Real-Time Vision-Based Gesture Learning 
for Human-Robot Interaction 
in Social Humanoid Robotics

Author: Boris Mocialov
Matriculation Number: H00180016, s1581976

Supervisor: Dr. Patricia A. Vargas
Co-Supervisor: Dr. Micael S. Couceiro
Second Marker: Dr. Kartic Subr

Report submitted in fulfillment of the requirements for the degree of MSc. in Robotics and Autonomous Systems

August 2016
Abstract

This report emphasises the importance of a research topic in the field of human-robot interaction. The research focuses on real-time learning paradigm and its application in social robotics. Learner robot learns how to react to demonstrator’s actions that convey emotional states of the demonstrator. The work focuses on a particular subset of real-time learning that deals with vision-based data acquisition in social robotics. This work is partly a continuation of the previous work in gesture recognition that gives a starting point for this project. The report enumerates the details of the project, such as objectives, assumptions, and risks, analyses the requirements, proposing the research plan that will lead to development of an integrated self-organising system. Various experiments with the developed system are described in the report and the results are compared to each other. It had been shown by the emulation that the real-time learning can be achieved using simple rules. Discussion and the implication of the results are presented in the report. The results obtained show that the chosen approach to real-time gesture learning is feasible and could be extended to more complex scenarios. Further research should tackle refinement of the system and use it to apply in human-robot interaction scenario.
Declaration of Authorship

I, Boris Mocialov, declare that this dissertation is my own original work that is being submitted to Heriot-Watt University, Scotland in partial of the Degree of Master of Science in Robotics and Autonomous Systems.
This work is submitted for the first time and has never been submitted to any university or institution of higher learning.

Signed Boris Mocialov

On 19th of August 2016.

Signature: ___________________________________________
Acknowledgements

I would like to thank my supervisor, Dr. Patricia A. Vargas, for the guidance and motivation and my co-supervisor, Dr. Micael S. Couceiro, for the technical assistance when it was needed. Also, I appreciate the help, received from Dr. Katrin S. Lohan, in the form of occasional paper, on the subject of gesture recognition, and a feedback on the report. In addition, I would like to help authors of the libraries used in this project for their bearably frequent response to my emails. Mostly I would like to thank my family for their selfless support during the tough times.
Contents

Abstract i

Declaration of Authorship ii

Acknowledgements iii

1 Introduction 1
  1.1 Project Aim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Hypothesis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.3 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.4 Report Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2 Learning by Interaction in Social Robotics through Evolutionary Robotics Techniques applied in Gesture Recognition 4
  2.1 Intelligent Robotics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    2.1.1 Evolutionary Robotics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  2.2 Social Robotics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  2.3 Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.4 Evolution of Real-Time Learning from Interaction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  2.5 Approaches to Real-Time Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.6 Data Acquisition in Real-Time Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  2.7 Real-Time Learning in Social Robotics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  2.8 Previous Work in Gesture Recognition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    2.8.1 Vision-Based Feature Extraction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
    2.8.2 String of Feature Graphs Encoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
    2.8.3 Affinity Matrix Construction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    2.8.4 Feature Detectors Evolution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
    2.8.5 Classifier Training . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3 Organisation 14
  3.1 Objectives . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  3.2 Assumptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  3.3 Requirements Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
    3.3.1 Functional Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
    3.3.2 Non-Functional Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  3.4 Research Plan . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  3.5 Performance Assessment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  3.6 Risk Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

4 Methods 22
  4.1 Scenario . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
  4.2 Hardware . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
  4.3 Methodologies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
    4.3.1 System Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    4.3.2 Development . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    4.3.3 Primitives as Innate Abilities . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    4.3.4 Communication between Robots . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    4.3.5 Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

5 Experiment with Reduced MNIST Dataset 26
  5.1 Experiment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
  5.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
    5.2.1 Fitness Evolution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
    5.2.2 Evolved Detectors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
    5.2.3 Classifier Training . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overall view of existing gesture recognition system</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>ROI Detection and Feature Extraction</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Project Plan</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Scenario Execution Sequence</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>NAO T14 Hardware</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>Reduced MNIST dataset recognition system</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>Average fitness development of detectors evolved for 250 generations for reduced MNIST dataset</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>Heatmaps of evolved for 250 generations detectors for reduced MNIST dataset</td>
<td>29</td>
</tr>
<tr>
<td>9</td>
<td>Classifier training with Resilient Backpropagation for reduced MNIST dataset</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>Confusion matrix for reduced MNIST dataset with leave-one-out strategy using one example</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>Generated affinity matrices for 4 self-made gestures dataset</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>Average fitness development of detectors evolved for 130 generations for 4 self-made gestures dataset</td>
<td>33</td>
</tr>
<tr>
<td>13</td>
<td>Heatmaps of evolved for 130 generations detectors for 4 self-made gestures dataset</td>
<td>35</td>
</tr>
<tr>
<td>14</td>
<td>Classifier training with Resilient Backpropagation for 4 self-made gestures dataset</td>
<td>36</td>
</tr>
<tr>
<td>15</td>
<td>Confusion matrix for 4 self-made gestures dataset with leave-one-out strategy using one example</td>
<td>37</td>
</tr>
<tr>
<td>16</td>
<td>Generated affinity matrices for 10 ChaLearn gestures dataset</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>Average fitness development of detectors evolved for 120 generations for 10 ChaLearn gestures dataset</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>Heatmaps of evolved for 120 generations detectors for 10 ChaLearn gestures dataset</td>
<td>40</td>
</tr>
<tr>
<td>19</td>
<td>Classifier training with Resilient Backpropagation for 10 ChaLearn gestures dataset</td>
<td>41</td>
</tr>
<tr>
<td>20</td>
<td>Confusion matrix for 10 ChaLearn gestures dataset with leave-one-out strategy using one example</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>Generated affinity matrices for 2 self-made NAO gestures dataset</td>
<td>44</td>
</tr>
<tr>
<td>22</td>
<td>Average fitness development of detectors evolved for 230 generations for 2 self-made NAO gestures dataset</td>
<td>45</td>
</tr>
<tr>
<td>23</td>
<td>Heatmaps of evolved for 230 generations detectors for 2 self-made NAO gestures dataset</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>Classifier training with Resilient Backpropagation for 2 self-made NAO gestures dataset</td>
<td>47</td>
</tr>
<tr>
<td>25</td>
<td>Confusion matrix for 2 self-made NAO gestures dataset with leave-one-out strategy using one example</td>
<td>48</td>
</tr>
<tr>
<td>26</td>
<td>Real-time learning algorithm example for 4 iterations</td>
<td>52</td>
</tr>
<tr>
<td>27</td>
<td>Real-time learning simulation for learning 2 reactions to 2 demonstrator's gestures</td>
<td>54</td>
</tr>
<tr>
<td>28</td>
<td>Real-time learning simulation for learning 5 reactions to 5 demonstrator's gestures</td>
<td>56</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Objectives</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Assumptions</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Requirements</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Prototypes Assessments</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Enumerated Risks</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Gesture classification accuracies after different experiments</td>
<td>48</td>
</tr>
</tbody>
</table>
List of Algorithms

1. Real-time learning algorithm .......................... 50
1 Introduction

Interest in real-time learning capabilities in software arisen after it had been noticed that systems can become more efficient by adapting dynamic behaviour. Efficiency, in the context of real-time learning systems, is improved with adaptation and self-organisation. Adaptation and self-organisation use existing capabilities of a system that are often implemented as simpler primitives used to adapt to changing environments by developing complex behaviours.

Ability of a system to learn in real time can benefit the user. The user, who is the most knowledgeable about the operation domain, otherwise known as task expert, can instruct a robot to perform certain desired actions or give a high-level description of the task that needs to be accomplished. Similarly, robot experts, developing the hardware and software for a robot, benefit from spending less to no time at all on collection of training data.

For the user to be able to instruct a robot to perform a new task, some form of communication must be established between the two parties. The aim in human–robot interaction is to empower the user and give as much control and freedom as possible when operating a robot. Therefore, the communication mechanism should be simple yet expressive.

Apart from the ability of explicit training of some new task, the robot should be able to track the behaviour of the user and become more stable and efficient in day-to-day operations on the known tasks.

Beyond strengthening the trained or user–taught behaviours, advanced sensing and processing capabilities on board of a robot should support lifelong learning from observations.

1.1 Project Aim

The aim of this project is to show that evolutionary robotics techniques and real-time supervised learning can complement each other when applied to sub-problems in HRI, such as real-time distributed learning of gestures in contrast to the whole-system’s approach.

1.2 Hypothesis

Evolutionary robotics and real-time learning approaches could complement each other when applied in HRI, tackling sub-problems, rather than developing a single model for a complex problem.

As opposed to the whole-system approach, where multiple factors are considered when designing a unified model, divide-and-conquer approach could be a better alternative.
Modularity, in general, simplifies problem solving, dividing complex problems into sub-problems and solving them individually. Therefore, it is hypothesised that splitting the complex problem into sub-problems and attacking each sub-problem with specific techniques, in particular, ER and real-time supervised learning techniques would result in a more tractable solution. ER techniques will be used in this project due to the personal interest in using genetic algorithms in various applications. Apart from the mere interest, it is necessary to explore ER generalisation capabilities in different domains. Ability to learn in real time as well as the advantages of having a robust and adaptable controllers had been discussed earlier in this chapter. Robots will be given the ability to learn in real time and to have a system on board that would be able to update itself in real-time sufficiently, despite the low processing power.

1.3 Motivation

Robots begin to appear more frequently among the general public on the street, at home, in shops, at workplaces. Moreover, the term ‘robot’ is becoming vague as robot-like devices accompany everyday life and almost every electronic device has some sort of intelligent behaviour. As all these devices interact with people in some way, they all rest under the human-robot interaction umbrella. Unfortunately, most of these devices have no idea about their users’ characters and habits, except for the device usage statistics. The device is not able to improve or suit its users changing preferences by not being able to analyse personal characteristics of the users. By inferring users’ characteristics, devices could self-organise and adapt to users, making the interaction pleasant for the user, which, in turn, could increase the popularity of that device. The only thing the device would require is an extra bit of memory for every user and very simple update rules to update the observed pre-defined characteristics. Apart from an active real-time learning capabilities, devices could allow their users to teach the devices new tasks through demonstrations and examples, which, in turn, would give extra control to the users, generation of new ideas through active user participation in domestication of a robot.

1.4 Report Outline

The report starts off by giving introduction to dynamic behaviour in intelligent programs and how this can benefit the user. The aim together with hypothesis and the general motivation for the project are presented after in the introduction chapter. The hypothesis lays out a exploratory path towards reaching the aims of this project.

Then literature review is split in two parts, first part gives a high-level view of such fields
as intelligent robotics, evolutionary robotics and emergence of complex behaviour, and social robotics with psychological affects it has on humans. The second part of the literature review focuses on learning, in particular, real-time learning and its evolution throughout the history. Different approaches to real-time learning and data acquisition are presented, compared, and discussed. Eventually, certain past research is identified that enables real-time learning in some way in social robotics. By the end of the literature review a short overview of previous work in gesture recognition is given, which can be reviewed in detail by following the references.

The main bulk of this report consists of the organisation, methods, and experiments chapters. Organisation chapter breaks down the aim of the project into individual objectives that are needed to achieve the overall goal. Organisation chapter also presents the assumptions and requirements for the project. By the end of the chapter, the project plan is given that tells when which task of the project had been performed. Performance assessment shows how and when the milestones from the project plan were assessed. Eventually, risk analysis describes the potential risks with the project and how these should be mitigated. The methods chapter introduces the scenario that will be used to achieve the aim of the project. Then, the hardware that was used for this project is shown and methodologies section tells which methods were employed for various milestones and tasks.

The next two chapters present conducted experiments and their respective results with different datasets. Chapters are split due to the nature of used datasets. The first dataset is the reduced MNIST dataset, which is used to test the system, while the second dataset tests recognition of different gestures. Extended discussion of both chapter results is given at the end of the second chapter, where results are compared and analysed.

At the end, the report shows results of performed real-time learning experiments. At the end of the chapter, a discussion tries to explain the obtained results, expressing some of the ideas for why certain results were achieved and what they represent.

By the end of the report, a conclusion summarises the project and gives an idea of what may be done in the future to improve this research. Appendices at the very end give links to the scripts/programs that were used to do the experiments.
2 Learning by Interaction in Social Robotics through Evolutionary Robotics Techniques applied in Gesture Recognition

2.1 Intelligent Robotics

Jarvis tells that adaptable intelligent robots should use sensors data when accomplishing useful tasks as opposed to robots that plan and actuate without using sensors as a feedback mechanism [32]. Even though intelligent robotics adds environment into the loop and solves the adaptation problem to dynamic environments by feeling and responding to forces in the environment, it leads to additional challenges posed by applications of intelligent robotics, such as (i) optimised kinematic/inverse kinematic derivations, (ii) advanced localisation methods, (iii) determining efficient navigation paths, (iv) advanced locomotion, (v) HRI (Human-Robot Interaction), and (vi) Emergence of collective intelligence.

Two paradigms of intelligent robotics that facilitate emergence of complex behaviour are (i) Behaviour-Based Robotics (BBR) and (ii) Evolutionary Robotics (ER). Both paradigms are bottom-up. BBR combines simpler innate abilities to achieve an emergent complex behaviour [14], while ER achieves emergence with the artificial process of evolution and survival of the fittest [39], [55].

2.1.1 Evolutionary Robotics

ER techniques are extensively employed in designing controllers and/or body morphologies that are used by intelligent robots. ER key component is evolutionary computation that evaluates a range of solutions for every generation and propagates some selected solutions into next generations. The evaluation of a potential solution is done with the fitness function (cost function), which tells how good the current solution is independent of other solutions in the population. In ER, evolved complex behaviour (which consists of simpler behaviours) is evaluated with fitness function, which factors are tailored to evaluate simpler underlying behaviours.

2.2 Social Robotics

Available to date robots are used widely as research platforms. Social robotics is a very broad subject that encapsulates research on robotic platforms among others on learning, adaptation, cognition, developmental psychology, embodiment design, system engineering,
societal organisation. On one hand, the aim of the various research on these platforms is to advance the algorithms on planning, navigation, and manipulation. The bigger picture, on the other hand, is obtained by studying the effects of these algorithms, implemented on the embodied agents in real environments, where humans or other robots are present. During the course of research, the robots are given certain roles, while operating in social settings. Robots may assume a role of a partner or an assistant within the bigger picture in social robotics. No matter what the role is, Terrence Fong et al. point out that a social robot has to be adaptable and flexible to spur humans to interaction and engage them to cooperate [22].

Following authors’ focus on research in learning in social robotics. Bandera Rubio and Juan Pedro define social robot in [42] as a robot that is able to learn from others in a society of other robots, humans, or both. More specifically, Cynthia Breazeal reduces social robotics to robots interacting with humans through the process of social exchange in [11]. One such approach can be studied through imitation games [12] during which a robot imitates behaviour of a human by which it explores its own motor abilities. Such bottom-up way of learning can be used to study how humans interact with robots and how robots explore their abilities through imitation and at the same time use new abilities to communicate with humans.

2.3 Learning

According to Stuart J. Russell and Peter Norvig as well as Richard S. Sutton and Andrew G. Barto, to be intelligent means being able to learn [44] [53]. Sutton and Barto, for example, identify tradeoff between exploration and exploitation in learning. The tradeoff describes the choice the agent must take of either exploring for better future actions or exploiting the learned actions. In this case, reinforcement learning technique attempts to model this tradeoff. Different techniques have different approaches to data modeling.

No matter what sort of learning is chosen, the aim of a learner robot is to map some inputs to the outputs by finding a function that would be able to generalise from a set of examples to all possible future examples. When the function is found, it can divide the input space into decision regions and identify boundaries for every decision. Once the input space is divided, the robot can make decisions regarding new unseen previously inputs. The challenge is to eliminate misclassifications altogether. The input space can be divided in few ways. Every decision in the space can be separated with an imaginary straight line, identifying different classes. Examples of these classifiers are Naive Bayes classifier, perceptron, etc. These methods assumes that the input space is linearly separable and classes are mutually exclusive, which is not always the case. When the input space is known not to be linearly separable, a simple approach could be to slice the input space with many straight lines (e.g. Boosting, k-nearest neighbour, decision trees, etc.). Otherwise, some transformation from inputs to a different base can allow splitting the input space in
non-linear fashion (e.g. Support Vector Machine with kernel substitution, Neural Network, etc.) [9].

This project aims at achieving learning from interaction with another robot.

2.4 Evolution of Real-Time Learning from Interaction

Real-time learning is a special type of learning. Learner is expected not only use the prior knowledge about the environment, but to identify the deviations from the prior knowledge about that environment and to incorporate these deviations into the current model without loosing too much information about the past knowledge.

There is no particular point in time when the real-time learning had become popular tool for social robots’ development. The idea evolved from the desire to generalise programs, written for robotic arms for the tasks involving object grasping, trajectory or motion planning. Initially, Bruno Dufay presents sequential or inductive learning approach for building programs that decide in real time which motions to execute for a certain task and generate a program to solve that particular task [20]. Next, programming by demonstration (PbD) approach is applied to real-time learning problems. Holger Friedrich and Rüdiger Dillmann claim that the problem with PbD at that time was that, although real time learning could be achieved by providing demonstration, many demonstrations were required to cover classes of tasks. Therefore, authors propose a method for generalisation of a single demonstration with the help of user intentions to cover one task class using STRIPS [21] programming language [23].

Nearly at the same time, Christopher Lee and Yangsheng Xu make a system that interacts with users using gestures and updates its knowledge about gestures [34]. Aude Billard develops a real-time learning system with loose sensory requirements and applies it on real robots to demonstrate communication grounding [6–8]. On PbD side, Paul Rybski and Richard Voyles give a mobile robot a set of basic capabilities and let it determine which of these capabilities are needed to replicate a demonstrated task [45]. Soshi Iba et al. adapts WYSIWYG (what you see is what you get) pattern when performing PbD, which empowers the user to coach the robot, by continuously being able to view the training results and, thus, being engaged in the process. John Demiris develops a system that allows a robot to match perceived movements with equivalent movements of its own, claiming that imitation can be used as a mechanism for bootstrapping further learning [18] with which Cynthia Breazeal later agrees, introducing imitative games as a mechanism for bootstrapping social behaviour in robots [12]. Stefan Schaal concludes this stream of research on real-time learning, achieved with imitation learning, on a pessimistic note, arguing that none of the existing to date approaches satisfy challenges raised with the imitation learning paradigm, such as appropriate feature extraction for perceptual data encoding, actuation primitives representation, and learning mapping between sensory input and actuation [46].
Later, Luc Steels develops action/language games used for self-organised communication grounding [49,52]. The idea of action/language/imitation games is not novel in respect to communication grounding and had been investigated and defined formally by Christo-

pher Nehaniv in [38] for evolution of sensory-motor loop, which is similar to the communication grounding as both agents establish the common meaning of needs or intentions.

More recent studies of real-time learning haven’t gone far from the initial research in PbD. In the application domain of the PbD, the tasks had become more complex, whereas the techniques remain the same [16, 42, 29, 57, 17]. Hidden Markov Models (HMM) together with Baum-Welch algorithm for parameter estimation [56] or raw sampling (e.g. position, speed, orientation) together with Dynamic Time Warping techniques (DTW) [5] for sampling comparison are used to represent and compare dynamic systems (e.g. gestures and actions). Recently Radial Basis Function (RBF) [15] have gained attention mod-

elling such systems. Many cognitive architectures are being built on top of artificial neural networks (ANN) that use either recurrent model (for short-term memory) or feedforward model for sensorimotor processing.

From the reviewed evolution of research on real-time learning, the trend tilts towards establishment of frameworks depending on whether learning is achieved sequentially, via imitation, or demonstration. Distinct frameworks tend to be biologically inspired as in the case of the neural networks that resemble the human brain or imitation learning that relates to mimicking during developmental stages in developmental psychology.

2.5 Approaches to Real-Time Learning

Many approaches to real-time learning can be distinguished throughout the identified liter-

ature. Because the field is not well-defined, the approaches are often ambiguous and contradictory in various publications. Following approaches were selected for real-time learning after careful literature walkthrough:

- Reinforcement Learning (RL)
- Artificial Neural Networks with Backpropagation (ANN BP)
- Genetic Algorithms (GA)
- Programming by Demonstration (PbD) through:
  - Imitation
  - Walk-Through/Lead-Through
  - Learning from Observation
- Sequential Learning
Iterative Learning

RL \cite{53} seeks for optimal policy based on experience. Temporal Difference (TD) learning happens without modeling the environment as opposed to Dynamic Programming (DP) techniques that assume that states of the environment satisfy Markov property and model using Markov Decision Processes (MDP). TD algorithms choose the next step based on the previously made choices. DP algorithms model the environment using Markovian Decision Properties (MDP), which assumes that the environment consists of states that satisfy Markov property. ANN BP trains neural network to adjust weights to obtain desired output given some input \cite{43}. GA applies genetic operators (mutation and crossover) to a population of data structures over a number of generations, where every data structure encodes a possible solution. The algorithm compares the desired output to the actual output (fitness evaluation) to guide the evolution \cite{25}. Both RL and GA operate in hypothetical landscape of possible solutions, trying to converge to a global optimal solution. The landscape is infinite and, therefore, there are infinite number of possible solutions. Algorithms use sophisticated approaches to ensure fast traversal of the search space. Unfortunately, finding the global optimal solution can take very long time and, in fact, not always achievable, which may be caused by premature convergence to local optima that may not even be a solution to the problem. To cope with this challenge, a set of heuristics and tricks exist that simplifies the traversal of the search spaces \cite{54}.

PbD paradigm reduces the search space of all possible solutions by guiding the traversal to the right directions on the landscape. In addition, PbD allows users to train robots to perform desired tasks, which makes this approach far more robust than RL and GA \cite{28}. PbD can be accomplished by imitation, where a robot roughly repeats a motion/action/gesture/etc. demonstrated to it \cite{46}. Another way is to either walk the robot through a particular task, physically moving the hardware or lead the robot through the task by instructing it a set of commands and intermediate points in space that it has to go through to accomplish that task. The advantage of walking the robot through the task is the chance of transferring the lifelong skill of the demonstrator onto a robotic platform.

Although both sequential learning and iterative learning may sound similar, they target real-time learning differently. Sequential learning (consecutive learning, incremental learning) evolves current knowledge at each step by consecutively seeing new training examples. Iterative learning exploits repetitiveness of batch processing training examples and uses knowledge obtained from previous batches to revise knowledge of newly arriving batches. Therefore, iterative learning operates on the training data itself, while sequential learning modifies learned model \cite{2}, \cite{47}, \cite{58}.

Overall, on one hand, RL solves a set of equations for every state to find global optimal strategy, ANN BP and GA are trained to achieve an optimal output on expected input. In PbD approaches, on the other hand, the knowledge is transferred through some medium and that knowledge is expected to be accommodated in some way by the recipient and be
used in the future. Sequential and iterative learning adjust the existing knowledge to fit the new coming knowledge. Therefore, in all the techniques, the model has to be specified by the designer and updated with the new information. There is no technique out of the listed ones that is capable of performing learning without any cognitive architecture that would already have a model capable of updating itself. The architecture would perform differently based on which learning technique is selected by the designer. In particular, ANN BP and GA would have to be trained on a set of examples, while RL together with sequential and iterative learning would have to be modelled well enough to explore the problem space. PbD approaches would have to be able to accept new knowledge and store it in their respective cognitive architectures.

Originally, selected approaches were mutually exclusive, although there is research that combines RL with ANN BP [3], RL with GA [37], PbD with GA [33].

2.6 Data Acquisition in Real-Time Learning

Soshi Iba et al. distinguish four data acquisition methods in [31] used in robot programming by demonstration, which are (i) vision–based, (ii) range sensing, (iii) external wearable devices, and (iv) tactile sensing. Vision–based sensing, compared to other, has the advantage of obtaining a lot of data from the environment. On the downside, to be able to build general frameworks, a standard has to be in place that would tell what data is important and what can be ignored. Unfortunately, the standard is unrealistic because one set of data may be crucial in one application and insignificant in another application and all the data from an image can not be used due to the processing speed/power limitations. Therefore, some information must be ignored.

In listed available approaches to real-time learning, the data/demonstrations have to be recorded using some sensor. Different approaches employed mean different sensory preferences. The following methods are available for obtaining the information:

- Teleoperation
- External devices-based motion tracking
- Vision-based motion tracking
- Kinaesthetic
- Shadowing

In teleoperation, the robot is recording operator’s input, while the operator is being relayed the haptic feedback [30]. Motion tracking external devices (e.g. data gloves, motion capture suit, etc.) provide less noisy data, but are obtrusive to the user. Kinaesthetic tracking had been mentioned previously as the walk-through procedure. During kinaesthetic
user’s input, the robot records changing parameters of own structure (e.g. orientation, position, joint angles, etc.). During shadowing, the robot records changing parameters, while mimicking the demonstrator. Many authors, who research real-time learning by imitation, observation, or demonstration with humanoid robots, like in [4], [8], [12], [17], [18], [45], [49-52], [57], focus on vision-based data acquisition method with an exception in [29], where Kinect is used additionally. The vision-based data can convey more information than other listed methods. In addition, vision-based sensing does not impede user’s operation and requires relatively cheap hardware.

Independent of what method is used for obtaining the information from the environment, the information must be condensed and encoded in a convenient for the task form that would strip the data from all the present noise. Real time domain implies that the acquisition and encoding of the incoming data must be done in real time since delays are obviously not desirable in real-time systems as they lead to confusion and frustration.

2.7 Real-Time Learning in Social Robotics

This literature review focuses on previous research in social robotics that uses vision-based sensing and looks at how it can be extended to improve human-robot interaction with the help of real-time learning.

Early research had been driven by the desire to make a system that would be able to learn in real time. Resulted frameworks with different designs, reported in [8], [18], and [45], showed that such systems are feasible both in simulations and on real robots. Ambitions for the future work ranged from extension of developed systems to accommodate object-related manipulations in human-robot interactions to application in physical robot-robot interactions.

Later, research had been more directed towards social robotics with the aim to build robots that would show social behaviour that would manifest in such phenomenon as shared attention and turn taking that are very common among humans. Most of the backbone algorithms are reused from previous research on cognitive architectures, while additional modules are added to facilitate social behaviour in robots [12], [49].

The latest research attempts an implementation of recently discovered spiking neuron system using available processing algorithms [17]. In addition, application becomes more complex, which requires development of tailored system [57].

Selected publications indicate that the research in real-time learning is headed towards complex novel applications of the technique. Unfortunately, sometimes cognitive systems are either not suitable or not desirable for new application areas, in which case new architectures are being developed.
2.8 Previous Work in Gesture Recognition

Proposed project utilises parts of previously implemented unpublished gesture recognition system [36]. The system consists of multiple layers, where every layer performs processing and transformation of visually perceived gesture data and outputs probability of an input for every learned gesture.

![Figure 1: Overall view of existing gesture recognition system](image)

- a) extracted features from sequence of frames in a video stream
- b) String of Feature Graphs (SFG) [24] for the video stream
- c) affinity matrix for the SFG
- d) HyperNEAT [48] evolved feature detectors (artificial neural networks)
- e) classifier artificial neural network
- f) resulting classification of the video stream in a)

The system is designed to detect and recognise gestures of a human, therefore, the system searches for human body parts (head and torso) by using Haar feature-based cascade classifier [40] from the standard OpenCV library. Once the head and the upper body are detected, the system performs background subtraction and searches for moving objects that originate from the upper body, such as limbs (e.g. arms).

The system lacked extensive testing at the start of the project. Therefore, considerable amount of time had been spent in the beginning on testing the system to build confidence in its performance before moving to the next step. All the tests are described in detail in Chapter 6.


The latest implementation of the recognition system can be found in Appendix E.

---

1OpenCV library [http://opencv.org/](http://opencv.org/)
2Python Evolutionary Algorithms [https://github.com/noio/peas/](https://github.com/noio/peas/)
2.8.1 Vision-Based Feature Extraction

Feature extraction begins with region of interest (ROI) detection, which detects where action in video is happening by performing background subtraction, followed by illumination reduction and edge enhancement with the help of erosion and dilation. Features are extracted from detected moving objects that originate from the upper body of identified human in a video. Features are represented by local kinematic features that contain the most relevant information needed for gesture representation. In theory, capturing kinematic features is less noisy than capturing more complex features, such as blobs (as in [10]) or salient spots (as in [13]) that contain collection of features, as the amount of noise is proportionate to the amount of data collected [35]. Potential limbs are analysed by looking at hull and convex defects to find break points (e.g. elbows) and smaller details (e.g. fingers).

![Figure 2: ROI Detection and Feature Extraction](image)
a) Torso b) Torso and limbs c) Torso, limbs, and limb details detection

2.8.2 String of Feature Graphs Encoding

Extracted features are encoded in feature graphs as vertices with few added edges, connecting these vertices (distance between features). One feature graph is generated per single frame and at the end of the video, all graphs are added together into a list, which becomes a string of feature graphs. Graphs is a convenient data structure for the features as it is easily extended if more features are needed to be encoded in a feature graph.
2.8.3 Affinity Matrix Construction

Symmetric affinity matrix is generated from string of feature graphs. Affinity matrix encodes similarities between all the feature graphs in the list. Every single feature graph is compared to other feature graphs in the list and the distance between the features is encoded in the affinity matrix in case if the distance is within some threshold. At the end, affinity matrix is resized to fit the detectors’ input size.

Affinity matrix construction can be viewed as a transformation step, where features are encoded using different base, which is the similarity measure between features.

2.8.4 Feature Detectors Evolution

Once affinity matrix is generated from string of feature graphs, it is used in evolution of detectors. Detectors are evolved offline and then plugged back into process similar examples in real time.

Detectors are neural networks with as many inputs as there are pixels in the affinity matrices and a single output without any hidden layers. The aim of every detector is to detect some feature within a affinity matrix and then output that feature in the form of a real-valued number.

Detectors are evolved using novelty search technique, which leads to finding detectors that search for distinct features in an affinity matrix. Eventually, evolved detectors focus on different parts of an affinity matrix.

The evolution of detectors is driven by the HyperNEAT algorithm. The same amount of processes is created as the amount of detectors needed. Every process runs an instance of the HyperNEAT evolutionary algorithm that is responsible for the evolution of a single detector.

2.8.5 Classifier Training

Outputs from all detectors are used to train the classifier. Classifier’s number of inputs corresponds to the number of detectors used and the number of outputs is the number of classes used in training. Same as detectors, classifier is trained offline and then plugged into the system. Originally, the classifier had been trained with backpropagation method. After number of experiments and multiple changes to the learning rate of the algorithm, it had been noticed that the method performs poorly. Backpropagation keeps high error and continuously misclassifies examples. Another technique will be used in this project, which will be given in Chapter 5 when presenting first experiments with the recognition system.
3 Organisation

The aim of the project, stated in the introduction is expanded in this chapter, describing in
details what is needed to achieve this aim.

3.1 Objectives

Table 1: Objectives

<table>
<thead>
<tr>
<th>#</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Primitive programming</td>
</tr>
<tr>
<td>2</td>
<td>Gesture recognition algorithm</td>
</tr>
<tr>
<td>3</td>
<td>Cognitive architecture capable of real-time learning</td>
</tr>
<tr>
<td>4</td>
<td>Integrated System</td>
</tr>
</tbody>
</table>

Gestures are a part of the scenario, given in Section 4.1. Robots have no pre-programmed
abilities to perform complex gestures as hugging or attempting to hit an opponent. There-
fore, these primitives will be programmed in advanced before proceeding onto development
of gesture recognition algorithm. The detailed description of primitives programming can
be found in Section 4.3.3.

Having the primitives, perceived gestures will be encoded and transformed into a uni-
form representation (affinity matrices) and the gesture recognition algorithm will be trained
to recognise different gestures.

Cognitive architecture will be created and updated in real time on board of a robot to
facilitate real-time learning.

All developed and tested components will be integrated in a single system and executed
on board of a NAO robot.

3.2 Assumptions

Since this project is a proof of concept, it is restricted by a set of assumptions, some of
which are identified in the table below.
Identified assumptions are split into three groups. First group consists of assumptions made for the demonstrator. It is assumed that the demonstrator has the full knowledge of the world and knows exactly what the learner must learn. It is assumed that the feedback from the demonstrator has no noise and that it is perceived perfectly by the learner every time. The reason for this is because the research for this project focuses on gesture recognition and the feasibility of the real-time learning, rather than the accuracy of the feedback. In the real-world scenario, the noise would be present and would have to be accounted for. In addition, it is assumed that at no time during the learning process will demonstrator change the intention of what has to be learned by the learner. It can be argued that this assumption is unreasonable for the real-time learning as the learner should account for the concept drifts and adapt to any change. Naturally, the complete prototype would have to account for concept drifts. At this stage, however, the problem is simplified and does not force the learner to account for everything that may be changed during the learning phase.

For both the demonstrator and the learner, red light means bad or fail, while the green light means good or success, which is useful for the interpretation of the feedback. Both robots must see each other and see face and upper body of the other robot in their fields of view, so that they can start tracking each other from the first moment the algorithm starts. It is also assumed that both robots know when a gesture begins and when it ends. The segmentation is performed manually, as the algorithm is fed cropped videos of robots performing gestures. All the gestures and the reactions are mutually exclusive. This means that two or more reactions must never be both correct to a single demonstrator’s gesture.

The environment is assumed to be static. Neither robots nor the surrounding objects change their positions throughout the experiments.

<table>
<thead>
<tr>
<th>#</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Demonstrator</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Full knowledge of the world</td>
</tr>
<tr>
<td>2</td>
<td>Feedback is noisy-less</td>
</tr>
<tr>
<td>3</td>
<td>No concept drifts</td>
</tr>
<tr>
<td><strong>Demonstrator &amp; Learner</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Red = bad, green = good</td>
</tr>
<tr>
<td>5</td>
<td>Visible face and upper body</td>
</tr>
<tr>
<td>6</td>
<td>Robots face each other</td>
</tr>
<tr>
<td>7</td>
<td>Perfect gesture segmentation</td>
</tr>
<tr>
<td>8</td>
<td>Gestures are mutually exclusive</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Static environment</td>
</tr>
</tbody>
</table>

Table 2: Assumptions
3.3 Requirements Analysis

Table 3: Requirements
H — High, M — Medium, L — Low

<table>
<thead>
<tr>
<th>#</th>
<th>Requirement</th>
<th>Objective</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Primitive execution</td>
<td>1</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>Extract features</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>Encode features</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td>Create uniform representation</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>Model training</td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>6</td>
<td>Model testing</td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>7</td>
<td>Correct classifications</td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>Create cognitive architecture</td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>Cognitive architecture update mechanism</td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>10</td>
<td>Single system</td>
<td>4</td>
<td>M</td>
</tr>
<tr>
<td>11</td>
<td>System executable on NAO</td>
<td>4</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td><strong>Non-Functional Requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Clear features</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>13</td>
<td>Consistent features</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>14</td>
<td>Simple encoding</td>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>15</td>
<td>Expressive encoding</td>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>16</td>
<td>Fast model training</td>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>17</td>
<td>Fast recognition</td>
<td>2</td>
<td>H</td>
</tr>
<tr>
<td>18</td>
<td>Fast cognitive architecture update</td>
<td>3</td>
<td>M</td>
</tr>
<tr>
<td>19</td>
<td>Instant reference table look-up</td>
<td>3</td>
<td>M</td>
</tr>
<tr>
<td>20</td>
<td>No execution delays</td>
<td>4</td>
<td>L</td>
</tr>
</tbody>
</table>
Compiled requirements are split into two groups with functional requirements in one group and non-functional requirements in another. Functional requirements usually describe the overall system in terms of its functions and are mostly prioritised higher. Non-functional requirements describe system from the perspective of usability and comfort. Although non-functional requirements that deal with delays and stability are important, these are usually prioritised lower when the system is in its conceptual phase.

Identified requirements are listed together with the objective that they belong to and the priority (high, medium, low). High priority requirements will be given the most attention as they decide the outcome of the final system.

### 3.3.1 Functional Requirements

Robot shall be able to execute a certain primitive on demand. Primitive are innate to robots and are encoded directly. Robot shall be able to extract features from the vision data. The system shall be able to encode extracted features in a suitable way and transform encoding into a uniform representation. The system should be able to train as well as to test a model using encoded features. Trained model should be able to correctly classify gestures. Cognitive architecture should be used to extract existing knowledge and be available for an update when new knowledge becomes available without losing the old existing knowledge. All the components of the system should be integrated into a single self-organising system and uploaded on the robot.

### 3.3.2 Non-Functional Requirements

Extracted features should be clear enough and not include any unnecessary noise that could potentially require more processing power, which is to be minimised. Certain features should be chosen by the designer and not changed at any time of the execution. The encoding of the features should be simple yet expressive to reduce the amount of processing power needed and still contain lots of information. The update of knowledge in cognitive architecture should be fast so that the system can function in real time.

### 3.4 Research Plan

Research consists of (i) primitives programming (ii) developing robot gesture recognition algorithm based on data from visual sensing, (iii) development of cognitive architecture that will be able to update the knowledge of the world in real time, (iv) extensive experimentation and data gathering for evaluation, (v) evaluation, and (vi) reporting.
Figure 3: Project Plan
The project starts with the first prototype that consists of programming a set of primitives (basic innate abilities) for robots that will be hard-coded and uploaded onto robots. Demonstrator is taught how to perform such actions as hitting and hugging, while the learner is taught how to evade by either moving to a side or performing blocking action and how to perform a hug.

The output of the second prototype is a gesture recognition algorithm. Gesture recognition consists of components, described in Section 2.8. Prior to training/testing, the data has to be collected. This will be done in parallel with the gesture recognition algorithm testing and debugging.

For the third prototype, a model will be developed that will play a role of a look-up table for a robot to decide what to output given some input. Update mechanism will be developed that would update the model given some new knowledge.

After all sub-components will be developed, they will be merged into a single self-organising system on board of a learner robot. Thereafter, the system is uploaded on to the robot, a set of test-cases from scenario, described in Section 4.1, will be executed and the data will be collected for analysis.

All the activities will be documented in the report, that will be submitted at the end of the project. The documentation will start after the prototype 2 will be available, describing the technical part of the implementation. The report will be updated alongside the project activities.

### 3.5 Performance Assessment

<table>
<thead>
<tr>
<th>#</th>
<th>Prototype</th>
<th>Date</th>
<th>Requirements</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Primitives programming</td>
<td>20th of May</td>
<td>1</td>
<td>Robots can perform every gesture on demand</td>
</tr>
<tr>
<td>2</td>
<td>Gesture recognition algorithm</td>
<td>17th of June</td>
<td>2-7,12-18</td>
<td>High accuracy on collected testing data set is high</td>
</tr>
<tr>
<td>3</td>
<td>Cognitive architecture</td>
<td>22nd of July</td>
<td>8-9,18-19</td>
<td>Demonstrate correct gesture classification</td>
</tr>
<tr>
<td>4</td>
<td>Integrated system</td>
<td>5th of August</td>
<td>10-11, 20</td>
<td>Demonstrate real time evolution of cognitive architecture</td>
</tr>
</tbody>
</table>

Primitives programming will be assessed by demonstrating the selected primitives (attacking to hit, hugging, and evading) are executed correctly every time they are requested.

Gesture recognition algorithm will be assessed by showing that certain accuracy over some chosen threshold is achieved, signifying correctness of the algorithm.
Cognitive architecture will be assessed by showing that it can store the knowledge and update existing knowledge and eventually learn and use the new knowledge.

Finally, integrated system will be assessed by demonstrating the learning of gestures in real time on NAO robot and that robot will be capable of associating gestures with responses.

### 3.6 Risk Analysis

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L</th>
<th>I</th>
<th>Resolution Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Robot breaks or does not work</td>
<td>L</td>
<td>H</td>
<td>Request a new robot meanwhile continue working in Webots simulator</td>
</tr>
<tr>
<td>2</td>
<td>Robot is not capable of extracting consistent features from visual cues</td>
<td>L</td>
<td>H</td>
<td>Choose other features or add an external camera/kinect</td>
</tr>
<tr>
<td>3</td>
<td>Classification is consistently wrong</td>
<td>M</td>
<td>H</td>
<td>Test the implementation for errors or choose other classification method</td>
</tr>
<tr>
<td>4</td>
<td>Training is too slow</td>
<td>H</td>
<td>M</td>
<td>Simplify the model or use additional resources</td>
</tr>
<tr>
<td>5</td>
<td>EA is too slow on the robot</td>
<td>H</td>
<td>M</td>
<td>Simplify the algorithm, use additional resources, or look for alternative solutions</td>
</tr>
<tr>
<td>6</td>
<td>Integration of sub-systems is impossible</td>
<td>L</td>
<td>M</td>
<td>Look for workarounds that would allow pseudo-integration</td>
</tr>
<tr>
<td>7</td>
<td>System is not executable on NAO</td>
<td>L</td>
<td>H</td>
<td>Fallback onto a PC solution, substitute components that are causing this behaviour</td>
</tr>
<tr>
<td>8</td>
<td>Lack of time to implement everything</td>
<td>M</td>
<td>M</td>
<td>Reduce the scenario and use existing solutions for sub-systems</td>
</tr>
</tbody>
</table>

Robot-related risks have low likelihood as the robots are relatively stable. In case of a possible issue, it will be reported and alternative hardware/robots will be used. If no other robots or hardware is available, simulators, such as Webots\(^4\) will be used as a fallback solution.

Risks with the functional requirements have medium likelihoods due to the difficulty of predicting the amount of work needed for a particular feature. The impact is medium because such issues can be resolved relatively fast with the help of the peers in the community or with extra timing, diverted from other project activities.

\(^4\)Webots development environment https://www.cyberbotics.com/overview
Qualitative risks are marked with high probability. Slow execution of the system is expected due to the low amount of resources present on NAO robots. The impact is marked from medium to high, because some of the requirements may not be met due to the limitation in processing power. Developed algorithms may be simplified in order to achieve the fastest possible processing on board of robots.
4 Methods

Supervised distributed learning approach is chosen for the following scenarios using two NAO robots (demonstrator and learner).

4.1 Scenario

Figure 4: Scenario Execution Sequence

Figure 4 shows scenarios execution sequence. Demonstrator communicates its emotional state to the learner with a gesture. Two emotional states are considered for the scenario: anger and happiness. The demonstrator is given a set of primitives (actions) to execute based on the current emotional state. The primitives for the angry state consist of the demonstrator raising hand up and approaching the learner (trying to hit). The primitives for the happy state consist of the demonstrator spreading arms and approaching the learner (trying to hug). Learner is expected to select appropriate reaction for the two emotional states, namely, anger and happiness and evade being hit or hug back. The demonstrator sees the reaction of the learner and signals back the feedback on whether the reaction is correct (green light) or wrong (red light). Therefore, the learner is expected to associate gestures with reactions, where lights have an innate meaning for the learner. Learner’s reaction primitives include evasion by moving to the side or blocking gesture (avoiding being hit by angry demonstrator) and spreading arms (hugging back the demonstrator).

Lookup table is used as a cognitive architecture of the learner and is updated in real time using simple update rules.

In case if risks, identified in Section 3.6, will have a major negative impact on the completion of the project, the scenario will be simplified. The simplified scenario will comprise an algorithm, capable of learning mapping between gestures and reactions in real time without being implemented on board of a robot. The algorithm can be tested on a PC and proven to be scalable to be implemented on real robots in the future.
4.2 Hardware

NAO humanoid robots are used for this project, built by Aldebaran-Robotics. Range of sensors is present on board of a NAO robot: 1. two axis gyrometer in the centre of the body, 2. one three axis accelerometer, 3. two sonar emitters and two receivers, 4. tactile sensor on the head, and 5. two video cameras in the head. The robot features fourteen degrees of freedom.

NAO robots are perfectly suitable for the social robotics experiments, because, as it is pointed out by Fong et al. in [22], a robot must have an embodiment that would allow it to interact with the environment in the same way living creatures do (humans in the case of social robotics) and see things humans find salient in the environment.

4.3 Methodologies

A set of methods are used throughout the project. Choice of methodologies is split into system design, development, primitives, communication, and evaluation categories.
4.3.1 System Design

System design falls back on biologically inspired approaches, such as ER, using evolutionary approaches. The evolution evolves a neural network that serves a role of a gesture recognition component used for feature extraction. Another neural network is used for gesture classification. Cognitive architecture takes the form of a look-up table.

4.3.2 Development

Prototypes serve as a proof of concept for sub-components of the scenario described previously. With every prototype a set of posed requirements are satisfied, which guides the development process.

4.3.3 Primitives as Innate Abilities

NAO robot does not have pre-programmed primitive behaviours that are required for the scenarios described at the beginning of this chapter. Therefore, the primitives, such as hit and hug are pre-programmed prior to experiments.

Gestures are programmed by specifying a point in space, where the robot is expected to move the tip of its end-effector. There is no need to specify every joint position and orientation as the system on board of the robot takes care of possible configurations in the configuration space. The static points in space for gestures had been selected based on empirical evaluation of the executed gestures after a set of trials.

Sometimes, the gesture is slightly corrupted by adding a random small random value to the point in space to where the end-effector should be moved. This is done for more robust recognition of gestures, which is described in detail in Section 6.

The code for the primitives is given in the Appendix A.

4.3.4 Communication between Robots

Described scenarios propose to use gestures as a communication medium between the robots.

Indirect communication is proposed as opposed to other alternative ways of communication, such as, for example, direct communication. Indirect communication has advantages over direct communication with which information is unambiguously passed between the robots. Gestures may encapsulate a lot of information or may disambiguate communicated information by, for example, pointing to a specific object using very simple gesture.
Sign language is a form of indirect communication and it allows constructing gestures with infinite possibilities, where gestures are combined together to create new meanings.

4.3.5 Evaluation

HCI techniques are used for the evaluation of functional and non-functional requirements. This will include empirical testing of prototypes and gathering of quantitative and qualitative data.

Quantitative data tells how many times the robot succeeded at recognising gestures and correctly associating colour with the perceived gesture. Overall, the quantitative data allows to measure the quality of the evolved controller and how good the controller is at remembering the sensori-motor mapping.

Qualitative data gives meaning to quantitative data. In particular, having collected the observed measurements of how reliable and satisfactory the controller is, qualitative data corrects these results by looking at the behaviour of robots.

Ultimately, having collected quantitative and qualitative data, it is put through statistical analysis to measure how sufficient the developed system is.
5 Experiment with Reduced MNIST Dataset

MNIST\textsuperscript{5} dataset is a set of handwritten digits images. It may seem unusual at first to see a gesture recognition system to be tested on digits, but there is a reasonable explanation. The link that allows comparisons between gestures and handwritten digits is the image representation of the two. Both gestures and handwritten digits can be represented as images. Section 2.8 explains briefly how gesture encoding is transformed (generating affinity matrix) before it is used for training the recognition system.

The reduced dataset that was generally used for assignment purposes was obtained from one of the course leaders at the Heriot-Watt University. The dataset consists of 750 instances and every instance has 64 fields (flattened 8×8 matrix) and one target class. Every field corresponds to a non-normalised pixel intensity value, whereas the target class indicates which digit the instance corresponds to. The reason for not using the full dataset (70000 instances) is the processing time. The bigger the dataset, the longer the processing time. Since the aim of this experiment is to test whether the recognition system is usable, the experiment completion time has higher priority than the accurate classification accuracy results, because the data will not be useful for the gesture recognition experiments later. The link to the dataset is given in the Appendix B.

Architecture, shown on Figure 6 has been used in this experiment. The architecture differs from the original architecture on Figure 1 in that the pipeline is shortened. Both feature extraction and SFGs encoding are skipped and affinity matrix is substituted with a handwritten digit representation (digit 6 on the figure).

The classifier neural network has a fixed size of 50 input neurons, 300 hidden neurons in

\textsuperscript{5}MNIST dataset http://yann.lecun.com/exdb/mnist/
first hidden layer, 300 hidden neurons in second hidden layer, and 10 output neurons. This size of the network had been chosen based on a number of initial trials and this architecture had best results. The size of the network is not the main focus of this research, therefore, little attention had been dedicated to finding the most optimal network size.

5.1 Experiment

The first step is to normalise the fields’ values by setting the values to be between 0 and 1 across the whole dataset. This is done by dividing every value in an instance by the maximum value in that instance. After normalisation, different YAML files have to be created for every single instance. The reason for converting into the YAML file type is because the OpenCV library is used to read the matrices into the memory and the library expects to work with this file format. Later, every single file has to be encoded into UTF-8 format, because the library has problems operating on other encodings, which may be set as default on different operating systems.

After the YAML files are generated, the algorithm uses them to train the detectors. Since detectors’ input size varies between experiments, the instance size has to be adjusted by resizing the matrix. This is done using the standard OpenCV library ‘resize’ function with ‘INTER_AREA’ flag that preserves the structure of the original matrix.

YAML files are split into two datasets, namely, training set (20% of the original dataset) and testing set (80% of the original dataset).

The program starts with the evolution of detectors using the training set of handwritten digits images. Few child processes are automatically created by the program, where every process controls the evolution of a single detector. The evolution is being run until the fitness development reaches plateau and no more fitness improvements can be seen.

After the evolution of detectors has reached plateau, it is stopped and final detectors are extracted from program’s memory. After the detectors are stored separately, the training data is processed through them and the output of every detector is collected and stored. Same is done with the testing set. Having outputs from detectors of training and testing sets, the output is used to train classifier using resilient backpropagation, obtained from MathWorks community website.

Previous classifier, described in Section 2.8, had been trained with backpropagation. During early experiments, it had been noticed that with backpropagation, the error rate remained high, no matter which parameters were modified (e.g. learning rate). Resilient backpropagation was quickly identified as a suitable alternative since the algorithm watches the development of the error over the time and adjusts weights adaptively.

---

5.2 Results

5.2.1 Fitness Evolution

![Figure 7: Average fitness development of detectors evolved for 250 generations for reduced MNIST dataset](image)

The fitness evolution for the reduced MNIST classification problem is shown on Figure 7. The graph shows that the detectors were evolved for 250 generations and that the fitness of detectors has evolved from 9% to 18%. The evolution is relatively stable at the beginning, but the evolution fluctuations become bigger by approximately 200th generation. This signifies that the evolution of detectors reached the point at which a small change to genotype results in big change in the phenotype and all subsequent evolved detectors would be deviating from the optimum.
5.2.2 Evolved Detectors

Figure 8: Heatmaps of evolved for 250 generations detectors for reduced MNIST dataset
Figure 8 presents evolved 50 detectors after 250 generations for the reduced MNIST dataset recognition. Detectors are presented without any order. Cells in every image correspond to a connection from input node to output node in the detector. Colours represent weights of the neural network. Red colour means positive weight, while blue colour means the weight is negative, and white colour means the weight is 0. Different intensities of colours correspond to weight intensities. The more intense the colour, the greater the weight of the connection.

Evolved detectors seem to be distinct with only few identical detectors, which means that the evolution managed to separate the affinity matrix into regions and let the detectors to focus on different parts of the affinity matrix.

5.2.3 Classifier Training

![Classifier training graph](image)

**Figure 9**: Classifier training with Resilient Backpropagation for reduced MNIST dataset
Figure 9 shows training of the classifier with the resilient backpropagation for 500 iterations. The training stabilizes at about 100 iterations with no more classification accuracy improvements. The training achieves 100% classification accuracy on training dataset and 62.08% of accuracy on test dataset.

5.2.4 Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>0.05</th>
<th>0.06</th>
<th>0.02</th>
<th>0.06</th>
<th>0.02</th>
<th>0.1</th>
<th>0</th>
<th>0</th>
<th>0.12</th>
<th>0.06</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.02</td>
<td>0.68</td>
<td>0.04</td>
<td>0.02</td>
<td>0.65</td>
<td>0.02</td>
<td>0.06</td>
<td>0.06</td>
<td>0.04</td>
<td>0.02</td>
</tr>
<tr>
<td>1</td>
<td>0.04</td>
<td>0.06</td>
<td>0.02</td>
<td>0.62</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>0.02</td>
<td>0.02</td>
<td>0.06</td>
<td>0.02</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.12</td>
</tr>
<tr>
<td>3</td>
<td>0.08</td>
<td>0.08</td>
<td>0.17</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.04</td>
</tr>
<tr>
<td>4</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.15</td>
<td>0.04</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>5</td>
<td>0.06</td>
<td>0.04</td>
<td>0.27</td>
<td>0</td>
<td>0</td>
<td>0.02</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 10: Confusion matrix for reduced MNIST dataset with leave-one-out strategy using one example

Confusion matrix in Figure 10 show how many instances of a class were predicted for every actual class in normalised form. It can be inferred from this confusion matrix that the algorithm has strong belief that the digit 0 and 1 are similar to the digit 8, digit 8 similar to 3, and digit 9 is similar to digits 3 and 7.

Section 6.4 discusses results from this chapter as well as the results from the next chapter on gesture recognition.
6 Gesture Recognition Experiments

6.1 Experiment with Self-Made Gestures Dataset

After having showed that the recognition system is operational on reduced MNIST dataset, it is tested on its ability to recognise gestures. One subject had recorded 20 videos of 4 different gestures (5 trials per gesture). The gestures included raising one arm up, raising another arm up, raising both arms up simultaneously and sitting still as depicted in Fig-

Figure 11: Generated affinity matrices for 4 self-made gestures dataset
Videos with gestures are recorded of about 1-2 seconds in length (approximately 20-40 frames). The algorithm begins with training the detectors. Number of child processes are created that control the evolutionary process. Every process generates a neural network that is loaded by the main program for evaluation. In order to evaluate generated neural networks, the program loads the videos with gestures and performs feature extraction, encoding of features into string of feature graphs, and generation of affinity matrices (one per video). After affinity matrices are generated, the same procedure is performed as with the reduced MNIST dataset to train the detectors and the classifier. In particular, affinity matrices are then resized to fit the detectors’ input size. Detectors are trained using affinity matrices. When detectors are finished training, they are plugged into the pipeline and the whole training set is passed through them, collecting detectors’ activations for training the classifier neural network. Classifier is then trained with resilient backpropagation and plugged back into the pipeline. After this procedure, the pipeline is ready to process test dataset.

6.1.1 Results

6.1.1.1 Fitness Evolution

![Fitness Graph](image)

**Figure 12:** Average fitness development of detectors evolved for 130 generations for 4 self-made gestures dataset
The fitness evolution appears to be steady and continues until 130th generation where it becomes apparent that it converges at around 31% fitness when the evolution is terminated manually.
6.1.1.2 Evolved Detectors

Figure 13: Heatmaps of evolved for 130 generations detectors for 4 self-made gestures dataset
Extracted detectors after 130 generations are presented in Figure 13. Most of the evolved detectors on the figure are slightly different. This shows that the evolutionary algorithm attempted to separate the detectors, but more evolutions are required to see bigger differences.

### 6.1.1.3 Classifier Training

![Figure 14: Classifier training with Resilient Backpropagation for 4 self-made gestures dataset](image)

Figure 14 shows training of the classifier for 100 generations, after which the training does not improve anymore. The training accuracy lies between 90% and 95%, while the testing accuracy achieved 75%.
6.1.1.4 Confusion Matrix

The experiment had been conducted using leave-one-out strategy on a single case for every gesture class. With this particular case, the recognition algorithm confused the raise one arm up gesture with the not raising arms up gesture. This may have happened due to the poor feature extraction on that particular case, where the arm may not have been detected by the algorithm.

Results will be discussed at the end of all experiments in Section 6.4.
6.2 Experiment with ChaLearn Dataset

![Generated affinity matrices for 10 ChaLearn gestures dataset](http://gesture.chalern.org/data/cgd2011)

Figure 16: Generated affinity matrices for 10 ChaLearn gestures dataset

Results from the experiment with the self-made gestures showed that the algorithm can be applied to human gesture recognition, which brings the algorithm closer to the scenario proposed in the beginning of this section. Before proceeding further, it is important to experiment how the recognition accuracy changes when more than 4 gestures have to be learned. In order to see whether the algorithm is scalable, ChaLearn\(^7\) dataset is used. In particular, 10 gestures, recorded by one subject were used in this experiment, showed in Figure 16. The dataset has 60 instances (6 instances per gesture) of length between 2 and 5 seconds. The dataset is organised in an unusual way and every recording assumes

\(^7\) ChaLearn Gesture Dataset (CGD 2011), ChaLearn, California, 2011 [http://gesture.chalern.org/data/cgd2011](http://gesture.chalern.org/data/cgd2011)
the segmentation in place. Since the segmentation is not part of this project, videos were segmented manually, creating 60 instances in total.

After all videos were segmented, the same process of feature extraction, encoding, and affinity matrix generation is repeated as in the experiments with the self-made gestures.

Since this is a publicly available dataset, previous work has been done on it in the context of gesture recognition in [26, 27]. Unfortunately, the results are not comparable, because the dataset had been manually segmented and because only one part of the dataset had been used in this experiment.

6.2.1 Results

6.2.1.1 Fitness Evolution

![Fitness Development Graph](image)

**Figure 17:** Average fitness development of detectors evolved for 120 generations for 10 ChaLearn gestures dataset

The fitness evolution for this experiment, shown on Figure [17] becomes jerky after about 60 generations and then tends to converge at around 31% fitness. The evolution is terminated after 120 generations.
6.2.1.2 Evolved Detectors

Figure 18: Heatmaps of evolved for 120 generations detectors for 10 ChaLearn gestures dataset
Figure 18 presents evolved detectors for ChaLearn dataset after 120 generations. Definite variety can be seen in the detectors, although the pattern is very similar for most of the detectors.

6.2.1.3 Classifier Training

Figure 19 shows training of the classifier. Accuracy of the training set lies somewhere around 70%, while the test dataset accuracy is no greater than 50%.

Low accuracy on the test dataset can be explained by the fact that the algorithm did not change at all between different experiments. It is common to adjust the feature extraction to accommodate different datasets, especially if there is a significant difference in the datasets.
6.2.1.4 Confusion Matrix

Figure 20: Confusion matrix for the ChaLearn dataset using leave-one-out strategy for a single example. It is apparent that the algorithm has many misclassifications. In particular, the classifier thinks that the first gesture is the fourth gesture. The gestures are indeed very similar with the only difference in another hand present during the gesturing of the third gesture. This can be explained by the poor feature extraction. Same holds for gesture number 7 being mixed with gesture number 4.

The results will be discussed in Section 6.4.

6.3 Experiment with Self-Made NAO Gestures Dataset

Specific experiment, as specified in the aim of this project (recognising gestures, performed by NAO robots) had been conducted and is described in this chapter. All the previous experiments targeted evaluation of correctness of the recognition algorithm and its performance on different datasets.

Prior to the experiment, additional functionality had to be added to the main program because the past experiments assumed a human subject showing gestures on the video. Now, as the subject is robot, the subject detection had to be changed to detect a NAO robot.

6.3.1 NAO Detection

NAO detection consists of two parts. First, robot’s head is detected and then the torso. The detection is done using Haar feature-based cascade classifier, which had to be trained on NAO body parts, collected with robot’s camera. For the head, 276 \((32 \times 32)\) examples were collected in different lightning and in different orientation with the face visible to the camera. Most of the examples contained both eyes, upper and lower camera holes as well as the head’s tactile sensor panel. For the torso, 101 examples of the torso in different
orientations had been collected and converted to (32×32) icons. Torso examples contained sonars, chest button, and the chest strip, where sonars are situated (see Figure 5). All examples were stripped of all the background information by cropping out in a such a way as to avoid including the background.

Few hundred images were taken of the surroundings, where the experiment had been conducted. All the images were sliced into 32×32 clips, which were used as negative samples during the training of the Haar classifier. The total of 13301 negative images had been used for the training of the classifier to detect NAO head and 13476 negative images for the training of the classifier to detect NAO torso. Head negatives included torso clips, while torso negatives included head clips.

Head classifier had been trained for 4 stages, while the torso classifier had been trained for 11 stages, achieving fairly high error rate with many false positives identified as either head or torso of a robot. One of the reasons for this is that not all the NAO embodiment parts were included in the negative images set. In addition, more negative images of the surrounding environment were needed. Instead of re-training the Haar feature-based cascade classifier, additional filtering has been used to correct detection of where the body parts are located in a frame. From the assumptions, a robot should have another robot in its field of view and see both torso and the face. Therefore, it is natural to assume that the face should be present in the upper part of the frame, while the torso should be present in the lower part of the frame. Thus, the frame is divided into two parts. Upper part is used to search for a face, while the lower part is used to search for torso. To reduce the number of false positives, a standard ‘HoughCircles’ function from OpenCV library was used. The function operates in the same way as looking for straight lines in Hough transform in an image [19], only ‘HoughCircles’ searches for circles in the image. Searching for circles is very convenient in the case of NAO, because all positive examples, used for classifier training, contain circles, e.g., eyes, camera holes, sonars, and chest button. All false positives are eliminated, where no circles were found. This approach slowed down the execution of feature extraction part of the overall pipeline.

Both head and torso cascade files are given in Appendix C together with the positive images that were used during classifier training.
6.3.2 Experiment

Figure 21: Generated affinity matrices for 2 self-made NAO gestures dataset

The experiment was executed in the exactly the same fashion as with both self-made gestures and ChaLearn datasets, described in Sections 6.1 and 6.2. As shown in Figure 21, feature extraction, encoding of features in string of feature graphs, and affinity matrix construction were performed on the raw video data. Total of 30 instances were used for training, 15 instances for every gesture. Later, affinity matrices were used to train the detectors, output from which, in turn, was used to train the classifier.
6.3.3 Results

6.3.3.1 Fitness Evolution

Figure 22: Average fitness development of detectors evolved for 230 generations for 2 self-made NAO gestures dataset

Figure 22 presents fitness evolution on encoded and transformed NAO gestures. The evolution is very unstable, but very slowly improving. The evolution is terminated after 230 generations.
6.3.3.2 Evolved Detectors

Figure 23: Heatmaps of evolved for 230 generations detectors for 2 self-made NAO gestures dataset
Figure 23 shows evolved detectors after 230 generations. Some detectors are similar, but overall some variety is noticeable.

6.3.3.3 Classifier Training

![Graph showing classifier training](image)

**Figure 24:** Classifier training with Resilient Backpropagation for 2 self-made NAO gestures dataset

Figure 24 shows the training of the classifier using leave-one-out strategy with a single example. Although the accuracy on training dataset is around 90%, the test dataset scores 100% in just few iterations. This can be explained by the fact that only two gestures are classified, so it should not be difficult to find some pattern that distinguishes corresponding affinity matrices.
6.3.3.4 Confusion Matrix

Figure 25: Confusion matrix for 2 self-made NAO gestures dataset with leave-one-out strategy using one example

Since the accuracy on the training set is 100%, confusion matrix shows that the predicted class for two gestures is always correct.

6.4 Discussion

Chapters 5 and 6 had showed different experiments performed on the recognition system. The reduced MNIST dataset had been used to show that the detectors and the classifier can be used to do classification on a simpler problem, such as handwritten digit recognition. Later, the complexity of the problem had been increased by performing classifications on gestures. Features had to be extracted, encoded and transformed from videos of people performing different gestures. Two relatively small datasets were created just for this project. One dataset contained 4 gestures and another dataset only 2 gestures. One publicly available dataset had been used for testing the system. Results for every experiment had been collected and will be analysed and compared in this chapter.

Table 6: Gesture classification accuracies after different experiments trying to classify different amount of gestures

<table>
<thead>
<tr>
<th>#</th>
<th>Gestures</th>
<th>Recognition Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

From Table 6 it can be seen that the classification accuracy drops down as the number of gesture classes increases (2 gestures - 100% accuracy, 4 gestures - 75% accuracy, 10 gestures - 50% accuracy). Complexity of the gestures was not expected to be a major factor in the recognition accuracy. The major factor that affects the accuracy, on the other hand, is the feature extraction, which is very simple and does not account for such gesture details as seen in ChaLearn dataset. During the experiments it had been noticed that the feature extraction should be tailored to every dataset due to the variations in camera positioning.
with respect to the subject, illumination, and others. Nevertheless, the system is robust enough, considering that nothing had been changed in the system between the different experiments.

Datasets that have been used in the experiments resembled the scenario, proposed in the aim for this project. With different datasets, which may include more gesture details (e.g. sign languages), the system would have to be improved by extending the feature extraction. Currently, the feature extraction looks only at the face, upper body and the limbs of the subject. If more details would be required, the feature extraction would have to extract detailed features, as it had already been attempted and described in Section 2.8.1.

It is important to note that neither classification accuracies nor confusion matrices represent the precise accuracies. The reason for this is that during testing on test dataset using leave-one-out strategy, a single example had been used. For example, in self-made gestures dataset 20 videos (5 per gesture) had been used for training the system and 4 videos (1 per gesture) used for testing. In case of ChaLearn dataset, 60 videos (6 per gesture) had been used during training and 6 videos (1 per gesture) used for testing. For self-made NAO gestures dataset, 30 videos (15 per gesture) had been used for training and 2 videos (1 per gesture) used for testing. For more accurate results, the system would have to be re-trained for every video that is used for testing by excluding it from the training set.

Fitness development for all the experiments involving the gesture recognition is very slow. With the self-made gestures dataset fitness improves by 11% over the time of 130 generations, while with the ChaLearn dataset fitness improves by 10% after 120 generations. In the case of self-made NAO gestures dataset, the fitness improves only by 4% after 230 generations. The biggest concern is the spike in fitness that is generated in the first generations when evolving detectors for the self-made NAO gestures dataset. The only explanation for this can be that the generated detectors performed good on the training set and then the evolutionary algorithm was not able to preserve the features of the generated detectors and the information about those detectors was lost in consequent generations. Developed detectors would never be able to reach 100% fitness values, because that would mean that the values in a affinity matrix are uniformly distributed, which is not the case with the generated affinity matrices. Nonetheless, it is hard to come up with the exact maximum fitness, therefore the maximum is taken as the case with all values uniformly distributed.

Although there are many distinct detectors in the set of 50 evolved detectors for all the experiments, most of the detectors seem to have the same pattern. Most of the time half of detector has negative weights and the other half has positive weights whether the separation is vertical, horizontal or side-way brush-like. The detectors were expected to be more random than patterned, but it may as well be a deliberate HyperNEAT behaviour. It would be beneficial for the project to study how many actually unique detectors can be generated for a specific dataset.
7 Gesture–Reaction Real-Time Learning

Learner robot learns how to react to demonstrator’s actions in real time by associating demonstrator’s feedback to a set of reactions. As it had been mentioned few times, the cognitive architecture is a simple lookup table. Rows of the table are associated with demonstrator’s gestures and columns are associated with learner’s responses.

7.1 Emulation Algorithm

This section presents an algorithm that enumerates steps to achieve the real-time learning. The algorithm is designed to emulate the learning and track the updates to the confidence table.

Algorithm 1. Real-time learning algorithm

```plaintext
1: GESTURE_RECOGNITION_ACCURACY = \ldots \%
2: LEARNING_ITERATIONS = \ldots
3:
4: GESTURES = \{gesture_hug, gesture_hit, \ldots\}
5: REACTIONS = \{hug_back, evade_attack, \ldots\}
6: FEEDBACKS = \{green, red\}
7:
8: CONFIDENCE_TABLE = \[
9: \begin{bmatrix}
10: 0.0 & 0.0 & \ldots \\
11: 0.0 & 0.0 & \ldots \\
12: \ldots
13: \end{bmatrix}
14: \]
15: \]
16: \]
17: \]
18: \]
19: for LEARNING_ITERATIONS do
20: \]
21: \]
22: \]
23: \]
24: \]
25: \]
26: \]
27: \]
28: end for
```

Algorithm 1 takes into consideration gesture recognition accuracy, the number of iterations (action-reaction-feedback exchanges between the demonstrator and the learner), a set of gestures available to the demonstrator, a set of reactions available to the learner, a set of feedback, and a confidence table. Confidence table is a lookup table, which learner uses to select a reaction for a perceived gesture from the demonstrator. The table is updated after
every iteration.

For every iteration of the algorithm, the demonstrator selects a random gesture (function f1) from the set of available gestures. Selected gesture is then perceived by the learner with the probability of \((100 - GESTURE\_RECOGNITION\_ACCURACY)\) that gesture being some other gesture from the set of available gestures (function f2). Perceived by learner gesture is then used to look up the corresponding reaction from the list of available reactions using confidence table. Learner looks at a specific row in the table that is designated for the perceived gesture and then looks for the maximum value in that row (maximum confidence) of chooses at random if all values in a row are the same. The column that is associated with the maximum value corresponds to the reaction the learner will select. The demonstrator perceives the reaction with the probability of \((100 - GESTURE\_RECOGNITION\_ACCURACY)\) that reaction becoming some other reaction from the set. Demonstrator then compares the perceived reaction with expected reaction and signals back an appropriate feedback: red if the reaction is wrong and green if the reaction is correct. The learner sees the feedback and updates the confidence table.

The update of the confidence table is done in the following way. If the feedback is positive (green light), the row of the table that corresponds to the perceived demonstrator’s gesture is updated. Since all gestures and reactions are mutually exclusive, the column that corresponds to the executed reaction is incremented by some constant, while all other values in the row are decremented by the same constant. If the feedback is negative, the value of the column that corresponds to the executed reaction is decremented while all other values are incremented.

Implementation of the algorithm is given in Appendix D.
Example in Figure 26 demonstrates the update of the confidence table in the course of 4 iterations, starting with all values in the confidence table being 0.

At the iteration 1, the demonstrator gestures hug gesture, learner perceives it as a hug gesture. The learner randomly selects hug back gesture due to both confidence levels being 0 and 0. The demonstrator perceives hug back gesture, compares to the expected reaction and signals back green light that means the reaction from the learner was correct. The learner sees the feedback from the demonstrator and increments confidence value for the hug back reaction for the hugging gesture at the same time decrementing the confidence level of the opposite reaction, which is evade attack. In this case no confusion had been detected when recognizing gesture/reaction. Same holds for the iteration 2, only this time the demonstrator shows hit gesture.

During iterations 3 and 4, one of the robots misclassifies the perceived gesture, which leads to confusion between the two robots and a wrong update to the confidence table.
7.2 Experiments

The aim of experiments was to find out whether the learner will be able to first associate correctly 2 reactions to 2 demonstrator gestures and then to scale the problem up to associating 5 reaction to 5 demonstrator gestures. Experiments were run with different gesture recognition accuracies (100%, 80%, 60%, 40%, and 20%) with 500 iterations per experiment.

It had been hypothesised that the lower the gesture recognition accuracy, the less likely it is that the learner would be able to associate reactions to demonstrator gestures, because there is too much confusion between the learner and the demonstrator, which prevents learner from learning the mapping.
7.3 Results

7.3.1 Learning 2 Reactions

Figure 27: Real-time learning simulation for learning 2 reactions to 2 demonstrator’s gestures. Green line corresponds to correct predictions, while red line corresponds to wrong predictions.

Figure 27 shows simulated scenario, where a learner tries to learn 2 reactions to 2 demonstrator’s gestures. Scenarios with different recognition accuracies had been simulated to show how gesture recognition accuracy affects learner’s ability to learn gestures in real time.
For every accuracy, evolution of confidence over the number of iterations (gesture-reaction-feedback loops) is presented in graphs. Green graph shows confidence evolution of the correct prediction for the actual class (e.g. actual class A, predicted class A or actual class B, predicted class B). Red graph shows confidence evolution of the wrong prediction for the actual class (e.g. actual class A, predicted class B or actual class B, predicted class A).

Graphs show that when the gesture recognition accuracy is 100% (gesture always perceived as demonstrated gesture), the learner is able to learn the mapping between the demonstrator’s gestures and reactions starting from the first iteration. Both red and green lines’ development is stable without confusions. Once the recognition accuracy drops, lines’ development becomes unstable. This means that the learner becomes confused about which reaction should map to which gesture. The more confusion is added (lower gesture recognition accuracy), the more noisy the confidence evolution becomes. Somewhere between 60% and 40% of the gesture recognition accuracy, the learner is no longer able to learn which reactions should follow after the demonstrator’s gestures.

Another measure, the confidence level for both correct and wrong predictions, is decreasing as the gesture recognition accuracy drops.
7.3.2 Learning 5 Reactions

Figure 28: Real-time learning simulation for learning 5 reactions to 5 demonstrator’s gestures. Green line corresponds to correct predictions, while red line corresponds to wrong predictions.
Figure 28 shows simulated scenario, where a learner tries to learn 5 reactions to 5 demonstrator’s gestures. Scenarios with different accuracies are simulated. In the case of 5 gestures and 5 reactions, the confidence development is different from that in 2 gestures and 2 reactions. The main differences are that even with 100% gesture recognition accuracy, the learner would require many more iterations (about 50) to successfully separate the gesture classes. Another major difference is that with 5 reactions to learn, the gesture accuracy must be somewhere between 60% and 80%, otherwise the learner would not be able to learn which reaction corresponds to which demonstrator’s gesture.

7.4 Discussion

Chapter 7 has demonstrated the emulation of two cases of the real-time learning. In both cases, the learner has to learn a mapping between a set of reactions and a set of gestures, demonstrated by the demonstrator. Lookup table data structure had been used to achieve the mapping between demonstrations and reactions. The lookup table has been called a confidence table because it stored values, representing confidence levels of a certain reaction for a certain demonstration. The confidence table has been updated after every demonstration-reaction-feedback loop. The feedback served as a way to reinforce the confidence in a certain reaction for a certain gesture.

As a result, the evolution of the confidence levels had been plotted in graphs to show how the confidence level changes over the time. It has been found that the evolution of the confidence level remain stable when the gesture accuracy is high. Once the accuracy drops down, the confidence evolution becomes noisy and eventually renders the learner unable to learn the mapping between demonstrations and reactions as the gesture recognition accuracy passes some minimum threshold (between 60% and 40% for learning 2 reaction and between 80% and 60% for learning 5 reactions).

With the proposed data structure (lookup table), it is useful to view the problem of real-time learning as a problem of separating clusters, where correct predictions belong to one cluster and wrong predictions belong to another cluster. Ideally, the cluster with correct predictions would have to be assigned a positive confidence values while the cluster with wrong predictions would have to be assigned negative confidence values.

It is worth to direct attention to the confidence values, first, for the graphs in Figure 27, showing the learning of 2 reactions. The maximum confidence level drops as the gesture recognition accuracy drops. Second, for the graphs in Figure 28, showing the learning of 5 reactions, the maximum confidence level drops as well as the gesture recognition accuracy drops, same as with learning of 2 reactions. What is interesting is that the maximum confidence values for learning 2 reactions are much higher where the accuracy is above 60% than the values for learning 5 reactions. Another interesting moment is that for learning 5 reactions, the confidence level for wrong predictions is positive when the learner is
not able to separate correct prediction and wrong predictions classes. When looking at the graphs that show learning of 2 reactions, the confidence value flips the sign making correct predictions have negative confidence and wrong predictions have positive confidence. This signifies that in order to tell whether the learner is able to learn in real time or not, two rules must be in place. First rule would check whether the correct prediction and wrong prediction clusters have been separated (having certain distance between the clusters). The second rule would check whether the correct predictions have positive confidence, while the wrong predictions have negative confidence. If both rules are satisfied, the learner is said to be able to associate demonstrator’s gestures with the reactions in real time.

Although it is possible for the learner to learn the mapping between perceived gestures and reactions, the learner does not understand the meaning neither of the gestures nor of the reactions. In particular, when the demonstrator shows hit gesture if it is angry, the learner may learn to evade being hit over the time, but would not be able to infer the meaning.

The real-time learning approach with the lookup table is simple yet powerful enough to map correctly actions to reactions and can be scaled to learning more reactions easily. The problem with the lookup table and the confidence values updates is that the convergence is very slow as the approach has a global view of the learning progress, whereas other techniques may have a local view of the learning that may speed up the mapping process between the gestures and the reactions. Additionally, other techniques may actually be able to infer the correct mapping even with lower gesture recognition accuracies than the ones presented in the results in Section 7.3.
8 Conclusion

Report described the work done on the real-time vision-based gesture learning. Along the presented work, the report presents the aim, hypothesis, and objectives of the project, motivates the reader in this area, and gives overview over techniques in literature review chapter. The description of the performed work is split in three parts, where the first part test the system on handwritten digit recognition dataset, the second part experiments with the gesture recognition system, using different gestures datasets that are closely related to the scenario, proposed for this project. The third part describes how real-time mapping between demonstration and reaction gestures can be learned in real time using simple cognitive architecture that resembles a lookup table. The results of all the experiments are discussed in two separate sections, comparing the results of different experiments and interpreting the results.

Overall, most of the objectives had been reached, except for one objective, namely, integration of the system on the real robots. The reason for this is that more mechanisms are needed to successfully perform such integration. In particular, gesture segmentation algorithm would be required, which was not included into objectives of this research initially. Despite the fact that the main objective was not reached, it was shown in emulation that the real-time learning would be possible under certain conditions.

Although simplistic prototypes had been presented as a result of this work and no comparisons were made to the parallel research in the similar fields, the work performed can be extended to compare to other research in the future.

8.1 Future Work

There is a substantial amount of work yet to be done to improve on current state of systems and to improve the results obtained. First, accuracies and confusion matrices should be generated for every single example, excluding it from the training set and then taking the average. Second, feature extraction should be improved and tailored to every single dataset it is being used on. Third, it should be investigated how many actually unique detectors are generated per training set as currently the minimum number is unknown and 50 detectors are chosen without any good reason behind it. Eventually, the system should be used on the full dataset (e.g. ChaLearn) to be able to compare to other published work. As for the real-time learning, an actual experiment with the real robots should be conducted to see whether results from the simulation will be the same as results from the real experiment. Also, a different technique to model the learning, instead of a confidence table, could be used to make it comparable to other similar published work as well as to get better results with lower gesture recognition accuracy. Lastly, the system should be studied more in detail.
with many more experiments to understand whether it is scalable to more detailed gestures, such as, for example, sign language gestures.
References


Appendix A


Both behaviours begin by connecting to the robot using IP address and the port number. After the connection is established, the robot is set to the default position. Later, the end effector is chosen, either left arm (LArm) or both arms (LArm and RArm). Finally, a path is specified by a set of points in space through which the end-effector should be moved and then the gesture is performed, after which the robot is returned to the default position.
Appendix B

Reduced MNIST dataset: https://github.com/mocialov/MSc-in-Robotics-and-Autonomous-Systems/tree/master/Datasets/reduced%20MNIST


All the datasets give a YAML file in UTF8 encoding with the data of the affinity matrix (or a handwritten digit in case of MNIST dataset) for a specific gesture (handwritten digit). Image files show a heatmap of the data in the YAML file.
Appendix C


Training of both classifiers had been done on hundreds of images.


Appendix D


Emulator shows the evolution of the confidence table. The user has to specify gestures and reactions, which must be the same amount. User must as well specify gesture recognition accuracy, number of learning iterations (demonstration-reaction-feedback loops) and update the size of the confidence table, filling it with 0’s.
Appendix E


The link contains the main algorithm that performs all the tasks related to the pipeline, except Resilient Backpropagation, which can be found separately at [https://github.com/mocialov/MSc-in-Robotics-and-Autonomous-Systems/tree/master/RPROP_2](https://github.com/mocialov/MSc-in-Robotics-and-Autonomous-Systems/tree/master/RPROP_2).

The main file for the gesture recognition system is ’stable_version_gesture_recognition.cpp’. Compilation information is at the top of the file.

Helper files that help extract fitness for every generation or extract detectors after certain amount of generations can be found at [https://github.com/mocialov/MSc-in-Robotics-and-Autonomous-Systems/tree/master/helpers](https://github.com/mocialov/MSc-in-Robotics-and-Autonomous-Systems/tree/master/helpers).