

Embodied Imitation and Behavioural Adaptation in Robot Collectives

MEHMET DINCER ERBAS

A dissertation submitted in partial fulfilment of the
requirements of the University of the West of England, Bristol
for the degree of Doctor of Philosophy

Faculty of Environment and Technology, University of the West of England, Bristol
April 2012

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Alan FT Winfield, for his continuous support and guide during my study. I have learned and benefited a lot from the discussions we had during our supervision meetings. I also would like to thank my second supervisor, Prof. Larry Bull, for helping me whenever I asked for.

I would like to thank the director of Bristol Robotics Laboratory, Prof. Chris Melhuish, for providing a friendly and well-ordered research environment. I also would like to thank my colleagues in the Swarm Intelligent Systems group, Dr. Wenguo Liu, Dr. Jan Dyre Bjorknes and Paul O'Dowd, for all their help. I am also grateful to the technicians of Bristol Robotics Laboratory for their efforts to build the infrastructure of my experiments.

The work presented in this thesis was financially supported by EPSRC (EP/E062083/1) - The Emergence of Artificial Culture in Robot Societies project. I would like to thank the partners and advisors to this project, Prof. Alistair Sutcliffe, Prof. Frances Griffiths, Prof. Jim Bown, Dr. Jenny Tennant Jackson, Dr. Robin Durie and Prof. Susan Blackmore, for their valuable discussions and suggestions during project meetings.

Finally, I would like to thank my parents, Selma and Adil Erbas, for their continuous love and support.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Abstract

Social learning, which enables the individuals to learn from others in a community, is an important mechanism for social animals. One of the most important types of social learning is imitation as it allows certain skills and behaviours to be transferred between individuals without language or other complex symbolic communication. This thesis investigates embodied imitation and behavioural adaptation in collective robotics. Firstly, by using robots to model social learning by imitation in a collective, we examine the adaptation of behaviours and their structure. Secondly, the link between imitation and individual learning is investigated and a method that uses imitation as an enhancing factor to increase the learning speed of artificial agents is developed and tested.

In this thesis, real robots are used to model imitation and a robot to robot movement imitation algorithm is implemented. The algorithm is embodied in the sense that robots use their on-board sensors to perceive the movement of other robots. Following this, a quality of imitation function is designed to quantitatively assess the fidelity of imitation. It is shown that although we have a homogeneous group of robots, variations that are caused by embodiment arise during imitation process. In experiments not previously undertaken in a group of real robots, it is shown that these variations allow certain behaviours, that appear to be more robust to uncertainties in robot's sensors, to emerge during multiple cycles of imitation. Therefore, as the robots share a similar perceptual context, the structure of the imitated behaviours adapts to the uncertainties in robot's sensors so that these emergent behaviours can be copied with high fidelity. This process is proposed as a possible abstract yet embodied model of cultural evolution that explains the emergence and propagation of shared behaviours by imitation in a group of social agents.

This thesis also presents an algorithm that combines individual learning and imitation. The algorithm uses imitation as an enhancing factor that increases the learning speed of agents that utilise a well known reinforcement learning method. The algorithm is novel as it attempts to use imitation of purely observed behaviours to enhance learning. As in nature, imitation provides

agents with model behaviours that can lead their individual learning process. Finding in what state or context these model behaviours are useful is determined by the learning process of the imitating agent. The algorithm is validated both in a simulation, and on real robots that use embodied imitation and learning speed is shown to increase in the robot collective.

Contents

1	Introduction	1
1.1	Main Contributions	3
1.2	Organisation of the Thesis and Relevant Publications	4
2	Background and Related Work	7
2.1	Introduction	7
2.2	Imitation	7
2.2.1	Definition of Imitation	8
2.3	Robot Imitation	10
2.3.1	Utility of Imitation in Robotics	10
2.3.2	Imitation Mechanisms in Robotics Research	12
2.3.3	Programming by Demonstration	14
2.3.4	Robot to Robot Imitation	15
2.4	Collective Robotics	18
2.4.1	Swarm Robotics	19
2.5	Reinforcement Learning	22
2.5.1	Reinforcement Learning in Robotics	24
2.6	Summary and Conclusion	26
3	Imitation on Real Robots	27
3.1	Introduction	27
3.2	Hardware Setup	28
3.3	Imitation Algorithm	32
3.4	Quality of Imitation	38
3.5	Summary and Conclusion	46

4	Emergence of Structure in Behaviours Adapted Through Embodied Imitation	48
4.1	Introduction	48
4.2	Experimental Setup	49
4.3	Experiments	53
4.3.1	Imitation with No Memory	54
4.3.2	Imitation with Unlimited Memory	55
4.3.3	Imitation with Limited Memory	61
4.4	Summary and Conclusion	65
5	Imitation-enhanced Reinforcement Learning	67
5.1	Introduction	67
5.2	Simulation Setup	69
5.3	Imitation-Enhanced Reinforcement Learning	70
5.4	Experiments	80
5.4.1	Imitating Predefined Paths	80
5.4.2	Agents Imitating Each Other	83
5.5	Summary and Conclusion	93
6	Imitation-Enhanced Q-learning in Continuous Space	95
6.1	Introduction	95
6.2	Experimental Setup	97
6.3	Experiments	99
6.3.1	Learner Robot Copying an Expert Robot	99
6.3.2	Two Robots Copying Each Other	101
6.3.3	On the Effects of Embodied Imitation	103
6.3.4	Summary and Conclusion	104
7	Conclusion	106
7.1	Summary of Contributions	108
7.2	Future Work	109
	Bibliography	111
A	Imitation on Real Robots	122

List of Figures

1.1	Adaptive processes of living organisms can be categorized into two main levels: Population level (evolution) and individual level (learning). Individual level adaptive processes can further divided into two subcategories: Individual learning and social learning. Adapted from (Eiben et al., 2007).	2
3.1	An e-puck with Linux board fitted in between the e-puck motherboard (lower) and the e-puck speaker board(upper). Note both the red skirt, and the yellow hat that provides a matrix of pins for the reflective spheres which allow the tracking system to identify and track each robot.	30
3.2	Robot arena with 6 robots.	31
3.3	Infrastructure of the experimental setup. Each robot is identified with a static IP. Robots are fitted with tracking hats which provide a matrix of pins for the reflective spheres that allow the tracking system to identify and track each robot. The dedicated swarm lab server works both as a logging pool for the experiments and as a router to bridge the robots and the local network. Diagram from (Liu and Winfield, 2011)	32
3.4	An image from the image sensor of the robots. In the figure, the blobfinder tool of Player has fitted a blob around the red skirt of the robot on the left.	33

- 3.5 Frame processing phase: Vertical distance (d_{robot}^y) and horizontal distance (d_{robot}^x) from the observing robot are calculated. (d_{robot}^x, d_{robot}^y) for each frame is stored in a linked list of positions. (d_{robot}^x, d_{robot}^y) is calculated by: *Previously calculated metrics: Size of the blob when $d_{robot}^{total} = 20\text{ cm} \rightarrow s_{robot} = 106\text{ pixels}$ Actual size of the blob $\rightarrow 3\text{ cm}$ At any distance: $d_{robot}^{total} = \frac{106 \times 20}{s_{robot}}$, $d_{robot}^x = \frac{3 \times p_{robot}^x}{s_{robot}}$, $d_{robot}^y = \sqrt{(d_{robot}^{total})^2 - (d_{robot}^x)^2}$ where: $d_{robot}^{total} = \text{Total distance between two robots, in cm}$ $d_{robot}^x = x\text{ component of the distance between two robots, in cm}$ $d_{robot}^y = y\text{ component of the distance between two robots, in cm}$ $p_{robot}^x = x\text{ component of the distance between two robots on the image frame, in pixels}$ $s_{robot} = \text{size of the blob covering observed robot on the image frame, in pixels}$ * 34
- 3.6 Data processing phase: Given a linked list of positions as input, the first three steps of data processing phase are detecting turns, line-fitting to determine straight line segments between turns and reconstructing (estimated) observed trajectory. These three data transformations are shown as light grey arrows. The input to detect turns and line fit are shown numerically with example values where each data pair is a relative position of the demonstrator robot to the imitating robot; the outputs for line fit and reconstruct trajectory are shown as vectors. 35
- 3.7 Regression line fitting: p_3 and p_4 are located on the same location of the arena which means that the robot was turning when these positions were recorded. The next such occasion is at p_8 and p_9 . So the robot was moving on a straight line between these two turns. Line L is the regression line fitted by using points p_4, p_5, p_6, p_7 and p_8 37
- 3.8 Plot of the trajectory of robots during an imitation run. The teacher robot moved in an equilateral triangular trajectory which was then copied by the imitator robot. 40
- 3.9 Movement patterns that the demonstrator robot followed during experiments. First movement pattern is a low-complexity pattern which consists of two moves of length 20 cm and a 180° turn between them. The second pattern is a medium-complexity pattern which consists of an equilateral triangular trajectory. The third is a high-complexity pattern which consists of a complex trajectory with 4 turns. 40

- 3.10 Mean quality value calculated at different distances between robots. The demonstrator robot has a low-complexity movement pattern which consists of a line. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 42
- 3.11 Mean quality value calculated at different distances between robots. The demonstrator robot has a medium-complexity movement pattern which consists of an equilateral triangle. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 42
- 3.12 Mean quality value calculated at different distances between robots. The demonstrator robot has a high-complexity movement pattern which consists of complex trajectory. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 43
- 3.13 Mean quality value calculated for different movement patterns when the distance between robots is 0.5 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 44
- 3.14 Mean quality value calculated for different movement patterns when the distance between robots is 1 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 44
- 3.15 Mean quality value calculated for different movement patterns when the distance between robots is 1.5 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$ 45
- 4.1 Each experiment presented in this chapter is performed in a 3 m by 3 m arena with 4 robots, placed 1 m apart and arranged as shown here. 51

4.2	Finite state machine of the controller of the robots. Once started the FSM loops indefinitely, alternating between demonstrator and observer modes. The two phases of the imitation algorithm take place in corresponding states as shown in the FSM.	51
4.3	The demonstrator robot randomly selects one of the trajectories in its memory and then the selected trajectory is converted into a set of motor commands that can be executed by the demonstrator robot (<i>enact</i>).	53
4.4	Plot of trajectories of each robot during the experiment with no memory.	55
4.5	Pattern progress map for a 4 robot experiment with no memory. Each node in the figure represents the demonstration of a movement pattern. If a pattern is demonstrated and imitated, the new copy of that pattern is linked to it by an arrow. For instance, pattern 2, the original square, was demonstrated by Robot A and was copied by two robots. The new (child) copies of pattern 2 are patterns 3 and 4. If the copy is of high quality (i.e. $Q_i \geq 0.85$), then the node has a red colour. Initial movement patterns are a triangle (1) and a square (2). The nodes are numbered according to the time that they were copied: if $x < y$, pattern x was copied before pattern y	56
4.6	Pattern progress map for a 4 robot experiment with unlimited memory. Initial movement patterns are a triangle (1) and a square (2).	58
4.9	Mean quality value calculated for all imitations in the experiment shown in Figure 4.6 (All copies) and mean quality value for the cluster formed by pattern 27 (Cluster). The quality of each imitation is shown by a circle and the horizontal line shows the mean.	58
4.7	Adaptation of pattern 27 in Figure 4.6. Pattern 27 is a descendant of the original equilateral triangle pattern. By following the imitation links on the pattern progress map for this experiment, we can see that there are 5 intermediate copies between the original triangle and pattern 27: the patterns numbered 5, 11, 18, 20, 26. All of these patterns, starting with the original triangle and ending with pattern 27, are shown here in order. All axes are marked in cm. Beginning of each movement pattern is marked with a circle.	59
4.8	The descendants of pattern 27 in Figure 4.6. Starting with pattern 27, its descendants (patterns 27, 36, 37, 46, 49, 50, 51, 55) are shown in order. All axes are marked in cm.	60

4.10	Pattern progress map for a 4-robot experiment with limited memory. Initial movement patterns are a triangle (1) and a square (2). The 20 patterns in the memory of all 4 robots at the end of the experiment are highlighted as diamonds.	62
4.11	Adaptation of pattern 12 in Figure 4.10. There is an intermediate copy (10) between the original triangle and pattern 12. All axes are marked in cm.	63
4.12	The descendants of pattern 12 in Figure 4.10. Starting with pattern 12, some of its high-quality copy descendants (patterns 12, 21, 22, 33, 34, 38, 43, 52, 53) are shown in order. All axes are marked in cm.	63
4.13	Mean quality value calculated for all imitations in the experiment shown in figure 4.10 (All copies) and mean quality value for the cluster formed by pattern 12 (Cluster). The quality of each imitation is shown by a circle and the horizontal line shows the mean.	64
5.1	Simulated arena. The arena is a 10 by 10 grid world. The agent is placed in the top-left corner and the goal item is placed in the bottom right corner.	70
5.2	Path given to the agent. It consists of 5 consecutive moves South East. Since by following this path, the agent gets close to the goal item, this path is beneficial for the agent. These moves are shown at relative positions.	72
5.3	Performance of the agent when a beneficial path is given, fixed $p_{imitate}$ value. Time is given in time units.	73
5.4	Path given to the agent. It consists of 5 consecutive moves East. Since by following this path, the agent does not get close to the goal item, this path is not beneficial for the agent. These moves are shown at relative positions.	73
5.5	Performance of the agent when a non-beneficial path is given, fixed $p_{imitate}$ value.	74
5.6	Performance of the agent when a beneficial path is given, path completion test.	75
5.7	Performance of the agent when a non-beneficial path is given, path completion test.	75
5.8	Performance of the agent when a beneficial path is given, average Q value test.	77
5.9	Performance of the agent when a non-beneficial path is given, average Q value test.	77
5.10	Arena with obstacle: The arena is a 100 by 100 grid world. The agents are placed in the top-left corner and the goal item is placed in the bottom-right corner. There is an obstacle in the middle of the arena.	80

5.11	8 Paths given to the agent. Each list of actions consist of 20 consecutive moves in one direction: E, SE, S, SW, W, NW, N and NE	81
5.12	Best path length for imitating (8 paths given) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals (note these are shown only at 10 point intervals). The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.	82
5.13	An example shortest path to the goal state. Any shortest path consists of a combination of 80 moves to SE, 20 moves to S and 20 moves to E directions.	83
5.14	The R_i values for each path given to the agent.	83
5.15	Best path length for imitating (copying an expert) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.	84
5.16	Best path length for imitating (two agents copying each other) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.	85
5.17	Best path length for imitating (copying an experienced agent) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.	86

5.18	Best path length for 4 scenarios. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.	87
5.19	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	88
5.20	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	88
5.21	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	89
5.22	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	89
5.23	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	90
5.24	An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.	90

5.25	Best path length for an internal-state-access (accessing an expert agent's Q values) agent and no-imitation/no-internal-state-access agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.	92
5.26	Best path length for an internal-state-access (two agents using each other's Q values) agent and no-imitation/no-internal-state-access agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values.	92
5.27	Best path length for an agent accessing another agent's Q values and imitating (two agents copying each other) agent. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.	93
6.1	The robot arena is divided into multiple compartments and a robot resides in each of them. The figure on the left shows two robots, robot A and robot B, in their own compartments, performing their individual learning. In the figure on the right, robot B is watching and copying actions performed by robot A.	97
6.2	Two robots performing individual learning in their separate compartments.	98
6.3	Best path length for Q-learning with imitation (copying an expert) and Q-learning with no-imitation robots. The results are mean best path length from 4 experiment runs along with 95% confidence intervals. The best path for each robot is calculated every 10 state transitions by using a greedy action selection algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments.	101

6.4	Best path length for Q-learning with imitation (two robots imitating each other) and Q-learning with no-imitation robots. The results are mean best length from 4 experiment runs along with 95% confidence intervals. The best path for each robot is calculated every 10 state transitions by using a greedy action selection algorithm in the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments.	102
6.5	Two paths that are copied and enacted highest number of times during an experimental run.	103
6.6	Two paths that are copied but enacted lowest number of times during an experimental run.	103
6.7	Two example movement patterns that were copied and then enacted the largest number of times. The shape on the left shows the original movement pattern of the observed robot and the figure on the right shows its perceived copy by the imitating robot.	104
A.1	A snapshot from the memory of a robot. Each line records the trajectory of a demonstrator robot, according to the calculations of the imitating robot. The structure of each line is: New angle / Length of straight move / New angle / Length of straight move (repeats). For instance the first line records a equilateral triangular trajectory: It starts with a 15 cm forward move, then a 120° counter clockwise turn, a 15 cm straight move, a 120° counter clockwise turn and a 15 cm straight move. Similarly the second line records a trajectory which consists of a 15 cm straight move, a 110° counter clockwise turn, a 15 cm straight move, a 121° counter clockwise turn and a 19 cm straight move.	122
A.2	A snapshot from the internal log file of a robot. It records state transitions and internal calculations with their timestamps.	123
A.3	A snapshot from the external log of an experiment. The structure of each line is: Robot ID, timestamp (in sec), X coordinate on the arena (in 0.1 mm), Y coordinate on the arena (in 0.1 mm), orientation of the robot (orientation in radians * 1000), robot state. The position and orientation of the robot is received from the tracking system. Internal state of the robots are sent by the robots.	123

List of Tables

3.1	Sensors of an e-puck robot	29
4.1	Results for 10 experimental runs with unlimited memory.	61
4.2	Results for 10 experimental runs with limited memory	65
A.1	The minimum length of distance change (R) that can be detected depending on the distance between two robots (D). For instance when the distance between two robots is 20 cm, one pixel change in size of the observed robot in the view of observing robot means the distance between two robots is altered by 0.1981 cm. .	124

Chapter 1

Introduction

All organisms have to adapt themselves to their environment and to changing conditions in order to survive. Life involves change at many time scales. Starting from their existence, living organisms have to react to those changes by a number of adaptive processes. There are two main levels of adaptive processes (figure 1.1): One of them is *evolution* that operates on the level of populations or species. Adaptation, in the evolutionary sense, is the process in which a population becomes better suited to its habitat (Darwin, 1859). Evolution takes place on a very long time period during which many generations pass by. With the increase in computational power and tools in recent years, there has been extensive research in robotics that is inspired by the evolution in nature (Nolfi and Floreano, 2000). Evolutionary algorithms such as *genetic algorithms*, *learning classifier systems* and *particle swarm optimization* have been applied to a number of problems in robotics. They use mechanisms inspired by biological evolution including *natural selection*, *adaptation* and *mutation*.

Secondly, we have individual level of adaptive processes which include lifetime learning of individuals. Compared to evolution, adaptation by learning takes place in a much shorter time period, limited to individual lifetimes. As Ackley and Littman stated (Ackley and Littman, 1991): “*An entire lifetime of learning is but one tick of the clock for evolution.*”. Learning activities that take place during the lifetime of individuals can be divided into two categories: *individual learning*, in which learning process purely depends on the individual activities, and *social learning*, in which individuals learn from others. In nature, these different adaptive mechanisms work in coordination. All living organisms learn how to use their motor system efficiently, which is inherited from their parents, by interacting with their environment. Similarly, social learning supplies individuals with successful model behaviours that can enhance the learning speed and success rate of

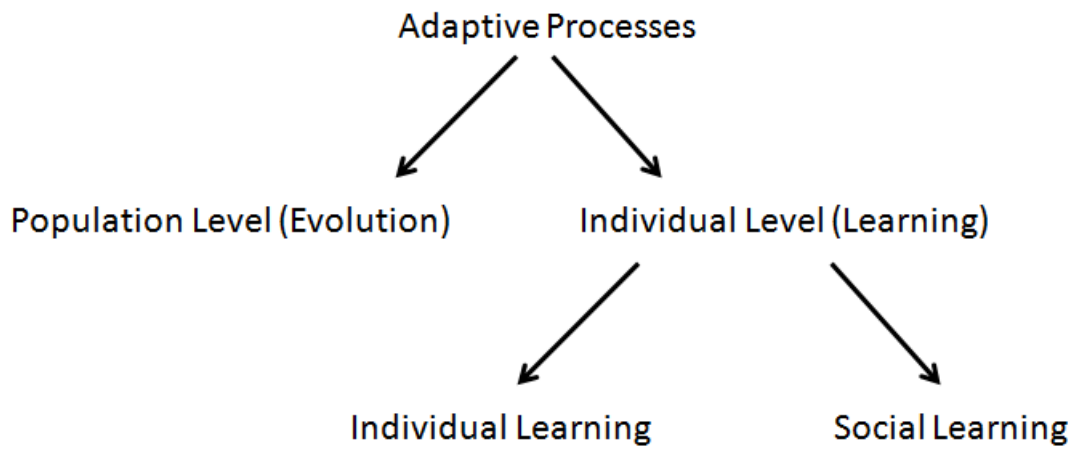


Figure 1.1: Adaptive processes of living organisms can be categorized into two main levels: Population level (evolution) and individual level (learning). Individual level adaptive processes can further divided into two subcategories: Individual learning and social learning. Adapted from (Eiben et al., 2007).

their individual learning activities. One of the most important types of social learning is *imitation* as it allows skills and behaviours to be transferred between individuals without language or other complex symbolic communication.

This thesis investigates embodied imitation and behavioural adaptation in collective robotics. Real robots are used for modelling imitation between artificial agents. The robots are able to observe and replicate each other's movement patterns, by using their on-board sensors; so the imitation is *embodied*. It is shown that variations that arise from embodiment allow certain behaviours, that appear to be more robust to the process of imitation, to emerge during multiple cycles of imitation. Therefore, as the robots share a similar perceptual context, the structure of the imitated behaviours changes so that these behaviours can be copied with high fidelity. As these adapted behaviours appear to be more robust to imitation, they can be transferred to other members of the group with higher fidelity. This process is proposed as a model that explains the emergence and propagation of shared skills and behaviours in a group of social agents. Furthermore, the connection between imitation and individual learning is examined. For this purpose, a method for using imitation as an enhancing factor that increases the individual learning speed of robots is developed. The method attempts to determine if the information gained by imitation is beneficial or not by checking its effects on the individual learning process.

Two hypotheses are proposed that stimulate this research:

Hypothesis 1 That is possible, in a group of robots, that are able to imitate each other, to observe both adaptation and emerging structure in the imitated behaviours, as they undergo multiple cycles of imitation (Chapter 4).

Hypothesis 2 That is possible, in a group of robots that are able to imitate each other, to use the information gained by imitation for speeding up the individual learning of robots (Chapter 5 and Chapter 6).

1.1 Main Contributions

The work presented in this thesis includes the following major and minor (indented) contributions:

- A robot to robot movement imitation algorithm is implemented on real robots. The algorithm completely depends on the visual data coming from the image sensor of the robots. Therefore the imitation is embodied, in the sense that the imitating robot uses only its on-board sensors to perceive the movement pattern of the demonstrator robot.
 - In order to quantitatively assess the fidelity of imitation, a quality of imitation function is defined. The success rate of the algorithm is determined in a scenario in which an observer robot copies the movement trajectory of a demonstrator robot. The effects of experiment settings, such as the distance between robots and the complexity of the movement patterns, are examined.
- An experimental set-up with robots is designed in order to model imitation between artificial agents that are able to copy each other's movement patterns. It is shown that variations that arise from embodiment allow certain behaviours, that are more robust to the process of imitation, to emerge during multiple cycles of imitation.
- The effect of memory capability on the adaptation of shared behavioural patterns during multiple cycles of imitation is investigated. It is shown that the agents should be able to store and retrieve their past observations to allow shared behaviours to emerge and spread in the group.

- A method is presented for using the information gained by imitation as an enhancing factor that increases the individual learning speed of agents. The method checks temporal changes in the individual learning process in order to determine if the information gained by imitation is beneficial or not. It is shown that the imitation of purely observed behaviours enhances learning.
 - The importance of having experienced individuals in a group of agents that can utilise social learning is investigated. It is shown that the learned skills of successful individuals can lead other members of the group by imitation.
 - The method is extended to continuous-time settings and it is tested on real robots that use embodied imitation. The effect of variance, that is introduced by embodiment, on the learning speed of robots, is examined.

This thesis mainly contributes to the fields of Collective Robotics and Multi-agent Systems. The embodied imitation algorithm that is developed on real robots contributes to the robot to robot imitation research. The imitation-enhanced reinforcement learning algorithm is a novel method that contributes to the field of Reinforcement Learning as it uses the imitation of purely observed behaviours to enhance learning.

1.2 Organisation of the Thesis and Relevant Publications

The rest of the thesis is organised in 6 chapters. Chapter 2 provides an overview of the topics related to this thesis. It starts with an overview of imitation and its functionality in social animal societies. The chapter then reviews the usage of imitation in robotics research. The expected utility of imitation and proposed mechanisms are presented, followed by examples of research on imitation in robotics. The chapter also includes a brief introduction of collective robotics. Some well-known problems that are widely examined by swarm robotics are listed. Finally, the last section briefly presents reinforcement learning and introduces some researches that used reinforcement learning in the robotics domain.

Chapter 3 presents the hardware and the infrastructure that are used in this thesis. Following this, a robot to robot movement imitation algorithm is presented and tested on real robots. The algorithm completely depends on the visual data coming from the image sensor of the robots as

no other type of communication is allowed between robots. Finally, a quality of imitation function is defined in order to assess the fidelity of imitation.

In chapter 4, real robots that are able to copy each other's movement patterns are used to model imitation between artificial agents. It is shown that variations in the robots' sensors allow certain behaviours to emerge during multiple cycles of imitation. These adapted behaviours can be copied with high fidelity as they appear to be robust to the errors in the imitation process. Three cases of different sizes of robot memories are experimentally tested in order to examine the effect of memory capability of individuals on the emergence of shared behaviours in the robot collective. The following publication is based on the content of this chapter:

- Erbas, M. D. and Winfield, A. F. T. (2011). On the Emergence of Structure in Behaviours Evolved through Embodied Imitation in a Group of Robots. In *Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems (ECAL 2011)*. T. Lenaerts, M. Giacobini, H. Bersini, P. Bourguine, M. Dorigo and R. Doursat, (eds.). pages: 219-226. MIT Press. [Lead Author wrote the first draft].

A more detailed inspection on the emergence of shared behavioural traditions in robot groups is presented in the following publication:

- Winfield A. F. T. and Erbas M. D. (2011). On Embodied Memetic Evolution and the Emergence of Behavioural Traditions in Robots. *Memetic Computing*. 3(4), pages: 261-270. [The author undertook all experimental work and contributed as second author].

Chapter 5 presents imitation-enhanced Q learning which is an attempt to use social learning (imitation) to increase the individual learning speed (reinforcement learning) of artificial agents. The method checks temporal changes in its individual learning process to determine if the information gained by imitation is beneficial or not. It is shown that the imitation-enhanced Q-learning agent learns faster than the same Q-learner without imitation. The following publication is based on the content of this chapter:

- Erbas, M. D., Winfield, A. F. T. and Bull, L. (2011). Towards Imitation-Enhanced Reinforcement Learning in Multi-Agent Systems. In *Proceedings of 2011 IEEE Symposium on Artificial Life*. pages: 6-13, Paris, France. [Lead author wrote the first draft].

In chapter 6, imitation-enhanced Q learning is extended to continuous time and space. The algorithm is tested on real robots that use embodied imitation. The effect of the variance, that arises from embodiment, is examined.

Chapter 7 concludes the thesis with a summary of contributions and a discussion about possible future extensions.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter provides an overview of the topics that are related to this thesis. It starts with an overview of imitation and its functionality in social animal societies. There is a discussion about several definitions of imitation and their meanings. Section 2.3 explains the usage of imitation in robotic research. It includes a discussion about the possible utilities of using imitation for robotic experiments, a list of proposed mechanisms to use imitation, and some example robotic researches that used imitation. Section 2.4 starts with a brief introduction to collective robotics. Some well-known problems that are examined by swarm robotics are listed. Section 2.5 presents reinforcement learning and introduces some research that used reinforcement learning methods in the robotics domain.

2.2 Imitation

Social learning, which enables individuals to learn from others, is a significant mechanism for social animals, including humans. Imitation, in which an individual observes and replicates another group member's actions, has been widely examined by both biologists and psychologists and is an important aspect of social learning. Biological research on imitation is largely focused on the adaptive value of imitation for the organism, while psychologists are mainly interested in the mechanisms and functions of imitation (Zentall, 2001). There is continuing debate on the definition of imitation and whether it is unique to humans: other species that have been reported to imitate include chimpanzees, monkeys, birds and rats (Bjorklund and Bering, 2003), (Bunyar and Huber, 1999), (Custance et al., 1998), (Campbell et al., 1999), (Akins and Zentall, 1998), (Heyes

et al., 1994). No matter its true definition, imitation plays an important role in the development of social cognition in humans. From birth, human beings are excellent imitators. Recent research has reported that human babies are born with the ability to imitate a wide range of actions, such as mouth opening and tongue protrusion (Meltzoff and Moore, 1977). Meltzoff (Meltzoff, 1988) argued that infants use imitation to enrich their understanding of people and their actions; that through imitation, humans become a part of a very complex social environment: human society. Therefore, Meltzoff suggests, a more appropriate species name for humans would be *homo imitans*: the man who imitates.

Because it allows information transfer between individuals, imitation has been seen as an important form of cultural learning (Tomasello et al., 1993). It has been argued by Blackmore (Blackmore, 1999) that imitation makes humans unique among animals, and might even have been evolutionarily selected as a fitness factor, because individuals who were good at imitation had a wider arsenal of learned cultural behaviours, such as tool making, or even language, at their disposal. Some researchers define any unit of cultural ideas, symbols and practices as a *meme*, which (by analogy with the role genes play in biological evolution) can be transferred between humans (Dawkins, 1976). They claim that imitation has an important function in cultural evolution, as it allows memes to be transmitted (Blackmore, 1999).

Experiments on macaque monkeys revealed neurons in their ventral cortex that are active both when the monkey performs an action and when the same action is performed by another animal (Gallese et al., 1996). These neurons have been labelled *mirror neurons* and are considered as a possible mechanism responsible for imitation. Demiris (Demiris, 2002) developed a distributed behaviour-based method for robot imitation that was inspired by the working mechanism of mirror neurons. However, there is no accepted computational model that describes the functionality of mirror neurons in cognitive activities such as imitation. Zentall (Zentall, 2001) claimed that mirror neurons fail to provide an adequate account of imitation, as their activity may be the end product of a complex cognitive process, or they may represent genetically predisposed hard-wiring of the nervous system.

2.2.1 Definition of Imitation

Before defining imitation, it may be beneficial to list some other types of social learning methods (Bjorklund and Bering, 2006). *Mimicry* is the copying of the physical appearance of one species by another; this behaviour is accepted as being genetically based. *Contagion* is the process in

which animals engage in similar species-typical behaviour, for example flocking and herding. *Social facilitation* occurs when the presence of a conspecific encourages similar behaviour: if the observer executes an action when a conspecific is present, although that action is not shown yet, this is deemed social facilitation. *Local enhancement* and *stimulus enhancement* are processes in which the attention of the observer is drawn to a location or an object respectively. *Imprinting* is an animal behaviour in which younger individuals follow an older member of the same group for protection. One could claim that a group of chicks following their mother are imitating her, but the act of following is here interpreted as a simple conditioning process for fear reduction (Kovach and Hess, 1963). In *discriminative following* or *matched-dependent behaviour*, the actions of the observed individual serve as a cue for the observing individual. For instance, if following is rewarded, rats in a maze can learn to follow a trained leader. This does not necessarily mean that the observing agent is actually paying attention to the demonstrator's actions; if the act of observing the demonstrator is followed by a reward, the observing individual may deduce a Pavlovian association between the observed actions and the food reward. This is called *observational conditioning*.

A number of definitions exist for imitation; each emphasizes a different aspect. A well known definition is Thorndike's (Thorndike, 1898):

"Learning to do some act from seeing it done"

This definition restricts imitation to the visual domain and so excludes other types, such as vocal imitation. It might also include other types of social learning methods, such as contagion or social facilitation, which are believed to be different from imitation. However, it emphasizes the direct link between imitation and learning.

Thorpe (Thorpe, 1963) defined imitation as:

"The copying of a novel or otherwise improbable act or utterance or some act for which there is clearly no instinctive tendency"

As this definition states that imitated behaviour should be novel, it might exclude cases in which an individual copies the behaviour of another individual when the behaviour is already in its repertoire. Therefore, it is necessary to define what should be counted as novel.

Bakker and Kuniyoshi emphasize the importance of the mechanism involved in imitation (Bakker and Kuniyoshi, 1996):

"Imitation takes place when an agent learns a behaviour from observing the execution of that action by a teacher"

This definition emphasizes the importance of a teacher or expert in the imitation process, so that the teacher's greater ability can be used to direct other members of the group. Hayes and Hayes used a *do-as-I-do* procedure to demonstrate the imitation abilities of chimpanzees (Hayes and Hayes, 2003). In their study, they tried to calculate the degree of matching of behaviour between subject and model, as a check on how much the imitated action resembles the original action.

Mitchell (Mitchell, 1987) claimed that imitation can be identified when:

- *Something C (the copy) is produced by an organism and/or machine, where C is similar to something else, M (the model).*
- *Registration (or perception) of M is necessary for the production of C.*
- *C is designed to be similar to M.*

This definition implies intent on the part of the imitator, as the copy is designed to be similar to the original.

In accordance with these definitions, a robot to robot movement imitation algorithm is developed in chapter 3. In chapter 5, a method is developed that uses imitation of purely observed behaviours to enhance the learning.

2.3 Robot Imitation

In recent years, the study of imitation in robotics has received cross-disciplinary attention. Unsurprisingly, much of the work has focussed on the problem of humanoid robots imitating humans, as this presents a method for programming a robot by demonstration rather than coding (Dautenhahn et al., 2003). There has been some research on robot to robot imitation, in which imitation has been proposed as a learning mechanism to propagate certain behaviours in a multi-robot group.

2.3.1 Utility of Imitation in Robotics

Imitation in robotics offers many benefits for increasing the performance of robotic agents. Demiris and Hayes (Demiris and Hayes, 1996) claimed that learning by imitation has certain desirable characteristics. It can be used to reduce the solution space for the tasks the agent is trying to solve. Using a suitable reinforcement function, an agent can learn the solution to any problem through reinforcement learning. However, the presence of an expert can be used to increase the learning

speed of the agent. The expert can demonstrate the solution and the learning agent can achieve the solution more quickly by imitation. Similarly, learning by imitation may provide the agent with novel solutions which it may not be able to achieve. In addition, learning by imitation does not require the expert to spend additional time or energy in teaching others, as it can continue to perform its task as the learners observe it. No explicit communication is needed, which is beneficial where communication is costly. Baker and Kuniyoshi (Bakker and Kuniyoshi, 1996) claimed an agent that has the ability to imitate has an increased level of adaptation to its environment. The observed actions are likely to be useful, as they were executed by another agent sharing the same environment. Imitation also provides agents with an efficient means of non-verbal communication. Agents can learn from other agents with different morphologies (hardware or software), because communication via imitation takes place at a higher level (in terms of actions) rather than a lower level (such as motor commands). In addition, they claimed that imitation can be used as an enhancing factor within existing learning strategies such as reinforcement learning, trial-and-error learning or symbolic induction schemes. In accordance with this approach, in chapter 5, we developed a method that uses imitation as an enhancing factor within a reinforcement learning algorithm. The method uses imitation of purely observed behaviours to enhance learning. Billard et al. (Billard et al., 2008) claimed that imitation learning is a powerful mechanism as it can be used to reduce the complexity of the search space for learning. In addition, it offers an implicit way of training a robot as it can minimize the effort that is needed for explicit and tedious programming by a human. Finally, the mechanisms, in which self organization of perception-action coupling arise during development, can be understood by studying and modelling perception-action coupling during imitation learning. Breazeal and Scassellati (Breazeal and Scassellati, 2002) claimed that imitation-inspired mechanisms have three main utilities in robotics research. First, imitation can be an easy way of programming robots to achieve novel actions by observing demonstrations of those actions. Second, imitation can be used for robot to robot or robot to human communications. Third, imitation can be used as an effective tool for efficient motor learning in high dimensional spaces, like in humanoid robots with many articulated joints.

Dautenhahn et al. (Dautenhahn et al., 2003) stated that the study of imitation in robotics holds the promise of overcoming the need to program every behaviour a robot might need to perform. A robot that is able to imitate can learn new actions by observing demonstrations of those actions. Billard (Billard, 1999) claimed that imitation can be used to enhance the learning of communication skills in autonomous robots. A perceptual context shared by the imitator and

imitatee can create the necessary social context in which language can develop. Billard devised experiments in which, using imitation, robotic agents were able to learn a basic synthetic proto-language for matching their environmental perceptions with observed actions. In this thesis, it is shown that, in addition to sharing an environmental context, if the robots share a similar perceptual context, the structure of the imitated behaviours changes during multiple cycles of imitation, so that these behaviours can be copied with high fidelity. Steels and Kaplan (Steels and Kaplan, 2001) stated that social learning can play a crucial role in initiating a humanoid robot into linguistic culture. He emphasized the importance of methods such as open-ended dialogues, in which social learning could be embedded. Alissandrakis et al. (Alissandrakis et al., 2004) claimed that social learning offers variability during imitation, which provides the evolutionary substrate for culture. Therefore, imitation in robotics can serve as the basis for the emergence of artificial culture in groups whose members comprise artificial agents.

2.3.2 Imitation Mechanisms in Robotics Research

A number of mechanisms are proposed to implement imitation on real robots. Demiris (Demiris, 1999) suggested that imitation should be divided into subtasks to be addressed when an agent is introduced to a new environment. In the order that they will be encountered, these subtasks are:

- *Picking the right agent to imitate:* the imitating agent should be able to select which agent to observe, based on its design and behaviours.
- *Identifying a beneficial behaviour to imitate:* the imitating agent should be able to select and extract the beneficial behaviours from the observed agent.
- *Matching the behaviour:* after selecting which behaviour to copy, the imitating agent should be able to translate those behaviours to its own actions.
- *Analysing the reproduced behaviour:* after the selected behaviour is reproduced, there should be an analysis of its outcome, to enable self-improvement.

Bakker and Kuniyoshi (Bakker and Kuniyoshi, 1996) presented a framework for robot imitation in which they claimed that for an agent to imitate an action, it must perform at least three steps:

- *observe* the action: the observing agent should identify a teacher and observe its actions.

- *represent* the action: the agent should choose an appropriate representation for those actions and convert them to its own internal representation.
- *reproduce* the action: the observed actions should be reproduced in an appropriate context.

The embodied imitation algorithm that is developed in chapter 3 of this thesis has a similar framework: A robot performs *observe*, *learn* and *enact* operations in order to imitate another robot.

Dautenhahn et al. (Dautenhahn et al., 2003) identified five questions to be addressed in designing experiments on imitation: *who to imitate*, *when to imitate*, *what to imitate*, *how to imitate* and *what comprises a successful imitation*. They suggested that to be able to implement imitation on real robots, there should be modules addressing these questions.

For an agent to be able to imitate, it has to match the observed behaviour of the demonstrator with a behaviour of its own. The problem of matching the actuators of the demonstrator robot to the imitating robot's actuators is presented as *the correspondence problem*. Nehaniv and Dautenhahn defined the correspondence problem (Nehaniv and Dautenhahn, 2002) thus:

'Given an observed behaviour of the model, which from a given starting state leads the model through a sequence (or hierarchy) of subgoals - in states, action, and/or effects, while possibly responding to sensory stimuli and external events, find and execute a sequence of actions using one's own (possibly dissimilar) embodiment, which from a corresponding starting state, lead through corresponding subgoals - in corresponding states actions, and/or effects, while possibly responding to corresponding events'

Based on the level of similarity between the bodies of the demonstrator and the imitator and the nature of perception of the imitator, there could be different types of correspondence problem (Nehaniv, 2007). A solution to the correspondence problem for robot to robot imitation, for our experimental context, is presented in chapter 5.

Demiris (Demiris, 1999) defined different levels at which imitation can occur. This work categorises imitation thus:

- *Basic Imitation*: reproduction of perceived stimuli, for example an agent copying another's sequence of movements.
- *Functional Imitation*: the agent replicates the effect of the perceived stimuli, instead of the exact stimuli.

- *Internal State Imitation*: the agent imitates the presumed internal state of the demonstrator, instead of the demonstrator's actions.

In accordance with this categorization, as we are interested in embodied imitation that leads to variations in the imitation process, our embodied imitation algorithm only copies the sequence of movements that are executed by a demonstrator robot.

Byrne and Russon (Byrne and Russon, 1998) made a distinction in the context of animal imitation, claiming there can be two levels of imitation: *action-level* and *program-level*. At the action-level, the imitator replicates the actions of the demonstrator, meaning the sequence and content of the actions would be similar to the those of the demonstrator. In program-level imitation, the imitator replicates the overall structure of the demonstrator's behaviour. As long as the overall structure is similar, the actions do not need to be identical. Mitchell (Mitchell, 1987) stated that there are five levels of imitation, based on the awareness necessary for the imitation. At the first level, the agents, neither model nor copy, have *no awareness* of each other. He claimed mimicry to be an example of first level imitation. The other levels are *awareness of the model*, *awareness of the differences between the model and the copy*, *awareness of the imitative act* and *awareness of another's awareness*. Saunders et al. (Saunders et al., 2008) compared three different theories of imitation from a robotic viewpoint. These theories are *Active Intermodal Mapping*, *Associative Sequence Learning* and *Extended Ideomotor Theory*. Each theory is examined based on the questions such as how the correspondence problem is solved, how the theory address the problem of perceptual motor translation and if the theory can be realistically implemented in a robotic system. Demiris and Meltzoff (Demiris and Meltzoff, 2008) made a comparative analysis between on studies of infant learning and the research on robot learning by demonstration. For that purpose, the initial conditions and developmental mechanisms for both infants and robot learners are compared. They claimed that both research areas can benefit from each other: robotic systems can develop methods analogous to the mechanisms observed in human infants while imitation research in robotics can help to better understand the details of developmental mechanisms in humans.

2.3.3 Programming by Demonstration

Although not directly connected to the work in this thesis, this section will give a brief introduction of Programming by Demonstration (PbD) research. PbD, in which a robot is taught new

behaviours by demonstration of these behaviours by humans or other robots, has been widely studied in Robotics research. The term is first applied to industrial robots during 1980s as it appeared to be a way to automate the tedious programming of robots (Billard et al., 2008). In these applications, robots are mostly controlled by *teleoperation* in which the robots are operated by a human teacher while reading its own sensory input and outputs. In doing so, the robots create its own states-actions mapping which is then applied to same task by the robot itself. Some research in PbD that use this technique are (Lozano-Perez, 1983), (Dufay and Latombe, 1984), (Segre and DeJong, 1985), (Ng et al., 2004) and (Browning et al., 2004). With the progress in machine learning techniques, the PbD methods improved from simple direct repetition of demonstrated behaviours to generalizing across sets of demonstrations. In order to generalize solutions that are learned by demonstration to new situations, machine learning methods such as *Artificial Neural Networks* ((Dillmann et al., 1995), (Billard and Hayes, 2003)) and Hidden-Markov models ((Lee and Xu, 1996), (Dixon and Khosla, 2004)) were utilised.

With the development of humanoid robots, recent PbD research had an increasingly interdisciplinary approach. It benefited from examining the neural mechanisms for imitation in primates and social sciences to emulate the ability of imitation in social animals (Billard et al., 2008). As humanoid robots are expected to adapt to changing environments, the flexibility of their control systems became crucial. As a result, the notion of PbD progressively replaced by a more biologically inspired label, *imitation learning*. Some example research that used imitation learning in order to train robots are (Gaussier et al., 1998), (Dillmann, 2004), (Breazeal et al., 2005), (Nicollescu and Mataric, 2007) and (Calinon and Billard, 2007).

2.3.4 Robot to Robot Imitation

The work in this thesis contributes to robot to robot imitation research. In some of the earliest research in robot to robot imitation, Hayes and Demiris (Hayes and Demiris, 1994) used imitative learning to train a robot. They used a learner robot that followed a teacher robot, detecting significant actions of the teacher and replicating those actions. The learner robot associated actions due to imitation with the environmental configuration that it perceived. For example, it learned to turn 90 degrees when it arrived at a corner. In this way, it learned the correct set of actions for each environmental configuration. Hayes and Demiris addressed the issues of how to map an observed action to the robot's own actions and how to replicate observed behaviour, and their experiments showed that such abilities can be achieved by imitation. Demiris (Demiris, 2002)

later attempted to implement a distributed behaviour-based architecture inspired by the properties of mirror neurons, that provided the simulated robots with an architecture for movement imitation ability. Dautenhahn's seminal 1995 paper proposed, for the first time, a larger context of social interactions among robots (Dautenhahn, 1995). In this research, robots were programmed to follow each other by seeking out and attaching themselves to a teacher robot in a hilly landscape. The robots learn to recognize suitable teachers by using learned experience and achieved movement through the environment by associating their actions with the local gradient of the hilly landscape.

Using a connectionist approach, Billard and Hayes (Billard and Hayes, 2003) proposed the *DRAMA* architecture (Dynamic, Recurrent, Associative Memory Architecture). They provided one case study, in which the teacher robot actively participated in the process of imitative learning. Using the *DRAMA* architecture, in other research Billard (Billard, 1999) claimed that the sharing of a similar perceptual context between the imitator and the demonstrator during imitation creates the necessary social context within which language can develop. Steels and Kaplan (Steels and Kaplan, 2001) stated that social learning can play a crucial role in initiating a humanoid robot into linguistic culture. They introduced the notion of language games, in which autonomous robots have been programmed to act according to a specific framework.

Alissandrakis et al. (Alissandrakis et al., 2002) compared different levels of imitation, in which low level was defined as replicating the same actions and high level as deriving a purpose and performing (possibly different) actions to achieve that purpose. They pointed out that sub-goal granularity will affect the performance of the imitation; the imitator can try to replicate every action the teacher performs, or it can just try to achieve the same results, ignoring the actual actions of the teacher. For example, an agent imitating a dance should use the first strategy, while an agent imitating the painting of a wall may use the latter. Alissandrakis et al. created a chess world simulation to examine these issues, in which chess pieces imitated each other's moves. Saunders et al. (Saunders et al., 2004) compared two social learning paradigms: following, or matched dependent behaviour, and observational learning. In their experiments, robots were trained to perceive the dynamic movement of a human or robot carrying a light source by observing either from a static location or while following. They stated that the two mechanisms raise different issues, as there is a trade-off between the positional accuracy obtained through static observation and the benefits of perception to action coupling obtained by following. In their experiments *Khepera-1* robots, that are equipped with 8 infra-red sensors, were used as both imitator and demonstrator. The imitating robot copied the movement of the demonstrator by using its infra-red sensors to

perceive the light source mounted on the demonstrator robot. Saunders et al. (Saunders et al., 2007a) extended their work to compare different methods for observation, namely *reactive following*, *static observation* and *dynamic observation*. As they use infra-red sensors to perceive the movements of the demonstrator robot, they implemented a learning phase, which uses a method named *environmental sampling*, in which the readings from the sensors are filtered in order to estimate the turning angles and move distances of the observed robot. They reported that especially for the static observation, the performance greatly depends on the choice of the threshold values in the learning phase. Compared to this work, the movement imitation algorithm that is developed in this thesis is vision based. The algorithm does not need a learning phase which simplified the analysis about the source of variations in the imitation process. We used static observation as the imitating robot needs to keep the demonstrator robot in its field of view; the imitating robot only rotates itself if the demonstrator robot gets out of its field of view. Additionally in our work, the movement patterns of the robots consist of straight line moves and turns. By simplifying the content of the movement trajectories, we were able to devise a quality of imitation metric which is used to analyse the adaptation of the movement patterns during multiple cycles of imitation. We also examined the adaptation of movement patterns during multiple cycles of imitation with a group of robots.

Following their 1994 work, Demiris et al. (Demiris and Hayes, 2002), (Demiris and Khadhouri, 2005), (Johnson and Demiris, 2004) developed the HAMMER architecture (Hierarchical, Attentive, Multiple Models for Execution and Recognition), which is based on the building block of paired inverse and forward models. The inverse model outputs commands which are fed to the forward model and the output of the forward model is compared with the input of the inverse model. So the robot is able ‘rehearse’ possible actions and compare them with its perception of the actions it is trying to imitate. In another research, Saunders et al. (Saunders et al., 2007b) investigated the concepts of *self-imitation* and *environmental scaffolding* in animals to apply them to a robotic system. They used a sort of high level teleoperation in which the robot’s actions are moulded by a human teacher and, by the process of scaffolding, the teacher ensures that the environmental context amplify the learning experience of the robot. The method was tested on a Pioneer 3-DX robot in which the robot was taught three simple task: point to an object, track and follow a human carrying that object and avoid obstacles.

Alissandrakis et al. (Alissandrakis et al., 2004) developed the ALICE architecture (Action Learning via Imitation between Corresponding Embodiments), to address the problem of imitation between dissimilar embodiments. They examined the rules of synchronization, looseness of perceptual matching and proprioceptive matching in a series of experiments in which a robotic arm with variable-size joints tried to imitate other robotic arms. Although not tested on real robots, ALICE contributes a powerful generalised solution to the correspondence problem for imitation between agents with different morphologies. It works by dynamically creating a correspondence library relating the actions of the demonstrator to the actions the imitator is capable of. The proposed architecture was tested on simulated robotic arms with variably-sized and numbered joints. As these were simulated agents, the only source of variation was the heterogeneities between the robotic arms. In this thesis, it is shown that, with embodied imitation on real robots, even in a homogeneous group of robots, variations arise that allow certain behavioural patterns to emerge during multiple cycles of imitation. These emergent behaviours can be copied with higher fidelity, as they appear to be more robust to the uncertainties in real robots' sensors and actuators.

2.4 Collective Robotics

Collective robotics studies the use of a team of autonomous robots to collectively fulfil predefined missions, seeking to use a collection of simple mobile robots to perform tasks which transcend the abilities of an individual robot (Melhuish, 2001). rather than defining the high level behaviours of each robot in the group, the solution depends on a collective behaviour emerging from the interactions among the robots and from between the robots and the environment in which they operate. Collective robotics research is highly inspired by natural collective systems, especially social insects and their communities. Social insect colonies are considered to be decentralised systems consisting of co-operative and autonomous individuals distributed in the environment: individual insects have limited abilities but the system is able to achieve complex tasks such as nest-building. Collective robotics examines such systems and applies similar principles to robotic systems.

2.4.1 Swarm Robotics

Swarm Robotics is the most well-known field of collective robotics. It aims to co-ordinate a large number of simple physical robots to achieve a purpose. Sahin (Sahin, 2005) defined swarm robotics as:

“the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from local interactions among the agents and between the agents and the environment.”

There are two underlying concepts in swarm robotic systems: *decentralised control* and *self-organisation*. In a swarm robotic system, no central control protocol shapes the behaviour of each individual robot. Instead, complex collective behaviour and patterns emerge from local interactions. Therefore, the robots' local interaction rules determine group behaviour. A form of self-organisation, *stigmergy*, an indirect mechanism of communication between agents, is another important aspect of swarm robotic systems. In stigmergy, the robots communicate and co-ordinate with each other through changes within the environment where they are operating. The use of pheromones is an example of natural stigmergic communication (Bonabeau et al., 1999).

Some well known problems extensively examined through swarm robotics are *flocking*, *aggregation*, *clustering*, *collective searching*, *co-operative transport* and *foraging*. *Flocking* was one of the first social animal behaviours to be studied through swarm robotics. In a flock of birds, each bird seems to be flying independently, although the entire shape of the flock is well-organised and coherent. Reynolds (Reynolds, 1987) simulated the flocking behaviour of birds in his famous 'boids' work, in which each boid (agent) was controlled by three simple local rules: *separation*, *alignment* and *cohesion*. The agents try to remain separate, to avoid collisions while remaining aligned towards the average position of their neighbours. At the same time, they steer towards the average position of their neighbours, to provide cohesion. Using these rules, a simulated flock is able to move in a realistic and organised way, reflecting behaviour seen in the natural world. Mataric (Mataric, 1994b) used a team of mobile robots to examine emergent flocking behaviour with simple local rules. The robots were controlled by a schema-based routine method which combined simple rules such as *aggregation* and *dispersion*. The robot team demonstrated an emergent flocking behaviour.

Nebrini (Nembrini, 2005) developed an emergent flocking behaviour based on only local communication between mobile robots. Each robot had a limited range communication ability in

which they exchanged some information between nearby neighbours. They were controlled by three simple behaviours, which were avoiding other robots if they got too close, turning back to recover connection and making a random turn to maintain aggregation. It was shown that flocking behaviour could be achieved by setting priorities to these simple rules. Hanada et al. (Hanada et al., 2007) proposed a flocking method for mobile robots in an environment with obstacles. In their experiments, robots dynamically selected two neighbouring robots in their sensing range and maintain a specific distance with them. They showed in a simulation that the robot managed to navigate in coordination by using only local interactions. Stanieri et al. (Stranieri et al., 2011) studied self-organized flocking behaviour in a swarm of robots. The swarm consists of two different types of robots, aligning and non-aligning, in which aligning robots had the ability to agree on a common direction while non-aligning robots did not have this ability. In simulation, they showed that the swarm could achieve flocking behaviour and they proposed a model for achieving motion in the self-organized flock of robots.

Beckers et al. (Beckers et al., 1994) studied collective clustering with a group of mobile robots. The collective task of the robots was to gather randomly distributed objects (pucks) and cluster them into a pile. Each robot was programmed with three simple behaviours: *moving in a straight line* (the default behaviour), *obstacle avoidance* and *puck-dropping*. Co-ordination between robots was achieved by stigmergy, as no other type of communication was allowed. Beckers et al. showed that the robot team was able, albeit in a long period of time, to create a single large cluster of pucks. Melhuish et al. (Melhuish et al., 1998) presented their robot swarms with several types of items, to be clustered into separate piles. Through slightly altering their behaviour depending on the type of puck they encountered, the robot team achieved emergent segregation of different items. In another research, Melhuish et al. (Melhuish et al., 2001) extended their method in order to sort multiple types of objects by adding a new sensor to robots that distinguished between different types of objects

Trianni et al. (Trianni et al., 2003) examined the emergent aggregation behaviour of simulated robots. Each robot was controlled by a neural network evolved by a genetic algorithm. The fitness of each robot was determined by calculating its distance from the centre of mass of all the robots. They observed that two types of aggregation behaviour emerged: *static clustering behaviour* and *dynamic clustering behaviour*. Static clustering behaviour enabled the robots to form a compact and stable aggregation, whereas dynamic clustering behaviour enabled the robots to form a less compact but more scalable aggregation.

Hayes et al. (Hayes et al., 2000) presented an experiment in which a team of mobile robots searched for an infra-red beacon. They compared a collaborative strategy, in which the robots used a simple binary signalling method to attract surrounding robots, to a non-collaborative strategy, reporting that the collaborative approach improved the performance of the robot team. Correll and Martinoli (Correll and Martinoli, 2006) used a swarm of robots in order to inspect the blades in a jet turbine. They captured the dynamics of the system by using probabilistic, discrete-time macroscopic models and they tried to find an optimal collaboration mechanism to minimize the time and energy consumption of the swarm. Pugh and Martinoli (Pugh and Martinoli, 2007) used *Particle-Swarm Optimization* (PSO) in order to implement a multi-robot search with mobile robots. They made some modifications on the PSO algorithm based on the properties of the robots and their environment. The algorithm was tested on *Webots* robot simulator and it was shown that the performance of the method could be enhanced by increasing the number of particles or the communication range of the robots.

Mataric et al. (Mataric et al., 1995) presented a co-operative box-pushing experiment, in which two robots pushed a box towards a ‘goal’. Their co-operative strategy took advantage of a simple communication protocol which compensated for the robots’ noisy sensors. They showed that two robots using the co-operative method outperformed both a single robot and an approach using two non-communicating robots. Kube and Bonebeau (Kube and Bonabeau, 1998) modelled co-operative transport in ants in experiments with a group of mobile robots that collectively pushed a large box towards a goal. The robots communicated by stigmergy, with no explicit communication.

Krieger and Billeter (Krieger and Billeter, 2000) presented an experiment in which a swarm of mobile robots engaged in a foraging task by collecting food items to maintain the energy level in their nest. They implemented a task allocation mechanism based on randomly chosen activation thresholds. Their experiments showed that such a mechanism achieves a self-organised task allocation in an efficient and decentralised way. Liu et al. (Liu et al., 2007) developed a probabilistic macroscopic model for a swarm of foraging robots in order to investigate the effects of group size and individual parameters on the performance of the swarm. They showed that a set of individual parameters could be determined that would enhance the performance of the swarm. Liu et al. (Liu et al., 2009) extended their model for a swarm of heterogeneous robots in another research. Campo et al. (Campo et al., 2010) investigated path selection in order to enhance the foraging behaviour

of a swarm of robots. To do that, they implemented *virtual ants* which provided local communication between robots by laying artificial pheromone inside a network of robots. They parametrized their method with a mathematical model and tested on a swarm of twenty real robots. Stirling and Floreano (Stirling and Floreano, 2010) developed strategies to deploy a swarm of flying robots in an unknown environment. In their experiments, a swarm of flying robots progressively built a dynamic sensor network to save energy. They tested different strategies based on the sensing and communication capabilities of the robots.

2.5 Reinforcement Learning

To survive or adapt themselves to changing conditions, all organisms must interact with their environment; perceiving their environment is crucial for any living organism. *Reinforcement Learning* is a computational approach that explores the use of interactions between an artificial agent and its environment to solve problems, mapping environmental situations to actions that the agent is able to perform to maximize a numerical reward. The learning agent must be able to sense the state of the environment to some extent and must be able to affect the environment by performing actions. In addition, it should have a goal to achieve. Therefore, the basic reinforcement learning formulation includes three components: sensation, action and goal (Sutton and Barto, 1998). The learning agent discovers which actions result in the most desirable outcome by testing them in a trial-and-error search.

Reinforcement learning is different from *supervised learning*. In supervised learning, the agent learns from examples provided by a supervisor that knows the best response to some situations. Supervised learning is not adequate for learning from interaction, as it is often impossible to obtain the best response that would apply in any given situation the agent may encounter. In reinforcement learning, correct responses to any situation are never given to the learning agent. Instead, the learning agent is expected to discover these responses by interacting with its environment.

As reinforcement learning purely depends on interaction with the environment, it involves finding a balance between *exploration* and *exploitation*. To obtain a high immediate reward, an agent would prefer an action that it has already tried and which produced a desirable outcome. However, it also has to try previously unselected actions, to discover which actions offer a high reward. Therefore, to make better action selections in the future, the agent needs to exploit its

current knowledge, while exploring other possible action. The exploration versus exploitation trade-off has been widely studied in reinforcement learning literature.

Beyond the basic elements, it is possible to identify four main components in a reinforcement learning system (Sutton and Barto, 1998): a policy, a reward function, a value function and a model of the environment. *Policy* defines the reaction of the agent in a given state. It maps the state of the environment to actions that can be executed in that state and therefore determines the behaviour of the agent following any environmental input. The *reward function* calculates the desirability of the state of the environment. It produces a single metric, *reward*, that maps each state to its outcome. The *value function* similarly calculates the cumulative reward that can be achieved from the starting state of the environment. Therefore it indicates the long-term desirability of certain environmental states. The main objective of the learning agent is to maximize its long-term reward it receives. *A model of the environment*, given a state and an action, predicts the next state of the environment; this is used to estimate the response of the environment to an action in a state.

If, for a reinforcement learning task, the effects of an action in a state depend only on that state and the action taken, then the learning task is called a *Markov decision process* (MDP). In reinforcement learning research, MDPs are particularly important as most methods try to solve problems which are MDP.

There are three main classes of methods for solving problems with reinforcement learning. The first class, *Dynamic Programming* (Bellman, 1957), is used to compute the optimal policies, given a perfect model of the environment, by iteratively estimating the value of each state. Classical dynamic programming algorithms are of limited utility because of their assumption of a perfect model of the environment and because of their high computational expense. The second class, *Monte Carlo Methods*, do not require a complete knowledge of the environment. Monte Carlo methods learn from experience by averaging sample returns. They are used for episodic tasks so that sample returns for a specific episode can be calculated. The third class, *Temporal Difference Learning*, is a combination of dynamic programming and Monte Carlo methods. Like Monte Carlo methods, temporal difference learning methods can learn directly from experience. They do not require a complete model of the environment. Like dynamic programming, they can update estimates based on previously learning. They do not need to wait until certain episodes are completed. Temporal difference methods take into account the fact that subsequent estimates are often correlated at some level.

There are two types of control methods based on temporal difference learning. *SARSA* (Rumery and Niranjan, 1994), is an *on-policy* control method. Given a policy, an agent using a *SARSA* algorithm will update its policy by interacting with its environment. The algorithm updates its estimates for the state-action pairs by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.1)$$

where s_t is the current state, a_t is the chosen action, r_{t+1} is the reward the agent gets when the action is performed, s_{t+1} is the new state of the agent, a_{t+1} is the action that will be chosen in the new state, α is the learning rate and γ is the discount factor. The quintuple $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}$ forms the acronym *SARSA*. As expressed in the formula, the Q value of a state-action pair is updated by an error that is adjusted by the learning rate, α .

The second control method based on temporal difference learning is *Q-learning* (Watkins, 1989), an *off-policy* control algorithm. It directly estimates the Q value of state-action pairs based on learned experiences, independent of the policy being followed. One step *Q-learning* is defined by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2.2)$$

Q-learning is one of the most important breakthroughs in reinforcement learning. It is able to compare the expected utility of the actions that can be executed in each state without the need for a model of the environment. It is highly suitable for real-time control of agents that are learning to achieve a task by interacting with their environment. An imitation-enhanced *Q-learning* algorithm is presented in Chapter 4, where its performance is tested in comparison with an original *Q-learning* algorithm in a multi-agent based simulation.

2.5.1 Reinforcement Learning in Robotics

As reinforcement learning enables artificial agents to learn from interaction with their environment, it has been widely used to train and control mobile robots in robotic research. Kaelbling (Kaelbling, 1990) used reinforcement learning to teach a robot to approach a light source while avoiding obstacles. She used immediate and delayed reinforcement, in which an estimate for the reinforcement of each input-action pair was stored. Mataric (Mataric, 1994a) presented a formulation for reinforcement learning in noisy and dynamic environments; her method used behaviours

and their associated conditions to minimize the state space. Reinforcement was calculated by using *heterogeneous reinforcement functions* and *progress estimators*. The method was validated with a multi-robot group learning a foraging task.

Mahadevan and Connell (Mahadevan and Connell, 1991) used Q-learning to train a mobile robot. They claimed there are some requirements for any learning algorithm that could be used on robots: *noise immunity*, *fast convergence*, *incrementality*, *tractability* and *groundedness*. They reported their behaviour-based method speeded up learning of the robot by converting the complex task into a set of simpler subtasks. They tested their method for a box pushing task on both a real behaviour-based robot, *OBELIX*, and in simulation.

Thrun and Schwartz (Thrun and Schwartz, 1993) stated that function approximators cause a systematic overestimation effect when they are recursively used for value estimation. To prevent this problem, Smart and Kaelbling (Smart and Kaelbling, 2000) introduced the *HEDGER* prediction algorithm, which is based on *locally weighted regression*. Locally weighted regression is a variation of the standard linear regression method, in which training points that are close to the query point have more influence on regression calculation. The algorithm was tested with real robots on two problems, the *mountain-car task* and *corridor-following*, in which they achieved a high performance.

Crites and Barto (Crites and Barto, 1996) examined the application of reinforcement learning techniques in an elevator dispatching task, in which the objective is to minimize the average waiting time for passengers. They applied a version of one-step Q-learning with some additional constraints that provided some prior knowledge to the system. A neural network was used for function approximation to represent the action-value functions. They reported their method achieved higher performance than the controller most frequently used in real elevator systems. Smart and Kaelbling (Smart and Kaelbling, 2002) presented a framework for using Q-learning on mobile robots. Their method uses some example trajectories to increase the learning speed of the robots. They introduced two distinct learning phases. During the first phase the robot is controlled by a human while the Q-learning system observes states, actions and rewards encountered by the robot. In the second phase, the Q-learning system gets full control of the robot. They tested their framework on some problems such as *corridor following* and *obstacle avoidance*. Peters et al. (Peters et al., 2003) attempted to apply reinforcement learning to high dimensional systems such

as humanoid robots. They presented the *Natural Actor-Critic algorithm* that depends on *natural policy gradient* instead of regular policy gradient methods. They tested their algorithm on a pole-balancing experiment in which it outperformed some other policy gradient methods.

There is some research that attempted to use imitation in connection with reinforcement learning methods. For instance Latzke et al. (Latzke et al., 2006) utilised Q-learning and used the experience of an expert agent as training data. The experience of the expert was provided as a sequence of states and action along with their reward. Bentivegna et al. (Bentivegna et al., 2004) used experience of a teacher in order to improve the performance of a learner agent through practice. They stored the actions of the teacher in a number of states which was then used by the learner agent during its practice period. The imitation enhanced Q-learning algorithm that is developed in this thesis is different from these methods as it uses the imitation of purely observed behaviours to enhance learning. There is no assumption about the learner agent having access to the internal state or experience of the teacher agent. The only information that is transferred between agents is the teacher agent's executed actions. Finding in what state or context these actions are meaningful is determined by the learning process of the agent.

2.6 Summary and Conclusion

This chapter provided an overview of the key existing work related to this thesis. It started with presenting imitation and its functionality in the social animals. In chapter 3, in accordance with the definitions given, an embodied movement imitation algorithm is presented and tested on a group of mobile robots. As stated, imitation has a social dimension, allowing individuals to become a part of a social community as it allows certain skills or behaviours to be shared by multiple individuals of the community. In chapter 4, we use real robots to model social learning between artificial agents. It is shown that variations that arise from embodiment allow certain behaviours that appear to be robust to the process of imitation to emerge during multiple cycles of imitation.

As this thesis contributes to the fields of Collective Robotics and Reinforcement Learning, the rest of the chapter briefly presented these fields. Some key works that are relevant to this thesis were listed and their connection to the work in this thesis was explained.

Chapter 3

Imitation on Real Robots

In this chapter, firstly the hardware and the infrastructure used for the work in this thesis are presented. The properties of the robots that constitute the artificial group of agents in the experiments are specified and a detailed description of the infrastructure for the experiments is given. Secondly, an embodied movement imitation algorithm is presented which completely depends on the on-board sensors of the robots. The algorithm is tested on robots with different settings and it is shown that uncertainties in the sensors of the robots cause the imitated movement pattern to differentiate from their originals.

3.1 Introduction

In the previous chapter, research that examines imitation and its links with learning new behaviours was outlined. Since this thesis explores imitation as a method of social learning, we need a system of embodied artificial agents, i.e. robots, in which the agents are able to learn socially from each other, by imitation. Throughout this thesis, when a robot imitates another, it performs the following operations:

- *observe*: The imitating robot watches the demonstrator robot with its on-board camera. Therefore the imitation is vision-based.
- *learn*: The imitating robot converts the observed movement patterns of the demonstrator robot into a set of moves and turns, i.e. a trajectory, by using the imitation algorithm explained in section 3.3. At the end of this step, the imitating robot saves the newly learned movement pattern to its memory so that it can be enacted by the robot later.

- *enact*: The learned movement pattern is converted into motor commands and these motor commands are executed by the imitating robot.

Using real robots instead of simulated agents or biological social entities for modelling is motivated as follows:

- Real robots, with their less than perfect perception and actuators, provide natural variation in the imitation process which allow new behaviours to emerge. Using simulated agents in a simulated environment, it would be possible to control the degree and types of heterogeneities and noise, but this may preclude or predetermine any emergent processes that are a part of imitation; the level and characteristics of emergence in a simulated environment would be limited to the level of variance that is artificially introduced.
- Data about the imitative activity, including the internal data and calculations of the robots, can easily be extracted and examined. This would not be the case if biological social entities (for example, people or monkeys) were used.
- The implementation of imitation on real hardware makes clear how theoretical assumptions and hypotheses regarding imitation can be operationalised.

In addition, robots are not allowed to transmit behaviours (i.e. a sequence of motor actions) directly from one to another. This means that the robots have to overcome essentially the same problems of inferring each others' behaviours from possibly unreliable first-person perceptions as any embodied agents (robots, animals or humans), yet at the same time the robots implement a rather minimal model of social learning by imitation. This embodied yet abstract model of social learning provides both a degree of biological plausibility and opportunities for unexpected emergence that would not be present in an agent-based simulation.

3.2 Hardware Setup

The artificial system that is used to model imitation consists of *e-puck* miniature robots (Mondada et al., 2009). E-puck robots are 7 cm in diameter and 5 cm in height. Table 3.1 lists the sensors on board of an e-puck robot. Importantly, the e-puck robots can sense and track the movement of other robots nearby (albeit imperfectly because of their limited sensors); thus the robots have the

Sensor type	Used in experiments?
2 stepper motors	Used for embodied imitation
2 wheels of 41 mm diameter	Used for embodied imitation
8 proximity sensors	Not used
CMOS image sensor	Used for embodied imitation
Accelerometer	Not used
Microphone and speaker	Not used
Ring of colour LEDs	Used for signalling between robots
LiION battery (3 hours of autonomy)	Used for embodied imitation
Bluetooth adapter	Not used

Table 3.1: Sensors of an e-puck robot

physical apparatus for imitation. Robots can signal to each other with movement and light, one to one or one to many, providing alternative methods for robot to robot interaction.

Despite these benefits, the original e-puck, with its default micro processor dsPIC30F6014A, lacks the computational power that is needed for the image processing during imitation. To overcome this limitation, the robots are enhanced with a Linux extension board (Liu and Winfield, 2011) based on the 32-bit ARM9 microcontroller with Debian/Linux system installed. The board has a USB extension port which is used for connecting a wireless network card and it is equipped with a MicroSD card slot. These additions to the standard e-puck robot have a number of benefits (Figure 3.1):

- **Increased processing power:** The ARM9 processor on the Linux extension board has a higher processing power compared to the dsPIC on the standard e-puck. Its processing power is 200 MIPS at 180 MHz compared to the 15 MIPS at 60 MHz of the dsPIC on the original e-puck.
- **Increased memory capacity:** The Linux extension board has significantly more memory than the original e-puck. It has 64 megabytes of RAM and additional 2 gigabytes of storage space on the MicroSD card compared to the 8 kilobytes of RAM and 144 kilobytes flash on the original e-puck.
- *Player* (Gerkey et al., 2003), a cross platform robot device interface and server, is installed on the Linux extension board. Running on the robot, it provides an interface to the robot's sensors and actuators over the IP network. It makes it possible to control a group of robots from a station on the same IP network and get their outputs in real-time. It communicates

with the dsPIC of the e-puck via SPI bus. The high level program runs on the Linux extension board and it sends necessary commands to the dsPIC in which a secondary controller collects these commands, activates the actuators of the robot if necessary and sends back the sensor readings to the linux extension board.



Figure 3.1: An e-puck with Linux board fitted in between the e-puck motherboard (lower) and the e-puck speaker board(upper). Note both the red skirt, and the yellow hat that provides a matrix of pins for the reflective spheres which allow the tracking system to identify and track each robot.

Since the bodies of the robots are transparent, in order to make it possible for the robots to 'see' each other, they are fitted with coloured skirts (figure 3.1). So the robots observe each other's movements by visually sensing their skirts. The experiments are performed in a 3m by 3m robot arena (figure 3.2). Figure 3.3 shows the infrastructure of the experimental setup. Each robot is identified with a static IP address. Robots connect to the swarm wireless subnet through a wireless router and they can be accessed via SSH protocol from any computer connected to this subnet. A dedicated swarm-server is configured as a data logging pool for the swarm of robots. It also functions as a router to bridge the swarm wireless subnet and the local network.

A vision tracking system from ViconTM (<http://www.vicon.com>) provides high precision position tracking for multi-robot experiments. The tracking system consists of four Vicon MX400 cameras, one Vicon MX and one HP xw6600 workstation. Each robot is fitted with a tracking hat which provides a matrix of pins for the reflective spheres that allow the tracking system to identify

and track each robot (figure 3.1). The tracking system is connected to the local network in which it broadcasts the real-time position of each robot over TCP/IP during experiments.

The author of this thesis, in collaboration with other researchers and technicians of Bristol Robotics Laboratory, worked for the planning and development of the infrastructure explained in this section. The Linux extension board was designed by Dr. Wenguo Liu, the data logging tool was implemented by Jean-Charles Antonioli and the construction of Vicon system and robot arena was supported by the technicians of the Bristol Robotic Laboratory. All robot coding for experiments, experimental design, development, execution and analysis were made by the author of this thesis.

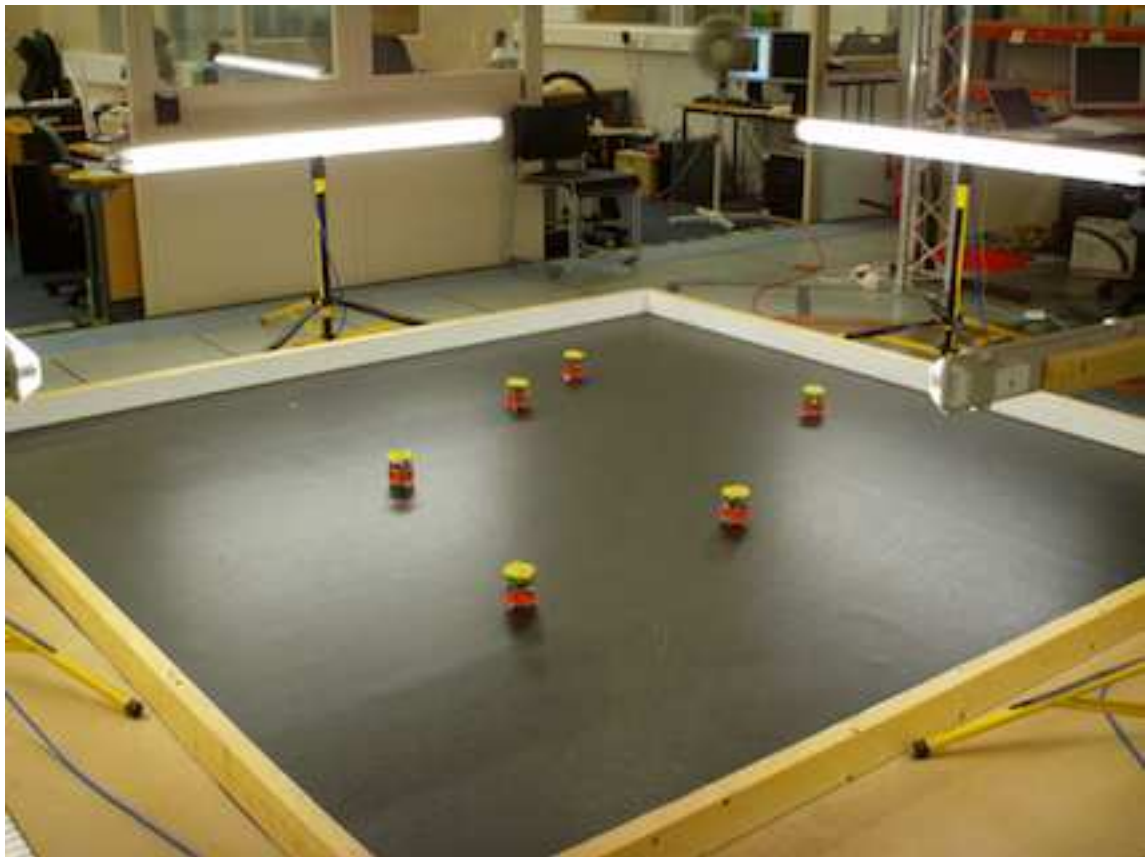


Figure 3.2: Robot arena with 6 robots.

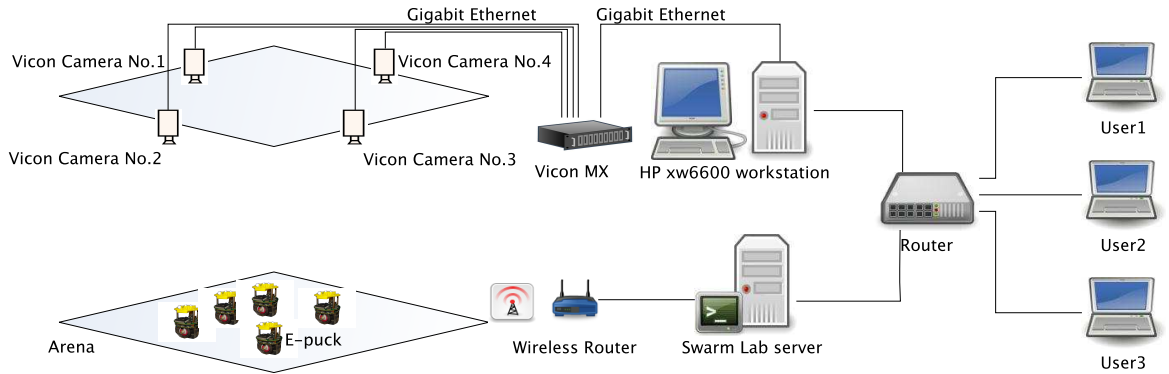


Figure 3.3: Infrastructure of the experimental setup. Each robot is identified with a static IP. Robots are fitted with tracking hats which provide a matrix of pins for the reflective spheres that allow the tracking system to identify and track each robot. The dedicated swarm lab server works both as a logging pool for the experiments and as a router to bridge the robots and the local network. Diagram from (Liu and Winfield, 2011)

3.3 Imitation Algorithm

In this chapter, an embodied movement imitation algorithm is implemented on the robots. The algorithm is embodied in the sense that it completely depends on the on-board sensors of the robot. The imitation algorithm has two phases: *frame processing* and *data processing*. The frame processing phase corresponds to the *observe* step, as defined in section 3.1. During this phase, the imitating robot observes the movement of the demonstrator robot by processing frames captured by its image sensor. The following operations are applied to each frame:

1. *The blobfinder* (Bruce et al., 2000) tool of Player determines the position and size of the skirts covering the observed robot. It takes a colour value as input and fits a rectangular blob around the regions with that colour (figure 3.4).

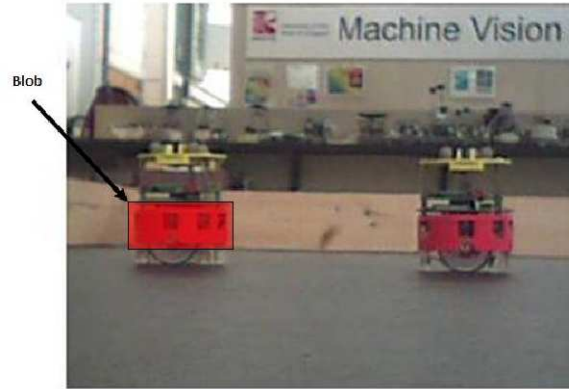


Figure 3.4: An image from the image sensor of the robots. In the figure, the blobfinder tool of Player has fitted a blob around the red skirt of the robot on the left.

2. The exact size and position of the blobs are determined by approaching from 4 sides to the approximate blob position and checking pixel values.
3. In each frame, the new position of the observed robot is determined by comparing it with its previous position and considering the speed of the robot. Every 0.2 sec, a new image frame is captured by the robot. The robot's speed is set to 5.2 cm/sec during experiments so the displacement of the robot between two consecutive image frames should be equal to (if the robot was moving) or smaller than (if the robot was turning) 1.04 cm . Therefore the observed robot should be inside a circle, around its previous position, with a radius of 1.04 cm , in the new frame. Any observed position outside this radius is ignored.
4. The relative position of the observed robot to the observing robot is calculated and stored in a linked list. Since the robots have only one camera and hence monoscopic vision, the observing robot determines the relative distance of the observed robot by previously calculated blob position and size (figure 3.5).

In this way, up to 5 frames per second are processed and the relative position information for each frame, $(d_{robot}^x, d_{robot}^y)$, is stored in a linked list. The pseudo-code for the frame processing phase is given in algorithm 1. During this phase, the observing robot rotates itself, if it is necessary, to keep the demonstrator robot in its field of vision.

After the observed robot's movement sequence is completed¹, the linked list of relative positions is processed during the data processing phase. The data processing phase corresponds to

¹The signalling protocol between robots that determines the beginning and completion of a movement sequence will be explained in the next chapter

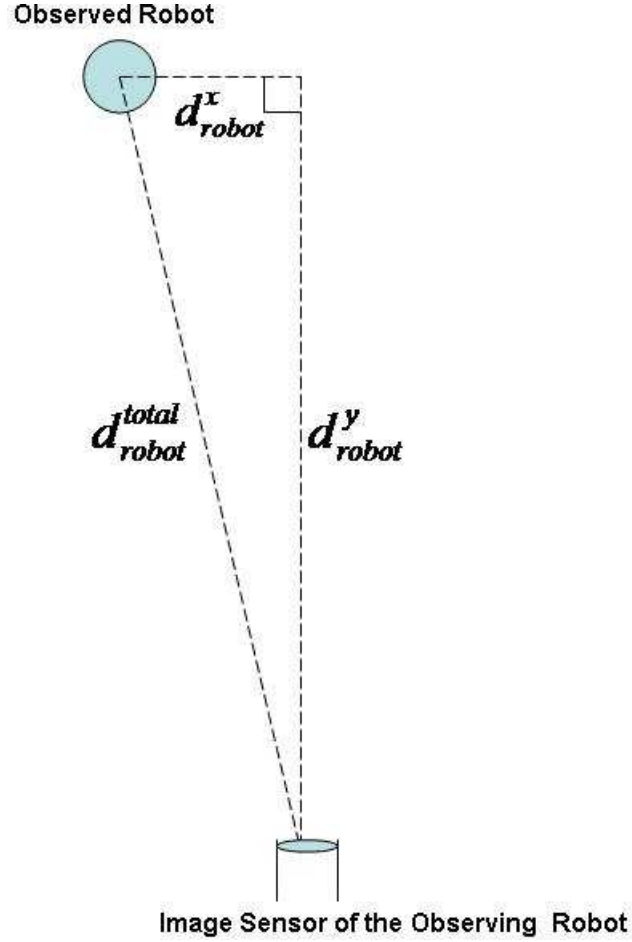


Figure 3.5: Frame processing phase: Vertical distance (d_{robot}^y) and horizontal distance (d_{robot}^x) from the observing robot are calculated. (d_{robot}^x, d_{robot}^y) for each frame is stored in a linked link of positions. (d_{robot}^x, d_{robot}^y) is calculated by:

Previously calculated metrics:

Size of the blob when $d_{robot}^{total} = 20\text{ cm} \rightarrow s_{robot} = 106\text{ pixels}$

Actual size of the blob $\rightarrow 3\text{ cm}$

At any distance: $d_{robot}^{total} = \frac{106 \times 20}{s_{robot}}$, $d_{robot}^x = \frac{3 \times p_{robot}^x}{s_{robot}}$, $d_{robot}^y = \sqrt{(d_{robot}^{total})^2 - (d_{robot}^x)^2}$

where:

d_{robot}^{total} = Total distance between two robots, in cm

d_{robot}^x = x component of the distance between two robots, in cm

d_{robot}^y = y component of the distance between two robots, in cm

p_{robot}^x = x component of the distance between two robots on the image frame, in pixels

s_{robot} = size of the blob covering observed robot on the image frame, in pixels

Algorithm 1 Pseudocode for frame processing phase of imitation algorithm**Input:**Robot being observed, $Rob_{observed}$ Linked list of relative positions of the observed robot, $PositionList \leftarrow \emptyset$ **while** observation continues **do** $f \leftarrow GetImageFrame()$ $BlobList \leftarrow GetBlobs(f)$ $RobotList \leftarrow GetRobots(BlobList)$ **for each** $rob \in RobotList$ **do****if** $rob = Rob_{observed}$ **then** $[d_{robot}^x, d_{robot}^y] \leftarrow GetRelativePosition(rob)$ $PositionList \leftarrow AddNewPosition([d_{robot}^x, d_{robot}^y])$ **end if****end for****end while**

learn step, as defined in section 3.1. The objective of this phase is to reconstruct the observed robot's movement pattern. The following operations are applied during this phase (figure 3.6):

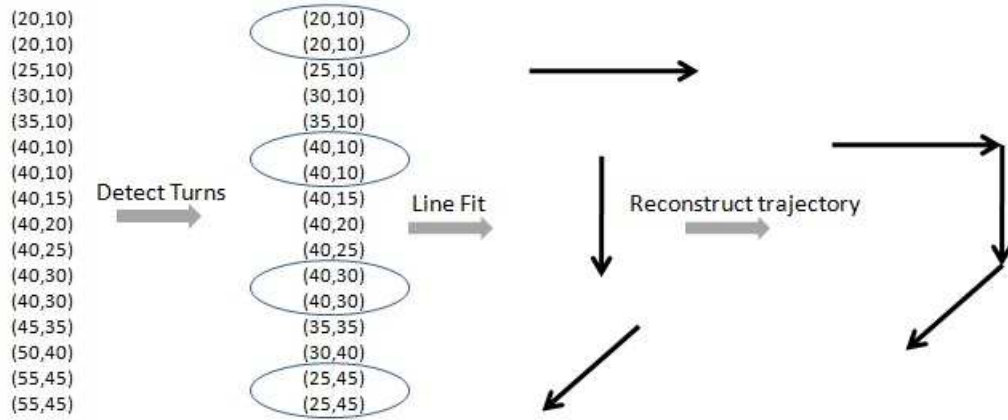


Figure 3.6: Data processing phase: Given a linked list of positions as input, the first three steps of data processing phase are detecting turns, line-fitting to determine straight line segments between turns and reconstructing (estimated) observed trajectory. These three data transformations are shown as light grey arrows. The input to detect turns and line fit are shown numerically with example values where each data pair is a relative position of the demonstrator robot to the imitating robot; the outputs for line fit and reconstruct trajectory are shown as vectors.

1. The robots' movement patterns consist of turns (rotations) and straight line segments. While the demonstrator robot is rotating itself to turn to a new direction, it would appear to be static to the observing robot as its relative distance and position stays unchanged. The Euclidean

distance between each consecutive relative position in the link list is calculated to determine the intervals that the observed robot is rotating itself in its current position. These intervals are marked as ‘turns’.

2. During each period between consecutive turns, the robot is moving in a straight line. To better estimate the direction of the line, a regression line fitting algorithm (Edwards, 1976) is utilised by considering all the relative positions that are recorded during that period. The regression line associated with n relative positions $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$ is calculated by the regression (best fit) line algorithm with the form of:

$$y = mx + b \quad (3.1)$$

in which

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n \sum (x^2) - (\sum x)^2} \quad (3.2)$$

$$b = \frac{\sum y - m(\sum x)}{n} \quad (3.3)$$

Assume there are 10 positions stored in the linked list from the frame processing phase, $p_1, p_2, p_3, \dots, p_{10}$. If p_3 and p_4 correspond to the same location on the arena, it means that the observed robot was turning during the period p_3 and p_4 was recorded. If the next such occasion is at p_8 and p_9 , the observed robot was moving on a straight line during the period that p_4 to p_8 was recorded. So the positions p_4 to p_8 are utilised for the line regression calculation to determine the length and direction of the movement (figure 3.7).

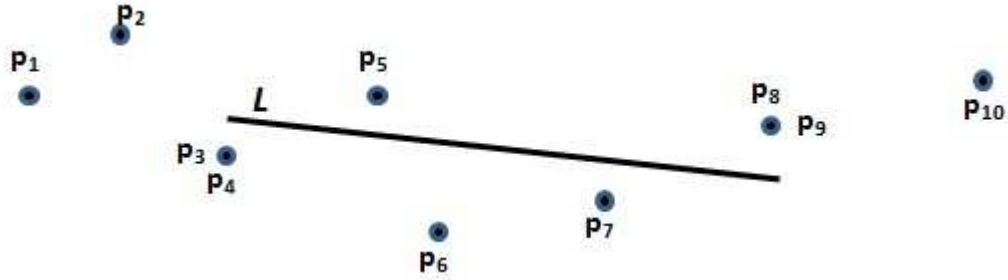


Figure 3.7: Regression line fitting: p_3 and p_4 are located on the same location of the arena which means that the robot was turning when these positions were recorded. The next such occasion is at p_8 and p_9 . So the robot was moving on a straight line between these two turns. Line L is the regression line fitted by using points p_4, p_5, p_6, p_7 and p_8 .

3. By combining the straight line segments and the turns between them, the observed trajectory is reconstructed. The pseudo-code for the data processing phase is given in algorithm 2.

Algorithm 2 Pseudocode for data processing phase of imitation algorithm

Input:

Linked list of relative positions of the observed robot, *PositionList*

List of moves, *MoveList* $\leftarrow \emptyset$

Turns $\leftarrow \text{DetectTurns}(\text{PositionList})$

for $i = 1 \rightarrow \text{size}(\text{Turns}) - 1$ **do**

LinePositionList $\leftarrow \text{GetPointsBetweenTurns}(\text{Turns}[i], \text{Turns}[i + 1])$

move $\leftarrow \text{LineFitBetweenTurns}(\text{LinePositionList})$

MoveList $\leftarrow \text{AddNewMove}(\text{move} \rightarrow \text{angle}, \text{move} \rightarrow \text{displacement})$

end for

At the end of the data processing phase, the imitating robot saves the newly observed set of moves and turns, i.e. the estimated trajectory of the observed robot, in its memory. A sample entry in the memory of the imitating robot might look like:

- (271 15) (21 15) (142 19)

which would mean that the observed robot, according to the calculations of the imitating robot, moved 15 cm in a 271° relative angle to the imitating robot, then turned 110° ² counter clockwise and moved 15 cm and then turned 121° counter clockwise and moved 19 cm. A sample memory of a robot is shown in figure A.1.

² 110° is calculated by $|271 - 360| + 21 = 110$

3.4 Quality of Imitation

To quantitatively assess the fidelity of imitation (that is, the similarity between the original movement pattern and its copy), a quality of imitation function needs to be defined. Since each movement pattern consists of straight moves and turns, there are 3 components to each pattern that can be copied: the number of segments (straight moves), the length of each move and the angle (turn) between each consecutive move. Therefore, the overall quality of a copy can be calculated by separately estimating 3 quality indicators. The quality of move length, Q_l , between the original path O and its copy C is calculated as follows:

$$Q_l = 1 - \frac{\sum_m |l_m^O - l_m^C|}{\sum_m l_m^O} \quad (3.4)$$

where l_m is the length of move m that is to be compared. Here, the ratio of move length differences between the original pattern and its copy, to the total move length of the original pattern is calculated. If the original movement pattern and its copy have different numbers of segments, N^O and N^C respectively, the sum is calculated only over the number of segments in the smaller: $\min(N^O, N^C)$. The quality of angle (turn) imitation similarly calculated as:

$$Q_a = 1 - \frac{\sum_m |a_m^O - a_m^C|}{\sum_m a_m^O} \quad (3.5)$$

where a_m is the turn angle following the move m . The quality of segment imitation simply compares the difference between the number of segments of the original pattern and its copy. It is calculated as:

$$Q_s = 1 - \frac{|N^C - N^O|}{N^O} \quad (3.6)$$

where N^O and N^C are the number of segments of the original path and its copy³. The overall quality of imitation, Q_i , is a combination of 3 quality indicators:

$$Q_i = \frac{LQ_l + AQ_a + SQ_s}{L + A + S} \quad (3.7)$$

where L , A and S are weighting coefficients.

³ N^O is always greater than 0 as there should be a movement in the original movement pattern so that it could be copied.

To test the performance of the algorithm, a demonstrator robot is programmed to follow sequences of moves and turns that describe different geometrical shapes while an imitator robot watches and tries to learn the movement pattern. Then the imitator robot performs its copy of the demonstrator's pattern (figure 3.8). By comparing these two movement patterns, the quality of imitation is determined. The same movement pattern is repeated and copied multiple times at different distances between robots. Movement patterns are classified according to their structure: if it has none or only one turn, it is defined as a *low-complexity* movement pattern. If it has 2 or 3 turns (a triangle or a square), it is defined as a *medium-complexity* movement pattern and if it has 4 or more turns it is defined as a *high-complexity* movement pattern. In accordance with this classification, 3 shapes that have different levels of complexity are used (figure 3.9) to test the algorithm. In the first set of experiments, the demonstrator robot is programmed to move in a line, forward and backward, which is a low-complexity movement pattern, while the imitator robot watches it from different distances. In the second set of experiments the demonstrator robot has an equilateral triangular trajectory which is a medium-complexity movement pattern. Finally in the third set of experiments, the demonstrator robot has a high-complexity movement pattern which consists of a complex trajectory with 4 turns. Once the experiments are completed, the quality of imitation is assessed by an external program which is used for the offline analysis of the experiments.

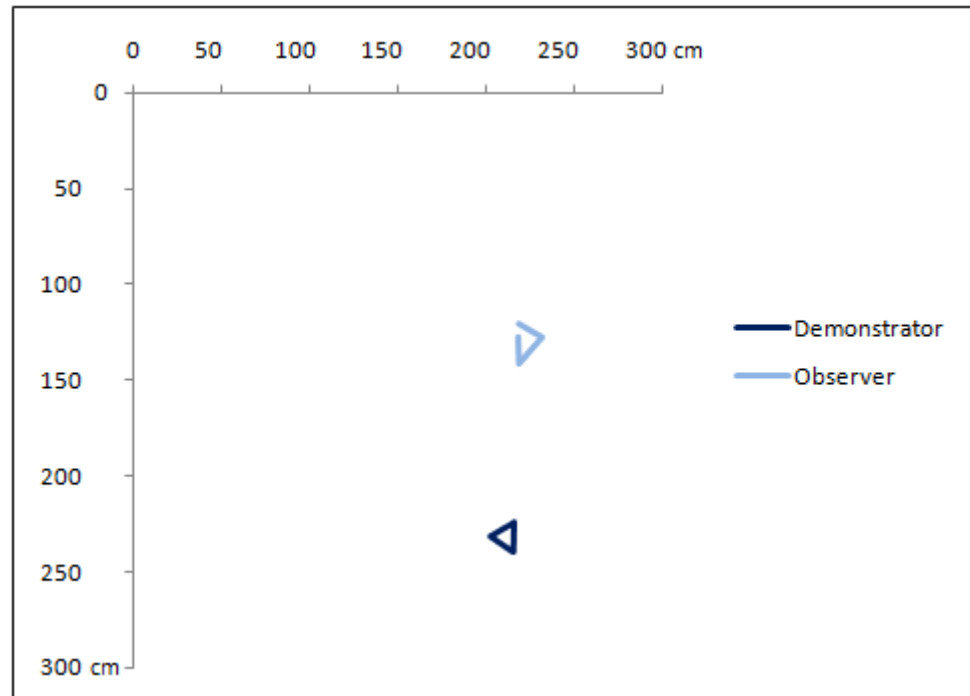


Figure 3.8: Plot of the trajectory of robots during an imitation run. The teacher robot moved in an equilateral triangular trajectory which was then copied by the imitator robot.

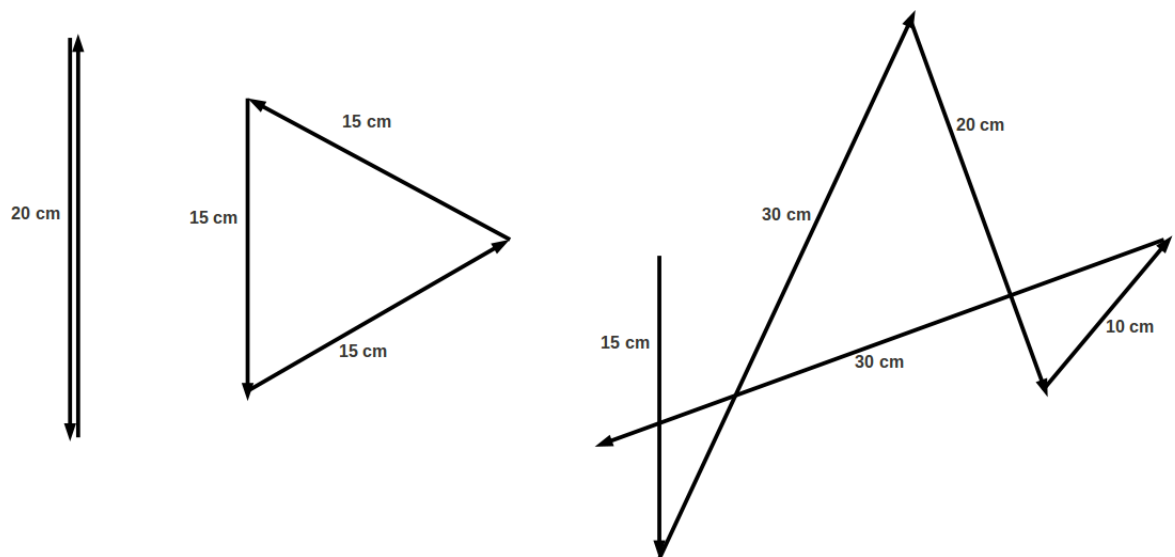


Figure 3.9: Movement patterns that the demonstrator robot followed during experiments. First movement pattern is a low-complexity pattern which consists of two moves of length 20 cm and a 180° turn between them. The second pattern is a medium-complexity pattern which consists of an equilateral triangular trajectory. The third is a high-complexity pattern which consists of a complex trajectory with 4 turns.

Figure 3.10 to 3.12 shows results for these experiments. As can be seen, for all 3 movement patterns, the highest quality is achieved when the distance between robots is 1 m. When the distance is increased (1.5 m or more), the mean quality of imitation starts to degrade. The reason is that the relative positional changes are calculated based on the size and the location of the demonstrator robot in the imitator robot's field of view. When the distance is high, the positional changes are harder to detect as they cause smaller variations in the perceived size of the demonstrator robot. Table A.1 shows the shortest displacement that can be detected at any distance. As a result, the second best mean quality achieved when the distance between robots is 1.5 m (figure 3.10 to 3.12). The lowest mean quality of imitation is observed when the distance between robots is 0.5 m. This is due to the fact that when the distance between robots is low (i.e. 0.5 m or less), the demonstrator robot leaves the field of view of the imitator robot many times, which forces the imitator robot to rotate itself each time. The imitator robot may then miss some moves and especially turns of the demonstrator robot while it is itself busy turning. This effect can be clearly seen when the observed robot has a high-complexity trajectory. As many of the turns are missed, the mean quality of imitation is very low when the distance is 0.5 m, compared to other cases. For imitation of low-complexity and medium-complexity patterns, the highest standard deviations are seen when the distance between robots is 0.5 m, which is a result of some low quality imitations.

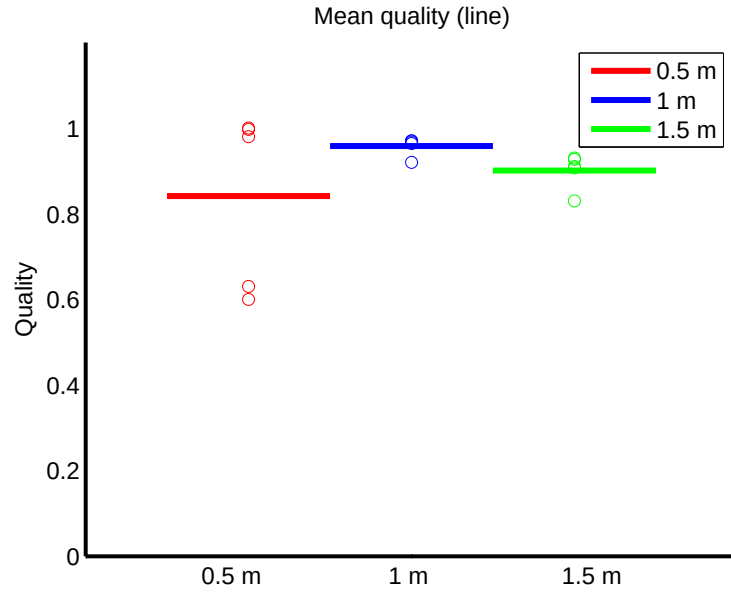


Figure 3.10: Mean quality value calculated at different distances between robots. The demonstrator robot has a low-complexity movement pattern which consists of a line. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

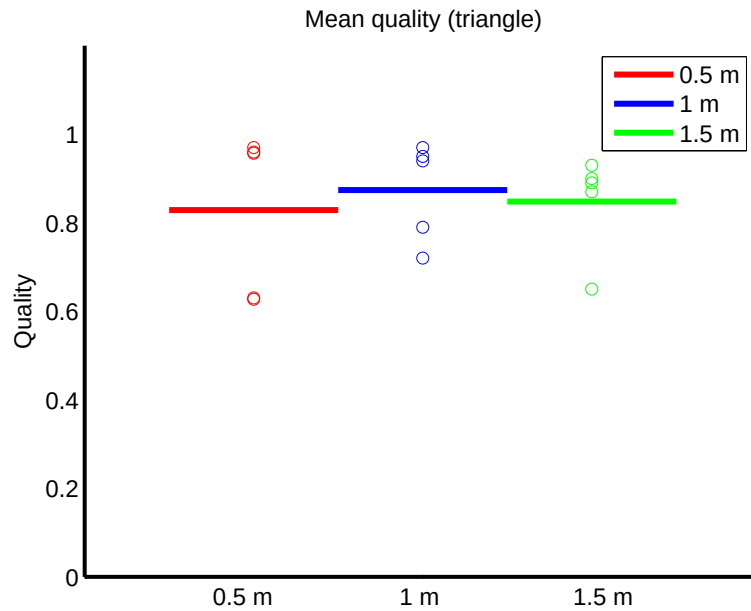


Figure 3.11: Mean quality value calculated at different distances between robots. The demonstrator robot has a medium-complexity movement pattern which consists of an equilateral triangle. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

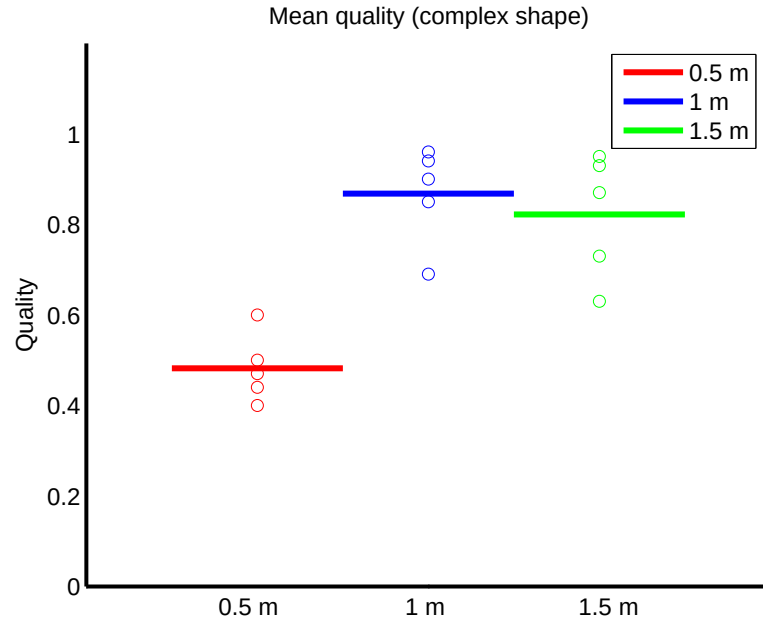


Figure 3.12: Mean quality value calculated at different distances between robots. The demonstrator robot has a high-complexity movement pattern which consists of complex trajectory. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

Figure 3.13 to 3.15 compares imitation of movement patterns with different complexity at each distance. As can be seen, the mean quality of imitating a low-complexity pattern is slightly higher than other two cases at all distances. However, a pair-wise t-test reveals that the differences between low-complexity and medium-complexity patterns are not statistically significant. The only statistically significant difference is between low-complexity and high-complexity when the distance between robots is 0.5 m and between medium-complexity and high-complexity when the distance between robots is 0.5 m.

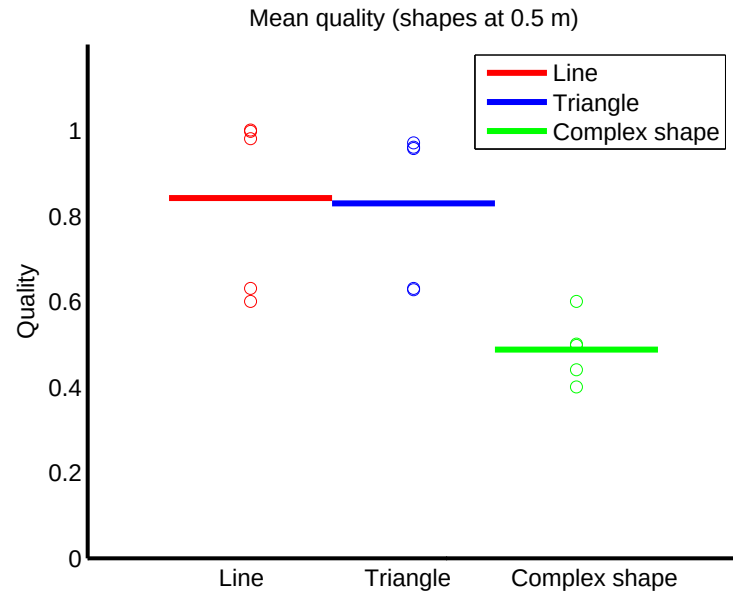


Figure 3.13: Mean quality value calculated for different movement patterns when the distance between robots is 0.5 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

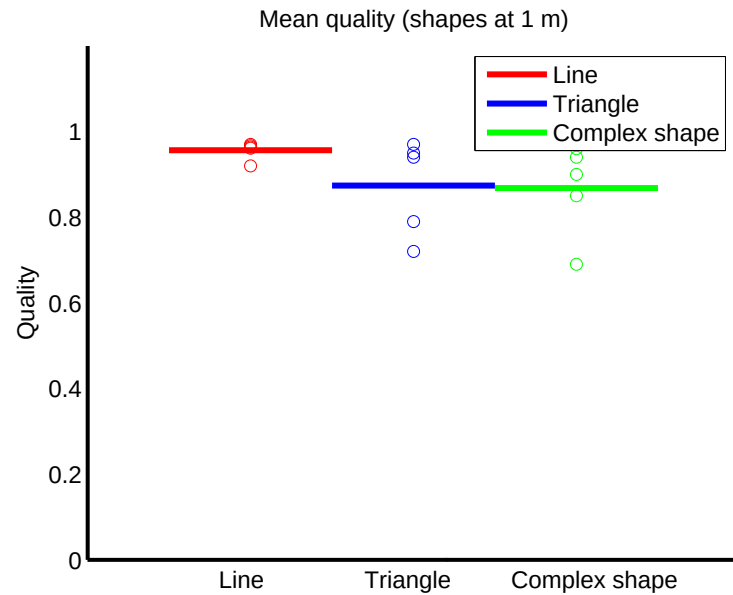


Figure 3.14: Mean quality value calculated for different movement patterns when the distance between robots is 1 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

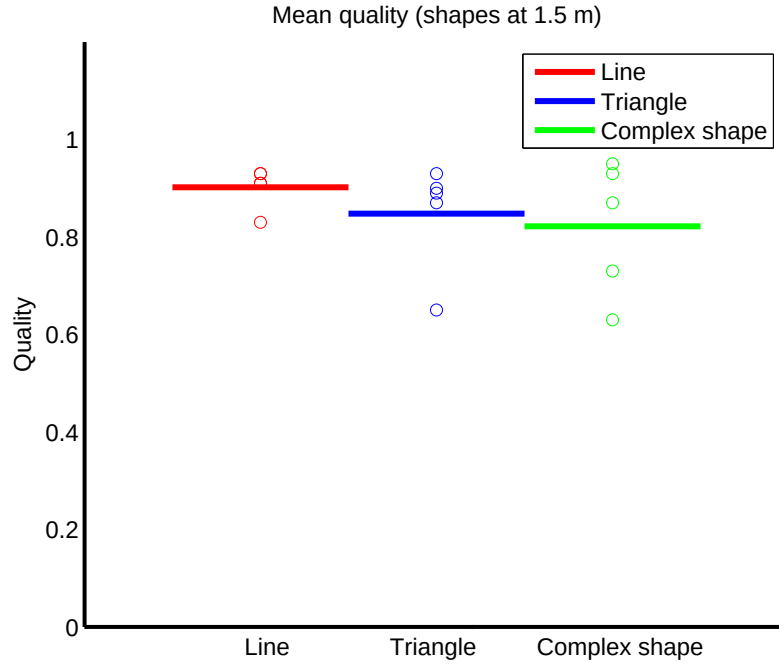


Figure 3.15: Mean quality value calculated for different movement patterns when the distance between robots is 1.5 m. The quality of each imitation is shown by a circle and the horizontal line shows the mean over 5 imitations. Each quality indicator was given equal weight: $L = A = S = 1$.

As can be seen in figure 3.11, when the demonstrator robot has a medium-complexity movement pattern, the mean quality of imitation is similar at all tested distances, compared to the imitations of low-complexity and high-complexity movement patterns. Therefore, imitation of medium-complexity movement patterns are relatively less dependant on the distance between robots. Furthermore, when a low-complexity pattern is copied, although sensor errors may occur, its copy is typically another low-complexity pattern. When a hand-sketched high-complexity movement pattern is demonstrated, the copy typically has a lower complexity. However, when a medium-complexity pattern is copied, we see that it may result in patterns with different levels of complexity at all distances. For these reasons, in the next chapter, in which the adaptation of movement patterns during multiple cycles of imitation is investigated, robots are initialised with medium-complexity patterns. As stated above, when a hand-sketched high-complexity movement pattern is demonstrated, the copy typically has a lower complexity. Therefore the results should generalize if the robots are initialised with more complex patterns. When the distance between

robots is 1 m, the quality of imitation was relatively higher and the difference in the quality of imitation for imitation of patterns with different complexity is smaller. For these reasons, the distance between robots is set to 1 m in the next chapter.

A copy which has $Q_i \geq 0.85$ is defined as a *high quality* copy. This value is determined as follows: In order to accept that a copy is of high quality, it should have a relatively high Q_i value and all of its 3 quality indicators should be higher than 0.5. To guarantee that the second condition is always true, the Q_i value should be higher than 83.33. By examining the results that are observed during the experiments presented in this section, a slightly higher value, 0.85 is selected as the threshold that defines a high quality copy.

3.5 Summary and Conclusion

In this chapter, the experimental hardware and infrastructure used for the work of this thesis was presented. The original e-puck robot, which lacks the computational power and memory requirements that are needed for the experiments that involve imitation, was introduced and a linux extension board to enhance the e-puck robot was outlined. Then, the infrastructure of the experiments, including the robot arena, linux extension board enhanced robots, the tracking system and the dedicated swarm server, was presented. The logging structure that allows offline analysis of imitation experiments was explained. Following this, an embodied movement imitation algorithm, that depends completely on the visual data coming from the image sensor of the robots, was implemented. A quality of imitation function was defined which determines the similarity between a demonstrated movement pattern and its perceived copy. Using this formula, the algorithm was tested in some experiments in which a demonstrator robot followed some movement patterns with different complexities while an imitator robot watched and copied it.

There are some limitations of the algorithm that could be further explored. The algorithm is vision-based so it needs good lighting over the arena. It has a working range in which if the distance between robots is less than 20 cm or more than 2 m, it does not function properly. The algorithm can only detect movement patterns that consist of straight line movements and turns. Also, it can not detect turns but it infers them from the straight line movements to different directions.

At this point, it is shown that variations that arise from embodiment, cause copied movement patterns to differ from their originals and hence reduce the quality of imitation. The quality of

imitation depends on both the complexity of the movement pattern that is copied and external factors such as the distance between the demonstrator and the imitator robots. In the next chapter, the effects of these variations on the emergence of shared behavioural patterns during multiple cycles of imitation is investigated.

Chapter 4

Emergence of Structure in Behaviours Adapted Through Embodied Imitation

In the previous chapter, it was shown that variations that arise from embodiment cause copied movement patterns to differ from their originals. This chapter examines the effects of these variations on the structure of copied movement patterns during multiple cycles of imitation in which a group of robots observe and imitate each other's movement trajectories. It is shown that movement patterns, that are more robust to noise and uncertainties in the robot's sensors, emerge and these adapted movement patterns can be copied with higher fidelity by the group members.

4.1 Introduction

As stated in chapter 2, imitation in robotics research holds the promise that we may be able to overcome the necessity to program every behaviour a robot may need to perform, as the robot can learn new behaviours through observing demonstrations of those behaviours. In addition to supporting skill transmission between individuals, imitation has a social dimension as it allows individuals to become part of a social community. As Alissandrakis et al. (Alissandrakis et al., 2004) stated, imitation may serve as a stepping stone towards the development of social cognition in artificial agents as it can form social integration with other artificial agents or with humans. Steels and Kaplan (Steels and Kaplan, 2001) stated that social learning can play a crucial role in initiating a humanoid robot into a linguistic culture. They used methods such as initiating open-ended dialogues among humans and robotic agents, in which social learning could be embedded. Billard (Billard, 1999) claimed that imitation can be used to enhance autonomous robots' learning of communication skills. The sharing of a similar perceptual context between the imitator

and demonstrator can create the necessary social context in which language can develop. Bilard devised some experiments in which robotic agents were able to learn a proto-language by using imitation to match their environmental perceptions with observed actions. In this chapter, it is shown that, as the robots share a similar perceptual context, the structure of the imitated behaviours changes so that these behaviours can be copied with high fidelity. Multiple robots are programmed to observe and imitate each other's movement patterns and the imitated behaviours undergo multiple cycles of copying, in which they mutate because of noise and uncertainties in the real robots' sensors and actuators.

In other research related to the work of this chapter, Alisandrakis et al. (Alissandrakis et al., 2004) developed the ALICE architecture (Action Learning via Imitation between Corresponding Embodiments) to address the problem of imitation between dissimilar embodiments. They examined the rules of synchronisation, looseness of perceptual matching and proprioceptive matching in a series of experiments in which robotic arms with variably-sized and numbered joints tried to imitate each other. They showed that patterns can be transmitted between simulated robotic arms and variations occur during imitation because of heterogeneities between the arms. They propose that these variations might provide the evolutionary substrate for culture, as new behavioural patterns may emerge and be transferred between agents. In this chapter, a series of experiments are presented in which real robots observe and imitate each other's movement patterns. It is shown that even with an homogeneous group of real robots, variations occur during the imitation process that allow certain behavioural patterns to emerge during multiple cycles of imitation. These adapted behaviours can be copied with higher fidelity, as they are more robust to uncertainties in the real robots' sensors and actuators.

4.2 Experimental Setup

As stated above, we are interested in the effects of variations on the structure of the imitated movement patterns. For that purpose, an experimental setup in which 4 robots copy each other's movement patterns is introduced. The robots are placed 1 m apart from each other on the arena, as shown in figure 4.1. They interact by copying each others' movement patterns using the imitation algorithm explained in section 3.3. Figure 4.2 shows the finite state machine (FSM) for the controller of the robots. Each robot runs the same FSM. Robots can be in one of two modes during the experiments: demonstrator or observer. When a robot enters demonstrator mode, it turns its

LEDs on for 35 seconds¹ to signal that it will start to demonstrate a movement pattern. During this period the demonstrator signals to get the attention of the other robots. After that, the demonstrator robot turns its LEDs off and executes a movement pattern that consists of straight-line moves and turns. When execution is complete, the demonstrator robot blinks its LEDs for 1.6 seconds² to signal ‘finish’. Then the demonstrator robot returns to its original start position and enters the observer mode. When a robot enters observer mode, it searches for a start signal by scanning the arena while rotating itself. When it detects a start signal, it waits for the demonstration to start. After completion of the demonstration, the observer robot records what it has observed and enters demonstrator mode. To setup starting conditions for the experiments, two of the robots are programmed to start in demonstrator mode while the other two are set to start in observer mode. Then the experiment is left free-running as the robots change roles while imitating each other. The pseudo-code for the controller of the robots is given in algorithm 3. Since the robots need to have a starting movement trajectory in their memory to be able to start the experiments, they are initialised with medium-complexity movement patterns³: two of them with an equilateral triangular trajectory, two of them with a square trajectory.

As we are interested in the adaptation of movement patterns during multiple cycles of imitation, regardless of their utility or context, no fitness value is attached to the movement patterns. Thus we choose the simplest selection strategy as follows: each time a robot demonstrates a movement pattern, it randomly selects one of the trajectories in its memory⁴, with equal probability. Once it is selected, the trajectory is converted to motor commands that can be executed by the demonstrator robot, which corresponds to the *enact* step, as defined in section 3.1 (figure 4.3).

¹35 seconds is the approximate time needed for an observer robot to complete two complete scans of the arena, in search for a demonstrator robot.

²In order to discriminate between a *start* and a *finish* signal, a finish signal is shorter than a start signal. A finish signal takes 1.6 seconds while a start signal should be observed for at least 2 seconds.

³The rationale in choosing medium-complexity movement patterns is explained in section 3.4.

⁴The different memory structures are explained later in this section.

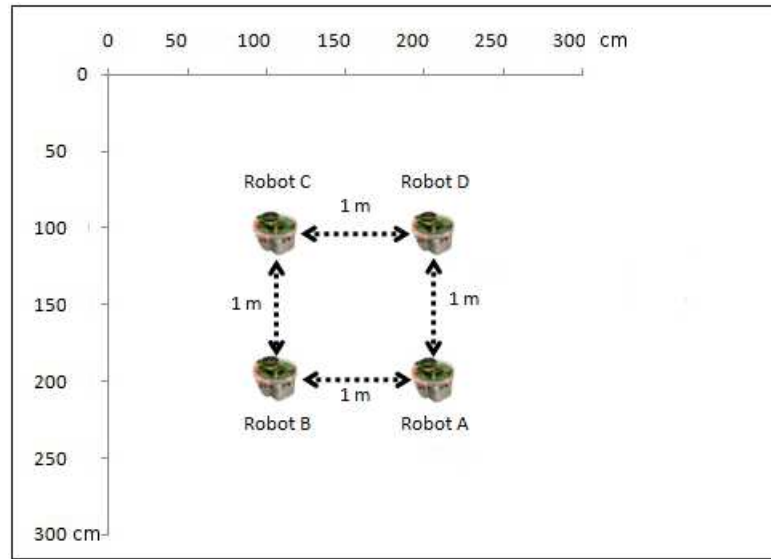


Figure 4.1: Each experiment presented in this chapter is performed in a 3 m by 3 m arena with 4 robots, placed 1 m apart and arranged as shown here.

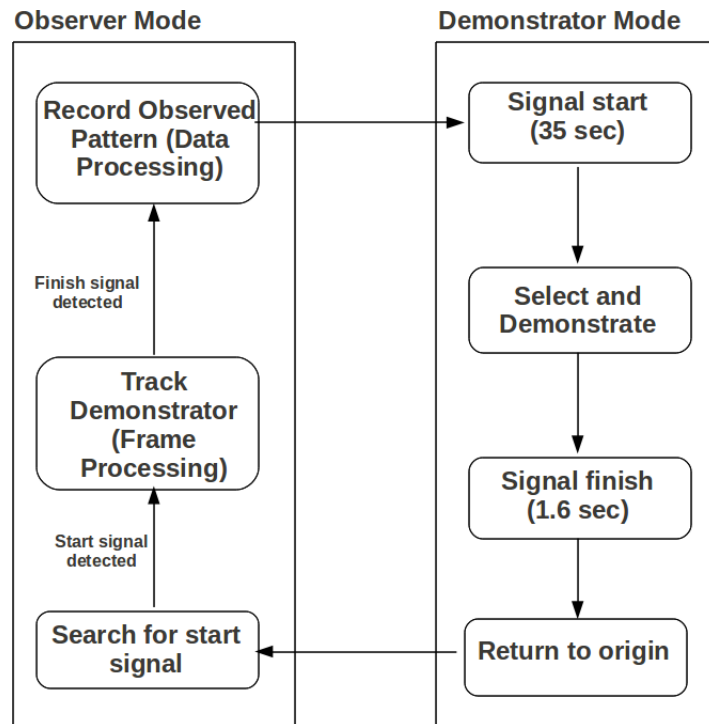


Figure 4.2: Finite state machine of the controller of the robots. Once started the FSM loops indefinitely, alternating between demonstrator and observer modes. The two phases of the imitation algorithm take place in corresponding states as shown in the FSM.

Algorithm 3 Pseudocode for robots' controller

Input: $PositionList \leftarrow \emptyset, Rob_{Observed} \leftarrow \emptyset, MoveList \leftarrow \emptyset, f \leftarrow \emptyset$ $RobState \leftarrow searchForStartSignal$ **while** *experiment continues* **do** **if** $RobState = searchForStartSignal$ **then** $GetRobotList()$ **for each** $rob \in RobotList$ **do** **if** $RobotSignalOn(rob) = \text{true}$ **then** $Rob_{Observed} \leftarrow rob$ $RobState \leftarrow waitForDemonstration$ **end if** **end for** **else if** $RobState = waitForDemonstration$ **then** $GetRobotList()$ $Rob_{Observed} \leftarrow FindObservedRobot(Rob_{Observed})$ **if** $RobotSignalOff(Rob_{Observed}) = \text{true}$ **then** $RobState \leftarrow trackDemonstrator$ **end if** **else if** $RobState = trackDemonstrator$ **then** $GetRobotList()$ $Rob_{Observed} \leftarrow FindObservedRobot(Rob_{Observed})$ **if** $RobotSignalOff(Rob_{Observed}) = \text{true}$ **then** $FrameProcess(Rob_{Observed})$ **else** $RobState \leftarrow recordObservedPattern$ **end if** **else if** $RobState = recordObservedPattern$ **then** $DataProcess(PositionList)$ $SaveNewPattern()$ $PositionList \leftarrow \emptyset, Rob_{Observed} \leftarrow \emptyset, MoveList \leftarrow \emptyset$ $RobState \leftarrow signalStart$ **else if** $RobState = signalStart$ **then** $SignalFor(35)$ $RobState \leftarrow demonstrate$ **else if** $RobState = demonstrate$ **then** $selectedPattern \leftarrow SelectMovementPatternFromMemory()$ $DemonstratePattern(selectedPattern)$ $SignalFor(1.6)$ $RobState \leftarrow returnToOrigin$ **else if** $RobState = returnToOrigin$ **then** $ReturnOriginalPosition()$ $RobState \leftarrow searchForStartSignal$ **end if****end while****function** $GetRobotList()$ $f = GetImageFrame()$ $BlobList = GetBlobs(f)$ $RobotList = GetRobots(BlobList)$ **end function**

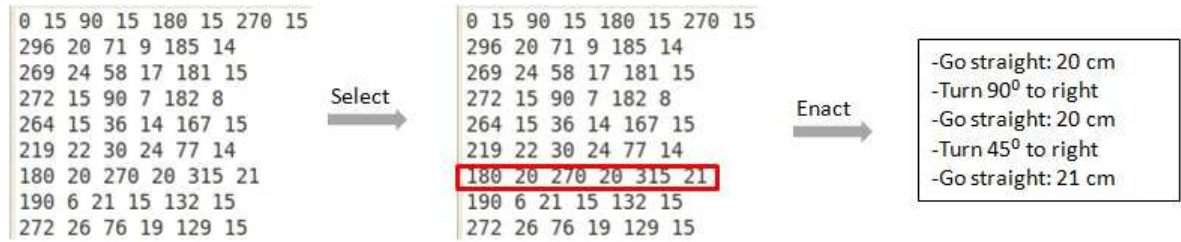


Figure 4.3: The demonstrator robot randomly selects one of the trajectories in its memory and then the selected trajectory is converted into a set of motor commands that can be executed by the demonstrator robot (*enact*).

In order to test the effect of memory on the structure of adapted behaviours, we tested three cases. First case is *no memory*, in which the robot remembers only the most recent observation so that any movement pattern that is previously observed is replaced by the new observation. Second case is *unlimited memory*, in which the robot remembers all its past observations as any newly observed pattern is saved to its memory. In the third case, the robot has a *limited memory* so that it can store only a fixed number of movement patterns to its memory. Once its memory is full and a new pattern is copied, the new observation replaces the oldest movement pattern. All three cases of memory structures are tested while the adaptation of movement patterns are examined in experiments that will be presented in the next section.

Robots keep an internal log file that records all state transitions and calculations made by the robot. A sample internal log file is shown in figure A.2. In addition, robots broadcast their internal states over the IP network while the tracking system broadcasts the position of robots in the arena. An external logging program running on the swarm server (figure 3.3) combines state information from the robots and position information from the tracking system to create an external log file. An example external log file is shown in figure A.3. These log files are used for the later analysis of the experiments.

4.3 Experiments

Three cases with different sizes of robot memories are experimentally tested in order to examine the effects of memory size on the structure of emerged behaviours. For each case, a sample run is presented in detail and the generalisation of results is discussed with multiple experiment runs.

4.3.1 Imitation with No Memory

In the first set of experiments, the robots are able to remember only the most recent movement pattern they have observed; any newly-observed pattern replaces the previous one. Figure 4.4 shows the plot of trajectories of the robots during one particular run of experiments with these settings. Figure 4.5 shows the pattern progress map of this experiment run. After the experiment is completed, an external program calculates the quality of imitation for each copied movement pattern and it creates pattern progress map of the experiment. In the figure, each node represents a pattern. If an arrow exists at a node, this means one of the robots executed that pattern and it was copied by another robot. The new copy is at the end of the arrow. If the copy is high-quality, ($Q_i \geq 0.85$) then the node has a red colour⁵.

We see that the original patterns deteriorate very quickly. At the beginning of the run, the robots that started in observer mode, by chance, copied the square trajectory and the triangular trajectory vanished from the experiment. The square trajectory also deteriorated rapidly, as any low quality copy can easily replace it. In this run, some bad copies, in which the observer robot missed some turns, caused the robots to eventually end up with a low-complexity movement pattern consisting of a single forward move. These low-quality copies do not occur very often but with the no-memory setting, one of them is sufficient to disrupt the progress of movement patterns. As explained in chapter 3, these low-complexity patterns can be copied with high quality, though we still observe some bad quality copies. In this experiment run, all patterns after pattern number 22 are low-complexity patterns with only one forward move without turns. All runs with these setting have the same characteristics. As variations that arise during the imitation directly affect the progress of the movement patterns, their adaptation is highly sensitive to errors. In all runs, the original movement patterns deteriorated very quickly and most runs ended up with a low-complexity movement pattern after some imitation cycles.

⁵The rationale in choosing this value for defining high-quality copies is explained in section 3.4.

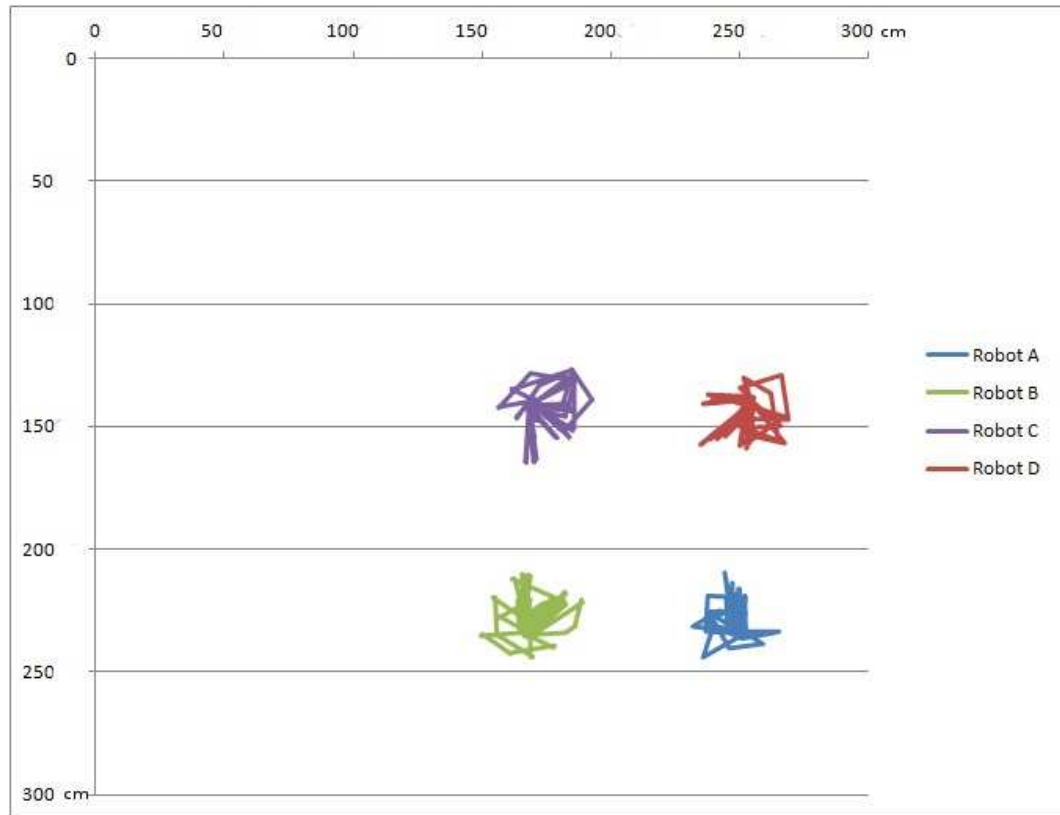


Figure 4.4: Plot of trajectories of each robot during the experiment with no memory.

4.3.2 Imitation with Unlimited Memory

In the second set of experiments robots have unlimited memory so they save all patterns that they have observed. As explained in section 4.2, when they enter demonstrator mode robots randomly select, with equal probability, one of the patterns in their memory and demonstrate it. Figure 4.6 shows the pattern progress map of one particular run with these settings. It is clear that, compared to the experiments with no memory setting, as each newly observed pattern is stored in memory, the original movement patterns are more likely to be preserved with variation. Low quality copies occasionally occur but as they do not replace previously observed patterns, they do not easily alter the adaptation of the movement patterns. Second, we see that as patterns change during multiple cycles of imitation, some patterns that are able to be copied with high quality emerge and propagate among robots. In this run, pattern 27 has this property. Figure 4.7 shows the adaptation of pattern 27. In this experiment, Robot A watched Robot C enact pattern 1 (the original equilateral triangle) and attempted to learn the movement sequence; the result is pattern 5. Then Robot D watched

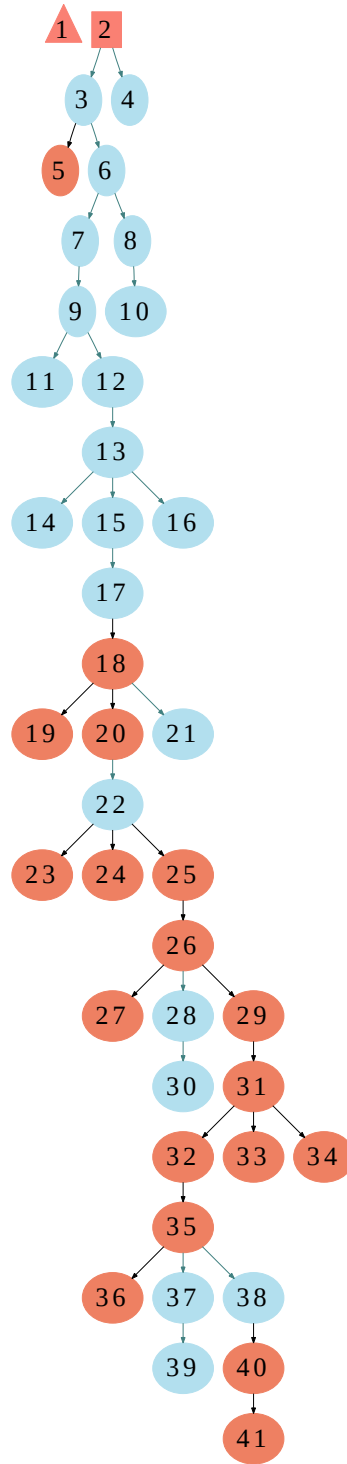


Figure 4.5: Pattern progress map for a 4 robot experiment with no memory. Each node in the figure represents the demonstration of a movement pattern. If a pattern is demonstrated and imitated, the new copy of that pattern is linked to it by an arrow. For instance, pattern 2, the original square, was demonstrated by Robot A and was copied by two robots. The new (child) copies of pattern 2 are patterns 3 and 4. If the copy is of high quality (i.e. $Q_i \geq 0.85$), then the node has a red colour. Initial movement patterns are a triangle (1) and a square (2). The nodes are numbered according to the time that they were copied: if $x < y$, pattern x was copied before pattern y .

pattern 3, enacted by robot A, and attempted to learn it; thus pattern 11 is an imitation of pattern 5. Pattern 27 is a descendant of the original equilateral triangle trajectory, and there are 5 intermediate copies between the original triangle and pattern 27: pattern 1 $\rightarrow$ pattern 5⁶ $\rightarrow$ pattern 11 $\rightarrow$ pattern 18 $\rightarrow$ pattern 20 $\rightarrow$ pattern 26 $\rightarrow$ pattern 27. As can be seen, pattern 1 $\rightarrow$ pattern 5, pattern 5 $\rightarrow$ pattern 11 and pattern 11 $\rightarrow$ pattern 18 are high fidelity imitations while pattern 18 $\rightarrow$ pattern 20, pattern 20 $\rightarrow$ pattern 26 and pattern 26 $\rightarrow$ pattern 27 are low fidelity imitations. Finally pattern 27 emerges and a sharp increase in quality of imitation can be observed after this point ($Q_i > 0.94$ for all of its descendants).

What makes this pattern and its descendants easily copiable? First, short moves are more prone to error, as a small mistake in perception can cause them to vanish; a pattern that can be copied with high quality typically does not include short moves. Second, the length of each move varies at each subsequent copy. Although estimating the relative size and position of the demonstrator robot is straightforward image processing, it is error-prone, because of the relatively low resolution of each robot's image sensor. A move directed towards or away from the observing robot can only be detected if it causes a perceptible change in the size of the demonstrator robot, i.e. a detectable change in number of pixels in the image of the demonstrator. At each copy, the observing robot stores what it infers from the demonstration, as perceived from its relative position and perspective. Therefore, the patterns tend to change into ones that can be more easily imitated. Figure 4.8 shows pattern 27 and its descendants. As can be seen, there is a high level of similarity between these patterns. At the end of the run, pattern 27 and its descendants form a cluster of similarly-shaped patterns in the robots' memories. A cluster is defined as a group of movement patterns, with 4 or more members, that are related to each other by a series of high quality imitations (i.e. $Q_i \geq 0.85$). Figure 4.9 shows the average Q_i value for this experiment in comparison with the average Q_i value for the cluster formed by pattern 27's cluster. As can be seen, a sharp increase in Q_i value can be observed after a pattern emerges that is more robust to uncertainties in the robot's sensors and the imitation process: the average Q_i value for the cluster that is formed by the descendants of pattern 27 is around 0.96, while the average quality of imitation for this experiment run is around 0.82.

⁶We use notation $A \rightarrow B$ as shorthand for B is a learned copy of A.

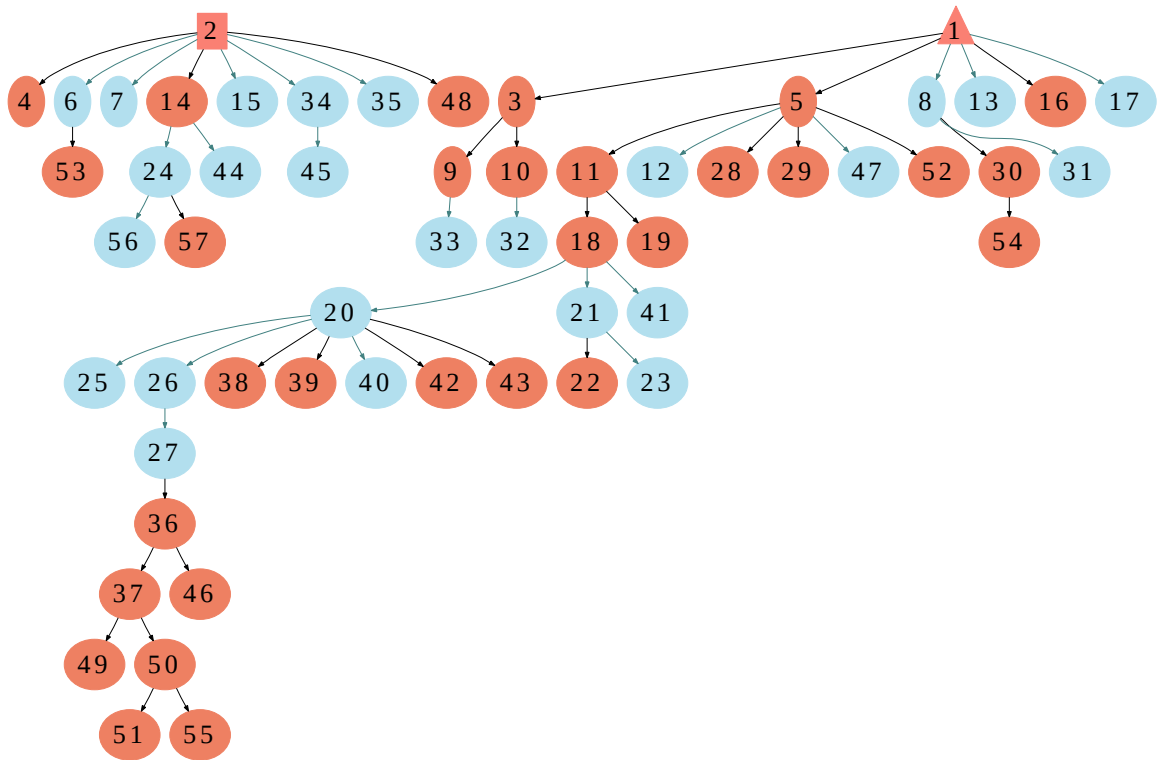


Figure 4.6: Pattern progress map for a 4 robot experiment with unlimited memory. Initial movement patterns are a triangle (1) and a square (2).

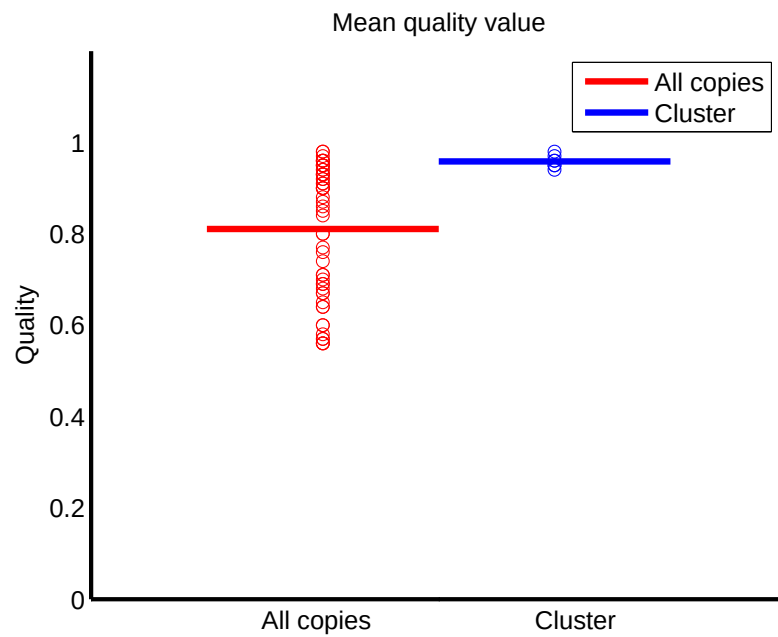


Figure 4.9: Mean quality value calculated for all imitations in the experiment shown in Figure 4.6 (All copies) and mean quality value for the cluster formed by pattern 27 (Cluster). The quality of each imitation is shown by a circle and the horizontal line shows the mean.

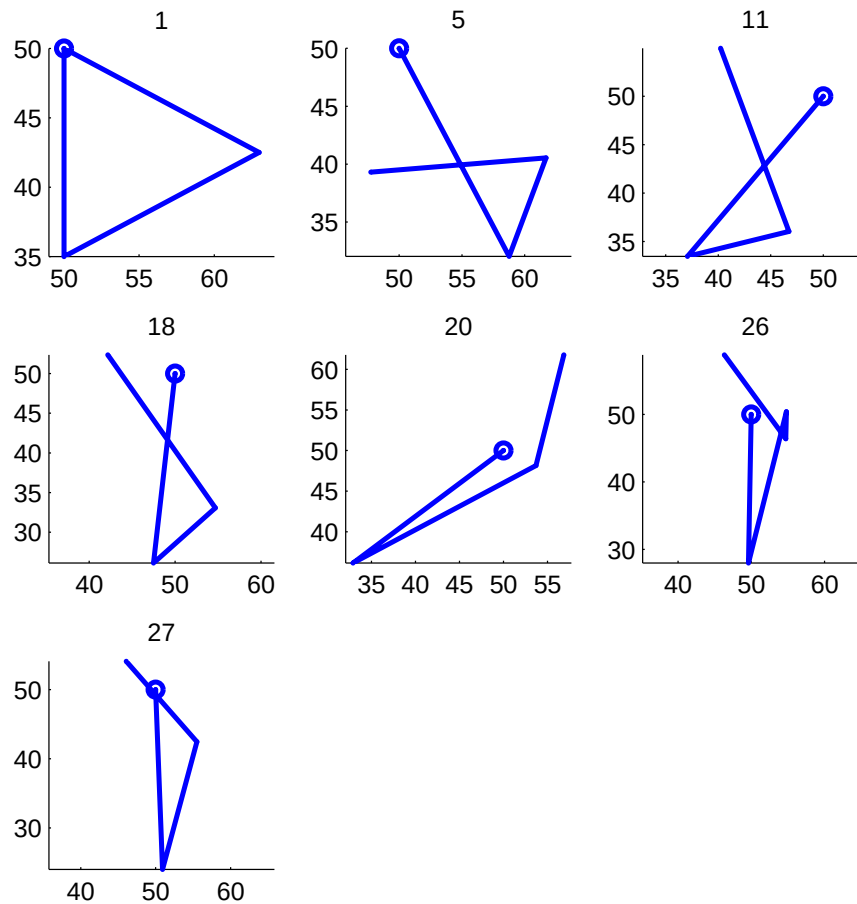


Figure 4.7: Adaptation of pattern 27 in Figure 4.6. Pattern 27 is a descendant of the original equilateral triangle pattern. By following the imitation links on the pattern progress map for this experiment, we can see that there are 5 intermediate copies between the original triangle and pattern 27: the patterns numbered 5, 11, 18, 20, 26. All of these patterns, starting with the original triangle and ending with pattern 27, are shown here in order. All axes are marked in cm. Beginning of each movement pattern is marked with a circle.

Table 4.1 shows results from 10 experiment runs with the unlimited memory setting. The table is created by determining the clusters of movement patterns that are related to each other by high quality imitations. As can be seen, in all runs, such clusters of highly similar movement patterns emerged in the memory of the robots. The average similarity between the members of these clusters is very high, 0.927. Note that members of these clusters may have low quality copies. For instance, in figure 4.6, patterns 20, 38, 39, 42 and 43 forms a cluster of size 5. Patterns 25, 26 and 40 are copies of pattern 20 but since they are low quality copies, they are not members of this

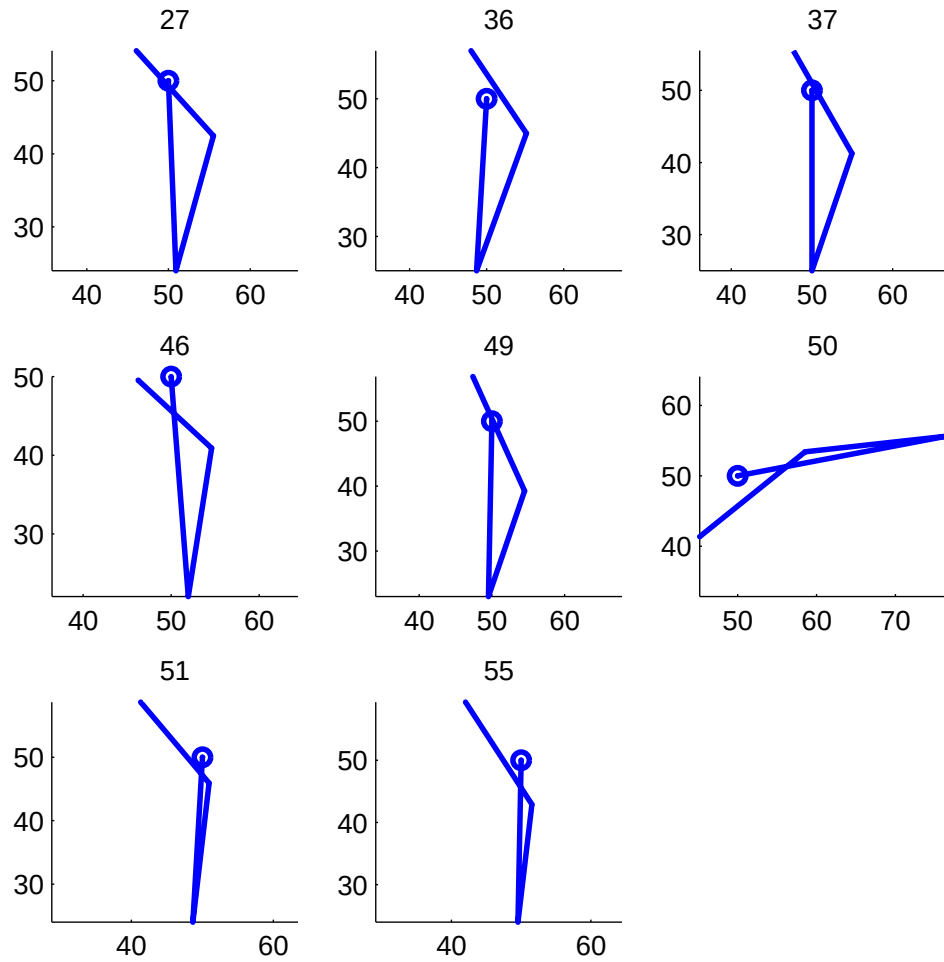


Figure 4.8: The descendants of pattern 27 in Figure 4.6. Starting with pattern 27, its descendants (patterns 27, 36, 37, 46, 49, 50, 51, 55) are shown in order. All axes are marked in cm.

cluster. Column 4 in table 4.1 shows the average quality of imitation for the members of clusters while column 5 shows the average quality of imitation for all copies of each run. A pair-wise t-test reveals that considering all runs, there is a statistically significant difference between these two values. these results suggest that the emergent movement patterns that constitute the clusters are more robust to the embodied imitation.

Number of clusters	Average size of clusters	Average similarity between the members of the clusters	Average quality of imitation for the members of the clusters	Average quality of imitation for all copies
4	5.25	0.94	0.85	0.81
4	6	0.94	0.86	0.8
8	4.88	0.93	0.83	0.75
2	7	0.93	0.85	0.72
4	9.5	0.92	0.82	0.78
4	6.25	0.93	0.82	0.78
5	6	0.92	0.82	0.8
3	5.33	0.93	0.79	0.72
2	12	0.92	0.81	0.79
3	6.6	0.91	0.78	0.73
3.9	6.88	0.927	0.823	0.768

Table 4.1: Results for 10 experimental runs with unlimited memory.

4.3.3 Imitation with Limited Memory

In the previous set of experiments it was shown that certain patterns, those that appear to be more robust to uncertainties in the real robots' sensors and the estimation process of imitation, can emerge during multiple cycles of imitation. As these emergent patterns can be copied with high quality, their descendants have similar, inherited characteristics. As a result, clusters of highly copiable patterns are formed in the memories of the robots. These clusters may grow larger with subsequent cycles of imitation if, by chance, members of these clusters are selected for demonstration. In the third set of experiments, robots have a limited memory in which they can store only the most recent 5 patterns observed. When their memory is full and a new pattern is copied, the oldest pattern in their memory is replaced with the new one. Once again, a typical run with these settings is shown here in detail. Figure 4.10 shows the pattern progress map for a particular run with limited memory. In this experiment, Robot C watched Robot B enact pattern 1 (the original equilateral triangle) and attempted to learn it; the result is pattern 10. Then robot D watched pattern 10, enacted by Robot C, and attempted to learn it; thus pattern 12 is an imitation of pattern 10. As can be seen, pattern 1 $\rightarrow$ pattern 10 is a high fidelity imitation and pattern 10 $\rightarrow$ pattern 12 is a low fidelity imitation. Once the V-shaped pattern, pattern 12, emerged, there is a sharp increase in the quality of imitation as all descendants of pattern 12 are high quality imitations. Figure 4.11 shows the adaptation of this path and Figure 4.12 shows some of its high-quality descendants. At the end of this run, 12 of the 20 patterns in the memory of all 4 robots are

descendants of this pattern. Since the robots randomly choose which pattern to demonstrate, there is now a 60% chance that one of the descendants of pattern 12 will be demonstrated again. Once it is selected and copied, the new copy is itself likely to be a high quality copy, and so, similar to pattern 12. This process will then increase the percentage of patterns in the memory that are similar to pattern 12. Therefore, with a limited memory, the emergent patterns and high quality copies, that are adapted through multiple cycles of imitation, can become dominant in the robots' collective memory. Figure 4.13 shows the average Q_i value for this experiment in comparison with the average Q_i value for the cluster formed by pattern 12's descendants. As can be seen there is a sharp increase in the average Q_i value for the pattern 12 cluster.

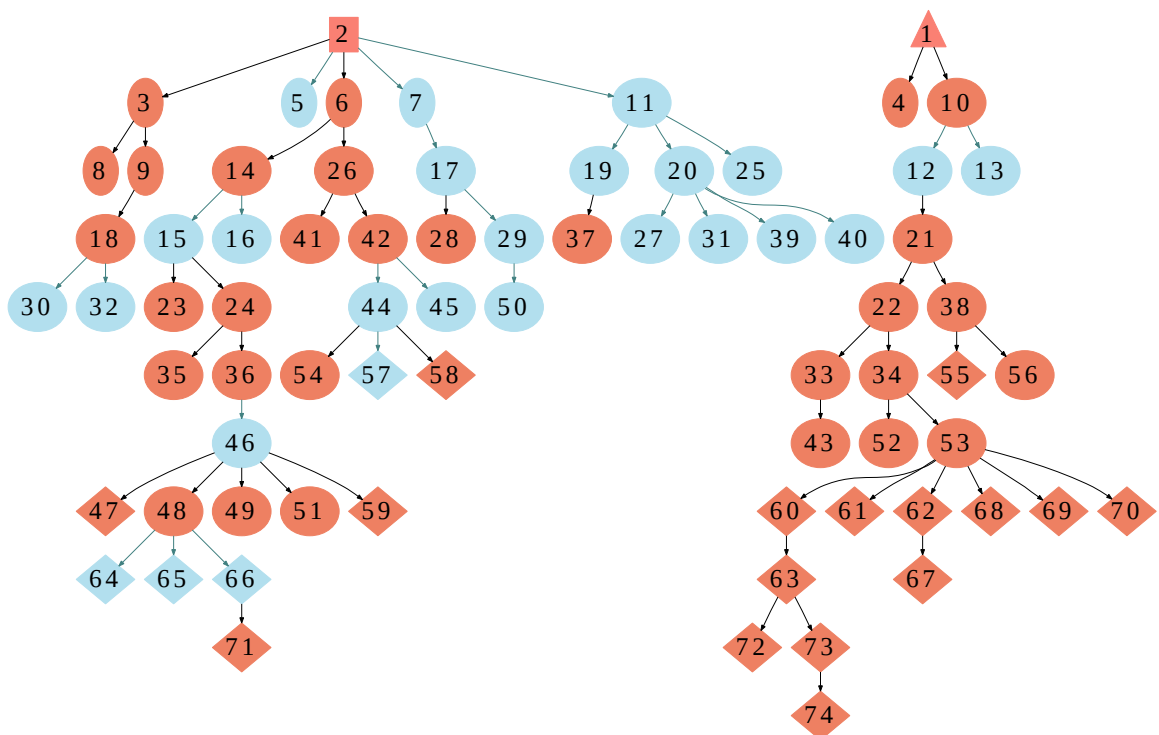


Figure 4.10: Pattern progress map for a 4-robot experiment with limited memory. Initial movement patterns are a triangle (1) and a square (2). The 20 patterns in the memory of all 4 robots at the end of the experiment are highlighted as diamonds.

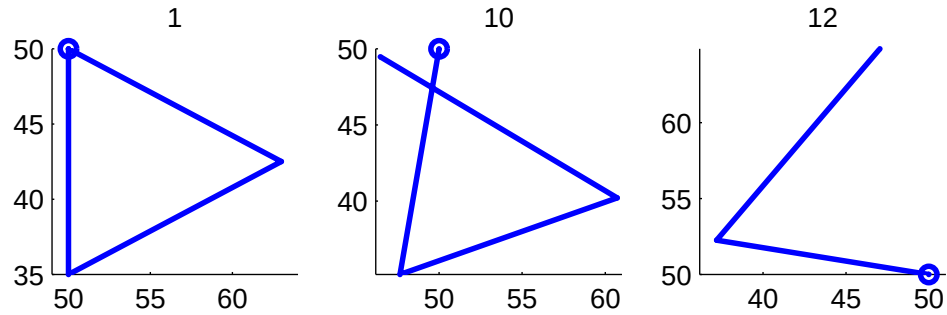


Figure 4.11: Adaptation of pattern 12 in Figure 4.10. There is an intermediate copy (10) between the original triangle and pattern 12. All axes are marked in cm.

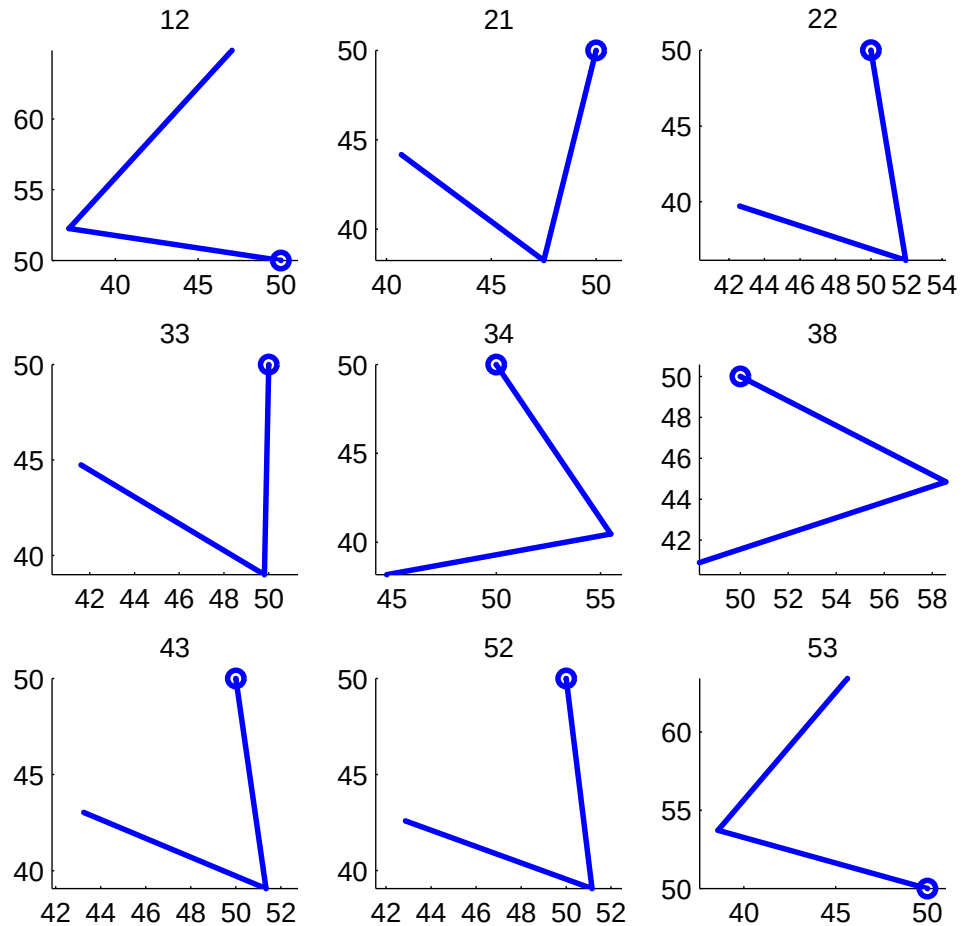


Figure 4.12: The descendants of pattern 12 in Figure 4.10. Starting with pattern 12, some of its high-quality copy descendants (patterns 12, 21, 22, 33, 34, 38, 43, 52, 53) are shown in order. All axes are marked in cm.

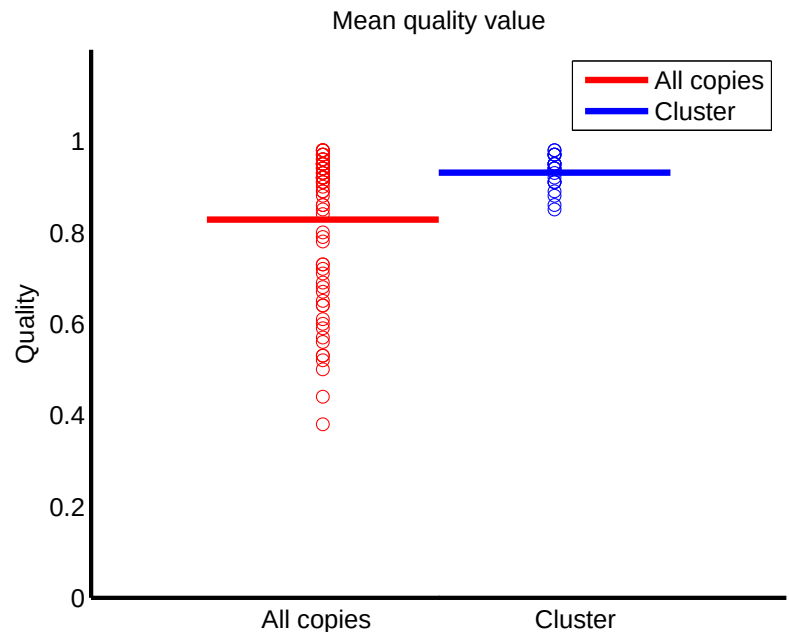


Figure 4.13: Mean quality value calculated for all imitations in the experiment shown in figure 4.10 (All copies) and mean quality value for the cluster formed by pattern 12 (Cluster). The quality of each imitation is shown by a circle and the horizontal line shows the mean.

Table 4.2 shows results from 10 experiments runs with limited memory setting. As can be seen, in all runs, clusters of highly similar movement patterns emerged in the memory of the robots. The average similarity between the members of these clusters is very high, 0.93. A pairwise t-test reveals that, considering all runs, there is a statistically significant difference between the average quality of imitation for the members of clusters (column 4) and the average quality of imitation for all copies (column 5). Therefore, as it was the case with unlimited memory, clusters of movement patterns appear more robust to uncertainties in the sensors of the robots, emerge during multiple cycles of imitation.

When we compare experimental results with unlimited memory and limited memory, as can be seen, in the limited memory case we see a smaller number of clusters (column 1 in tables 4.1 and 4.2) and the average size of clusters is larger (column 2). This is due to the fact that in some runs with limited memory, the collective memory of robots shows signs of being dominated by emergent patterns that formed the clusters. Each time a pattern that can be imitated with high fidelity emerged, in a sense, it starts to compete for the domination of the robots' memory. If the members of their clusters are, by chance, demonstrated, the new copies are likely to be similar,

thus increasing the size of their clusters. As a result of this process, we have fewer clusters but their size is larger in the limited memory case.

Number of clusters	Average size of clusters	Average similarity between the members of the clusters	Average quality of imitation for the members of the clusters	Average quality of imitation for all copies
5	6.4	0.94	0.86	0.82
4	5	0.93	0.81	0.77
4	6.75	0.92	0.81	0.78
1	20	0.92	0.84	0.73
3	6	0.93	0.81	0.74
2	5.5	0.92	0.85	0.77
3	6.6	0.91	0.81	0.76
3	5.6	0.92	0.77	0.72
1	14	0.96	0.92	0.73
2	7.5	0.95	0.81	0.75
2.8	8.33	0.93	0.829	0.757

Table 4.2: Results for 10 experimental runs with limited memory

4.4 Summary and Conclusion

In this chapter, real robots were used to model imitation between artificial agents. It was shown that variations in the real robots' sensors allow certain behaviours to emerge during multiple cycles of imitation. These adapted movement patterns appeared to be more robust to the uncertainties of the real robot embodied imitation process and so they could be imitated with higher fidelity. As the robots shared a similar perceptual context, the structure of the imitated behaviours adapted to the uncertainties in the robot's sensors so that these behaviours can be copied with high fidelity.

Three cases with different sizes of robot memories were experimentally tested: no memory, unlimited memory and limited memory, in order to examine the effects of memory capability on the adaptation of movement patterns. In the no memory case, the adaptation of movement patterns was extremely sensitive to any instance of poor quality imitation, which meant that the original movement patterns very quickly deteriorated. In the unlimited memory case, patterns emerged that could be easily copied but were less likely to then become dominant, as the number of patterns in the robots' collective memory grew larger with each new imitation cycle. However, in the case with limited memory, these adapted patterns could become dominant if they and their descendants were, by chance, chosen for demonstration. For simplicity of analysis, we had carried out our

limited memory experiments with a small memory size (5 patterns per robot). With a larger (but still limited) memory, multiple patterns that could be imitated with higher fidelity can emerge and form clusters of similarly-shaped patterns in the robots' collective memory. In this way, the robots can collectively adapt an ensemble of patterns that can be copied between them with higher fidelity.

Chapter 5

Imitation-enhanced Reinforcement Learning

Chapter 4 investigated the emerging structures in behaviours as they change over multiple cycles of imitation. This chapter examines the connection between imitation and individual learning, by presenting a way of using imitation to enhance the learning speed of individual agents¹. The algorithm is firstly tested in simulation with multiple simulated agents and it is then tested on real robots in the following chapter.

5.1 Introduction

In robotics research, a number of adaptive learning algorithms have been used in recent years including Reinforcement Learning (Barto et al., 2004), Evolutionary Algorithms (Nolfi and Floreano, 2000) and Supervised Learning (Rumelhart et al., 1986). These algorithms differ in the type of supervision they provide and they have different mechanisms for shaping the strategy of the learning agent. However, these learning approaches do not typically include social learning, which clearly plays an important role in social animals and humans. As stated in chapter 2, the ability to learn from others is a powerful mechanism as it allows certain skills to be transferred from one group member to others. *Imitation* is one example of social learning in which an individual observes and copies another's actions. As stated by Bakker and Kuniyoshi (Bakker and Kuniyoshi, 1996), imitation can be used as an enhancing factor with existing learning strategies such as reinforcement learning. The objective of this chapter is to present a way of using imitation as a way of enhancing the learning speed of individual agents in a multi-agent group. The

¹The term 'agent' is used when simulated agents perform experiments instead of real robots.

agents employ a well known reinforcement learning algorithm, namely *Q-learning*, and imitation provides enhanced learning.

An important aspect of imitation is that the only information that is transferred between agents is the set of observed actions. In accordance with our embodied imitation approach in chapter 3 and 4, an agent imitating another may not know anything about the internal state and structure of the other agent. Similarly, the imitating agent only observes the actions executed by the other agent and that information is free from its own internal state and structure. Therefore imitation is different from other types of learning in that supervision can not be used to directly influence internal processes. This chapter presents a simple method for linking reinforcement learning with imitation.

As stated in chapter 2, the problem of matching the actuators of the observed robot to the robot's own actuators is presented as the correspondence problem (Nehaniv and Dautenhahn, 2002). A solution to the correspondence problem for robot to robot imitation, for our experimental context, is presented in this chapter in the following way: The problem can be divided into two separate issues, (1) how to replicate the observed actions and (2) how to find an appropriate context (or states) for which those actions are meaningful. The first problem is solved by programming intrinsically, as developed in chapter 3, that is to say if an agent observes that the other agent turns to its left then goes forward for some time, it knows that in order to replicate those actions, it has to turn to its left and go forward. The second issue must be part of the learning process of the imitating agent so that it will infer in what context (or states) the observed actions are useful through, for instance, a trial and error mechanism. Thus imitated actions that cause an increase in the performance in a state will be more likely to be learned. Similarly, any negative outcome will make the observed actions unlikely to be associated with that state.

There is some research that attempts to use imitation in connection with individual learning. Abbel and Ng (Abbeel and Ng, 2004) used an expert in order to learn to perform a task in which the reward function is not known. They tried to obtain the unknown reward function which is supposed to be implicitly followed in the expert's behaviour. The value of a policy is defined as a linear function of the features of the environment and the learning agent has an estimate of the expert's feature expectations. Their algorithm is proved to attain performance close to that of the expert agent. Latzke et al. (Latzke et al., 2006) utilized Q-learning which uses the experience of an expert agent as training data. The imitating agent has full access to the experience of the expert and these experiences are provided as sequences of states and actions along with their rewards.

Bentivegna et al. (Bentivegna et al., 2004) used a modified version of Q-learning in order to improve the performance of an agent through practice. Their algorithm stores the actions of a teacher agent in a number of states. During the practice period, the learner agent selects one of the data entries based on the distance between the state recorded in that entry and the current state. The value of selecting that entry in the current state is then updated, based on the reward observed after the action stored in the selected data entry is performed.

The approach presented in this chapter is different from those outlined above for two reasons. Firstly, there is no assumption about the learner agent having access to the internal state or expectations of the imitated agent. The only information that is transferred to the learner agent is the imitated agent's executed actions. The learner agent does not know anything about in what state these actions were executed or what were the outcomes of these actions. Secondly, the case in which the observed actions are not useful at all is also considered; in this case the learner agent must discriminate between useful and useless actions.

In this chapter, an adaptive algorithm in which imitation is used as a method for enhancing learning is presented. As developing the algorithm purely on real robots would take a long time, it is firstly checked in a simulation with simulated agents in this chapter. It is shown that the agents that use imitation as an enhancing factor are able to learn faster. Once the algorithm is proved to be effective, it is then tested on real robots in the next chapter.

5.2 Simulation Setup

To examine the effects of imitation on learning strategy, an agent that employs Q-learning in order to reach a target location is simulated. For the first set of experiments, the arena in which the agent operates is a 10 by 10 grid world (figure 5.1). The agent starts from the top-left corner of the arena and, in each time unit, it moves to one of the eight neighbouring cells. One simulation run ends when the agent reaches the goal item, which is placed in the bottom-right corner of the arena.

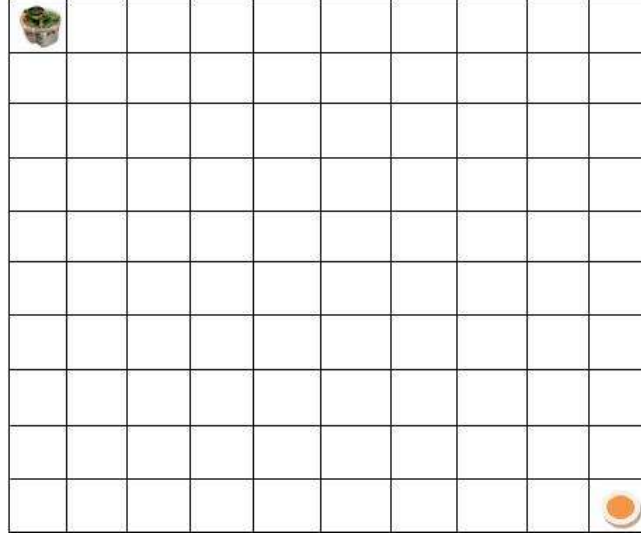


Figure 5.1: Simulated arena. The arena is a 10 by 10 grid world. The agent is placed in the top-left corner and the goal item is placed in the bottom right corner.

The learning agent uses an ϵ – *greedy* algorithm in which at each time step it finds the action that has the highest Q value estimate for its current position. With probability $1 - \epsilon$, it chooses that action and with a probability of ϵ , it chooses a random action. In this way, it updates the Q value for its current state and chosen action:

$$Q(s_t, a_c) = Q(s_t, a_c) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_c)] \quad (5.1)$$

in which a_c is the chosen action and r_{t+1} is the reward for getting to state s_{t+1} . In one experiment run, the agent gets to the goal state 100,000 times and so the Q values for each state-action pair have adequate time to converge to their final values. Every 10 time units, the shortest path to the goal item that is learned by the agent is determined by using a greedy action selection method on the current Q values.

5.3 Imitation-Enhanced Reinforcement Learning

Only the executed actions are observable and hence copied when an agent imitates another. To simulate this process, for the first set of experiments a set of actions is given to the learning agent. At each time step, the agent may start to replicate the given set of actions with a probability equal to $p_{imitate}$. When the agent starts to imitate the path, all actions that constitute the path are executed one by one by the agent. If these actions contradict the Q values of the agent, i.e. have a low

value, then the agent stops following the imitated path and acts based on its original $\epsilon - greedy$ algorithm. Thus the controller of the agent with imitation is updated as shown in Algorithm 4.

Algorithm 4 Pseudocode for the controller of the agent with one path

Input:

Set $Q(s, a) \leftarrow 0$ for all state-action pairs

A set of observed actions: $actionList \leftarrow \{a_1, a_2, \dots, a_n\}$, $actionIndex \leftarrow 0$

Repeat 100.000 times

Place the agent in the top-left corner

while $s \neq GoalState$ **do**

if $actionIndex \neq 0$ **then**

$actionIndex \leftarrow actionIndex + 1$

$a \leftarrow actionList[actionIndex]$

if $ListCompleted(actionList) = true$ **then**

$actionIndex \leftarrow 0$

end if

else

if $p_{imitate} > random()$ **then**

$actionIndex \leftarrow 1$

$a \leftarrow actionList[actionIndex]$

else

$a \leftarrow \max_{a'} Q(s, a')$

if $\epsilon > random()$ **then**

$a \leftarrow SelectRandomAction()$

end if

$actionIndex \leftarrow 0$

end if

end if

if $actionIndex > 0$ **then**

if $\exists a' : Q(s, a') > Q(s, a)$ **then**

$a \leftarrow \max_{a'} Q(s, a')$

if $\epsilon > random()$ **then**

$a \leftarrow SelectRandomAction()$

end if

$actionIndex \leftarrow 0$

end if

end if

$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \max_{a'} Q(s', a') - Q(s, a)]$

$s \leftarrow s'$

end while

But how do we select, $p_{imitate}$, the starting imitation probability? For the first try, $p_{imitate}$ is set to a fixed value. In the first experiment run, the agent is given a path that moves it towards the goal item (figure 5.2) which is a beneficial path, so that the agent has this path in the memory and is able to enact it according to its controller. Figure 5.3 shows the performance of imitation

enhanced learning agent with different $p_{imitate}$ values. As can be expected, there is a performance increase in the learning speed of the agent compared to a no-imitation agent. But what would happen if the path given is not beneficial? To test this scenario, the path in figure 5.4 is given to the agent in the second experiment run. Figure 5.5 shows the performance of the agent when this path is given, which is clearly not a beneficial path. As can be seen, there is a performance loss when the path given is not directing the agent towards the goal item. It is clear that there should be an adaptive $p_{imitate}$ calculation which can determine if an observed path is beneficial or not.

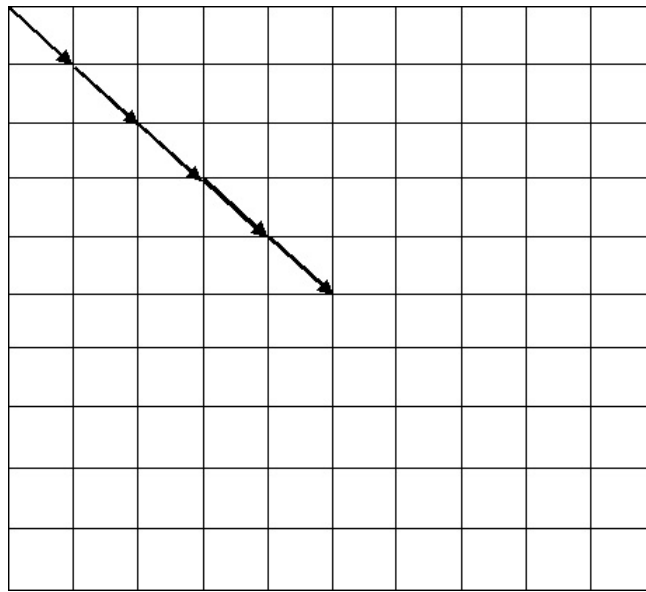


Figure 5.2: Path given to the agent. It consists of 5 consecutive moves South East. Since by following this path, the agent gets close to the goal item, this path is beneficial for the agent. These moves are shown at relative positions.

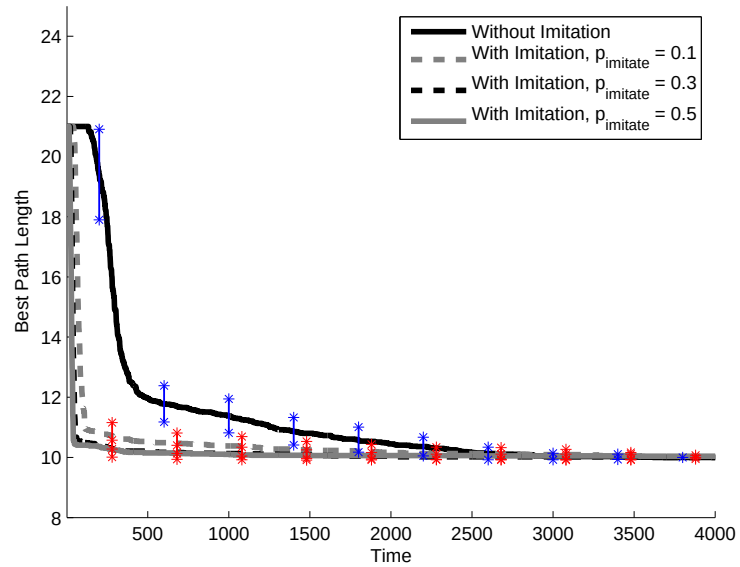


Figure 5.3: Performance of the agent when a beneficial path is given, fixed p_{imitate} value. Time is given in time units.

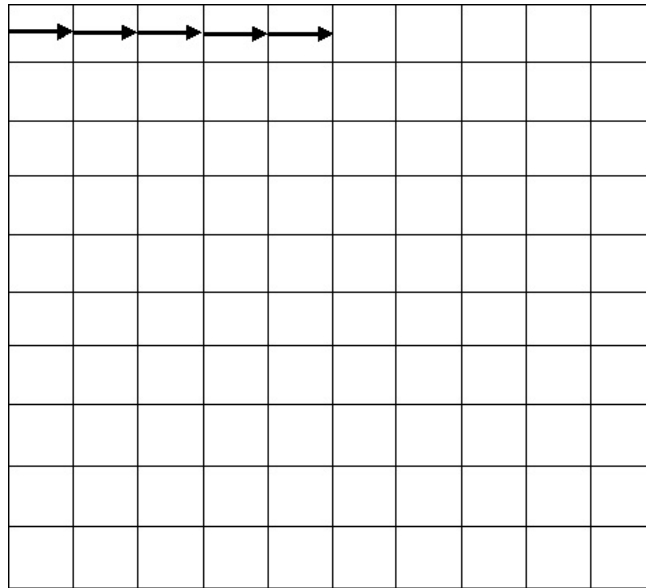


Figure 5.4: Path given to the agent. It consists of 5 consecutive moves East. Since by following this path, the agent does not get close to the goal item, this path is not beneficial for the agent. These moves are shown at relative positions.

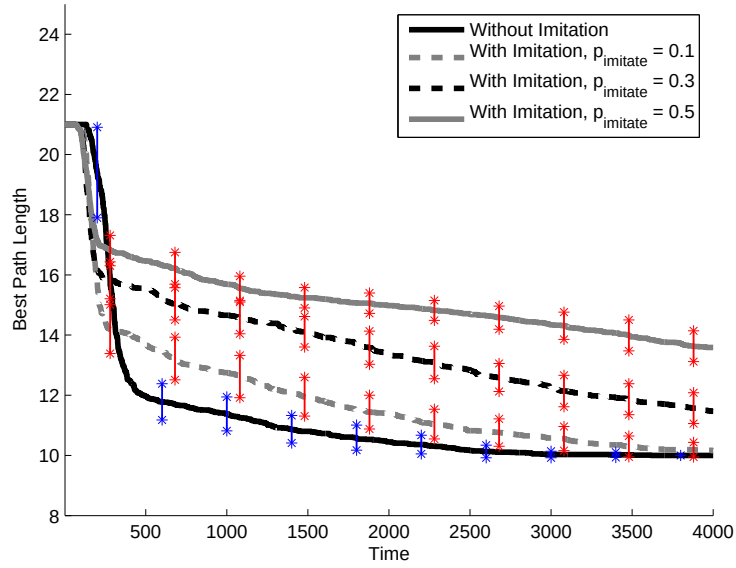


Figure 5.5: Performance of the agent when a non-beneficial path is given, fixed $p_{imitate}$ value.

The agent can also check if a path can be completed. If the imitated path is being abandoned continuously (because it is rejected by the Q values or because of the physical boundaries of the arena) whenever it is started, it may be a sign of a non-beneficial path. So for the second try, the $p_{imitate}$ value is calculated by:

$$n_{completed} = \text{number of times the observed path is completed}$$

$$n_{replicated} = \text{number of times the observed path is replicated}$$

$$p_{imitate} = \beta \frac{n_{completed} + 1}{n_{replicated} + 1} \quad (5.2)$$

in which β is a constant that regulates initial imitation probability. Therefore by using this formula, the agent will tend to imitate paths more if they can be completed. Figure 5.6 shows the performance of the imitation-enhanced learning agent for different β values when the beneficial path in figure 5.2 is given. Once again there is a clear performance increase in the learning speed of the agent when the imitated path is beneficial. Figure 5.7 shows the performance of the imitation-enhanced agent when the path in figure 5.4 is given. Although it is slightly better than the case with a fixed $p_{imitate}$, there is still a performance loss when the path given is not beneficial.

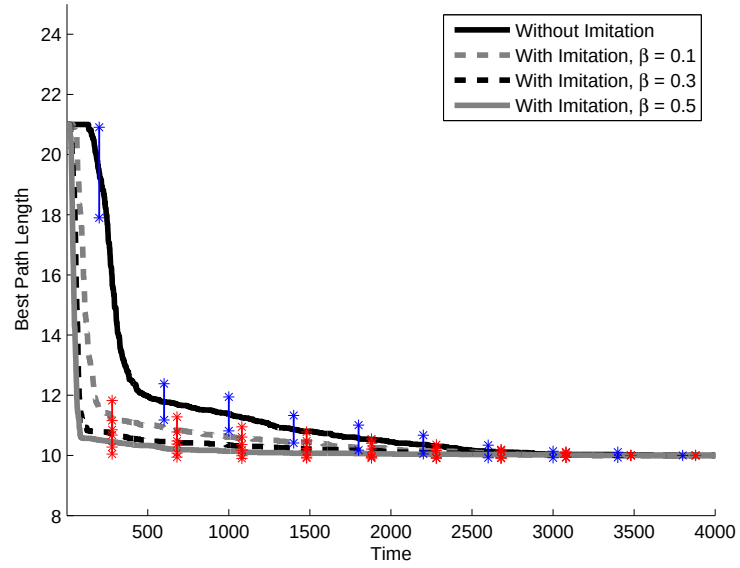


Figure 5.6: Performance of the agent when a beneficial path is given, path completion test.

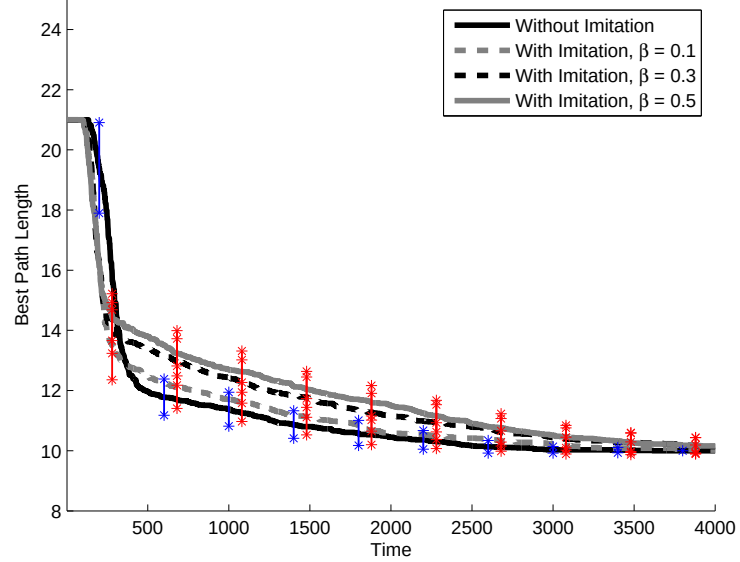


Figure 5.7: Performance of the agent when a non-beneficial path is given, path completion test.

Another method can be examining the effects of imitation on the Q values of the agent. For this purpose, we check if the observed actions cause any change in Q-value of the related state-action pairs. If they do not, this means that the observed actions are causing the agent to explore some parts of the state-action space which have already converged to their final values or the agent is

exploring some parts of the state action-space which do not contribute to its overall performance. If the observed actions do cause a change in Q-values then it is possible that imitation may be beneficial. This is determined as follows: the average Q value is recorded at the start of each imitation sequence and is compared to the average Q value when the imitated path is completed or abandoned. So $p_{imitate}$ is calculated by:

n_{Q+} = number of times the average Q value increased when the observed path is replicated

$n_{replicated}$ = number of times the observed path is replicated

$$p_{imitate} = \beta \frac{n_{Q+} + 1}{n_{replicated} + 1} \quad (5.3)$$

Figure 5.8 shows the performance of the imitation-enhanced agent for different β values when the beneficial path in figure 5.2 is given. Once again there is a clear performance increase in the learning speed of the agent when the imitated path is beneficial. Figure 5.9 shows the performance of the imitation-enhanced agent when the path in figure 5.4 is given. As can be seen, the performance loss due to imitating a non-beneficial path is minimal compared to the previous two tries. When the β value is set to 0.1, the difference between an imitation-enhanced agent and a no-imitation agent is not statistically significant. For higher β values (0.3 and 0.5) the difference between the two is much smaller than the case with a fixed $p_{imitate}$ and with the path completion test. Therefore, by checking temporal changes in Q values, it is possible to determine if imitating an observed path is beneficial or not.

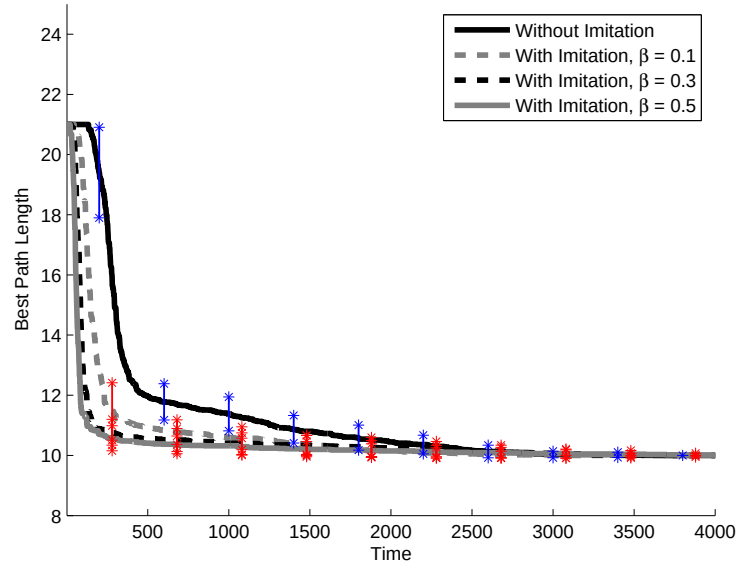


Figure 5.8: Performance of the agent when a beneficial path is given, average Q value test.

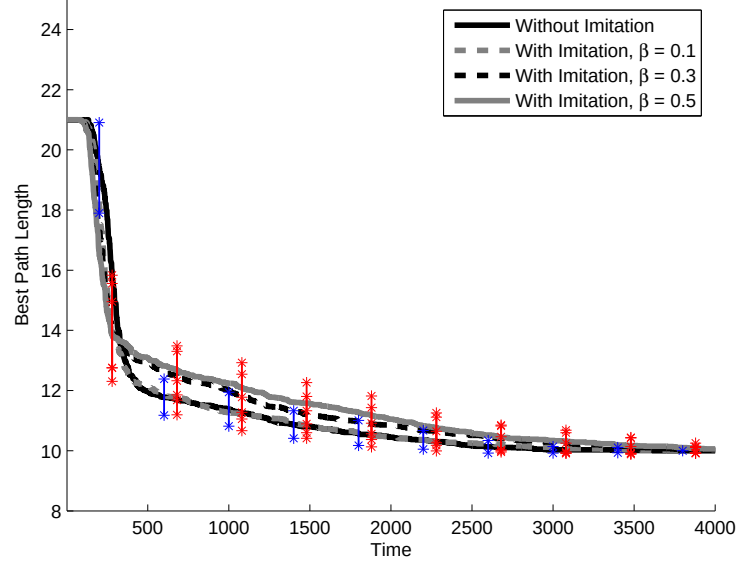


Figure 5.9: Performance of the agent when a non-beneficial path is given, average Q value test.

As the imitation-enhanced agent is able to determine if a path is beneficial or not by checking temporal changes in Q values, when it observes multiple paths, it should be able to select the ones that are useful and avoid others. For this purpose, the ratio of the times the average Q value increased is calculated for each path i as:

n_{Q+}^i = number of times the average Q value changed when path i is replicated

$n_{replicated}^i$ = number of times path i is replicated

$$R_i = \frac{n_{Q+}^i + 1}{n_{replicated}^i + 1} \quad (5.4)$$

R_i is used for both regulating the imitation probability of each path observed and choosing which path to imitate. If the agent has n distinct paths, the probability of choosing path i is calculated by:

$$P_{choose}^i = \frac{R_i}{\sum_{k=1}^n R_k} \quad (5.5)$$

The probability of imitating path i once it is chosen is:

$$p_{imitate}^i = \beta R_i \quad (5.6)$$

in which β is a constant that regulates the initial imitation probability. Thus, the controller of the agent with multiple paths is updated as shown in Algorithm 5.

Algorithm 5 Pseudocode for the controller of the agent with multiple paths**Input:**

Set $Q(s, a) \leftarrow 0$ for all state-action pairs
 A set of observed paths $pathList \leftarrow \{Path_1, \dots, Path_n\}$
 $pathChosen \leftarrow 0, actionIndex \leftarrow 0$
 Repeat 100.000 times
 Place the agent in the top-left corner
while $s \neq GoalState$ **do**
 if $pathChosen \neq 0$ **then**
 $actionIndex \leftarrow actionIndex + 1$
 $a \leftarrow pathChosen[actionIndex]$
 if $PathCompleted(pathChosen) = true$ **then**
 update $R_{pathChosen}$
 $pathChosen \leftarrow 0$
 end if
else
 $pathChosen \leftarrow SelectProbabilistically(PathList)$
 if $p_{imitate}^{pathChosen} > random()$ **then**
 $actionIndex \leftarrow 1$
 $a \leftarrow pathChosen[actionIndex]$
 else
 $a \leftarrow \max_{a'} Q(s, a')$
 if $\epsilon > random()$ **then**
 $a \leftarrow SelectRandomAction()$
 end if
 $pathChosen \leftarrow 0$
 end if
 if $pathChosen > 0$ **then**
 if $\exists a' : Q(s, a') > Q(s, a)$ **then**
 $a \leftarrow \max_{a'} Q(s, a')$
 if $\epsilon > random()$ **then**
 $a \leftarrow SelectRandomAction()$
 end if
 update $R_{pathChosen}$
 $pathChosen \leftarrow 0$
 end if
 end if
 $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \max_{a'} Q(s', a') - Q(s, a)]$
 $s \leftarrow s'$
end while

5.4 Experiments

In order to test the generalization of the algorithm and to make the problem more complex, the next set of experiments are performed in a 100 by 100 grid with an obstacle in the middle (figure 5.10). Once again the learning agents start from the top-left corner and try to reach the goal item which is placed in the bottom-right corner.

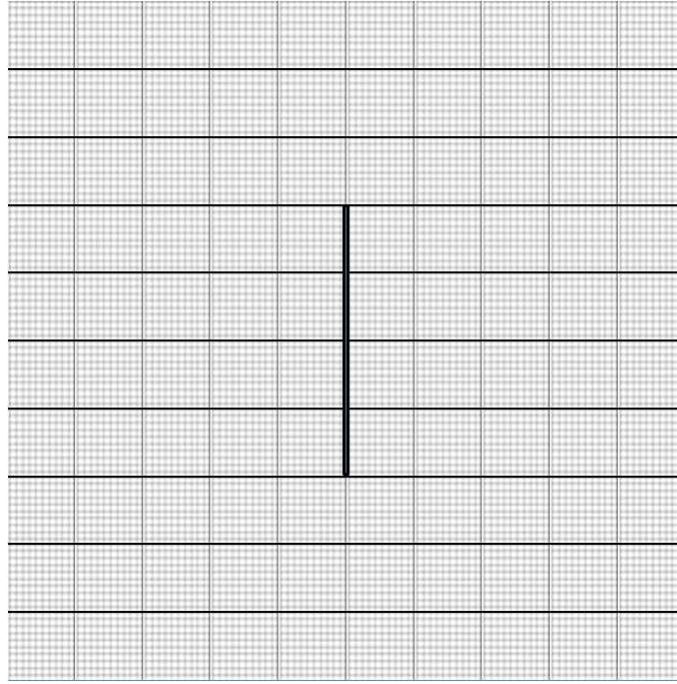


Figure 5.10: Arena with obstacle: The arena is a 100 by 100 grid world. The agents are placed in the top-left corner and the goal item is placed in the bottom-right corner. There is an obstacle in the middle of the arena.

5.4.1 Imitating Predefined Paths

To check if the algorithm is able to select useful paths and improve the learning speed by exploiting them, the learning agent is given 8 paths. These are chosen to each represent 20 consecutive moves in the eight different directions of the compass, as shown in figure 5.11. The agent has these paths in its memory and it is able to enact them according to its controller. During the experiment, the agent has the following action sets in its memory:

- $Path_1$: E-E-E-E-E....-E (20 moves)
- $Path_2$: SE-SE-SE-SE-SE-SE....-SE

- $Path_3$: S-S-S-S-S....-S
- $Path_4$: SW-SW-SW-SW-SW-SW....-SW
- $Path_5$: W-W-W-W-W-W....-W
- $Path_6$: NW-NW-NW-NW-NW-NW....-NW
- $Path_7$: N-N-N-N-N-N....-N
- $Path_8$: NE-NE-NE-NE-NE-NE....-NE

If the corresponding R_i values for each path are $R_1, R_2, \dots, R_8$, the probability of selecting each path would be equal to $R_i / (R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7 + R_8)$. Assume the agent is in state s_t and it selects $Path_1$ and starts it. If there exists $Q(s_t, a) > Q(s_t, E')$, the path is rejected and the agent moves according to its ϵ -greedy Q-learning. Otherwise the agent starts to follow $Path_1$, making the same check at each step. Whenever a path is rejected or completed, its R_i value is updated.

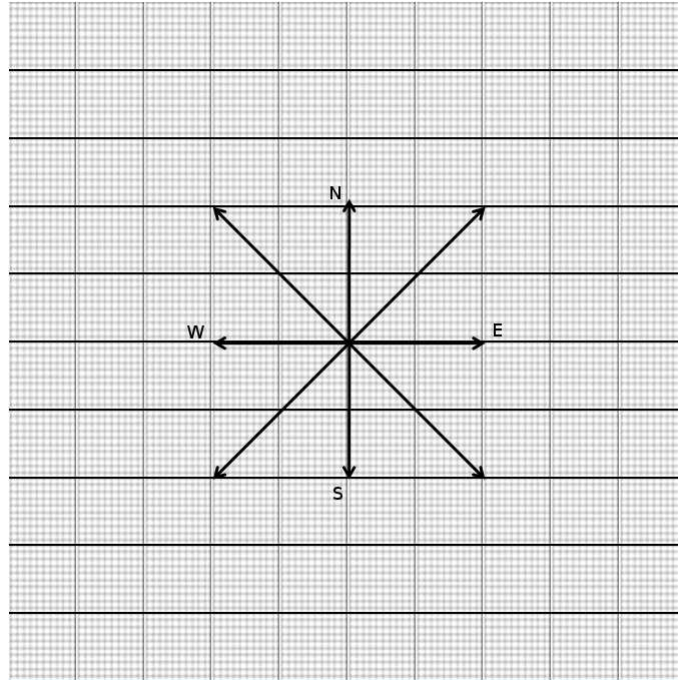


Figure 5.11: 8 Paths given to the agent. Each list of actions consist of 20 consecutive moves in one direction: E, SE, S, SW, W, NW, N and NE

Figure 5.12 shows the performance of the imitating agent averaged over 100 experimental runs in comparison with a no-imitation agent. The imitation-enhanced learning agent was able

to choose the beneficial paths and improve its performance. A pair-wise t-test reveals that the difference between the two is statistically significant until around time step $15.5m^2$. Figure 5.13 shows an example shortest path in our arena. Any shortest path consists of a combination of 80 moves south-east, 20 moves east and 20 moves south. Figure 5.14 shows the R_i values for each of the 8 paths averaged over 100 runs. As can be seen, the path to the SE has the highest R_i value which makes it more likely to be chosen. The paths E and S have relatively higher R_i values compared to the other 5 paths. This confirms that the agent was able to select the 3 paths that are more likely to improve its performance.

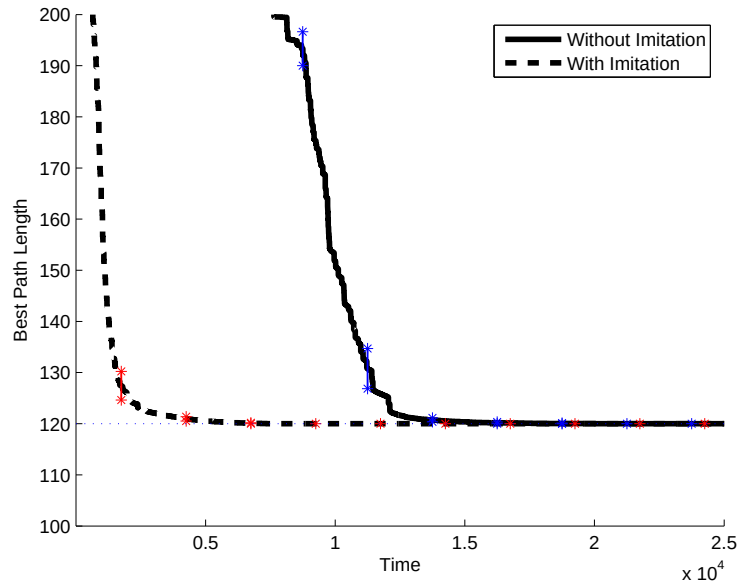


Figure 5.12: Best path length for imitating (8 paths given) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals (note these are shown only at 10 point intervals). The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.

²15.5m means 15,500,000 time steps

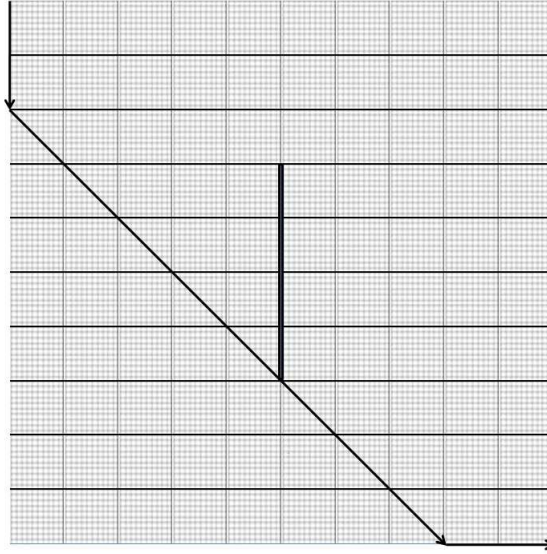


Figure 5.13: An example shortest path to the goal state. Any shortest path consists of a combination of 80 moves to SE, 20 moves to S and 20 moves to E directions.

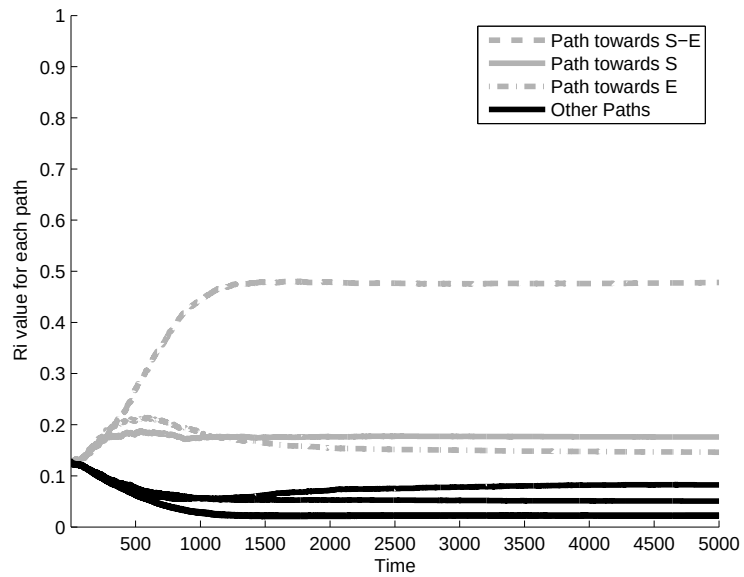


Figure 5.14: The R_i values for each path given to the agent.

5.4.2 Agents Imitating Each Other

An Agent Copying an Expert Agent

As stated above, an important aspect of imitation is the possibility of a teacher or expert whose experience can be transferred, by imitation, to other individuals. To simulate an expert in the arena,

an agent is programmed to follow the shortest path in Figure 4 indefinitely. Then a second agent is presented in the arena that starts learning and is able to watch and copy the expert agent, following the imitation-enhanced Q-learning algorithm. The second agent starts with an empty memory and once every 100 simulation runs, it copies a random consecutive portion of the expert's trajectory. Note that although this is a part of an ideal (i.e. shortest) path, the imitating agent does not know anything about in which state (or location in the arena) these actions are meaningful. The imitating agent can store up to 10 paths. After its memory is filled, when a new path is copied, of the paths in its memory the one with the lowest R_i is erased and the new path is recorded. Since each experiment consists of 100,000 simulation runs, the agent will imitate and test 1000 portions in each experiment run. Figure 5.15 shows the performance of the agent imitating an expert. As can be seen, the imitation-enhanced learning agent finds the shortest path much earlier than the agent with Q-learning only. A pair-wise t-test reveals that the difference between the two is statistically significant until around time step 15.5m.

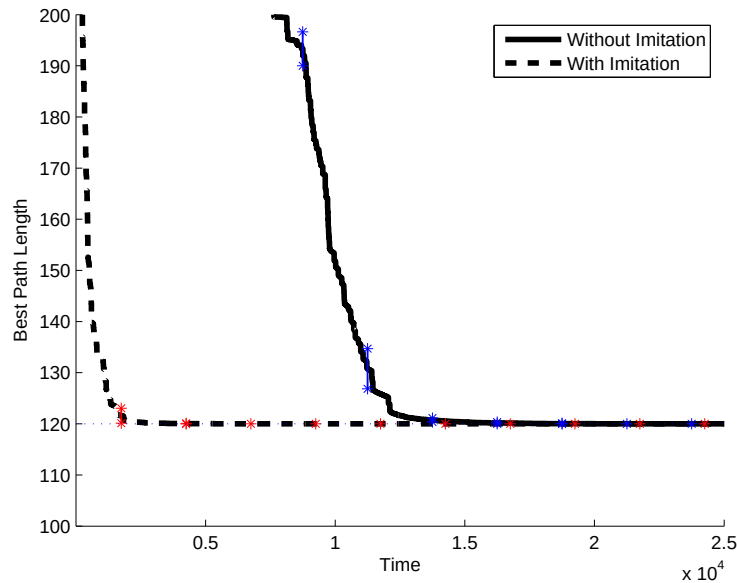


Figure 5.15: Best path length for imitating (copying an expert) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.

Two Agents Copying Each Other

But what would happen if there is no expert agent to share its experience? In order to simulate this scenario, two agents are presented in the arena, both starting to learn at the same time and able to watch each other. Once again they store 10 action lists and they copy (i.e. imitate) a new path from each other every 100 simulation runs. Figure 5.16 shows results for this experiment. As can be seen, the imitating agents have a better performance than a no-imitation agent initially. But the difference between them vanishes after 5m time steps. This is due to the fact that the two imitating agents influence each other throughout the experiment. At the start, the one that has slightly better performance is able to improve the other but as time passes both start to converge to the same performance. After some point, their continuous effect on each other causes less improvement which explains why they get similar performance to a no-imitation agent. Pair-wise t-test shows that the imitating agent has a statistically better performance between the time steps 4m and 9.5m. Note also that some runs of the no-imitation agent converge to a shortest path earlier than the imitating agent which makes the no-imitation agent statistically better between the time steps 12.5m and 19.5m, although the difference between the two is very small.

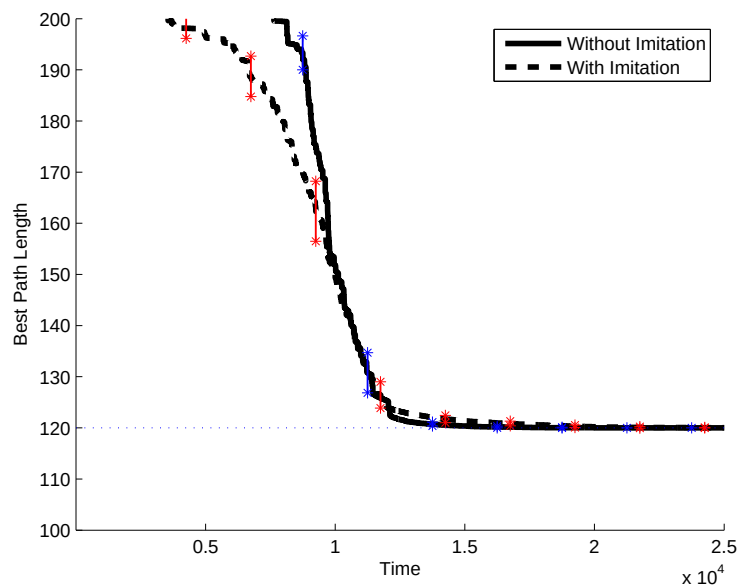


Figure 5.16: Best path length for imitating (two agents copying each other) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.

An Agent Copying an Experienced Agent

Finally, the effect of the existence of a slightly experienced agent instead of an expert is tested. A single agent is trained in the arena for some time (hence no-imitation) and a second agent is presented later. The second agent is able to watch and copy the trained agent, following the imitation-enhanced Q-learning algorithm. Once again the imitating agent copies a new path every 100 simulation runs and has a memory of 10 action lists. Figure 5.17 shows the results for this experiment. The agent that is watching an experienced (but not expert) agent has a faster learning rate compared to a no-imitation agent. A pair-wise t-test reveals that the difference between the two is statistically significant until around time step 15.5m.

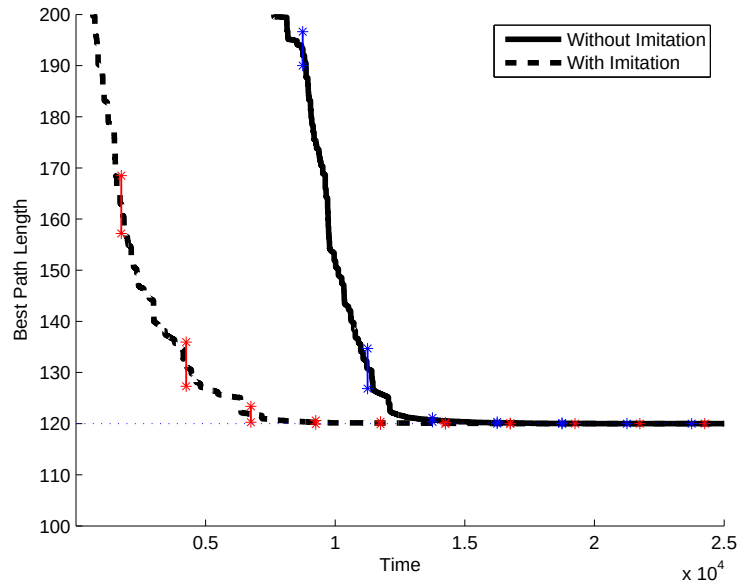


Figure 5.17: Best path length for imitating (copying an experienced agent) and no-imitation agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments. Time is given in 100 time units.

Figure 5.18 compares all scenarios: an agent given 8 paths (figure 5.12), an agent watching an expert (figure 5.15), two agents watching each other from the beginning (figure 5.16) and an agent watching an experienced agent (figure 5.17). As expected the agent copying an expert outperforms all others although the agent that is given 8 paths achieves the same performance after around 5m time steps. The agent that is watching an experienced (but not expert) agent achieves

the same performance with these two after around 8m time steps. All three have significantly better performance than the worst case of two agents copying each other.

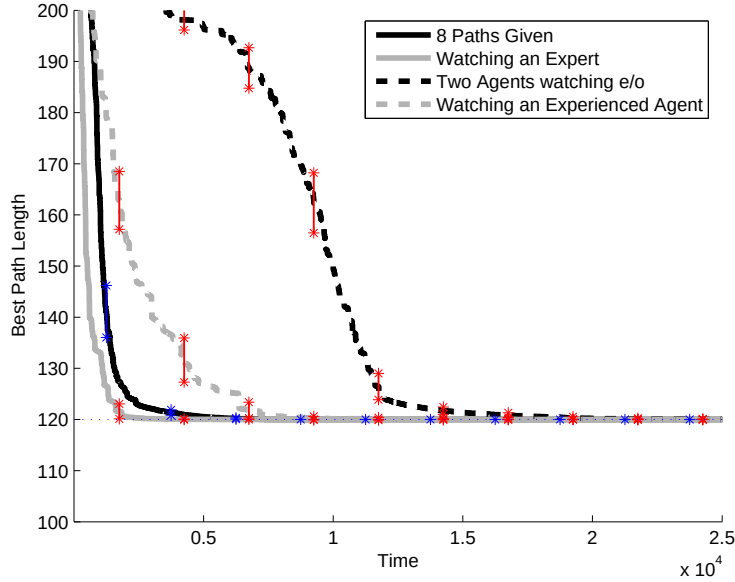


Figure 5.18: Best path length for 4 scenarios. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.

To understand what kind of paths are being copied by a learning agent while it is watching an experienced agent, the paths that are imitated the highest number of times during experiments are examined. Note that although the experienced agent may have already achieved an optimum path, there can still be variation in its moves as it has a random move probability equal to ϵ . Two types are mostly chosen by the imitating agent. One takes the agent SE. All runs included this kind of path imitated many times. Figures 5.19 to 5.21 show 3 example paths of this kind, along with the region of the arena they might be useful. The second type of path that is imitated the largest number of times allows an agent to avoid the obstacle and/or explore the region around it. Figures 5.22 to 5.24 show some examples of this kind of path together with the region in which they might be useful.

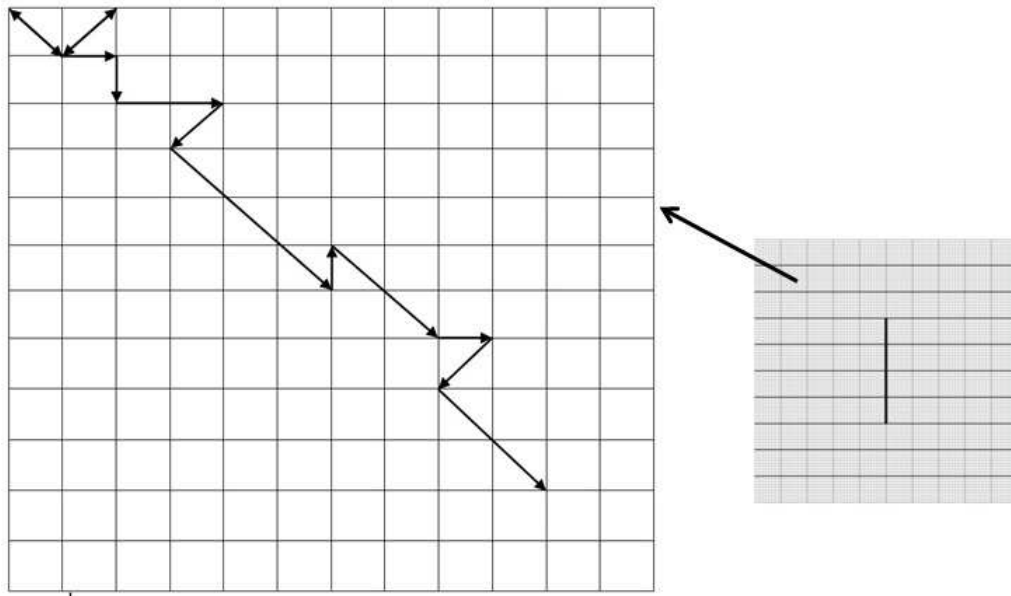


Figure 5.19: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

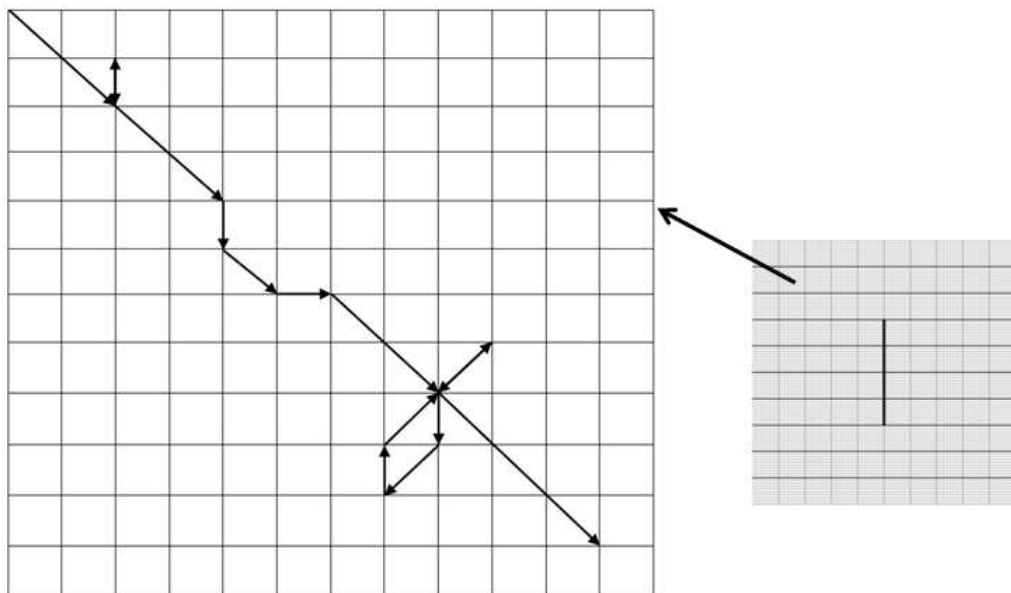


Figure 5.20: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

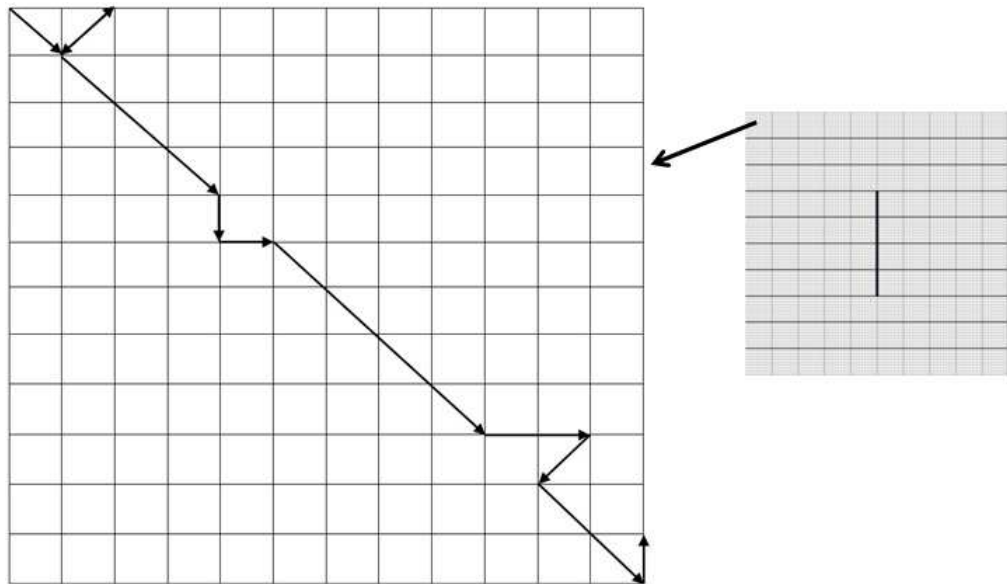


Figure 5.21: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

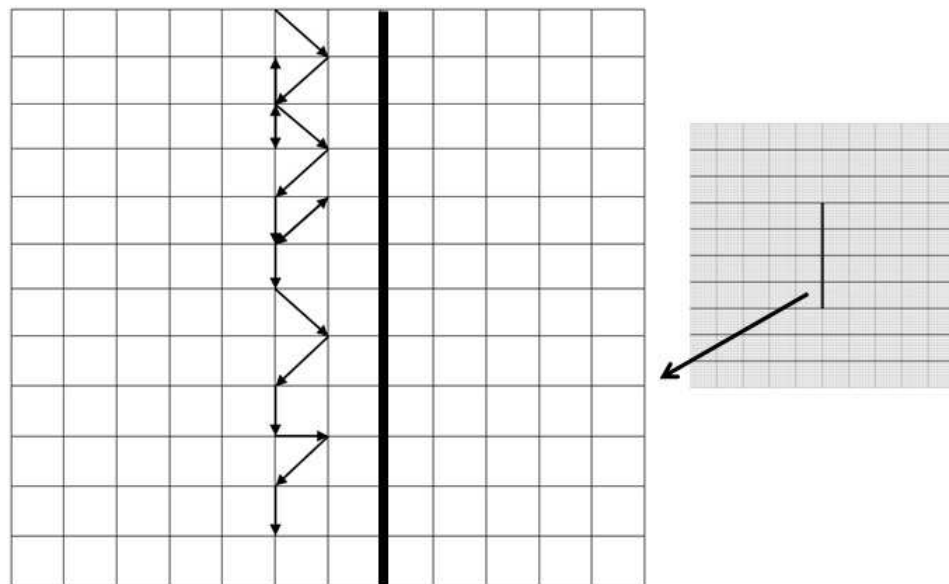


Figure 5.22: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

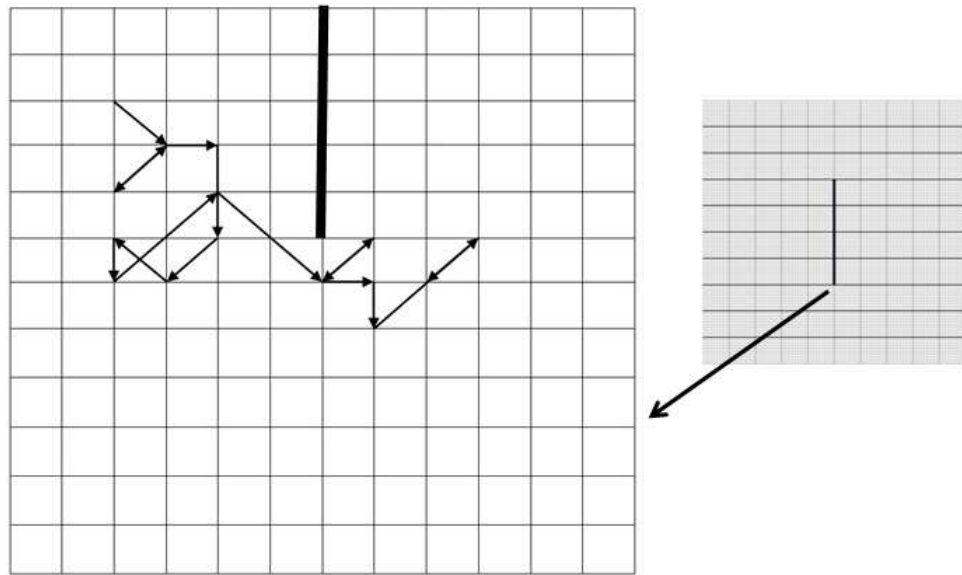


Figure 5.23: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

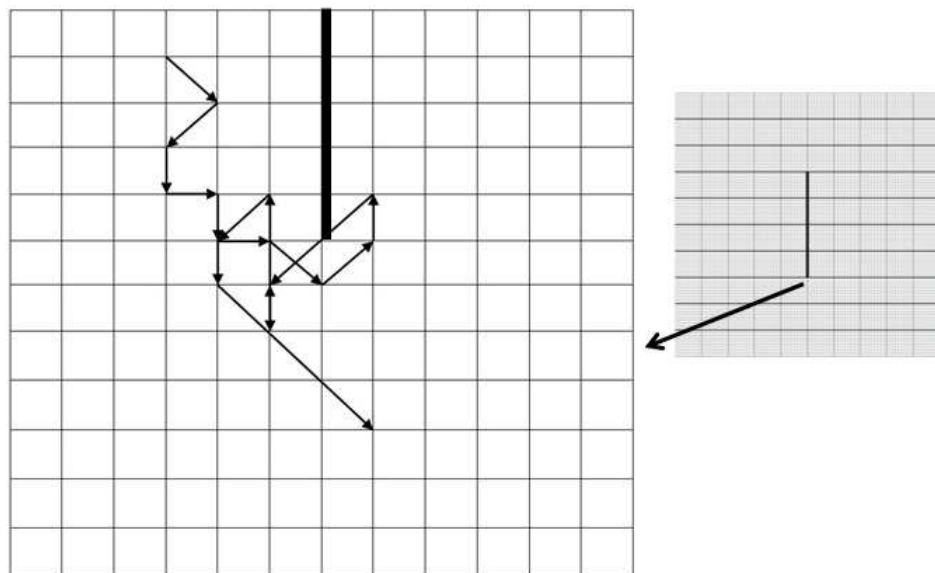


Figure 5.24: An example of the paths that are imitated the highest number of times. The small figure on the right side shows the complete arena and points to the region in the arena this path may be useful. The big figure on the left is the zoomed version of that region and shows the copied path.

Beyond Imitation; Agents Having Access to Each Other's Internal State

Based on the related work discussed in section 5.1, the performance of the agents when they are able to access the internal state of the other agent is tested. In this case the agents have complete access to the Q-table of the other agent, i.e. including its Q values for all state-action pairs. This essentially means that the agent being observed makes available its estimate of the best action for the current state of the observing agent. Therefore the observing agent, instead of copying actions executed by the other agent from potentially any state, determines a path by repeatedly selecting the action with the highest Q value of the other agent in the current state. Based on the previous mechanism, if the suggested action contradicts its own Q values, i.e. it has a lower value, the observing agent stops imitating.

Figure 5.25 shows the results of this scheme when copying from an expert. As can be seen, the learning agent becomes optimal after a very short time interval. This is perhaps not unexpected, since the scenario is similar to telepathy. But what if there is no expert? Figure 5.26 shows the results when two agents, starting to learn at the same time, are able to access each other's internal state. As can be seen, it almost has the same performance as the $\epsilon - greedy$ Q Algorithm.

Figure 5.27 compares two agents copying each other (figure 5.16) and two agents copying each other with access to each others internal state (figure 5.26). As can be seen, the agent with the algorithm introduced in this chapter learns more quickly as a pair-wise t-test reveals that the agent utilising imitation-enhanced Q learning is statistically better than the other scenario between time steps 4m to 9.5m. Therefore, it can be deduced that there is more scope for exploration using the overt action sequences which explains this difference, although this result may be parameter specific.

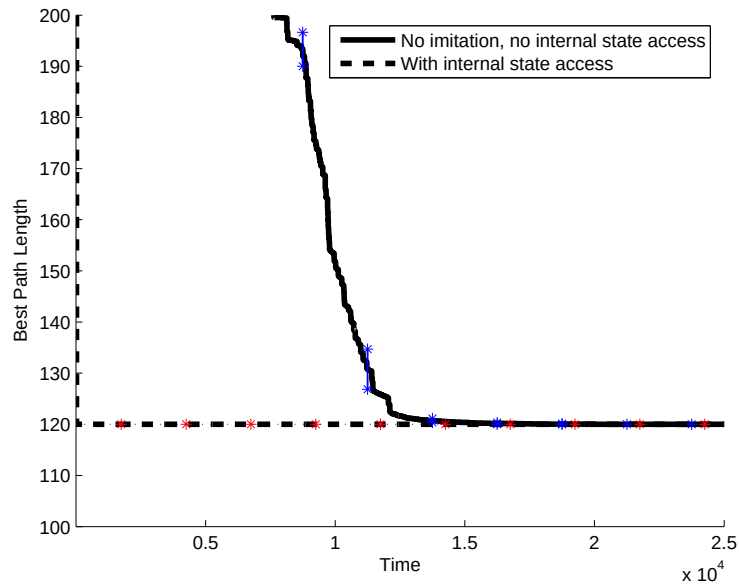


Figure 5.25: Best path length for an internal-state-access (accessing an expert agent's Q values) agent and no-imitation/no-internal-state-access agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.

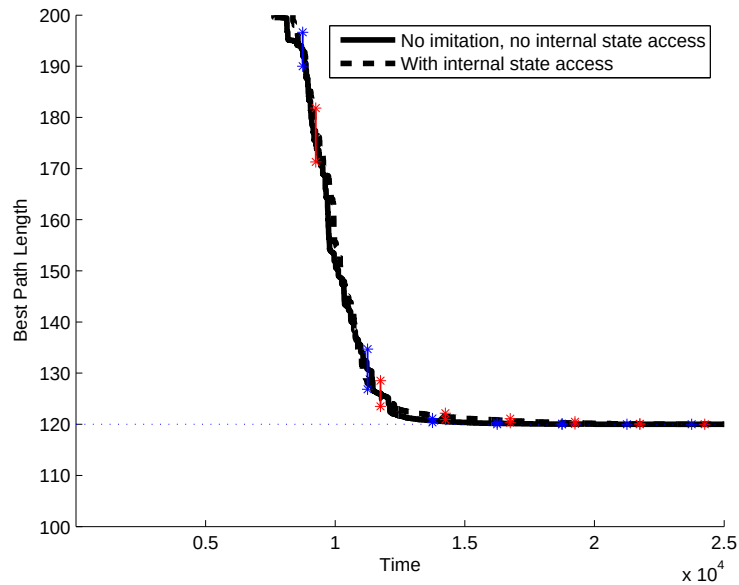


Figure 5.26: Best path length for an internal-state-access (two agents using each other's Q values) agent and no-imitation/no-internal-state-access agents. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values.

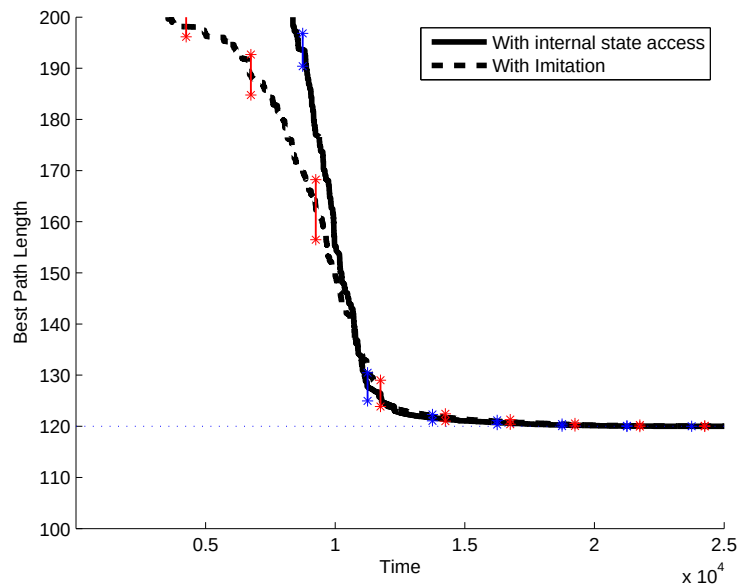


Figure 5.27: Best path length for an agent accessing another agent's Q values and imitating (two agents copying each other) agent. The results are mean best path length from 100 experiment runs along with 95% confidence intervals. The best path for each agent is calculated every 1000 time units by using a greedy algorithm on the current Q values. Time is given in 100 time units.

5.5 Summary and Conclusion

In this chapter, a method was presented for using imitation as an enhancing factor that increased the speed of learning for the well known reinforcement learning algorithm, Q-learning. It was shown that information gained by imitation could be used within the Q-learning algorithm. By checking temporal changes in Q values it was possible to determine if imitating some observed paths was beneficial or not. For a simple abstract model in simulation it was shown that the imitation-enhanced Q-learning agent learned faster than the same Q-learner without imitation. Compared to other research that used imitation with reinforcement learning, our method used imitation of purely observed behaviours to enhance learning as there was no internal state access or sharing of experiences between agents. The only information that was transferred between agents was the imitated agent's executed actions. Finding in what state or context these actions were meaningful was determined by the learning process of the imitating agent.

Although the algorithm was tested on a simple model, it can be applied to a range of problems including individual and social learning in real robots. There was no assumption about the properties of the two agents that were involved in imitation. For instance, it may be possible that a robot

can utilize imitation-enhanced Q-learning by watching a human. The only condition is that there should be a way of translating the actions of the agent that is being imitated into a set of actions that can be executed by the imitating agent.

The most important simplification in this chapter was that the algorithm was tested on simulation as it was suitable for discrete-time tasks. In the next chapter, the algorithm is extended so that it can be used for continuous-time tasks and it is tested on real robots.

Chapter 6

Imitation-Enhanced Q-learning in Continuous Space

In this chapter, imitation-enhanced Q-learning algorithm is tested on real robots¹ that have a continuous time learning task. The robots utilise embodied imitation, that is implemented in chapter 3, in which they use their on-board sensors to copy the actions executed by other robots.

6.1 Introduction

In the previous chapter, a method for using imitation to enhance the learning speed of agents employing Q-learning is presented. The method checks the temporal changes in Q values for state-action pairs to determine if the information gained by imitation is useful or not. It has been shown that for a simple abstract model in simulation, the imitation-enhanced agent learns faster than the Q-learner agent without imitation.

The most important simplification in the previous chapter is that the algorithm was tested in simulation. The agents do not possess any physical structure and they interact based on an abstract model of perfect imitation by copying each other's trajectory. In this chapter, the imitation-enhanced Q-learning algorithm is tested on real robots. The robots use an extended version of Q-learning that is adapted for continuous time. The effects of imitation on the learning speed of robots is evaluated in experiments as the robots learn to achieve a simple task. In chapter 4, it was shown that embodied imitation has certain effects on the content of the movement patterns that are transferred by imitation. The effect of embodied imitation on the learning speed of the robots is examined in this chapter.

¹The term 'robot' is used when real robots perform experiments instead of simulated agents.

There are a number of mechanisms proposed in order to use reinforcement learning methods in continuous time and space (Santamaria et al., 1998), (Smart and Kaelbling, 2000), (Doya, 2000), (Strosslin and Gerstner, 2006), (Peters and Schaal, 2006). Although we have a continuous-time problem, the learning robot has to make choices only infrequently. Such systems are called *semi-Markov Decision Processes* (SMDP) (Sutton and Barto, 1998). They can be treated as a Markov process by taking the reward on each discrete transition as an integral of the cumulative reward on the continuous time interval passed before that transition. Bradtke and Duff (Bradtke and Duff, 1995) introduced algorithms, based on reinforcement learning, adopted to the solution of SMDPs. They applied the SMDP version of Q-learning to the problem of determining the optimal control for a queueing system. Using a similar approach, Crites and Barto (Crites and Barto, 1996) applied the SMDP version of Q-learning, with some additional constraints that provide prior knowledge to the system, to an elevator dispatching task. They used a neural network for function approximation to represent the action-value function. Since we wanted to apply imitation-enhanced Q-learning algorithm that was developed in chapter 5 to continuous time and space, and our system has the property of SMDP, we used the method that is developed by Crites and Barto, by dividing our arena into 10x10 grids. Assume that the robot is in state s and it selects action a at time t_1 . If the next state transition will happen at time t_2 and the next state will be s' , then the update for the Q value for the semi-Markov decision process is calculated by (Crites and Barto, 1996):

$$Q(s, a) = Q(s, a) + \alpha \left[\int_{t_1}^{t_2} e^{-\beta(\tau-t_1)} r_\tau d\tau + e^{-\beta(t_2-t_1)} \max_{a'} Q(s', a') - Q(s, a) \right] \quad (6.1)$$

in which $e^{-\beta(t_2-t_1)}$ acts as a variable discount factor, similar to the γ parameter in the discrete time formula, and it depends on the time between state transitions. Each time the robot makes a decision and moves to its next state, the Q value of its state-action pair is updated based on this formula.

In this chapter, experiments are presented in which real robots utilise imitation-enhanced Q-learning, that is developed in chapter 5, to achieve a task. The robots are able to observe and imitate each other using the embodied imitation algorithm explained in chapter 3. In addition, the effects of variation in the copied paths due to embodied imitation are investigated in a continuous time experiment.

6.2 Experimental Setup

To examine the effects of embodied imitation on the learning speed, robots that employ imitation-enhanced Q-learning to reach a target location are used in this chapter. The robot arena is divided into multiple compartments, of size 120cm x 120cm and a robot resides in each of them (figure 6.1). Figure 6.2 shows two robots in their separate compartments performing individual learning. Similar to the previous chapter's settings, each compartment is divided into 10x10 grids and a look-up table method is used to store Q values for each state-action pair. The tracking system in figure 3.3 acts as a GPS server and so it broadcasts the location of each robot over the network. Upon receiving this information, the robots determine their current state and decide on their next action. They start from the top-left corner of their compartment and they can move to eight different directions of the compass to get to their next state. An experiment track ends when the robot reaches its target location which is placed in the bottom-right corner of the compartment. One experiment run ends when 15000 moves are enacted by the robot. Compared to a simulation with the same settings, the real robot experiments need a much longer time period as an experimental run requires around 10 hours.

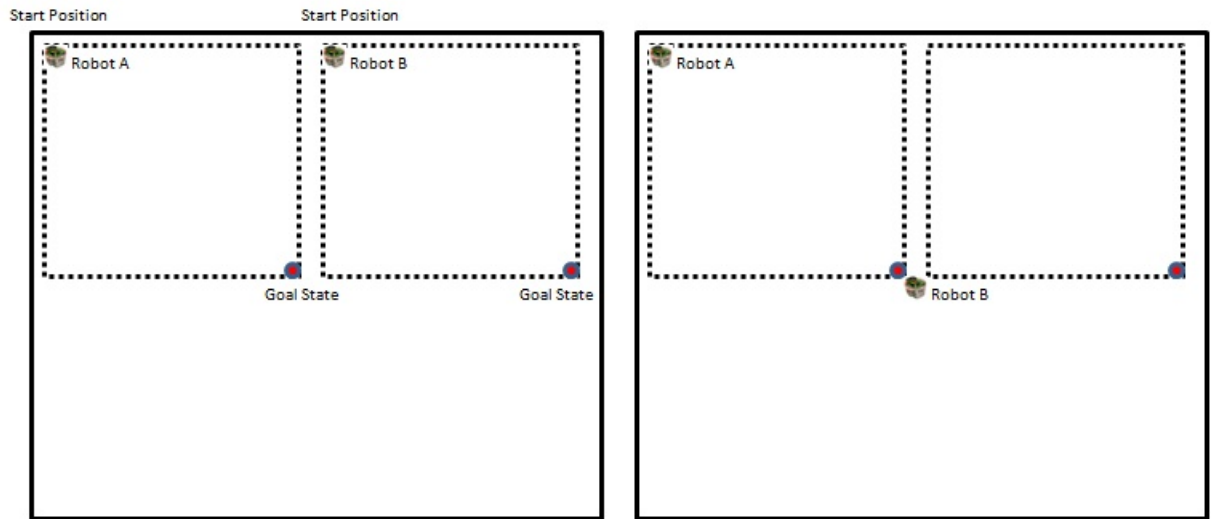


Figure 6.1: The robot arena is divided into multiple compartments and a robot resides in each of them. The figure on the left shows two robots, robot A and robot B, in their own compartments, performing their individual learning. In the figure on the right, robot B is watching and copying actions performed by robot A.

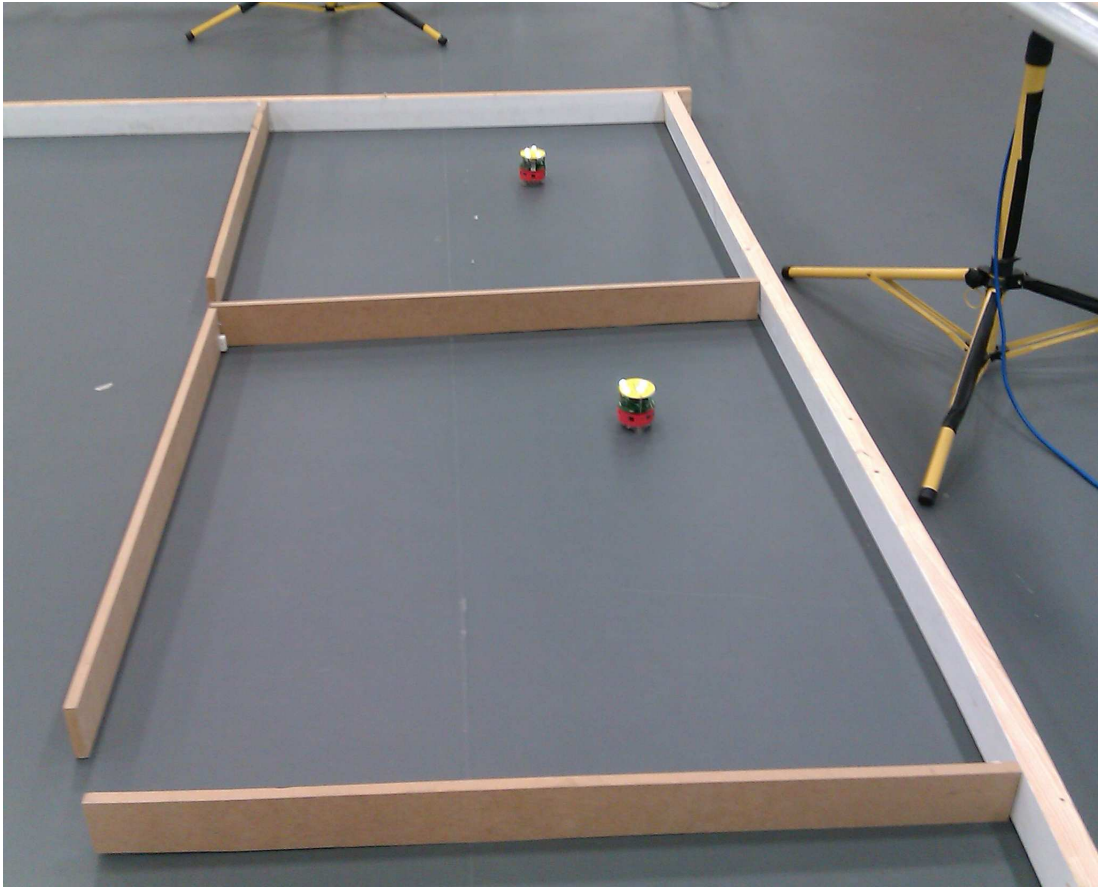


Figure 6.2: Two robots performing individual learning in their separate compartments.

The learning robot uses an ϵ -greedy algorithm in which, at every state transition, it finds the action that has the highest Q value estimate for its current state. With probability of $1 - \epsilon$, it chooses the action with the highest Q value and, with probability of ϵ , it chooses a random action. In this way, it updates the Q value for its current state and selected action using the update formula explained in the previous section. After every 10 moves, the shortest path that is learned by the robot is determined by using a greedy action selection method on the current Q values.

The robots are able to watch and copy the actions that are enacted by other robots, using the embodied imitation algorithm presented in chapter 3. In accordance with the experiment scenario in chapter 5, each time a robot needs to observe another robot, it moves to the bottom-right corner of the compartment of the robot that it plans to imitate. The imitating robot then watches the actions enacted by the observed robot for approximately 30 seconds. Once its observation finishes, the observing robot returns to its compartment and continues its individual learning.

6.3 Experiments

6.3.1 Learner Robot Copying an Expert Robot

In the first set of experiments, the learning speed of a robot, that is able to watch and copy the actions that are executed by an expert robot, using the embodied imitation algorithm presented in chapter 3, is examined. In order to do that, a robot (expert) is programmed to follow the shortest path to the target location indefinitely in its compartment. A second robot (learner) is then placed in another compartment and starts learning using the imitation-enhanced Q-learning method. The pseudo-code of the controller of the learning robot is given in algorithm 6. At the beginning of the experiment, the learning robot moves to the bottom-right corner of the compartment of the expert robot and randomly copies 5 consecutive moves of the expert. It then moves back to its own compartment and starts its individual learning. At the end of every 10 experiment tracks, the learning robot copies a new path of the expert robot. The learning robot has a memory of 10 paths and once filled, no more paths are copied.

Figure 6.3 shows the performance of robots imitating an expert, in comparison with robots without imitation. As can be seen, the imitation-enhanced robot finds the shortest path much earlier than the robot with Q-learning only. During the experiments, copying errors occasionally occur but the learning robot is able to exploit the information coming from the expert by using imitation-enhanced Q-learning algorithm.

Algorithm 6 Pseudocode for the controller of the robot**Input:**Set $Q(s, a) \leftarrow 0$ for all state-action pairs $pathList \leftarrow \emptyset, pathChosen \leftarrow 0, actionIndex \leftarrow 0, trackCounter \leftarrow 0$ $moveCounter \leftarrow 0, s \leftarrow \emptyset$ **while** $moveCounter < 15000$ **do** **if** $trackCounter \% 10 = 0$ **&&** $trackCounter < 100$ **then** $newPath \leftarrow CopyNewPath();$ $PathList \leftarrow AddNewPath(newPath);$ **end if** $trackCounter \leftarrow trackCounter + 1$ $MoveToTopLeftCorner();$ $s \leftarrow GetState();$ **while** $s \neq GoalState$ **do** **if** $pathChosen \neq 0$ **then** $actionIndex \leftarrow actionIndex + 1$ $a \leftarrow pathChosen[actionIndex]$ **if** $PathCompleted(pathChosen) = true$ **then** update $R_{pathChosen}$ $pathChosen \leftarrow 0$ **end if** **else** $pathChosen \leftarrow SelectProbabilistically(PathList)$ **if** $p_{imitate}^{pathChosen} > random();$ **then** $actionIndex \leftarrow 1$ $a \leftarrow pathChosen[actionIndex]$ **else** $a \leftarrow \max_{a'} Q(s, a')$ **if** $\epsilon > random();$ **then** $a \leftarrow SelectRandomAction();$ **end if** $pathChosen \leftarrow 0$ **end if** **end if** **if** $pathChosen > 0$ **then** **if** $\exists a' : Q(s, a') > Q(s, a)$ **then** $a \leftarrow \max_{a'} Q(s, a')$ **if** $\epsilon > random();$ **then** $a \leftarrow SelectRandomAction();$ **end if** update $R_{pathChosen}$ $pathChosen \leftarrow 0$ **end if** **end if** $Q(s, a) = Q(s, a) + \alpha [\int_{t_1}^{t_2} e^{-\beta(\tau-t_1)} r_\tau d\tau + e^{-\beta(t_2-t_1)} \max_{a'} Q(s', a') - Q(s, a)]$ $PerformAction(a)$ $s \leftarrow GetState(), moveCounter \leftarrow moveCounter + 1$ **end while****end while**

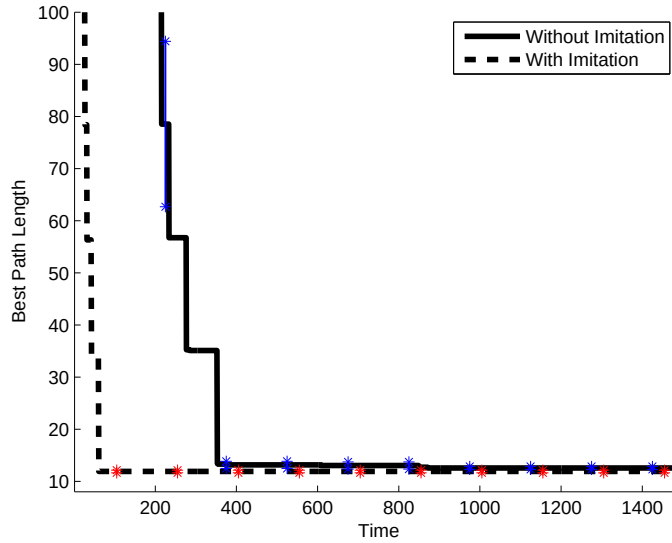


Figure 6.3: Best path length for Q-learning with imitation (copying an expert) and Q-learning with no-imitation robots. The results are mean best path length from 4 experiment runs along with 95% confidence intervals. The best path for each robot is calculated every 10 state transitions by using a greedy action selection algorithm on the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments.

6.3.2 Two Robots Copying Each Other

In the second set of experiments, two robots are placed in separate compartments, both starting to learn at the same time and able to watch each other. At the beginning of the experiment, robots, in turn, move to the bottom-right corner of the compartment of the other robot and copy 5 consecutive moves of it. They then move back to their own compartment and start their individual learning. Once again, at the end of every 10 experiment tracks, the robots copy a new path of the other robot. They have a memory of size 10 paths and once it is filled, no more paths are copied. Figure 6.4 shows results for this experiment. As can be seen, the imitating robots have a better learning speed, on average, as they are able to achieve better solutions earlier than the robots with Q-learning only. However, as expected, the difference between the two cases is smaller compared to the previous set of experiments. A close examination of the results reveals that if one of the robots makes a better start in its individual learning, by chance, then it is able to achieve better solutions sooner than the other robot. As that robot improves its performance, the other robot is able to learn from it by imitation. Therefore, imitation has a positive effect on the collective success of the robots in this scenario. On the other hand, if none of the robots make a good start,

imitation does not improve the learning speed of robots as there is nothing to be gained by copying the other robot.

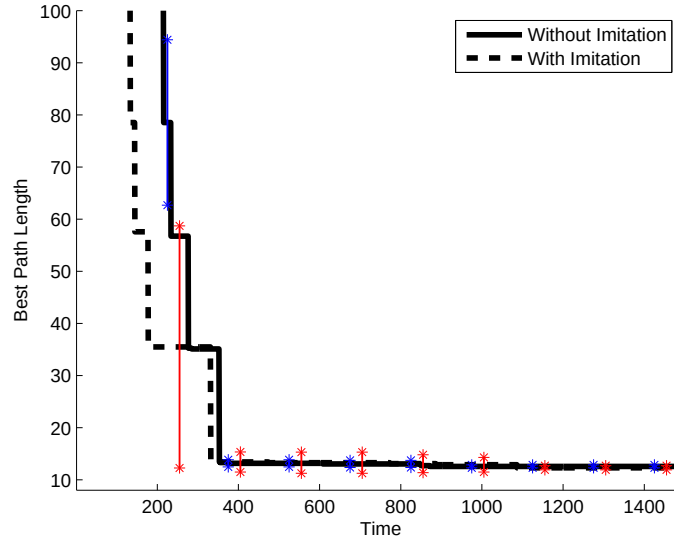


Figure 6.4: Best path length for Q-learning with imitation (two robots imitating each other) and Q-learning with no-imitation robots. The results are mean best length from 4 experiment runs along with 95% confidence intervals. The best path for each robot is calculated every 10 state transitions by using a greedy action selection algorithm in the current Q values. The β value that regulates the initial imitation probability is set to 0.5 for these experiments.

During these experiments, the robots are able to choose the paths that are more beneficial to their learning speed, by using the selection method that checks temporal changes in the Q values. Figure 6.5 shows two example paths that were copied and then enacted the highest number of times during an experimental run. As imitating these paths improved the performance of the robot, they were more likely to be enacted at the end of the experimental run and so they have the highest R_i values. Figure 6.6 shows two example paths that were copied but enacted the smallest number of times for the same experimental run. These two paths had the smallest R_i values which made them less likely to be enacted.

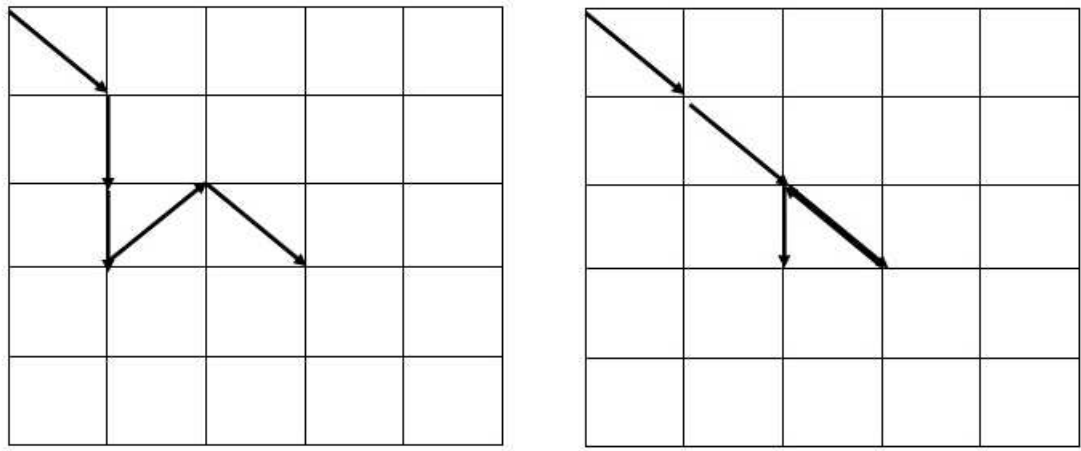


Figure 6.5: Two paths that are copied and enacted highest number of times during an experimental run.

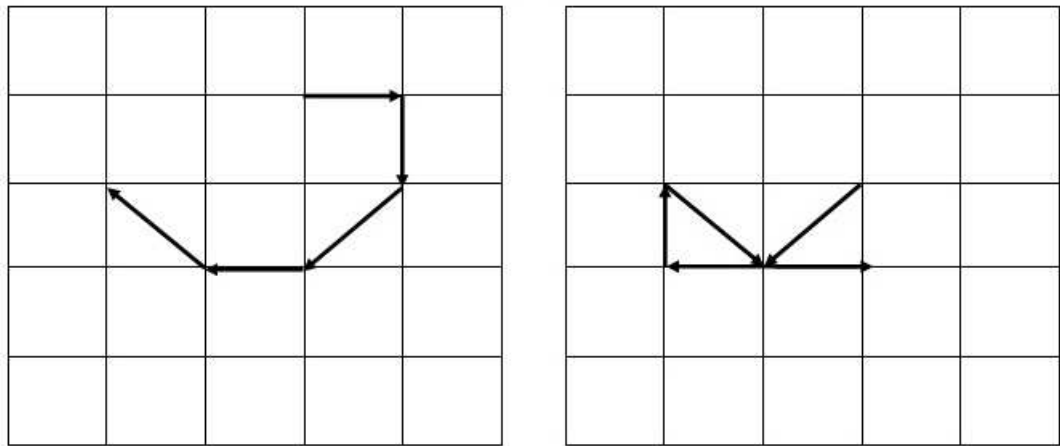


Figure 6.6: Two paths that are copied but enacted lowest number of times during an experimental run.

6.3.3 On the Effects of Embodied Imitation

As stated above, the robots use embodied imitation in which they utilise their on-board sensors to watch and copy each other's actions. Copying errors that arise from embodiment provide variation in the imitation which cause copied paths to differ from their originals. Figure 6.7 shows two example paths that were copied and then enacted the largest number of times during an experimental run, together with the original paths. As can be seen, they are not perfect copies. As the most enacted movement patterns were the result of imperfect imitations (in some experimental runs), it can be deduced that these errors may have a positive effect on the learning of robots.

Variations that arise from embodiment may have a functionality somewhat similar to the that of mutations in a genetic algorithm in that they allow different movement patterns to emerge. If these variations are by chance beneficial, they can be selected by the path selection method of the imitation-enhanced Q-learning algorithm, as in fact observed in some experimental runs.

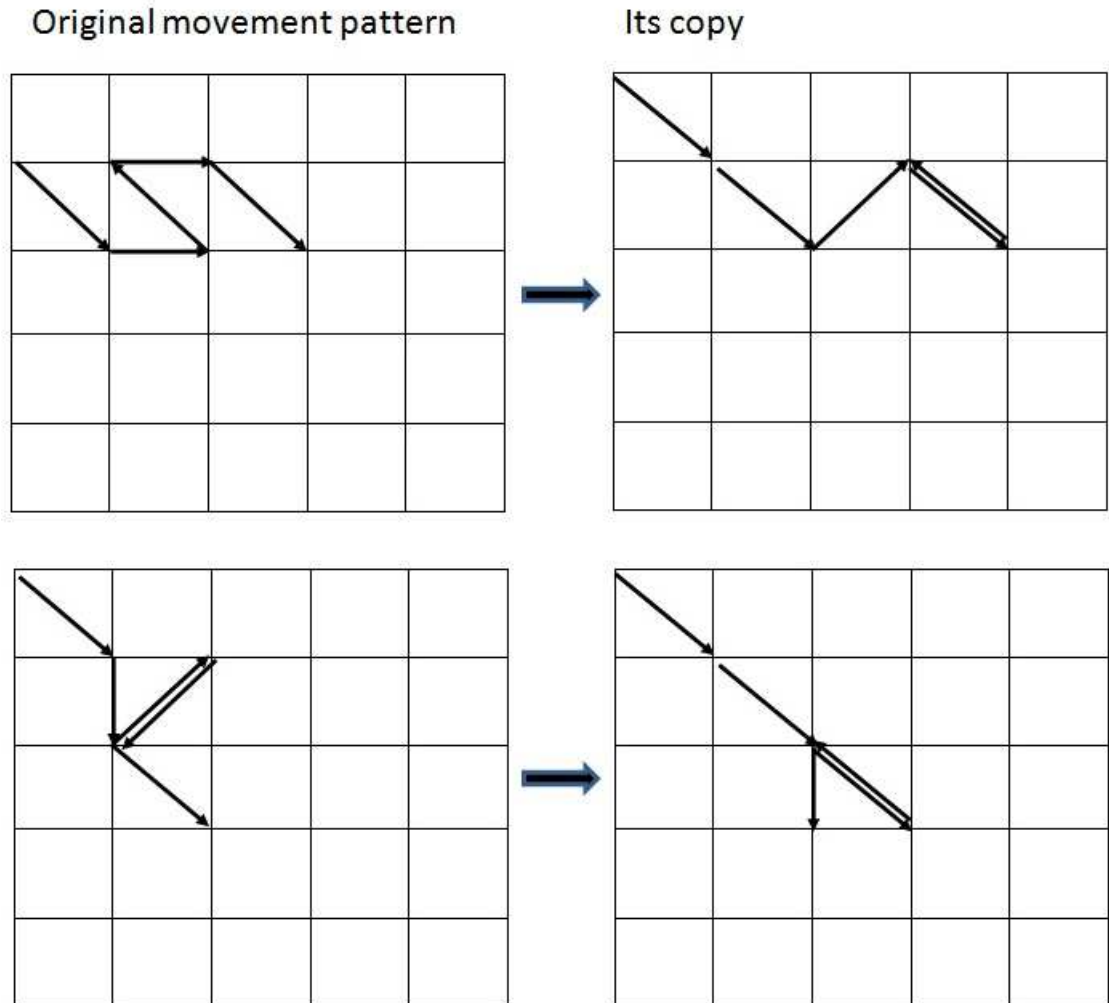


Figure 6.7: Two example movement patterns that were copied and then enacted the largest number of times. The shape on the left shows the original movement pattern of the observed robot and the figure on the right shows its perceived copy by the imitating robot.

6.3.4 Summary and Conclusion

In this chapter, a new imitation-enhanced Q-learning algorithm was tested on real robots. It was shown that information gained by imitation could be used within the Q-learning algorithm in a continuous time learning task. Robots were able to exploit imitated actions of an expert or experienced robot to improve their individual learning speed. In the case of two robots copying

each other, imitation improved the collective learning speed of the robots given that, generally, one of the robots made a better start than the other.

The effects of embodied imitation on learning speed was also examined. It was shown that variations that result from copying errors could allow different solutions to emerge. If these variations lead to a beneficial movement pattern, they could be selected by the path selection method of the imitation-enhanced Q-learning algorithm.

As it was explained in section 6.1, applying reinforcement learning methods to tasks in continuous time and space is an open research area and there are many mechanisms for different applications. Since the task our robots solved could be formalised as an SMDP, we used the method that was developed by Crites and Barto (Crites and Barto, 1996) by dividing the compartment of each robot into 10x10 grids. This simple solution allowed us to test the imitation-enhanced Q-learning algorithm on real robots. The algorithm can be further extended so that it could be used for tasks in continuous time without dividing the robot arena into grids and so allowing any type of move and turn in continuous time and space.

Chapter 7

Conclusion

Social learning, which enables individuals to learn from others in a community, is an important mechanism for social animals. One of the most important types of social learning is imitation, as it allows certain skills and behaviours to be transferred between individuals without language or other complex symbolic communication. This thesis investigated embodied imitation and behavioural adaptation in collective robotics. Firstly, by using robots to model social learning by imitation in a collective, we examined the adaptation of behaviours and their structure. Secondly, the link between imitation and individual learning was investigated and a method that used imitation as an enhancing factor to increase the learning speed of artificial agents was developed and tested.

In the first part of this work, we had a novel approach in which real robots that were able to imitate each other were used to model social learning between individuals. Our motivation for doing this was that real robots, with their less than perfect perception and actuators, provided natural variations in the imitation process which allowed new behaviours to emerge. Using simulated agents in a simulated environment, it would be possible to control the degree and types of heterogeneities and noise, but this may preclude or predetermine any emergent process that is a part of behavioural adaptation. We believe that variation in imitation is itself an emergent property of many complex interactions and any insights claimed from a simulation that attempts to identify individual sources of variation would be weakened by the assumptions made about the sources of variation.

A robot to robot movement imitation algorithm was implemented which depends on visual data coming from the image sensor of the robots. The algorithm is embodied in the sense that it completely depended on the on-board sensors of the robots as no other type of communication was

allowed between robots. Therefore the method precluded robot to robot “telepathy”, i.e. one robot transferring its trajectory or a part of its controller to another. The algorithm included *observe* and *learn* steps of imitation, in which the robot performed the following operations:

1. *observe*: The robot watched the movements of the demonstrator robot, with its on-board camera.
2. *learn*: The robot converted the observed movements of the demonstrator robot into a set of moves and turns, i.e. a trajectory, and saved the newly learned movement pattern to its memory.

Following this, a quality of imitation function was designed to assess the fidelity of imitation between robots. The quality function estimated 3 quality indicators, quality of move lengths, quality of turns and quality of number of segments, in order to calculate overall quality of an imitated movement trajectory. Although a different metric could be used to determine the quality of imitation, we believe that our quality function successfully determined the level of similarity between the original movement pattern and its copy.

Following the work of Alissandrakis et al. (Alissandrakis et al., 2004), it was shown that even with a homogeneous group of real robots, variations that arise from embodiment occurred during multiple cycles of imitation. These variations provided the evolutionary substrate for shared behaviours as they allowed new behaviours to emerge and propagate in the group of robots. It was shown that these behaviours, adapted through multiple cycles of imitation, appeared to be more robust to the process of imitation and so they could be transferred between individuals with higher fidelity. We tested three cases with different sizes of robot memories: no memory, unlimited memory and limited memory, in order to examine the effects of memory capacity on the adaptation of movement patterns. In the no memory case, the adaptation of movement patterns was extremely sensitive to any instance of poor quality imitation, which meant that the original movement patterns very quickly deteriorated and eventually vanished from the collective. In the unlimited memory case, patterns that could be easily copied emerged and spread among the robots but were unlikely to become dominant as the number of patterns in the robots’ collective memory grew larger with each new imitation cycle. However, in the case with limited memory, the adapted patterns could become dominant if they and their descendants were, by chance, chosen for demonstration. In each experiment run, we observed movement patterns that could be copied

with higher fidelity emerging. Once they emerged, they started to, in a sense, compete for dominance in the robots' collective memory. We presented an example experiment run in which a pattern and its descendants were able to spread through all robots of the group and become dominant. This process was proposed as a possible abstract yet embodied model of cultural evolution which explained the emergence and propagation of shared behaviours by imitation in a group of social agents.

In chapter 5, a method that combines individual learning and social learning of agents was presented. As in nature, imitation provided agents with model behaviours that could lead their individual learning processes. Finding in what state or context these model behaviours were useful was determined by the learning process of the imitating agent. The learning agent inferred in what context the observed behaviours were useful through a trial and error mechanism. Thus, imitated actions that caused an increase in the performance in a state were more likely to be learned. Similarly, a negative outcome made the observed behaviours less likely to be associated with that state. The only information that was transferred between agents was the set of observed actions. As there were no assumptions on the properties of the agents that were involved in imitation, it was possible that a robot could utilise imitation-enhanced Q-learning by copying the behaviours demonstrated by another robot that had a completely different internal structure and learning algorithm.

In chapter 6, imitation-enhanced Q-learning algorithm was tested on real robots. It was shown that information gained by imitation could be used within the Q-learning algorithm in a continuous time learning task. As the robots used the embodied imitation algorithm, which was implemented in chapter 3, it was shown that variations that resulted from copying errors could allow different solutions to emerge.

7.1 Summary of Contributions

This thesis investigated imitation in collective robotics. The collective dimension of imitation and its connection with individual learning of agents were examined. The contributions fall into three categories and can be summarised as follows:

In the design and implementation of imitation on real robots:

- A robot to robot movement imitation algorithm was implemented on real robots. The imitation algorithm was embodied in the sense that the imitating robot uses only its on-board

sensors to perceive the behaviours of the demonstrator robot. A quality of imitation metric was defined to assess the fidelity of imitation.

In modelling social learning:

- A novel approach that used real robots to model the imitation between social agents was implemented. It was shown that even in a homogeneous group of robots, variations that arise from embodiment allowed certain behavioural patterns to emerge during multiple cycles of imitation. Various memory lengths for the robots were tested. It was shown that the robots should be able to store, retrieve and enact their past observations to allow shared behaviours to emerge and spread in the group of robots.

In the usage of imitation to speed up individual learning in collective robotics:

- A method that uses information gained by imitation as an enhancing factor that increased the individual learning speed of agents was presented. It checked temporal changes in individual learning process of agents to determine if imitating the observed set of behaviours was beneficial or not. It was shown that imitation of purely observed behaviours enhances learning. The method was validated both on a simulation and on real robots that used embodied imitation.

7.2 Future Work

There are a great number of research questions that could be further explored following the work presented in this thesis. These include further examination of the value of the abstract yet embodied model of behavioural (cultural) evolution by, for instance, running more complex experiments with larger groups of robots. These will be discussed in the rest of this section:

On the imitation algorithm and the dynamics of imitation

1. The robot to robot movement imitation algorithm should be extended to provide feedback to the demonstrator robot, and thus allow selection mechanisms based on the success of imitation to be included.
2. The robot to robot movement imitation algorithm presented in this thesis imitated actions only. The algorithm can be extended to include the imitation of responses to sensed inputs;

this will allow the imitation of interaction so that interactions between robots can be imitated and propagated across the robot collective.

3. In the experiments presented in this thesis, we used groups of at most 4 robots as we discovered that it is the minimum size of robot group in which new behaviours that appear to be robust to variance in imitation process can emerge yet also small enough for ease of analysis. Increasing this number may help us to further explore the dynamics of behavioural adaptation in the group of robots. We could have multiple groups of robots in physically separated regions of the environment (i.e. robot villages) so that effects of environmental factors, such as region size or light intensity, on the emergence of shared behaviours can be modelled.
4. In chapter 4, imitated patterns were not linked to a task or environmental context. It should be possible and testable that using the embodied imitation approach presented, associating imitated behaviours with tasks that have utility can increase the efficiency of the robot collective. Furthermore, following the work of Billard (Billard, 1999), associating adapted behaviours with their environmental context might lead to the emergence of non-verbal communication between robots.

On the relationship between individual learning and imitation

1. Imitation-enhanced Q-learning algorithm was tested on both a simulation and real robots, with groups of two. By checking the algorithm with a larger groups, it would be possible to examine the effects of algorithm on the overall fitness of the robot collective. The algorithm checks temporal changes in the individual learning process of the agents to determine if imitating an observed path is beneficial or not. With a larger group, the problem of which agent to imitate would need to be solved. With the algorithm, the agents can select individuals to observe randomly and then the selection method of imitation-enhanced Q-learning can determine the most beneficial set of copied actions. however, in nature, agents can use environmental cues to decide on which individual to imitate. A successful member of the group would possess properties (i.e. higher energy than others) which would make it more desirable to be imitated. Adding such environmental cues that can be perceived by the learning agents can further enhance the learning speed of agents.

2. In chapter 6, it was shown that in the case of two robots copying each other start to learn at the same time, imitation improved the collective learning speed of the robots if one of the robots made a better start than the other. With a larger group, the probability of one member of the group making a better start to another would be higher. Therefore, the positive effect of imitation on the collective learning speed of robots might be more apparent if the algorithm is tested in a larger group of robots. In addition, it is shown that, with real robots, variations that result from copying errors can allow different solutions to emerge. In the experiments, the robots have a simple task with only one best solution. On a complex task with several solutions, the robots can achieve novel solutions by using the set of actions that emerge with variation during multiple cycles of imitation in the robot collective.

Bibliography

- Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of ICML 2004*.
- Ackley, D. H. and Littman, M. L. (1991). Interaction between learning and evolution. In Langton, C. G., Taylor, C., Farmer, D., and Rasmussen, S., editors, *Proceedings of the Second Conference on Artificial Life*, Reading, MA. Addison-Wesley.
- Akins, C. K. and Zentall, T. R. (1998). Imitation in japanese quail: The role of reinforcement of the demonstrators response. *Psychonomic Bulletin and Review*, 5:694–697.
- Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2002). Learning how to do things with imitation. In *AAAI Fall Symposium on Learning How to Do Things*, pages 1–6.
- Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2004). Towards robot cultures? learning to imitate in a robotic arm test-bed with dissimilar embodied agents. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 5(1):3–44.
- Bakker, P. and Kuniyoshi, Y. (1996). Robot see, robot do : An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11.
- Barto, A. G., Bradtke, S. J., and Singh, S. P. (2004). Learning to act using real-time dynacim programming. *Artificial Intelligence*, 6:105–122.
- Beckers, R., Holland, O. E., and Deneubourg, J. L. (1994). From local actions to global tasks: Stigmergy and collective robotics. In Brooks, R. and Maes, P., editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems*. Cambridge, MA, MIT Press.
- Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton.

- Bentivegna, D. C., Atkeson, C. G., and Cheng, G. (2004). Learning from observation and practice using behavioral primitives: marble maze. *Robotics*, pages 551–560.
- Billard, A. (1999). Imitation: A means to enhance learning of a synthetic proto-language in an autonomous robot. In *Imitation in Animals and Artifacts*, pages 281–311. MIT Press.
- Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Robot programming by demonstration. *Springer Handbook of Robotics*, pages 1371–1394.
- Billard, A. and Hayes, G. (2003). Drama: A connectionist architecture for control and learning in autonomous robots. *Adaptive Behaviour*, 7(1):35–63.
- Bjorklund, D. F. and Bering, J. M. (2003). A note on the development of deferred imitation in enculturated juvenile chimpanzees. *Developmental Review*, 23:389–412.
- Bjorklund, D. F. and Bering, J. M. (2006). Imitation: definitions, evidence, and mechanisms. *Animal Cognition*, 9:335–353.
- Blackmore, S. (1999). *The Meme Machine*. Oxford University Press.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.
- Bradtke, S. J. and Duff, M. O. (1995). Reinforcement learning methods for continuous-time markov decision problems. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pages 393–400. MIT Press, Cambridge, MA.
- Breazeal, C., Buchsbaum, D., Gray, J., Gatenby, D., and Blumberg, B. (2005). Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots. *Artificial Life*, 11(1-2):31–62.
- Breazeal, C. and Scassellati, B. (2002). Robot that imitate humans. *Trends in Cognitive Sciences*, 6(11):481–487.
- Browning, B., Xu, L., and Veloso, M. (2004). Skill acquisition and use for a dynamically balancing soccer robot. In *Proceedings of the 19th National Conference on Artificial Intelligence*.
- Bruce, J., Balch, T., and Veloso, M. (2000). Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of IROS 2000*, pages 2061–2066.

- Bunyar, T. and Huber, L. (1999). Pushor pull: An experimental study on imitation in marmosets. *Animal Behaviour*, 54:817–831.
- Byrne, R. W. and Russon, A. E. (1998). Learning by imitation: a hierarchical approach. *Behavioral and Brain Sciences*, 21:667–709.
- Calinon, S. and Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of ACM/IEEE International Conference on Human-Robot Interaction*, pages 255–262.
- Campbell, F. M., Heyes, C. M., and Goldsmith, A. R. (1999). Stimulus learning and response learning by observation in the european starling in a two-objective/two-action test. *Animal Behaviour*, 58:151–158.
- Campo, A., Gutierrez, A., Nouyan, S., Pinciroli, C., Longchamp, V., Granier, S., and Dorigo, M. (2010). Artificial pheromone for path selection by a foraging swarm of robots. *Biological Cybernetics*, 103(5):339–352.
- Correll, N. and Martinoli, A. (2006). Towards optimal control of self-organized robotic inspection systems. In *Proceeding of the 8th International IFAC Symposium on Robot Control*.
- Crites, R. H. and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Proceedings of the Neural Information Processing Systems 8*.
- Custance, D. M., Whiten, A., and Fredman, T. (1998). Social learning of artificial fruit capuchin monkeys (*cebus apella*). *Journal of Comparative Psychology*, 113:13–23.
- Darwin, C. (1859). *On the Origin of Species*. John Murray, London.
- Dautenhahn, K. (1995). Getting to know each other - artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 15:333–356.
- Dautenhahn, K., Nehaniv, C. L., and Alissandrakis, A. (2003). Learning by experience from others - social learning and imitation in animals and robots. In Khn, R., Menzel, R., Menzel, W., Ratsch, U., Richter, M. M., and Stamatescu, I. O., editors, *Adaptivity and Learning: An Interdisciplinary Debate*, pages 217–241. Springer Verlag.
- Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

- Demiris, J. (1999). *Movement Imitation Mechanisms in Robots and Humans*. PhD thesis, University of Edinburgh, Scotland, UK.
- Demiris, J. and Hayes, G. (1996). Imitative learning mechanisms in robots and humans. In *5th European Workshop on Learning Robots*, pages 9–16.
- Demiris, J. and Hayes, G. (2002). Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In Nehaniv, C. and Dautenhahn, K., editors, *Imitation in Animals and Artifacts*, pages 327–362. MIT Press.
- Demiris, Y. (2002). Mirror neurons, imitation and the learning of movement sequences. In *Proceedings of the Ninth International Conference on Neural Information Processing*, pages 111–115.
- Demiris, Y. and Khadhour, B. (2005). Hierarchical, attentive, multiple models for execution and recognition (hammer). In *Proceedings of ICRA 2005*.
- Demiris, Y. and Meltzoff, A. (2008). The robot in the crib: A developmental analysis of imitation skills in infants and robots. *Infant and Child Development*, 17:43–53.
- Dillmann, R. (2004). Teaching and learning of robot tasks via observation of human performance. *Journal of Robotics and Autonomous Systems*, 47(2-3):109–116.
- Dillmann, R., Kaiser, M., and Ude, A. (1995). Acquisition of elementary robot skills from human demonstration. In *Proceedings of the International Symposium on Intelligent Robotics Systems*, pages 1–38.
- Dixon, K. R. and Khosla, P. K. (2004). Learning by observation with mobile robots: A computational approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, 12:219–245.
- Dufay, B. and Latombe, J. C. (1984). An approach to automatic robot programming based on inductive learning. *The International Journal of Robotics Research*, 3(4):3–20.
- Edwards, A. L. (1976). *An Introduction to Linear Regression and Correlation*. W. H. Freeman Company.

- Eiben, A. E., Griffioen, A. R., and Haasdijk, E. (2007). Population-based adaptive systems: concepts, issues, and the platform new ties. In *Proceedings of ECCS 2007*.
- Gallese, V., Fadiga, L., Fogassi, L., and Rizzolatti, G. (1996). Action recognition in the premotor cortex. *Brain*, 119:191–202.
- Gaussier, P., Moga, S., Banquet, J. P., and Quoy, M. (1998). From perception-action loop to imitation process: a bottom-up approach of learning by imitation. *Applied Artificial Intelligence*, 12(1):701–729.
- Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323. Los Alamitos: IEEE Computer Society Press.
- Hanada, Y., Lee, G., and Chong, N. Y. (2007). Adaptive flocking of a swarm of robots based on local interactions. In *Proceedings of IEEE Swarm Intelligence Symposium*, pages 340–347.
- Hayes, A. T., Martinolo, A., and Goodman, R. M. (2000). Comparing distributed exploration strategies with simulated and real autonomous robots. In Parker, L. E., Bekey, G., and Bahren, J., editors, *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems*, pages 261–270. New York, Springer.
- Hayes, G. and Demiris, J. (1994). Robot controller using learning by imitation. In *2nd International Symposium on Intelligent Robotic Systems*.
- Hayes, K. J. and Hayes, C. (2003). Imitation in a home raised chimpanzee. *Journal of Comparative and Physiological Psychology*, 108:450–459.
- Heyes, C. M., Jaldow, R., and Dawson, G. R. (1994). Imitation in rats: Conditions of occurrence in a bidirectional control procedure. *Learning and Motivation*, 25:276–298.
- Johnson, M. and Demiris, J. (2004). Abstraction in recognition to solve the correspondence problem for robot imitation. In *Proceedings of Towards Autonomous Robotic Systems (TAROS)*.
- Kaelbling, L. P. (1990). *Learning in Embedded Systems*. PhD thesis, Stanford University.
- Kovach, J. K. and Hess, E. H. (1963). Imprinting: Effects of painful stimulation upon the following response. *Journal of Comparative and Physiological Psychology*, 56:461–464.

- Krieger, M. J. B. and Billeter, J. B. (2000). The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84.
- Kube, C. R. and Bonabeau, E. (1998). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101.
- Latzke, T., Behnke, S., and Benewitz, M. (2006). Imitative reinforcement learning for soccer play in robots. In *Proceedings of 10th International Robocup Symposium*.
- Lee, C. and Xu, Y. (1996). Online, interactive learning of gestures for humanoid/robot interfaces. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2982–2987.
- Liu, W. and Winfield, A. F. T. (2011). Open-hardware e-puck linux extension board for experimental swarm robotics research. *Microprocessors and Microsystems*, 35(1).
- Liu, W., Winfield, A. F. T., Chen, J., and Dou, L. (2007). Towards energy optimisation: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305.
- Liu, W., Winfield, A. F. T., and Sa, J. (2009). A macroscopic probabilistic model of adaptive foraging in swarm robotics systems. In *Proceedings of 6th Vienna International Conference on Mathematical Modelling (Mathmod 2009), Special Session on Modelling the Swarm*.
- Lozano-Perez, T. (1983). Robot programming. In *Proceedings of IEEE*, volume 71, pages 821–841.
- Mahadevan, S. and Connell, J. (1991). Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*.
- Mataric, M. (1994a). Reward functions for accelerated learning. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*.
- Mataric, M. J. (1994b). *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, USA.
- Mataric, M. J., Nilsson, M., and Simsarian, K. T. (1995). Cooperative multi-robot box-pushing. In *IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*.

- Melhuish, C. (2001). *Strategies for Collective Minimalist Mobile Robots*. Professional Engineering.
- Melhuish, C., Holland, O., and Hoddell, S. (1998). Collective sorting and segregation in robots with minimal sensing. In *Proceedings of the 5th International Conference on Simulation of Adaptive Behavior, From Animals to Animats 5*, pages 465–470. Cambridge, MA, USA, MIT Press.
- Melhuish, C., Wilson, M., and Sendova-Franks, A. (2001). Patch sorting: Multi-object clustering using minimalist robots. In *Proceedings of the 6th European Conference on Artificial Life*, pages 543–552.
- Meltzoff, A. N. (1988). The human infant as homo imitans. In Zentall, T. R. and Galef, B. G. J., editors, *Social learning: Psychological and biological perspectives*, pages 319–341. Erlbaum, Hillsdale, NJ.
- Meltzoff, A. N. and Moore, M. K. (1977). Imitation of facial and manual gestures by human neonates. *Science*, 198:75–78.
- Mitchell, R. W. (1987). A comparative-developmental approach to understanding imitation. In Bateson, P. and Klopfer, P., editors, *Perspectives in Ethology*, volume 7, pages 183–215. Plenum, New York.
- Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klapotocz, A., Magnenat, S., Zufferey, J. C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *9th Conference on Autonomous Robot Systems and Competitions*, page 59:65.
- Nehaniv, C. L. (2007). Nine billion correspondence problems. In Nehaniv, C. L. and Dautenhahn, K., editors, *Imitation and Social Learning in Robots, Humans and Animals*, pages 35–46. Cambridge University Press.
- Nehaniv, C. L. and Dautenhahn, K. (2002). The correspondence problem. In Nehaniv, C. L. and Dautenhahn, K., editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press.
- Nembrini, J. (2005). *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots*. PhD thesis, University of the West of England, Bristol.

- Ng, A., Coates, A., Diel, M., Ganapathi, J., Schulte, J., Tse, E., Berger, E., and Liang, E. (2004). Inverted autonomous helicopter flight via reinforcement learning. In *Proceedings of International Symposium on Experimental Robotics*.
- Niculescu, M. and Mataric, M. J. (2007). Task learning through imitation and human-robot interaction. In Nehaniv, C. L. and Dautenhahn, K., editors, *Imitation and Social Learning in Robots, Humans and Animals*, pages 407–424. Cambridge University Press.
- Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics*. MIT Press.
- Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In *Proceedings of the International Conference on Artificial Neural Networks*.
- Peters, J., Vijayakumar, S., and Schaal, S. (2003). Reinforcement learning for humanoid robotics. In *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots*.
- Pugh, J. and Martinoli, A. (2007). The cost of reality: Effect of real-world factors on multi-robot search. In *Proceeding of the IEEE International Conference on Robotics and Automation*, pages 397–404.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 25–34. New York, NY, USA. ACM Press.
- Rumelhart, D. E., Hinton, G. E., and William, R. J. (1986). Learning representation by back-propagation. *Nature*, 323:533–536.
- Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. Technical Report Technical Report CUED/F-INFENG/TR, Engineering Department, Cambridge University.
- Sahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Proceedings of SAB 2004 Workshop on Swarm Robotics*, pages 10–20. Lecture Notes in Computer Science Vol.3342, Springer-Verlag, Berlin.
- Santamaria, J. C., Sutton, R. S., and Ram, A. (1998). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behaviour*, 6(2):163–218.
- Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2004). An experimental comparison of imitation paradigms used in social robotics. In *Proceedings of 2004 IEEE Internatinal Workshop on*

- Robot and Human Interactive Communication*, pages 691–696.
- Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2007a). Experimental comparisons of observational learning mechanisms for movement imitation in mobile robots. *Interaction Studies*, 8(2):307–335.
- Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2008). What is an appropriate theory of imitation for a robot learner. In *Proceedings of ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, pages 9–14.
- Saunders, J., Nehaniv, C. L., Dautenhahn, K., and Alissandrakis, A. (2007b). Self-imitation and environmental scaffolding for robot teaching. *International Journal of Advanced Robotics Systems, Special Issue on Human Robot Interaction*, 4(1):109–124.
- Segre, A. B. and DeJong, G. (1985). Explanation-based manipulator learning acquisition of planning ability through observation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 555–560.
- Smart, W. D. and Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of the 17th International Conference on Machine Learning*, pages 903–910.
- Smart, W. D. and Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Steels, L. and Kaplan, F. (2001). Aibo’s first words: The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32.
- Stirling, T. and Floreano, D. (2010). Energy efficient swarm deployment for search in unknown environments. In *Proceedings of the 7th international conference on Swarm intelligence (ANTS’10)*.
- Stranieri, A., Ferrante, E., Turgut, A. E., Trianni, V., Pinciroli, C., Birattari, M., and Dorigo, M. (2011). Self-organized flocking with a heterogeneous mobile robot swarm. In *Proceeding of ECAL 2011*, pages 789–796.
- Strosslin, T. and Gerstner, W. (2006). Reinforcement learning in continuous state and action space. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Sutton, R. S. and Barto, G. B. (1998). *Reinforcement Learning*. MIT Press.

- Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative process in animals. *Psychological Review Monograph*, 2:551–553.
- Thorpe, W. H. (1963). *Learning and Instinct in Animals*. Methuen, London.
- Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the fourth Connectionist Models Summer School*.
- Tomasello, M., Kruger, A. C., and Ratner, H. H. (1993). Cultural learning. *Behavioral and brain sciences*, 16(3):495–552.
- Trianni, V., Gross, R., Labella, T., Sahin, E., and Dorigo, M. (2003). Evolving aggregation behaviors in a swarm of robots. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J., and Ziegler, J., editors, *Proceedings of the 7th European Conference on Artificial Life (ECAL)*. Volume 2801 of Lecture Notes in Artificial Intelligence, 865-874, Springer Verlag, Heidelberg, Germany.
- Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University.
- Zentall, T. R. (2001). Imitation in animals: Evidence, function, and mechanisms. *Cybernetics and Systems*, 32:63–96.

Appendix A

Imitation on Real Robots

```
0 15 120 15 240 15
271 15 21 15 142 19
275 18 94 3 29 15 178 14
271 16 35 15 159 14
274 18 37 17 154 16
272 12 27 17 175 15
262 15 83 3 185 8
271 15 15 14 91 4 150 18
320 8 269 7 91 3 4 13
265 18 92 3 196 12
```

Figure A.1: A snapshot from the memory of a robot. Each line records the trajectory of a demonstrator robot, according to the calculations of the imitating robot. The structure of each line is: New angle / Length of straight move / New angle / Length of straight move (repeats). For instance the first line records a equilateral triangular trajectory: It starts with a 15 cm forward move, then a 120° counter clockwise turn, a 15 cm straight move, a 120° counter clockwise turn and a 15 cm straight move. Similarly the second line records a trajectory which consists of a 15 cm straight move, a 110° counter clockwise turn, a 15 cm straight move, a 121° counter clockwise turn and a 19 cm straight move.

```

1284650484: signaling
1284650514: imitating
1284650515: Read Path List
1284650515: 0 15 120 15 240 15

1284650515: Path Chosen to be executed
1284650515: 0 15 120 15 240 15

1284650533:
1284650535: foraging
1284650558: checking
1284650562: waiting
1284650564: tracking
1284650569: recording
1284650570: New Path Stored
1284650570: signaling

```

Figure A.2: A snapshot from the internal log file of a robot. It records state transitions and internal calculations with their timestamps.

```

9,1284650484.369793,21809,19074,14,6
9,1284650484.425788,21808,19075,15,6
9,1284650484.485785,21809,19075,15,6
9,1284650484.546031,21808,19075,16,6
9,1284650484.607776,21809,19075,16,6
9,1284650484.666698,21808,19075,15,6
9,1284650484.726101,21808,19075,15,6
9,1284650484.785801,21808,19075,17,6
9,1284650484.846769,21808,19075,16,6
9,1284650484.905825,21808,19075,15,6
9,1284650484.967473,21808,19075,15,6
9,1284650485.026759,21808,19075,16,6
9,1284650485.086020,21809,19075,16,6
9,1284650485.146072,21808,19075,16,6
9,1284650485.205849,21808,19075,15,6
9,1284650485.265857,21808,19075,15,6

```

Figure A.3: A snapshot from the external log of an experiment. The structure of each line is: Robot ID, timestamp (in sec), X coordinate on the arena (in 0.1 mm), Y coordinate on the arena (in 0.1 mm), orientation of the robot (orientation in radians * 1000), robot state. The position and orientation of the robot is received from the tracking system. Internal state of the robots are sent by the robots.

Table A.1: The minimum length of distance change (R) that can be detected depending on the distance between two robots (D). For instance when the distance between two robots is 20 cm, one pixel change in size of the observed robot in the view of observing robot means the distance between two robots is altered by 0.1981 cm.

D (cm)	R (cm)	D	R	D	R	D	R	D	R
8	0.0340	38	0.6991	68	2.2134	98	4.5767	128	7.7882
9	0.0425	39	0.7358	69	2.2784	99	4.6707	129	7.9114
10	0.0519	40	0.7736	70	2.3441	100	4.7642	130	8.0321
11	0.0623	41	0.8123	71	2.4114	101	4.8591	131	8.1566
12	0.0736	42	0.8519	72	2.4795	102	4.9554	132	8.2781
13	0.0858	43	0.8925	73	2.5484	103	5.0531	133	8.4034
14	0.0991	44	0.9340	74	2.6178	104	5.1520	134	8.5324
15	0.1132	45	0.9764	75	2.6889	105	5.2493	135	8.6580
16	0.1283	46	1.0198	76	2.7601	106	5.3505	136	8.7873
17	0.1443	47	1.0642	77	2.8329	107	5.4496	137	8.9206
18	0.1613	48	1.1094	78	2.9070	108	5.5525	138	9.0498
19	0.1792	49	1.1557	79	2.9815	109	5.6561	139	9.1827
20	0.1981	50	1.2028	80	3.0562	110	5.7604	140	9.3110
21	0.2179	51	1.2509	81	3.1328	111	5.8651	141	9.4429
22	0.2387	52	1.3001	82	3.2103	112	5.9701	142	9.5785
23	0.2604	53	1.3501	83	3.2884	113	6.0753	143	9.7087
24	0.2830	54	1.4010	84	3.3681	114	6.1843	144	9.8522
25	0.3066	55	1.4529	85	3.4483	115	6.2933	145	9.9900
26	0.3311	56	1.5056	86	3.5298	116	6.4020	146	10.1215
27	0.3566	57	1.5593	87	3.6114	117	6.5104	147	10.2669
28	0.3830	58	1.6142	88	3.6941	118	6.6225	148	10.4058
29	0.4104	59	1.6697	89	3.7779	119	6.7340	149	10.5374
30	0.4387	60	1.7265	90	3.8625	120	6.8493	150	10.6081
31	0.4679	61	1.7841	91	3.9494	121	6.9638	151	10.6788
32	0.4981	62	1.8423	92	4.0355	122	7.0771	152	10.7495
37	0.6632	67	2.1492	97	4.4843	127	7.6687	157	11.1030