without loss of generality, the experiment is conducted with a single object to simplify the setup and make the object identification more robust.

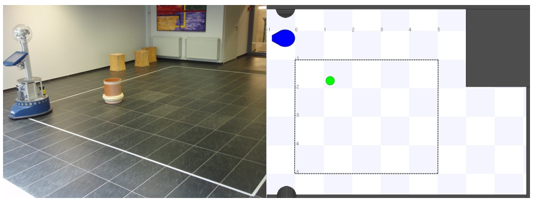

Fig. 2. The physical experimental setup of the corridor *(left)* and the simulation in Player/Stage *(right)*. Dotted line represents the desired area to be covered.

To benchmark our approach, the experiment consist of three sets of test runs with different "modes" (sets of explicit rules) of the safety module to compare their performance:

- "reversibility" – without rules, the decision and a post-action is generated by reversibility module
- "obstacle avoidance" – rules allow only the actions that don't move the object
- "no safety" – all actions are allowed.

The experiment starts with the data acquisition part to collect statistical data about the robot and the environment. It is followed by the data processing, that calculates statistics and prepares candidate cycles for $C(s, a)$ calculation. When data is collected and processed, the actual test-runs are executed to measure the covered area and the distance between the initial and the final positions of the object.

### C. Data Acquisition

In this experiment the robot can choose between the following actions:

- "f" – move 0.15 metres forward
- "F" – move 0.60 metres forward
- "b" – move 0.15 metres backward
- "B" – move 0.60 metres backward
- "L" – rotate 60 degrees left
- "R" – rotate 60 degrees right.

During the data acquisition part of the experiment the robot moves pseudo-randomly to collect statistics for the actions. For every execution of the action, the initial and final coordinates and orientation of the robot are saved for both local and global odometry. The object's movement data in the robot reference frame is also collected to gather statistics on how robot's actions influence the object.

### D. Data Processing

After data is acquired it is processed before the test-runs. First, for each of the actions, robot movement statistics is calculated from the collected data – average and standard deviation of covered distance along robot's X and Y axises and rotated angle. Then this data is used to create the internal model of the environment. This model is then tweaked to simulate resulting object movements as precisely as possible.

Experienced pushes of the object are compared with the simulated ones using different parameters; the best parameters are chosen.

Afterwards, the tweaked model is used to prepare the candidate cycles for on-line $C(s,a)$ calculation during the test run. It is done similarly to [10], but in this experiment hundreds of relative position of the object in robot reference frame are analyzed. The area around the robot is divided into cells of size $0.03$ meters and centers of the cells are candidate points to be analyzed for action reversibility when the object is at that position. The search for cycles is conducted for those points where the object position changes as a result of the action; there are separate sets of such points for every action. On our test machine (Intel Core i7-920 CPU, NVIDIA Tesla C1060 GPU, 6GB of RAM) the calculation takes approximately one hour for the simulated and four hours for the real environment.

### *E. Test Runs*

The final part is the actual experiment, which consists of multiple test-runs with different modes of the safety module. In the beginning of each test run, the object is placed randomly inside the desired area.

The area coverage algorithm for all three "modes" is the same, it queries the safety module before each action. If the action is prohibited, an alternative action is chosen and the query is repeated. If the new action is allowed, it is executed, together with the post-action, supplied by the safety module.

In "reversibility" mode no explicit safety rules are applied and the reversibility module is responsible for safety assessment. The cycles, generated in the off-line data processing part of the experiment, are used during the on-line assessment of action reversibility.

The analyzed point, closest to the current position of the object, is selected from action's set of such points. Each analyzed point has a set of the previously identified cycles associated with it. These candidates path are used to calculate $C(s,a)$ and select the associated cycle as the post-action.

The paths are simulated internally in $RMMai$, taking possible immovable obstacles into account. $P_a(s,s')$ is calculated by executing the internal simulation of the actions. Possible next states with probabilities are calculated by analyzing 10 different outcomes of the actions with normally distributed random error, using standard deviation obtained in the data acquisition part of the experiment. $R_a(s,s')$ is calculated as 0 minus the length of the movement in meters or rotation angle in radians. If collision with a wall is detected by the simulation code, then the $R_a(s,s')$ is additionally multiplied by 100.

The "obstacle avoidance" mode uses the same internal model of the environment to predict collisions. The action is prohibited if collision is predicted, allowed otherwise; the post-action is always empty. The "no safety" mode is the most naive approach – all the actions are allowed.

### *F. Technical Details*

The test area is 5 by 4 meters – the dashed rectangle in Fig. 2 (right). It is divided into $80$ cells of size $0.5$ metres. The cell is considered to be covered, if the center of the robot's round compartment enters it. The moving algorithm tries to visit all the cells in a lawnmower fashion. The safety module is queried each time before the desired action is executed. If the action is prohibited, a new action is generated and the process repeats. The experiment stops when all the cells are visited or manually, if it is clear that no more cells can be visited.

Robot coordinates in the global reference frame are provided by AMCL driver of the Player/Stage project [11]. Object identification is simplified by the round shape of the object and the IAV method [12] is used to identify the object in laser rangefinder scans.

In our experimental setup the object is the only round item in the environment and its approximate radius is known – $0.15$ metres. The round object's size and its position in robot's coordinates are identified from laser rangefinder scans, filtering out the occasional "wrong" objects by radius threshold; laser sensor position and settings are known.

State space's 2D position component is divided into $0.1\ m$ cells and orientation component is divided into $0.173\ rad$ sectors. Since the object is round, its orientation is always 0.

Threshold values are set as follows: $C_{min} = -2$, $C_{rev} = -20$. Maximum number of actions in a path/cycle is set to 14. The search for cycles is a plain iteration over the possible paths of given length. During $C(s,a)$ calculation, a maximum of $8192$ path candidates are tried. Run-time limits are not enforced, since execution times are limited appropriately by the previously described parameters.

## VI. RESULTS

The experiment showed that the ability to undo actions is very useful for the task at hand. Fig. 3 shows the results of the test-runs made in simulated environment (on the right) as well as on the real Scitos G5 robot (on the left). Each sub-figure contains the results of 5 test-runs made with each of the 3 different modes of the safety module.

Red triangles represent the results of the trials, when all the actions are allowed – the "no safety" mode. It is easy to see that this mode is the least safe of all the three – in the end the object is in average 1.5-1.9 metres away from its initial position. As expected, this mode performs very well in terms of area coverage – all the 80 cells are visited during both real and simulated test-runs. In the real environment the object is moved further away from its initial position than in the simulated environment. This is due to the difference in physical parameters of robot-object interaction.

The results of the "obstacle-avoidance" strategy, where only the actions that don't move the object are allowed, are shown as blue diamonds. This mode is in many ways the opposite of the "no safety": it is the safest mode with almost no object movements, but with incomplete area coverage – at least one and up to four cells remain unvisited. It is worth noting, that real environment exhibits small movements of the object in this mode. This is due to imperfect obstacle avoidance prediction and sensor noise.

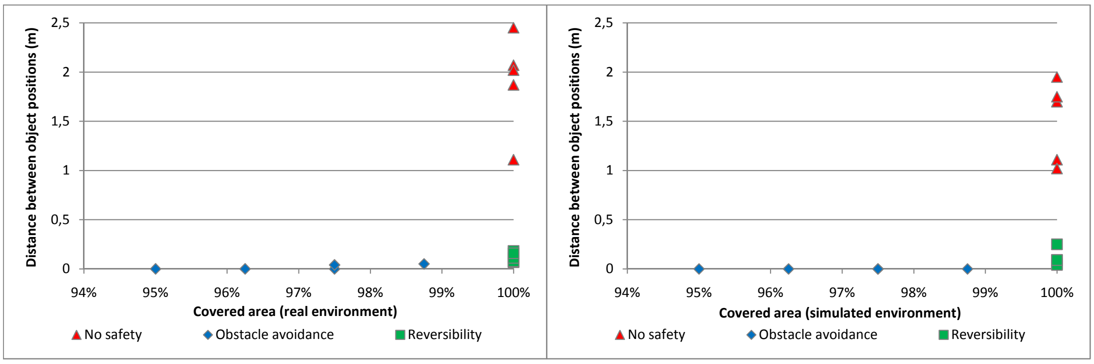

Fig. 3. Results of the test-runs in the real *(left)* and the simulated *(right)* environments.

Green squares represent the test-runs made in "reversibility" mode – when decision on the action and the post-action were generated by the reversibility sub-module. This mode performs on a par with the "no safety" mode in terms of area coverage – all the 80 cells of the area in question are covered. Also, this mode is safe – in the end the object is very close to its initial position after both real and simulated test-runs.

With respect to total navigation time, the fastest mode is predictably the "no safety", since the robot drives straight through the cells in the "lawnmower" fashion. The "reversibility" mode is only 3-10% slower than the "no safety" – additional time is spent only when driving around the obstacle and pushing it back. The slowest mode is the "obstacle-avoidance" – it is at least 50% slower than the "no safety" mode, because in this mode the next unvisited cell is tried many times from different positions, before it is marked as unvisitable and skipped.

## VII. CONCLUSIONS AND FUTURE WORK

### A. Conclusions

Based on the results of the experiment we conclude that the proposed safety architecture, consisting of the safety module and its reversibility sub-module can be used for practical purposes. The experiment showed that the principle of reversibility can be successfully used to ground safe object pushing behavior to increase covered area without compromising safety. The task of "robot area coverage with movable objects inside it" can be identified as one of the tasks, that benefit from application of such behavior.

Experiments showed that reversibility-based safety module exhibits better overall performance than the "no safety" and the "obstacle avoidance" pre-programmed approaches to cover the desired area. However, the experiment setups are somewhat idealized and further work must be done to make the approach applicable to the real-life scenarios.

### B. Future Work

Our ultimate goal is a multi-purpose personal robot-assistant with intrinsically safe autonomous decisions. We plan to use the same approach of using abstract principles to develop and ensure safe behaviors, but moving further towards more complex and real-life problems/scenarios.

## VIII. ACKNOWLEDGMENTS

This research was supported in part by scholarship from Estonian Information Technology Foundation.

## REFERENCES

[1] (2010, Sept.) IFR Statistical Department. [Online]. Available: http://worldrobotics.org/downloads/PR_2010-09-14_service_EN.pdf
[2] C. Urmson and others, "Autonomous driving in urban environments: Boss and the Urban Challenge," *J. of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
[3] M. Zinn, B. Roth, O. Khatib, and J. K. Salisbury, "A new actuation approach for human friendly robot design," *Int. J. of Robotics Research*, vol. 23, no. 4-5, pp. 379–398, 2004.
[4] A. D. Santis, B. Siciliano, A. D. Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253 – 270, 2008.
[5] S. Haddadin, A. Albu-Schaeffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *Int. J. of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.
[6] R. J. Sawyer, "Robot ethics," *Science*, vol. 318, no. 5853, p. 1037, 2007.
[7] R. C. Arkin, "Governing lethal behavior: Embedding ethics in a hybrid deiberative/reactive robot architecture," Georgia Tech, Tech. Rep. GIT-GVU-07-11, 2007.
[8] Y. Gavshin and M. Kruusmaa, "Comparative experiments on the emergence of safe behaviours," in *TAROS 2008*, pp. 65–71, 2008.
[9] R. Arkin and P. Ulam, "An ethical adaptor: Behavioral modification derived from moral emotions," in *CIRA 2009*, pp. 381–387, 2009.
[10] Y. Gavshin and M. Kruusmaa, "Assessing safety of object pushing using the principle of reversibility," in *HAIS 2011*, Part I, LNAI 6678, pp. 313–320, 2011.
[11] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *ICAR 2003*, pp. 317–323, 2003.
[12] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *ICRA 2005*, pp. 3930–3935, 2005.

# Lühikokkuvõte

Käesolev doktoritöö kirjeldab uudset turvaarhitektuuri, mis võimaldab traditsioonilisi ohutusprotseduure kombineerida tegevuste pööratavusel põhineva ohutuse hindamisega kui varulahendusega ootamatuteks olukordadeks. Selle arhitektuuri raames käsitletakse ohutuse hindamiseks kahte meetodit, mis kasutavad pööratavuse põhimõtet – “Ära tee asju, mida ei saa olematuks teha”.

Uurimuse ajendiks on robotite sulandumine meie igapäevaellu, mille tõttu on ohutus muutumas nende kasutamise üheks olulisemaks tahuks. Robotite autonoomia kasvab kiiresti ning on ainult aja küsimus, millal robotid hakkavad ise otsustama. Uurimustöö eesmärgiks on välja arendada raamistik, mis võimaldaks tuleviku robotitel suures hulgas erinevates olukordades ohutult käituda, kombineerides kavandatud ja iseeneslikke turvaprotseduure.

Robootikas vaadeldakse ohutust tavaliselt roboti keha ja robotkäte mehhaanilise ohutuse ning nende tavapärase, olukorrapõhiste käitumisreeglitega juhtimise taustal. Teisest küljest vaadeldakse ohutust eetika, seaduslike õiguste ning roboti projekteerija vastutuse kontekstis. Teadustöö autonoomsete robotite tegevuste uurimisel, põhinedes sisemistel motiividel ja abstraktsetel põhimõtetel, ei hooli tekkiva käitumise turvalisusest.

Käesolevas doktoritöös kirjeldatakse lähenemist, kus roboti projekteerija annab ohutusreeglid tuttavate olukordade jaoks ning pööratavusel põhinevat ohutuse hindamist rakendatakse tõeliselt iseseisvatele otsustele – otsustele juhtudel, kui ükski olukorrapõhine reegel pole kohaldatav. Sel viisil saab kirjeldatud turvaarhitektuuri siduda traditsioonilise roboti juhtsüsteemiga, olles tagavaralahenduseks ootamatutes olukordades. Säärastel juhtudel on ainult pööratavad tegevused iseeneslikult turvalised ning pöördumatuid tegevusi peab alla suruma, et sundida robotit ohutult käituma.

Käesolev väitekiri pakub välja kaks meetodit ohutuse hindamiseks kirjeldatud hübriidses turvaarhitektuuris. Maailmamudelivaba meetod kasutab olekuvaheliste kauguste mõõtusid, leidmaks sobivaid andmeid ning mõõtmaks tegevuse pööratavust. Maailmamudelil põhinev meetod arvutab tegevuse pööratavuse hindamiseks tegevuse tagasi võtmise hinna; robot simuleerib enda potentsiaalseid tegevusi, et arvutada algsesse olekusse naasmise hinda.

Simuleeritud keskkondades erinevate robotitega korraldatud katsed näitavad, et maailmamudelivaba pööratavuse hindamise meetodil juhitud robot käitub takistuste vältimisel turvaliselt. Erinevat tüüpi ja erinevate pikkustega tegevusi simuleerivad katsed näitavad, et maaimamudelivaba meetodit saab kasutada teineteist tühistavate tegevuste tuvastamiseks.

Simuleeritud ning reaalsetes liigutatava takistusega keskkondades korraldatud katsed näitavad, et maailmamudelil põhineva pööratavuse hindamisega hübriidne turvaarhitektuur võimaldab robotil suurendada katsepiirkonna kaetust, kasutades ennistatavat objektide manipuleerimist.

# Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor Professor Maarja Kruusmaa. Her encouragement, understanding, guidance, advice, criticism and knowledge have been of a great value for me and made this thesis possible.

I wish to thank all my colleagues from the Centre for Biorobotics for a very productive and enjoyable environment to work and make good research.

I express gratitude to Professor Bruno Siciliano for accepting to be my opponent and thoroughly reading this thesis.

Financial support from Estonian Information Technology Foundation and Tiger University programme (scholarship No. 09-03-99-03), Estonian Science Foundation and Archimedes Foundation is gratefully acknowledged.

I owe my loving thanks to my family. Without their encouragement it would have been impossible for me to finish this work. My dear wife Anna-Maria for her love, patience, understanding and bringing joy and sense to my life. My parents, brothers and sisters, for their loving support throughout my entire life.

# Curriculum Vitae

## Personal Data

| | |
|---|---|
| Name: | Juri Gavšin |
| Birth date and place: | 13.09.1981, Narva, Estonia |
| Citizenship: | Estonian |
| Marital status: | married |

## Contact Data

| | |
|---|---|
| Address: | Tuukri 58-17, 10120 Tallinn, Estonia |
| Phone: | +372 53405310 |
| E-mail: | yuri.gavshin@gmail.com |

## Education

| | | |
|---|---|---|
| 2007–2011 | Tallinn University of Technology | PhD studies |
| 2005–2007 | University of Tartu | MSc (computer science) |
| 1999–2005 | University of Tartu | BSc (computer science) |

## Language competence/skills

| | |
|---|---|
| English | fluent |
| Estonian | fluent |
| Russian | native |

## Professional employment

| | |
|---|---|
| 2011 - ... | Tallinn Univ. of Tech., Centre for Biorobotics; engineer |
| 2007 - ... | Karanar LLC; CEO, software developer |
| 2003 - 2007 | Codewiser LLC; senior software developer |
| 2003 - 2003 | Optimist LLC; software developer |

## Defended theses

| | |
|---|---|
| MSc thesis (2007): | “Using the concept of reversibility to develop safe behaviours in robotics”, (sup) Ahto Buldas, Maarja Kruusmaa |
| BSc thesis (2005): | “Internet structure visualisation”, (sup) Ahto Buldas |

## Research Interests

Robot learning, autonomous decision making, safety in robotics, cognitive robotics.

# Elulookirjeldus

## Isikuandmed

| | |
|---|---|
| Nimi: | Juri Gavšin |
| Sünniaeg ja -koht: | 13.09.1981, Narva, Eesti |
| Kodakondsus: | Eesti |
| Perekonnaseis: | abielus |

## Kontaktandmed

| | |
|---|---|
| Aadress: | Tuukri 58-17, 10120 Tallinn, Eesti |
| Telefon: | +372 53405310 |
| E-posti aadress: | yuri.gavshin@gmail.com |

## Hariduskäik

| | | |
|---|---|---|
| 2007–2011 | Tallinna Tehnikaülikool | doktoriõpe |
| 2005–2007 | Tartu Ülikool | teadusmagister (arvutiteadus) |
| 1999–2005 | Tartu Ülikool | teadusbakalaureus (arvutiteadus) |

## Keelteoskus

| | |
|---|---|
| English | kõrgtase |
| Estonian | kõrgtase |
| Russian | emakeel |

## Teenistuskäik

| | |
|---|---|
| 2011 - ... | Tallinna Tehnikaülikool, Biorobootika Keskus; insener |
| 2007 - ... | Karanar OÜ; tegevjuht, tarkvara arendaja |
| 2003 - 2007 | Codewiser OÜ; vanem tarkvara arendaja |
| 2003 - 2003 | Optimist OÜ; tarkvara arendaja |

## Kaitstud lõputööd

| | |
|---|---|
| Magistritöö (2007): | "Pööratavuse kontseptsiooni kasutamine robootikas ohutu käitumise õppimiseks", (juh) Ahto Buldas, Maarja Kruusmaa |
| Bakalaureusetöö (2005): | "Interneti struktuuri visualiseerimine", (juh) Ahto Buldas |

## Teadustöö Põhisuunad

Robotõppimine, autonoomsete otsuste tegemine, ohutus robootikas, kognitiivne robootika.

## DISSERTATIONS DEFENDED AT TALLINN UNIVERSITY OF TECHNOLOGY ON *INFORMATICS AND SYSTEM ENGINEERING*

1. **Lea Elmik**. Informational Modelling of a Communication Office. 1992.

2. **Kalle Tammemäe**. Control Intensive Digital System Synthesis. 1997.

3. **Eerik Lossmann**. Complex Signal Classification Algorithms, Based on the Third-Order Statistical Models. 1999.

4. **Kaido Kikkas**. Using the Internet in Rehabilitation of People with Mobility Impairments – Case Studies and Views from Estonia. 1999.

5. **Nazmun Nahar**. Global Electronic Commerce Process: Business-to-Business. 1999.

6. **Jevgeni Riipulk**. Microwave Radiometry for Medical Applications. 2000.

7. **Alar Kuusik**. Compact Smart Home Systems: Design and Verification of Cost Effective Hardware Solutions. 2001.

8. **Jaan Raik**. Hierarchical Test Generation for Digital Circuits Represented by Decision Diagrams. 2001.

9. **Andri Riid**. Transparent Fuzzy Systems: Model and Control. 2002.

10. **Marina Brik**. Investigation and Development of Test Generation Methods for Control Part of Digital Systems. 2002.

11. **Raul Land**. Synchronous Approximation and Processing of Sampled Data Signals. 2002.

12. **Ants Ronk**. An Extended Block-Adaptive Fourier Analyser for Analysis and Reproduction of Periodic Components of Band-Limited Discrete-Time Signals. 2002.

13. **Toivo Paavle**. System Level Modeling of the Phase Locked Loops: Behavioral Analysis and Parameterization. 2003.

14. **Irina Astrova**. On Integration of Object-Oriented Applications with Relational Databases. 2003.

15. **Kuldar Taveter**. A Multi-Perspective Methodology for Agent-Oriented Business Modelling and Simulation. 2004.

16. **Taivo Kangilaski**. Eesti Energia käiduhaldussüsteem. 2004.

17. **Artur Jutman**. Selected Issues of Modeling, Verification and Testing of Digital Systems. 2004.

18. **Ander Tenno**. Simulation and Estimation of Electro-Chemical Processes in Maintenance-Free Batteries with Fixed Electrolyte. 2004.

19. **Oleg Korolkov**. Formation of Diffusion Welded Al Contacts to Semiconductor Silicon. 2004.

20. **Risto Vaarandi**. Tools and Techniques for Event Log Analysis. 2005.

21. **Marko Koort**. Transmitter Power Control in Wireless Communication Systems. 2005.

22. **Raul Savimaa**. Modelling Emergent Behaviour of Organizations. Time-Aware, UML and Agent Based Approach. 2005.

23. **Raido Kurel**. Investigation of Electrical Characteristics of SiC Based Complementary JBS Structures. 2005.

24. **Rainer Taniloo**. Ökonoomsete negatiivse diferentsiaaltakistusega astmete ja elementide disainimine ja optimeerimine. 2005.

25. **Pauli Lallo.** Adaptive Secure Data Transmission Method for OSI Level I. 2005.

26. **Deniss Kumlander**. Some Practical Algorithms to Solve the Maximum Clique Problem. 2005.

27. **Tarmo Veskioja**. Stable Marriage Problem and College Admission. 2005.

28. **Elena Fomina**. Low Power Finite State Machine Synthesis. 2005.

29. **Eero Ivask**. Digital Test in WEB-Based Environment 2006.

30. **Виктор Войтович**. Разработка технологий выращивания из жидкой фазы эпитаксиальных структур арсенида галлия с высоковольтным p-n переходом и изготовления диодов на их основе. 2006.

31. **Tanel Alumäe**. Methods for Estonian Large Vocabulary Speech Recognition. 2006.

32. **Erki Eessaar**. Relational and Object-Relational Database Management Systems as Platforms for Managing Softwareengineering Artefacts. 2006.

33. **Rauno Gordon**. Modelling of Cardiac Dynamics and Intracardiac Bio-impedance. 2007.

34. **Madis Listak**. A Task-Oriented Design of a Biologically Inspired Underwater Robot. 2007.

35. **Elmet Orasson**. Hybrid Built-in Self-Test. Methods and Tools for Analysis and Optimization of BIST. 2007.

36. **Eduard Petlenkov**. Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach. 2007.

37. **Toomas Kirt**. Concept Formation in Exploratory Data Analysis: Case Studies of Linguistic and Banking Data. 2007.

38. **Juhan-Peep Ernits**. Two State Space Reduction Techniques for Explicit State Model Checking. 2007.

39. **Innar Liiv**. Pattern Discovery Using Seriation and Matrix Reordering: A Unified View, Extensions and an Application to Inventory Management. 2008.

40. **Andrei Pokatilov**. Development of National Standard for Voltage Unit Based on Solid-State References. 2008.

41. **Karin Lindroos**. Mapping Social Structures by Formal Non-Linear Information Processing Methods: Case Studies of Estonian Islands Environments. 2008.

42. **Maksim Jenihhin**. Simulation-Based Hardware Verification with High-Level Decision Diagrams. 2008.

43. **Ando Saabas**. Logics for Low-Level Code and Proof-Preserving Program Transformations. 2008.

44. **Ilja Tšahhirov**. Security Protocols Analysis in the Computational Model – Dependency Flow Graphs-Based Approach. 2008.

45. **Toomas Ruuben**. Wideband Digital Beamforming in Sonar Systems. 2009.

46. **Sergei Devadze**. Fault Simulation of Digital Systems. 2009.

47. **Andrei Krivošei**. Model Based Method for Adaptive Decomposition of the Thoracic Bio-Impedance Variations into Cardiac and Respiratory Components. 2009.

48. **Vineeth Govind**. DfT-Based External test and Diagnosis of Mesh-like Networks on Chips. 2009.

49. **Andres Kull**. Model-Based Testing of Reactive Systems. 2009.

50. **Ants Torim**. Formal Concepts in the Theory of Monotone Systems. 2009.

51. **Erika Matsak**. Discovering Logical Constructs from Estonian Children Language. 2009.

52. **Paul Annus**. Multichannel Bioimpedance Spectroscopy: Instrumentation Methods and Design Principles. 2009.

53. **Maris Tõnso**. Computer Algebra Tools for Modelling, Analysis and Synthesis for Nonlinear Control Systems. 2010.

54. **Aivo Jürgenson**. Efficient Semantics of Parallel and Serial Models of Attack Trees. 2010.

55. **Erkki Joasoon**. The Tactile Feedback Device for Multi-Touch User Interfaces. 2010.

56. **Jürgo-Sören Preden**. Enhancing Situation – Awareness Cognition and Reasoning of Ad-Hoc Network Agents. 2010.

57. **Pavel Grigorenko**. Higher-Order Attribute Semantics of Flat Languages. 2010.

58. **Anna Rannaste**. Hierarcical Test Pattern Generation and Untestability Identification Techniques for Synchronous Sequential Circuits. 2010.

59. **Sergei Strik**. Battery Charging and Full-Featured Battery Charger Integrated Circuit for Portable Applications. 2011.

60. **Rain Ottis**. A Systematic Approach to Offensive Volunteer Cyber Militia. 2011.

61. **Natalja Sleptšuk**. Investigation of the Intermediate Layer in the Metal-Silicon Carbide Contact Obtained by Diffusion Welding. 2011.

62. **Martin Jaanus**. The Interactive Learning Environment for Mobile Laboratories. 2011.

63. **Argo Kasemaa**. Analog Front End Components for Bio-Impedance Measurement: Current Source Design and Implementation. 2011.

64. **Kenneth Geers**. Strategic Cyber Security: Evaluating Nation-State Cyber Attack Mitigation Strategies. 2011.

65. **Riina Maigre**. Composition of Web Services on Large Service Models. 2011.

66. **Helena Kruus**. Optimization of Built-in Self-Test in Digital Systems. 2011.

67. **Gunnar Piho**. Archetypes Based Techniques for Development of Domains, Requirements and Sofware. 2011.