Cooperation via Indirect Communication

Author: Boris MOCIALOV
Matriculation Number: H00180016

Supervisor: Dr. Patricia A. VARGAS
Co-Supervisor: Dr. Micael S. COUCEIRO
Second Reader: Dr. Alasdair J. G. GRAY

A thesis submitted in fulfilment of the requirements for the degree of MSc. in Software Engineering in the School of Mathematical and Computer Sciences

August 2015
Declaration of Authorship

I, Boris MOCIALOV, declare that this thesis titled, 'Cooperation via Indirect Communication' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Boris Mocialov

Date: August 20th, 2015
Acknowledgements

I would like to thank my supervisor, Dr. Patricia A. VARGAS, for the guidance and my co-supervisor, Dr. Micael S. COUCEIRO, for the technical assistance when these were needed.

Most of all I would like to express my deepest gratitude to my family for everything.
Abstract

Evolution of indirect communication between two embodied agents is the research topic addressed in this thesis. Indirect communication is a niche that should be more common in the field of intelligent robotics as it is essential in most areas where humans and robots are present in the same environment. Apart from human–robot interaction, this communication paradigm can extend traditional communication approaches. As a result, indirect communication can serve as either a supporting or a substitute tool for other means of communication. Evolutionary approach to gesture recognition will be investigated, as one agent will train another agent to understand the meaning of gestures given by the former. Cognitive architecture of the trainer will be partly evolved using HyperNEAT algorithm. The evolved controller will be tested in reduced RoboCup environment with one agent being a goalie and a trainer and another - a kicker. This report summarises studied literature on the topic, plans the implementation of the system and guides through developed prototypes that reflect the aims of the project and lead towards a unified system for gesture recognition. Devised system requires further evaluations to build the confidence in this approach to gesture recognition.

Keywords: agent, learning, environment, information, system, algorithm, behaviour, artificial neural network, robotics, communication, artificial intelligence, fitness function, knowledge, evolution, interaction, emergence, modularity, self-organisation, indirect communication, cooperation, architecture, feedback, reasoning, robocup, gesture, embodiment, cognition, hypernet, training, webots, nao
## Contents

Declaration of Authorship ........................................ i

Acknowledgements .................................................. ii

Abstract ........................................................................ iii

Contents ....................................................................... iv

List of Figures .......................................................... viii

List of Tables .............................................................. viii

List of Equations ........................................................ viii

List of Algorithms ..................................................... ix

List of Source Code ................................................... ix

1 Introduction ................................................................ 1
   1.1 Context and Motivation ........................................... 2
       1.1.1 Context ....................................................... 2
       1.1.2 Motivation ................................................... 2
   1.2 Aims, Objectives and Contributions ........................... 3
       1.2.1 Objectives ................................................... 4
       1.2.2 Contributions .............................................. 5

2 Literature Review .................................................... 6
   2.1 Artificial Intelligence ............................................. 6
       2.1.1 Cognitive architecture .................................... 7
       2.1.2 Artificial Neural Networks .............................. 8
       2.1.3 Learning ..................................................... 11
## Contents

2.1.4 Learning paradigms .................................. 12
2.1.5 Evolving Artificial Neural Networks ................. 14
2.1.6 Embodiment ........................................... 15
2.1.7 Relevant information ................................ 16
2.1.8 Quantification of relevant information ............... 16
2.1.9 Communication ....................................... 17

2.2 Robot Platforms ....................................... 19
  2.2.1 NAO .................................................. 19
    2.2.1.1 Software ...................................... 20
    2.2.1.2 Hardware ...................................... 20
    2.2.1.3 Laser Head .................................... 21
    2.2.1.4 Laser Sensor Issues ......................... 22

2.3 RoboCup ............................................... 23

2.4 Intelligent Robotics .................................. 23
  2.4.1 Evolutionary Robotics .............................. 25
    2.4.1.1 Fitness Evaluation ........................... 26

3 Professional, Legal, Ethical, and Social Issues 28
  3.1 Professional Issues ................................... 28
  3.2 Legal Issues ......................................... 29
  3.3 Ethical and Social Issues ............................ 30

4 Requirements Analysis .................................. 31
  4.1 Objectives & Requirements ............................ 31
    4.1.1 Objective 1 - Gesture Recognition Program ...... 31
    4.1.2 Objective 2 - Agent Training .................... 32
    4.1.3 Objective 3 - (Re)action Programs ............... 32
    4.1.4 Objective 4 - Controller Evolution ............... 33
  4.2 Requirements Analysis ................................ 34
    4.2.1 Gesture Representation .......................... 35
      4.2.1.1 Region of Interest .......................... 35
      4.2.1.2 Features .................................... 36
      4.2.1.3 Encoding ................................... 36
    4.2.2 Gesture Matching ................................ 36
    4.2.3 Training ......................................... 37
      4.2.3.1 Training algorithms ........................ 37
      4.2.3.2 Training Data Set ......................... 38
## Contents

4.2.3.3 Agent Training ............................................. 38
4.2.3.4 Test Data Set ............................................. 38
4.2.3.5 Test Classification Mechanism ......................... 38
4.2.4 Show Gesture .................................................. 39
  4.2.4.1 Gestures Definitions .................................. 39
4.2.5 Locate & Track Agent ........................................ 39
4.2.6 Detect & Approach Ball ..................................... 39
4.2.7 Kick Ball ...................................................... 40
4.2.8 Controller Evolution ......................................... 40
  4.2.8.1 Fitness Function ....................................... 40
  4.2.8.2 Parameters ............................................. 40
4.3 Environment .................................................... 41
4.4 Performance Assessment ....................................... 41
  4.4.1 Prototype 0 Assessment .................................. 42
  4.4.2 Prototype 1 Assessment .................................. 42
  4.4.3 Prototype 2 Assessment .................................. 42
  4.4.4 Prototype 3 Assessment .................................. 43
4.5 Risk Assessment ................................................ 44
  4.5.1 Risk 1, Risk 2 & Risk 3 - Sensory Data .................. 45
  4.5.2 Risk 4 - Framework/Package Programming Interface .... 45
  4.5.3 Risk 5 - Framework/Package Interaction ............... 46
  4.5.4 Risk 6 - Controller Training ............................. 46
  4.5.5 Risk 7 & Risk 8 - Evolution ............................. 46
  4.5.6 Risk 9 - Hardware ....................................... 46
4.6 Project Plan ..................................................... 47

5 Methodology ....................................................... 48
5.1 Prototype 0: Dynamic Scene Representation from Laser Head Data ................................................. 49
  5.1.1 Design ....................................................... 49
  5.1.2 Implementation ............................................ 49
  5.1.3 Evaluation .................................................. 51
  5.1.4 Discussion ................................................ 53
5.2 Prototype 1: Dynamic Scene Representation from Video Data ......................................................... 54
  5.2.1 Design ....................................................... 55
  5.2.2 Implementation ............................................ 56
    5.2.2.1 ROI Detection ....................................... 57
5.2.2.2 Feature Extraction ........................................ 58
5.2.2.3 Feature Graph Representation ......................... 59
5.2.3 Evaluation ...................................................... 60

5.3 Prototype 2: Dynamic Scene Recognition ............... 61
5.3.1 Design .......................................................... 62
5.3.2 Implementation ............................................... 67
  5.3.2.1 Affinity Matrix Construction ....................... 67
  5.3.2.2 Spectral Clustering .................................. 68
  5.3.2.3 Dynamic Time Warping .............................. 70
5.3.3 Evaluation ...................................................... 72
5.3.4 Discussion .................................................... 72

5.4 Prototype 3: Evolutionary Approach to Dynamic Scene Recognition 73
5.4.1 Design .......................................................... 73
  5.4.1.1 Detectors Evolution ................................ 75
5.4.2 Classifier Training .......................................... 76
5.4.3 Implementation ............................................... 77
  5.4.3.1 Sub-Systems Communication ....................... 77
  5.4.3.2 Detectors Evolution ................................ 80
  5.4.3.3 Classifier Training .................................. 82
5.4.4 Evaluation ...................................................... 83
5.4.5 Discussion .................................................... 84

6 Discussion .......................................................... 85

7 Conclusion .......................................................... 88
  7.1 Future Work .................................................... 89

References ............................................................. 90

Appendix A Prototype 0 ............................................... 101
Appendix B Prototype 1 & 2 ........................................ 102
Appendix C Prototype 3 ............................................... 103
List of Figures

2.1 NAO H25 Hardware ......................................................... 20
2.2 NAO Laser head ............................................................. 21

4.1 Project Plan ................................................................. 47

5.1 Prototype 0 - Waving Gesture ........................................ 52
5.2 Prototype 1 - Pre-processed Scene .................................. 56
5.3 Prototype 1 - ROI Detection (Segmentation) ....................... 58
5.4 Prototype 1 - Concatenated String of Feature Graphs .............. 60
5.5 Prototype 2 - Affinity Matrix Heatmap ....................... 68
5.6 Prototype 2 - Principal Eigenvector of Affinity Matrix .............. 69
5.7 Prototype 2 - Dynamic Time Warping Heatmap with Cheapest Path 70
5.8 Prototype 3 - Overall View of Evolved System for Gesture Recognition ................................................................. 75
5.9 Prototype 3 - Sub-Systems Execution and Communication Sequence Diagram ................................................................. 79
5.10 Prototype 3 - Detectors Fitness Evolution ............................ 82

List of Tables

4.1 Requirements ................................................................. 34
4.2 Environment’s Description ................................................. 41
4.3 Prototypes Assessment ...................................................... 41
4.4 Enumerated Risks ............................................................ 44

5.1 Detector’s Evolution, HyperNEAT Parameters .............................. 81
List of Equations

5.1 Affinity Matrix Definition ........................................... 63
5.2 SFGs Distance Definition .............................................. 65
5.3 Cost Matrix Definition .................................................. 66

List of Algorithms

5.1 Affinity Matrix Calculation Algorithm ............................... 67
5.2 Searching for Cheapest Path over DTW–Generated Cost Matrix .. 71

List of Source Code

A.1 Prototype 0 ................................................................. 101
B.1 Prototype 1 & 2 ........................................................... 102
C.1 Prototype 3 ............................................................... 103
Chapter 1

Introduction

The report starts off by setting a problem context, giving general aims for the project followed by higher-order objectives. Contributions in the area alongside the literature review give very short introductions to the underlying concepts, identifying aspects that will be taken on to the implementation phase. Robot platform that will be used for the project is then presented reminding of a more recent thinking in terms of perception-actuation loops. Gradually perception-actuation paradigm is being substituted by the Darwinian-inspired evolutionary approach to robotics, which is a focus of this research.

After giving an extended introduction to the field, professional, legal, ethical and social issues relevant to the project are identified and acknowledged. Right after the identification of the important issues, practical requirements for the system are listed and described promptly. After requirements are specified, slightly more managerial details then give an overview of the environment, associated risks with the project, and performance assessment provides a description of how implemented prototypes will be assessed. All the workload is then shown on the diagram.

The main body of the report provides technical details of the implemented prototypes that are worked on to satisfy the aims of the project. Chosen methodology is described for every prototype, followed by the implementation and evaluation. A short discussion then presents thoughts on the evaluation results.

Last pages are dedicated to summary and discussion of the main body of the report followed by the conclusion and a short proposition for the future work.
Chapter 1. *Introduction*

1.1 **Context and Motivation**

1.1.1 **Context**

Understanding actions and behaviours without any sort of explicit communication is essential in most human-related tasks, such as autonomous surveillance, medical diagnosis, human-machine interaction (HMI), and even sports analysis (Aggarwal and Park, 2004). For instance, in football, group actions lead to a collective behaviour that is reflected on the playground, such as which team is attacking and who protects the goals (Davids et al., 2005). Yet, while most of the literature around the human movement science in football has been focusing on pattern recognition to increase sports analysts’ perception of the game (Kong et al., 2008), only a few works in the field of robotic soccer have been trying to converge towards a more realistic approach by modelling this sort of football knowledge (Dylla et al. 2008; Bogdanovych et al. 2012). In fact, most of the robotic football strategies and tactics available in the literature, namely the ones related to the RoboCup competition (Kitano et al., 1997), are either hard-coded or present rather simplistic behavioural frameworks that are far from being inspired by the behaviour of real football players (Bezek et al. 2006; Nakashima et al. 2006). In this project, the aim is to bridge the gap between robotic football and real (human) football, so as to pursue the ultimate goal aimed by researchers working on the field: to develop a team of autonomous robots capable of defeating a team of human players (Kitano et al., 1997). An important step towards this direction is to reproduce the way players understand others’ actions and (re)act accordingly.

1.1.2 **Motivation**

Despite the general availability of immediate communication devices, present on boards of autonomous robotic agents, a range of factors that can either be of external or internal nature can compromise their reliability. Physical damage of a communication device usually leads to a permanent loss of ability to communicate, while a failure to establish a communication link between agents’ results in temporal communication impairment (Parasuraman et al., 2013).
Apart from improving the quality of communication, action understanding is the key to automatic monitoring of behaviour in public areas to ensure safety, where real-time video streams are annotated according to observed activity. Smart homes employ gesture recognition methods to advocate safety for families and elderly people by offering a gesture interaction interface between residents and the system (Poppe 2010; Aggarwal and Ryoo 2011).

In sports, existing systems currently require extensive calibrations prior their execution to somewhat exhibit autonomy on the playground, while displaying very restricted human-like behaviour (Bogdanovych et al., 2012). Evolved controllers had been recognised to posses higher degree of robustness, as evolutionary algorithms handle better the interaction between multiple attributes, present in the environment, than other algorithms (Freitas 2003; Lehman et al. 2012).

1.2 Aims, Objectives and Contributions

The aim of this project is to enable indirect communication between two embodied agents sharing the same environment.

- Every agent (Franklin and Graesser, 1997) is an instance that should be able to process input and provide output that, in turn, will actuate agent’s embodiment towards problem solving

- Specific embodiment (Quick et al., 1999) allows agents to physically interact with the environment. Ability to physically interact with the environment is crucial for this project, since the aim is to enable communication by means of the embodiment of an agent

- Gestures will serve as an infrastructure to facilitate indirect communication (Parker, 2008) between the two agents in the environment by transmitting relevant information to the recipient

- Environment is everything that is outside an agent and what agent interacts with.

By manipulating the communication (gestures) between two agents and observing (re)actions, the project aims to confirm a hypothesis that
"One agent (goalie) can train another agent (kicker) to perform a kick according to the wish of the former using indirect communication."

Throughout the project, indirect communication will take the form of gestures and an investigation will be conducted whether gestures can represent intended actions and behaviours of players. Players will be evolved to understand planned action of other players by interpreting perceived gestures. If successful, this development will support and enrich collective behaviour of a team.

1.2.1 Objectives

The higher-order objectives are as follows:

- Develop gesture recognition program
  - Develop suitable gesture representation component
  - Develop gesture matching component

Gesture recognition program is central to the project. It will require extensive testing to ensure that the program can detect and recognise most of the learned gestures presented to the agent.

- Train agent to recognise certain gestures

An agent should be trained a set of gestures prior the recognition process to be able to relate perceived information from the environment to the already learned data.

- Develop (re)action programs
  - Show gesture
  - Locate and track another agent
  - Detect a ball
  - Approach detected ball
  - Kick detected ball
A specific (re)action program will be activated as a consequence of a recognised gesture

- Evolve controller for the integrated system

Evolutionary approach to robotics is the focus of this project

1.2.2 Contributions

This project focuses on evolutionary approach to gesture recognition. Despite existing successful attempts to gesture recognition using other approaches, such as space–time trajectory representation or sequential matching (Aggarwal and Ryoo 2011; Poppe 2010), evolutionary approach had not yet been investigated. This work shows how the horizons of the evolutionary techniques can be extended to a new area, namely, gesture recognition.

The project will look into the use of 2D laser for gesture recognition, as laser sensors do not occur throughout the research done in the context of RoboCup competition for the real-time gesture recognition.
Chapter 2

Literature Review

2.1 Artificial Intelligence

There are as many definitions of Artificial Intelligence as there are books that at least slightly touch on this wide multidisciplinary field. Nevertheless, most of these definitions include such terms as: reasoning, behaviour, rationality, planning, and learning.

Due to the topic of this thesis, definition, given by Russell and Norvig (2009) is the most appropriate. It states that Artificial Intelligence is the field that tries to find the best agent program for a given architecture. Looking back at the aim and objectives of this project, it becomes obvious that this is precisely what is to be investigated in the course of this research work. The keyword ‘best’ from the definition refers to the intended program that will meet the defined aim. The book distinguishes between two types of AI: weak and strong. Weak AI corresponds to the notion that machines act as if they were thinking, whereas strong AI is on the path towards discovery of machines that are actually intelligent. The task of strong AI is to look for a universal algorithm for learning and acting in any environment. An algorithm, in the context of this project, consists not only of agent’s program and its underlying mechanisms, but also of an additional external or internal, explicit or implicit evaluation module that guides learning and acting.

Throughout this report, the term ‘algorithm’ will have this exact meaning as defined above.
2.1.1 Cognitive architecture

Since learning and acting are embedded within some process, the algorithm requires an encoding to hold the state information of agent’s program. Cognitive architecture can provide the needed infrastructure for the algorithm to work with. Russell and Norvig (2009) gives a definition for a cognitive architecture, writing that it is a model of human reasoning. This definition should, of course, be considered metaphorically. In this way the definition can be interpreted as: cognitive architecture is a structure that supports reasoning of agent’s program.

It is worth noting that the performance of algorithm, speed of learning and acting depend greatly on the underlying structure of the agent. Duch et al. (2008) defines cognitive architecture in terms of memory and learning, where memory holds information about the external and internal states of the agent and learning moulds these states that represent the overall knowledge. Cognition becomes a composition of the underlying aspects of the cognitive architecture together with functions that are responsible for planning, reasoning and self-organisation. Possible cognitive architectures are divided into three classes, namely, symbolic (Newell, 1990), emergent (Rumelhart et al., 1986) and hybrid (Sun and Alexandre, 1997). For symbolic architectures, which are used to process high-level declarative knowledge, directed graphs are used, where nodes represent symbols and their attributes and edges represent relationships between symbols. Author identifies that the main uses for these types of architectures are semantic networks (Allen and Frisch, 1982) and conceptual graphs (Sowa, 1976). Emergent architectures use activations of processing units guiding activation signals throughout a network. This class of architectures is inspired by connectionist models of intelligent systems and relies on emergent self-organisation. Two approaches to memory organisation are distinguished, namely, globalist and localist. General neural networks (Haykin, 2007) process information globally, which means that all parameters of a network influence the output and a small change can lead to catastrophic consequences for the global knowledge. In localist networks, the output depends only on a subset of activated units within that network. Hybrid architectures are combinations of the previously mentioned architectures and the paper discusses in details the drawbacks of such combinations.
Although, authors like Russell and Norvig (2009), point out that both symbolic and emergent models are complementary and may both be considered for a candidate cognitive architecture, the practical choice should be made based on type of the data that the model will represent and intentions with that data.

A symbolic architecture, on one hand, would require an abstract view of the knowledge and its interactions present in the environment. Constructing such an abstract view would benefit cognitive architecture by reducing amount of data for the architecture to operate upon leading to faster processing speed. Nevertheless, the construction of the abstract view would not have been possible while in operation. The reason for that is an exceptional amount of predictions and assumptions based on the previous knowledge while constructing an abstract view of the environment. As an alternative, the abstract knowledge could be designed prior system’s execution, but this would eventually render the system as nonrobust. A limited amount of unbiased abstractions of the highest confidence can could still benefit any cognitive architecture by combining related concepts into single definitions.

Emergent networks, on the other hand, can operate on the raw data from the environment without the need for abstractions and relations between the data. First steps towards abstractions can be made first, by refactoring the data and extracting only needed information (specialisation) and second, combining extracted information to create related bundles of the knowledge within the environment (combination). Unfortunately, this would lead to a biased modelling of the environment.

2.1.2 Artificial Neural Networks

When making a rational decision regarding a suitable cognitive architecture for this project, artificial neural networks were considered as an optimal option to model cognitive architecture of an agent. Profanter (2012) presents comparison between cognitivist (symbolic) and emergent encodings for cognitive architecture modelling that supported the decision. Author points out that although symbolic approaches require less parameter tuning, they suffer from a crucial problem - symbols should be designed and pre-programmed and therefore cannot be generalised.
Therefore, the system can be modelled using self-organising artificial neural network, which is a common approach to generalisation over uncertain input data. Given the above, Haykin (2007) describes a situation, where a set of neural networks process divided input data cooperatively, which resembles the way human brain operates. This emphasises modularity that refers to localist approach to memory organisation described earlier.

Russell and Norvig (2009) writes that a neural network is simply a connection of separate units - neurons that are connected together in some fashion to support connectionism. What is more important is the topology of the network itself and the properties of neurons that together make up a big set of parameters that has to be thought of when designing an architecture.

Drew and Monson (2000) highlights the most important developments in artificial neural networks throughout the history and presents a classification of various neural networks according to their applications: (i) associative memory, (ii) optimisation, (iii) classification, (iv) pattern recognition, (v) general mapping and (vi) prediction. Russell and Norvig (2009) describes that neurons are interconnected in some fashion that constitutes topology of a networks and that connections have synaptic weights associated with them that encode and store acquired knowledge. Such structure has been deemed to be able to learn, which is now referred to as Hebbian learning (Schatz, 1992).

As one of the objectives of this project is to employ a cognitive model, the choice has to be made regarding which neural network is to be used to fulfil aims of the project.

Aim of the project implies the use of some classification mechanism that will classify input information based on the learned patterns. Before the classification can take place, agent must be able create patterns out of feedbacks provided by the goalie. Kicker shall be able to associate input with learned patterns.

Drew and Monson (2000) distinguish neural networks based on their topology, self-organisation properties and training approaches. They identify

- Adaptive Resonant Theory, as a theoretical network that is able to stabilise and avoid poor minimum problem by mapping between arbitrary input to a particular classification in real time. The network makes appropriate changes to weights without corrupting the old knowledge and
accommodating the new knowledge. This approach tackles problem with feed-forward networks, when a small change to weights may have a catastrophic consequence for previously obtained knowledge (Henrique et al., 2007).

- Extension of Rosenblatt’s perceptron (Rosenblatt, 1957), Multilayer Perceptron (MLP) is capable of solving linearly non-separable problems, which its predecessor was not capable of doing. MLP has one or more hidden layers, the network is highly interconnected and activation function of each neuron is differentiable. MLP network is trained using supervised backpropagation (BP) algorithm. The algorithm has two phases: (1) Forward phase: inputs are propagated through the network; (2) Backward phase: error signal is generated at the output of the network and the error signal is propagated backwards through the network, adjusting the weights of the network (Haykin, 2007).

- Cascade Correlation (CCN) performs supervised training (Sathya, 2013) faster than MLP with BP by starting off with a simple neural network and gradually adding new layers and new neurons to appropriate layers while analysing the error signal (Fahlman and Lebiere, 1990).

- Self-Organising Map (SOM) trains network in an unsupervised (Sathya, 2013) fashion by detecting features in the input. Network is capable of finding patterns in random data by providing a structural representation of the input data and utilising a winner-takes-it-all strategy, where one part of the network is activated and the rest fades out. Some applications of the neural network include classification of visual similarities and construction of a spatial relationship of objects present on one image (Haykin, 2007).

- Hopfield is a recurrent neural network, which means that neurons can have feedback connections either to themselves (self-feedback), other neurons (no self-feedback) in the network, or both. The network is defined in terms of the energy function, which the network tries to keep at minimum (balanced state) as it evolves. The main use of Hopfield network is to create a content-addressable memory (Pagiamtzis and Sheikholeslami, 2006). The model for the content-addressable memory operates in two stages: (1) storage phase and (2) retrieval phase. The primary application of a content-addressable memory is to retrieve stored information based on noisy incomplete input data (Haykin, 2007).
Drew and Monson’s identified network models have different learning paths that dictate the extent of self-organisation and the topological variance. Mentioned network’s training approaches compete for the training speed and the size of the resulting network, where the two are inversely proportional properties. Nothing is being explicitly said about the different learning paths. When one would choose supervised over unsupervised learning? From the description, the context dictates the choice of the learning path. The designer should have a perception of the environment and of what data is expected to be obtained. In addition, the intentions with the raw data should be made clear prior constructing the cognitive architecture.

If the data is random and the intention is to classify that data, unsupervised learning should be used. The results of the unsupervised learning are limited due to the limited functionality of the unsupervised learning path.

If the data is known to be structured, it can be classified using expected outcomes. Expected outcomes should be identified prior the classification. When trained, the system would become bounded by the identified outcomes restricting architecture’s domain.

Since one of the objectives of this project is to exploit connectionist paradigm, the emergent architecture with localist organisation approach will be explored in greater details. Supervised training will be utilised as the outcomes are bounded by a set of gestures that will be used to train the cognitive architecture.

2.1.3 Learning

Russell and Norvig (2009) quote McCarthy et al. (1955), who strongly imply that learning is a feature of intelligence and that intelligence can be described precisely for a machine to simulate it. This strong hypothesis have had great impact on the Artificial Intelligence ever since and still has major focus on the research in the field. Since the algorithm described in the beginning of this section requires being intelligent to produce reasonable and useful results, it should be able to learn.

Russell and Norvig (2009) state that information gathering is important for rationality and distinguish two cases: first, when the environment in completely known a priori and second, when this is not the case. In the former, agent could act correctly at all times without perceiving or learning anything. This approach
is not scalable to the new or changing environments. In contrast, in the latter case, agent can become independent of its prior knowledge of the environment and begin to exhibit some autonomous behaviour.

To emphasise more on the importance of learning, Sutton and Barto (1998) state that an agent should be able to learn from own experience because learning from interaction is the fundamental idea of learning and intelligence in general. The biggest challenge mentioned is the trade-off between exploration and exploitation - agent should explore for better action selections in the future while exploiting what it already has learned.

This project aims to construct an algorithm that would be able to operate in partially unknown environments and be capable of generalisation and application of learned information to unseen situations. Ideally, the algorithm should be able to accommodate the unseen data and build on top of the existing knowledge. This would require the algorithm to be trained every time the new knowledge becomes available, which would not be achievable during operation of the algorithm.

2.1.4 Learning paradigms

Hertzberg and Chatila (2008) define learning as an ability to improve performance or knowledge of agent based on experience and discriminate between learning schemes by the approach used to produce new knowledge, which can be accomplished either by deduction, induction or analogy. Deductive learning methods, on one hand, employ statistical learning that is mainly used for classification of patterns and is a basis for unsupervised learning. Inductive learning, on the other hand, is based on the approach of inferring new knowledge from already existing knowledge and is therefore mainly used in supervised learning.

Drew and Monson (2000) mention Rosenblatt, who enhanced Hebb’s learning method and identified two categories of learning: (i) competitive learning (self-organisation) and (ii) forced learning. In competitive learning, which also is the unsupervised learning, agent learns patterns from the input it is supplied. For Haykin (2007), unsupervised learning is actually a part of a more general
paradigm - *learning without a teacher*, which is further divided into *reinforcement learning* and *unsupervised learning*.

In addition to self-organisation, unsupervised learning employs *winner-takes-it-all strategy* in which only one neuron is activated at the end and the rest of the network fades out. Reinforcement learning uses different strategy - the learning is achieved through the interaction with the environment and series of reinforcements, either rewards or punishments. The aim of reinforcement learning is to discover a function that maps input to output actions while maximising the reward.

Supervised learning is an advanced variation of forced learning, in which a function that maps from input to output is learned provided examples given by a teacher.

Summarising identified learning paradigms, it should be noted that in supervised learning, a mapping function is being transferred from teacher to a learner. In reinforcement learning, mapping function is being guessed by the learner. In unsupervised learning, no explicit mapping function exists, the learner needs to group and classify inputs according to some criteria that the learner itself discovers.

*Drew and Monson* (2000) mention that many learning algorithms exist that can be classified as either one or combination of two of the learning paradigms described earlier. Learning algorithm should be chosen based on the application of a particular neural network. The main problem that was mentioned earlier is the variety of parameters that have to be dealt with. Author brings up the genetic approach to definition of various parameters to avoid the unnecessary speculations regarding, for example, number of hidden layers, amount of neurons in hidden layers, biases and other. *Genetic algorithm* (Beyer and Schwefel, 2002) can be employed for discovering the most optimal neural network for a particular problem situation.

Previously it had been identified that the supervised learning approach will be taken since the knowledge within the environment is bounded by a set of gestures that will be used for indirect communication (specific gestures will be presented later during implementation). Therefore, a mapping between input and output is expected to be transferred from trainer to the trainee. Trainer would possess the knowledge regarding which gesture maps to which reaction from
the trainee’s side and the trainee would be expected to learn the mapping and be able to react on request.

2.1.5 Evolving Artificial Neural Networks

As it was mentioned, genetic algorithm can be used to search for an optimal neural network. To be able to generate a search space, neural network has to be encoded in some way. Two common ways exist to encode a neural network, namely, direct and indirect encoding. In direct encoding, either connection weights or the topology itself is encoded directly. If the topology of a directly encoded neural network is expected to grow as evolution proceeds, this will cause the growth of the search space as well. The growth of the search space is undesirable effect as this slows down the evolution and potentially creates more local optimal solutions that complicate the searching process.

Indirect encoding, in contrast, uses some system that can generate a neural network and the genetic algorithm operates on that system instead.

One of the most promising indirect encodings that compresses the search space is the hypercube. The hypercube encoding is used by the HyperNEAT algorithm (Stanley et al., 2009), which consists of a set of functions (CPPNs) that evolves neural network topology (composition of functions can generate geometric patterns) and underlying weights. Verbancsics and Stanley (2011) present a figure that clearly shown the connection between neural network topology and a CPPN function. The paper also gives a basic HyperNEAT algorithm.

To encourage localist approach and modularity of the neural network, which is often one of the objectives, either encoding or fitness function can be tailored to facilitate discovery of modularities. HyperNEAT can tackle the issue of modularity by introducing thresholds to discourage connectivity between independent functionalities. Verbancsics and Stanley (2011) investigate three variants of thresholding in HyperNEAT to achieve modularity, regularity and hierarchy in evolved neural networks, reporting that satisfying results can be achieved using a particular type of thresholding.

Along with the HyperNEAT algorithm, Sher (2010) identifies other novel approaches: (i) EPN uses back-propagation for evolution of weights and mutation for topological evolution; (ii) GNARL uses evolutionary algorithm for both
weight and topological evolution. An additional variable parameter is used to determine the intensity of mutations; (iii) NEAT uses evolutionary algorithm to mutate (through small perturbations) both weights and topology of the network; (iv) in CoSyNe, neurons are grouped and groups are then permuted to find the best combinations. Approach does not evolve topology; EANT2 uses exploration through standard topological mutations and exploitation through CMA-ES, which does not evolve topology.

Similar to the artificial neural networks and their corresponding training algorithms, evolutionary algorithms are specific to the application area and the desired degree of variety in resulting neural networks. Some applications may restrict resulting neural networks to a single specific topology, while others may restrict the resulting network to a range of specific weights.

HyperNEAT is a highly customisable algorithm and is therefore an optimal choice for the project. The algorithm allows specification of both the resulting topology and weight ranges prior the evolution.

2.1.6 Embodiment

Nolfi and Floreano (2000), citing Arkin, write that in order to further advance an agent with cognitive architecture, that agent should be given ability to physically interact with the environment to enable embodied cognition to strengthen connection between perception and action. In addition, agent’s behaviour should be viewed as a result of agent’s actions in its environment, not simply by analysing the internal states of the agent, which may be intractable. Author points to Braitenberg’s vehicles that are the best examples of emergent behaviour as a result of structural coupling of an agent with its environment.

Embodiment in the project takes the form of the physical robot body. An agent uses cognitive architecture to process sensory data coming from the environment and reacts according to the data by actuating physical body of the robot. In this way the algorithm operates in terms of perception-actuation loops, where perception is based on the sensory input and actuation is the ability of the robot to show gestures.
2.1.7 Relevant information

While it is important for an agent to learn from its own experiences, the challenge is to distinguish that, what is worth agent’s attention and what can be ignored since it may not carry any significant information. The information theory, developed by Shannon (2001), provides quantification and means for interpretation of information that an agent is able to obtain from the interaction with its environment.

One approach, mentioned by Polani et al. (2001) is to correlate internal states of the agent with the current or future states of the environment. Nolfi and Floreano (2000), in turn, cites Brooks, writing that there are two ways to view internal representations of an agent - by explicitly making internal states of an agent represent the external world (the environment) or by making internal states of an agent to partially represent states of the world, where those partial representations of the external world would only include the most relevant information, which would help agent to achieve its goals.

Agent in this project will be required to extract relevant information from the environment by the means of pre-processing and encoding.

2.1.8 Quantification of relevant information

To be able to quantify and reason about the new semantically enriched abstraction over general information, additional mechanisms are required. Polani et al. (2001) propose a rigorous model for quantification of relevance in any system. As an alternative to building complex frameworks, a much simpler abstraction can be developed to provide a quantification of relevance of the information, like a suitable fitness function that would reflect the sensory evolution or a metaphorical model, which could resemble something like an emotional state of the agent, where the agent or a set of agents would try to maintain a balanced emotional state in the environment, resembling homoeostatic state of the system (Allan et al., 2013).
2.1.9 Communication

Communication for Russell and Norvig (2009) is an alternative to predefined constraints on agent’s behaviour achieved by a convention. Same as a convention, communication is adopted to achieve a joint plan between agents. Authors provide examples of an agent explicitly communicating a plan to another agent or, through action, notifying another agent of the intention of the action by means of shared signs (gesture, language, etc.).

Parker (2008) distinguishes three communication paradigms, namely, (1) implicit (e.g. by means of stigmergy), (2) passive/indirect (by observing another agent through sensors) and (3) explicit/direct (relevant information is communication directly and intentionally). Every paradigm has its advantages and disadvantages: implicit is usually simple and does not require standardisation, but limited by agent’s perception of the world and sensor availability; passive/indirect has no communication channel bandwidth restrictions and can be invincible against faults, but is limited by agent’s perception and sensor availability; explicit/direct is most common due to the ease of data transmission that does not require perception or additional sensors, but it can become faulty and unreliable due to internal or external factors.

The aim of this project is to use indirect communication paradigm, therefore mentioned advantages of the indirect communication paradigm should be exploited, while avoiding the disadvantages. The main concern is agent’s perception of the world, which may be insufficient to support a perceived intention of another agent. Steels (2003) talks about bootstrapping of the shared meaning from information and what prerequisites both agents must meet to support pragmatic feedback synthesis, feedback perception and interpretation. Bootstrapping is achieved by language games, an approach described by Steels (2011), in which consensus between two agents is reached by the means of invention, adoption and alignment. To sustain homoeostasis of the overall system and to introduce quantification of the relevant information, Steels (2003) introduces motivational system.

As communication may be described rigorously, Nehaniv (1999) takes refuge in interaction games that are, as claimed by the author, the basis for emergence of communication between agents and environment. Author extends the ideas of Wittgenstein regarding affordance and claims that the meaning is
acquired through the interaction games using agent’s perception. Interaction games utilise communication channels that should be created and evolved to facilitate prediction of behaviour.

Klyubin et al. (2004) talk about the indirect communication between agents and an approach of measuring information flows between the agents. For that, sensor evolution through *perception-action loop* using Claude Shannon’s *information theory* is used. It becomes possible to off-load information onto the environment to facilitate the exchange of the information as well as a chance to relieve agent from some extra load and be able to ‘pick it up’ for later use.

Chersi (2012) shifts attention to *mirror neuron system* for better relevant information transfer and shows an implementation on an agent that repeats manipulative sequences performed by humans. The aim of the system is to achieve a goal, not to mimic every movement of demonstrator. To do so, agent must understand intentions underlying demonstrated manipulations.

As it can be seen from previous research done on communicating relevant information between agents, the issue of achieving a common perception of the world is universal and can be tackled by aiming for the whole system approach to solve a problem, where a single system (e.g. centralised) has some control over agents perceptions.

As for the RoboCup environment and gesture interaction between agents, Trigueiros et al. (2015) propose a general framework that is vision-based. Authors identify the key steps needed in any gesture recognition system and incorporate these steps in the proposed framework. Key steps are: (i) data acquisition and pre-processing, (ii) data representation and feature extraction and (iii) classification or decision-making. Approaches for feature extraction differ for static and dynamic gesture recognition, where dynamic gesture recognition requires additional module for sampling between translations. Freelan et al. (2014) point out that the constituent part of sports is coaching/training and that personal training is extremely rare in RoboCup research. Used HiTAB method views simpler behaviours as *modular parts of more complex behaviours* that can be bootstrapped by coaching/training, giving raise to *behavioural bootstrapping*. Behaviours are robust, meaning that the coach/trainer is able to add, remove or change existing behaviours of an agent if that behaviour does not produce desired complex behaviour.
2.2 Robot Platforms

2.2.1 NAO

Paper by Gouaillier et al. (2008) presents NAO as an autonomous, interactive and programmable humanoid robot built by Aldebaran-Robotics, and describes the early generation of the robot in terms of hardware and software. Since the beginning, the company concentrated on designing lightweight, low-cost, high-performance, modular robots with an open architecture. The key objectives that designers followed creating the first version were:

- Rich, smooth navigation.
- Embedded feature/face recognition, self-localisation, and autonomous operations in the environment.
- Actuator modularity.
- Easy to use architecture.

Apart from sophisticated actuation system, NAO runs a software framework, NAOqi that manages execution of user binaries. This framework supports sequential, parallel and event-driven executions.

Some of uses of the NAO robot include:

- Research and education
- Successor to Sony AIBO in RoboCup league since 2008
- Developer community program to encourage programmers to create custom applications for general public
- Support learning process for autistic children in schools
- Robot workers
2.2.1.1 Software

The underlying NAOqi framework allows developers community to improve robot’s functionality by writing custom programs that would make robot do a certain task. The release of the new version of the framework is expected to facilitate exchange of applications between robots that are being built by the company (Aldebaran, 2014b).

2.2.1.2 Hardware

As seen on the Figure 2.1, the modern NAO robot is equipped with a system of sensors that includes: (i) eight force-sensing resistors on soles of robot’s feet; (ii) two axis gyrometer in the centre of the body; (iii) one three axis accelerometer; (iv) two sonar emitters and two receivers; (v) tactile sensor on the head, three on each hand and one on tip of each foot and (vi) two video cameras in

![Figure 2.1: NAO H25 Hardware (Aldebaran, 2014a)](image-url)
the head; (vii) an optional laser head shown on Figure 2.2 can be installed on NAO’s head.

The robot features twenty-five degrees of freedom allowing it to successfully perform a wide range of complex movements and object manipulations.

2.2.1.3 Laser Head

![Figure 2.2: NAO Laser head (Aldebaran, 2014a).](image)

Laser head module contains a laser sensor (Hokuyo URG-04LX-UG01). As stated in HOKUYO AUTOMATIC CO. (2009), the light source of the laser is infrared and it can be used for area scanning purposes. Laser’s scanning range is 4 meters and scanning angle is 240 degrees. Maximum divergence of the laser sensor is 40 millimetres at 4 meters range.

Detection and tracking of other actors in the environment requires some tracking sensors engaged. Lots of research dedicated to the object and actor tracking
within a RoboCup environment have utilised vision-based approaches. This is, without arguing, a justifiable choice as humans rely a lot on the vision-based sensory systems and so should robots. Today this choice poses additional constraints on environments, such as, for example, an introduction of an external camera, as embedded cameras in robots are extremely poor and require sophisticated algorithms to be able to extract relevant information about constantly moving objects. On the other side is the statement made by Steels (2012), saying that the focus on extended data acquisition from chosen sensors most of the time is too intense for the task the agent is trying to achieve. Therefore, simpler and more reliable sensors should be used to be able to obtain only crucial data at low cost for a given task.

As a possible alternative, Becker et al. (2007) presents the use of 2D laser scanner for obstacle detection, obstacle classification and obstacle tracking as well as path prediction of moving obstacles in car-like mobile robots. A general structure of the tracker algorithm for obstacle detection and obstacle tracking is provided. This approach reduces the noise coming from the environment and introduces a low-cost, faster approach for agent to sense its environment. Mertz et al. (2013) describes an algorithm for a system that uses 2D scanner or a combination of two or more sensors as primary sensors used for detection and tracking of moving objects.

### 2.2.1.4 Laser Sensor Issues

Kneip et al. (2009) report drift effects and different influences of the environment as well as distance and angle on the performance of a more advanced sensor of the same category. Some of the reported deviations include faulty measurements and the continuous drop of the measurement performance due to the heat of the sensor. Lightning conditions do not result in any drastic changes in sensor’s measurements. In conclusion, authors report that the measurement accuracy depends on the surface brightness.

Although issues were identified on a similar sensor, the results of the performed benchmarking should be remembered and consulted if similar problem would occur with the given sensor.
2.3 RoboCup

The RoboCup competition is a *testbed* for embodied cognition, described in Section 2.1. Many implementations have been realised with some of objectives to be able to carry out a *shared cooperative/collaborative tasks*, *(self-)*localisation, predicting trajectories of fast-moving objects, playing strategies analysis/synthesis, team/individual coaching/training, with all the mentioned objectives spanning homogeneity boundaries (Gupta et al. 2005; Yasui et al. 2014; Trigueiros et al. 2015; Freelan et al. 2014).

Apparent challenges that sometimes cause unforeseen complications that were reported by various sources include the accuracy of navigation or sensing due to the noise in the environment and/or deviations in joint kinematics, complicated algorithms that slow down reasoning and planning.

Due to the mentioned challenges, RoboCup situation is sometimes reduced to, for example, a penalty kick situation, where a solution to a particular problem can be scaled up once challenges are resolved.

RoboCup environment (either virtual or real) is relevant for this project as it allows the designer to test different cognitive architectures, algorithms, and theories tying to solve the same problem.

In the case of a successful outcome of the project, future projects may use different algorithms or cognitive architectures to achieve similar goals and evaluate results to find whether another approach is better or worse than the one presented in this project.

The ultimate goal of a RoboCup setting, as stated by Kitano et al. (1997), is for a humanoid type robot to be able to perform all the activities that the human player performs on the field while playing football and, finally, to defeat the human players.

2.4 Intelligent Robotics

Intelligent robotics is a field that emerged from Artificial Intelligence as interchangeability of perception, reasoning, and actuation became a driver for an
embodied agent on a quest of achieving useful tasks. Jarvis (2008) splits development of robotics into three generations: (1) not re-programmable (2) re-programmable and (3) adaptable robots. The third generation relies on the sensory feedback to support adaptability to variations in the environment and, therefore, can be viewed as intelligent. Gruver (1994) adds that appropriate decisions are being made based on sensory information, learning and reasoning algorithms. Any assumptions made about the environment become unnecessary. The former paper lists directions and challenges that are being researched by the present situation in intelligent robotics. These challenges include (i) optimised (inverse) kinematics derivations (ii) intelligent localisation methods (iii) determining efficient navigation paths (iv) more advanced locomotion (v) human-robot interaction and (vi) emergence of collective intelligence.

Gruver (1994) identifies three areas that benefit from using intelligent robotics: manufacturing, rehabilitation, and service. In manufacturing, a robot must make use of different components to perform complex intelligent joint operations. When making use of these components, robot must be able to ‘feel’ and respond adequately to various forces affecting the environment. As for non-functional requirements, robot must handle demands for flexibility, performance, and complexity in manufacturing processes. Intelligent robotics in rehabilitation can either be used as training devices to accommodate for functional disabilities or to act as a device that would remediate conditions that cause temporal disabilities. Service is a wider commodity of everyday life and robots that are employed in this area must incorporate sophisticated control, locomotion, world-modelling, and sensing capabilities.

In definition for intelligent robotics, Murphy (2000) adds that the type of embodiment is not important as long as robot can function autonomously. Author includes plasticity of the robot’s structure itself that should not affect the level of autonomy and provides the Terminator movie as an example.

In Maes (1993), a question of how to achieve an autonomous agent is discussed. The article claims that while traditional Artificial Intelligence concentrated on hypothetical problems, expert systems (knowledge-based systems) were designed to perform better than human experts within a particular domain. Russell and Norvig (2009) add that Artificial Intelligence concentrated more on the problem definition rather than knowledge representation and therefore suited more for abstract problem situations, while knowledge-based systems
reasoned about internal knowledge.

Guiding principle for the emergence of autonomy, stated in Maes (1993), says that interaction leads to emergence of complex behaviour. To support the emergence of complex behaviour, two paradigms are common for intelligent robotics: (1) **behaviour-based robotics** (Brooks, 1991a) and (2) **evolutionary robotics** (Nolfi and Floreano, 2000). Both paradigms belong to a more general bottom-up organisation, where underlying innate abilities are combined together to achieve some new emergent functionality.

Evolutionary robotics is inspired by Darwinian postulate that holds survival of the fittest as its main rule. Behaviour-based robotics approach exploits innate, simpler behaviours of a robot to produce more complex behaviours.

The project will be based on the evolutionary robotics principles utilising evolutionary algorithm to evolve cognitive architecture of the agent.

### 2.4.1 Evolutionary Robotics

Evolutionary robotics is an approach to designing control systems and/or body morphologies that are used by intelligent robots. This approach uses evolutionary techniques that, through generations of evolution, generate optimal solutions.

Finding an optimal solution can be seen as an optimisation problem that resides in some imaginary multidimensional search space. Search space usually has one dimension that is used as a quantification metric, while other dimensions correspond to various parameters in a solution. A candidate solution (*genotype*) is encoded so that every part of the *encoding* conforms to a particular parameter (*gene*).

Harvey et al. (1997) divide general *evolutionary algorithm* into five stages:

1. **Initial population** with random individual solutions spanning the search space is generated

2. **Evaluation** of every individual solution in an environment. Individual solution is given a score (*fitness*) that tells how good that particular solution had been
3. Selection mechanism picks some individual solutions from the evaluated population using selection criteria

4. At breeding stage picked individual solutions are allowed to pass on parts of their genotype to the next generation

5. The next generation consists of some individual solutions from the previous generation and offsprings produced during the breeding stage

Nolfi (1998) argues that in evolutionary robotics, complex behaviour is a result of self-organisation and adaptation rather than explicit design. This is further being proved by splitting the definition of a complex behaviour into two perspectives: distal (that of designer) and proximal (that of robot), where author argues that it is hard to analyse a complex behaviour from distal perspective. In contrast, it is relatively easy to break a complex behaviour into simpler ones when viewing a complex behaviour from proximal point of view. Therefore, in evolutionary robotics, desired complex behaviour is evolved by the means of evolutionary techniques and evaluated using fitness criteria applied to the overall resulting behaviour.

2.4.1.1 Fitness Evaluation

Any task for a robot can be expressed as an optimisation problem, where chosen parameters of the system will have to be either maximised or minimised dependent of the aims of the task. Fitness function evaluates phenotype resulting from a particular genotype and assigns it a numerical score. The main challenge in formulating a fitness function is to be able to describe desired behaviours on an abstract level without low-level implementation details. Robot's phenotypes are evaluated using explicitly specified abstract evaluation criteria. Nolfi and Floreano (2000) differentiate between implicit and explicit fitness functions, where the former is a strict constraint usually applied as a threshold to the crucial behaviours of a robot (i.e. not to run out of energy), while the latter constraint is less strict and may consist of variable factors that affect the overall fitness measurement (i.e. change in velocity).

To cope with challenges traversing search spaces, Harvey and Ezequiel (2014) share some practical information when applying evolutionary techniques to achieve more robust exploration of the search space. Some of the tricks are:
• Initial population initialisation tactics.
  – Random initialisation is appropriate in situations when the search space size is not known in advance nor is the approximate location of the solution.
  – Random initialisation does not have any bias on what candidate solutions may be.
  – In case when some of the prior knowledge is available, initialisation may be biased and initial population distributed over a region where potential solution may be found.

• Genetic operators tactics.
  – When upper and/or lower limits of gene values are known, the genetic operators should be constrained not to assign a higher or lower value than the limit.

• Activation function limits.
  – Summed input values to activation functions should not be close to the bounds as the system ceases to be sensitive to the variations in inputs. Even changes in weights would not affect much the activation.

• Reshape the search space.
  – Introduction of *intermediate targets* allows for a wide start and then gradually narrow down to more optimal solutions.

• Make crucial factors more salient.
  – Behaviour that is important for the overall behaviour can be distinguished and even given a different role for the early stages of the evolution.

For all optimisation problems where evolutionary techniques can be used, the crucial aspect is to find a balance between exploration and exploitation to achieve a steady state traversal of the search space.
Chapter 3

Professional, Legal, Ethical, and Social Issues

3.1 Professional Issues

The aims and objectives of this project did not address general public, as the project had been conducted in a fully controlled environment. All contributions made by the community had been acknowledged in the report. No confidential information had been obtained during the course of the project. No data shown in the report had been misrepresented or withheld, all findings were documented and much effort had been put into unambiguous and detailed explanations.

In the case of any possible extensions to support human-machine interaction, appropriate measures will have to be taken that would prevent a robot from harming a human being during the time of the interaction. In addition, necessary precautions will have to be taken to protect any confidential information so that it would not become available beyond the bounds of the controlled environment.
3.2 Legal Issues

In the context of this project, legal aspects of computing are important to identify and be aware of. Computing law is an encapsulation of underlying categories, namely, contract, intellectual property, data protection, and computer misuse and computer evidence laws.

Contract law explicitly specifies what each party has a right for and what is expected from them. If one of the parties is unwilling to comply with some specified points in a contract, the negotiations can be cancelled without any consequences for both parties.

For this project, the contract is inherited from the institution (Heriot-Watt University) and does not bear any additional constraints. Contractual duties do overlap with professional issues, in particular, with the sense of duty to the profession. Intellectual property law refers to the ownership of any work done under the name of the university. In the case of this project, the student conducting the research has all the rights for the outcomes of the project.

Under the data protection law, which should not be brought up unless the project will be extended to support human interaction, the collected data then must be protected at all times and no action towards seeking the profit for the collected data should be taken.

The university that provides facilities for the project execution enforces computer misuse law. In addition, personal computer had been cleared before beginning the project and only licensed or free software had been used to support the research.

Computer evidence law had been enforced by carefully monitoring the system for any changes and, in case of a suspicious act, the system would have been shut down and a professional invited for the investigation.
3.3 Ethical and Social Issues

The nature of the project is to perform a proof of concept. Therefore, the project poses no ethical or social issues. If the research is to be extended further, then such issues will have to be identified and certain precautions taken into consideration.

As the society is mainly divided into two parts, where one is extremely eager to introduce new technologies into the modern world, the other part is more conservative and sees robotic solutions as one possible threat to the humanity, the research in the area of robotics must therefore move at a slow pace, giving the opportunity for the general public to try out and get used to the robotic solutions to existing problems.
Chapter 4

Requirements Analysis

4.1 Objectives & Requirements

Prototypes will serve as a proof of concept for the objectives of the project listed in Chapter 1.2.1. Every prototype will aim to satisfy a set of requirements. Prototype approach had been chosen due to the exploratory nature of the project. Every prototype will be used to decide whether prototype results are satisfactory and whether chosen design can meet aims of the project. If prototype is suitable, its implementation details are passed further to the next prototype that would reuse the most of the previous prototype. This inductive approach is expected to lead the project forward.

4.1.1 Objective 1 - Gesture Recognition Program

The gesture recognition program is the key component of the overall system. The program consists of two sub-components, namely, gesture representation and gesture matching.

Gesture representation should be able to encode presented to the agent gesture into an analysable data. Resulting encoded gesture data should be standardised and made tractable for further manipulations (e.g. gesture matching sub-component). Gesture representation should capture most relevant information about perceived gesture without missing the essence of the presented
gesture and evading the noise. This particular requirement may introduce a trade-off during the implementation between the relevance and the noise of the information of perceived gestures.

Gesture matching should operate on the data returned by the gesture representation sub-component. Gesture matching should be able to take two encoded gestures and return their similarity probability or learn a set of gestures in advance and be able to decide which class a perceived gesture belongs to. Gesture matching should be fast and operate in real time without delays.

4.1.2 Objective 2 - Agent Training

Agent should be trained to be able to recognise gestures. It should be investigated which training algorithm gives best results on the test data. A separate testing data set should be created to test training results. Training set should be of the same standardised format as results from gesture representation sub-component so that the gesture matching component would operate on same formats and be able to match them. Training set should contain multiple different entries for every gesture to allow robust recognition. At the same time, training set should not contain all possible variations of the same gesture to avoid overfitting issue. Training data set should be shuffled so that the training algorithm would be able to generalise input during testing.

4.1.3 Objective 3 - (Re)action Programs

Agent should be able to act or react in response to a recognised gesture. As a possible response, agent should be able to kick detected ball. Kicking detected ball can include additional procedures that would ensure that the agent found a ball on the field, approached the ball and positioned itself in front of the ball. Additional requirements are optional and the situation can be restricted to the agent being positioned in front of the ball prior the gesture recognition. All developed (re)action programs should be tested on the achievement of the ball-kicking task separately from other components of the system.

Agent has to be able to show a gesture to the other agent. A range of gestures can either be encoded into agent’s controller or dynamically synthesised from a
set of available joints settings. It would be simpler to decide on a set of gestures in advance and encode them directly into the controller. Later this model can be extended to allow for dynamic gesture synthesis.

### 4.1.4 Objective 4 - Controller Evolution

All developed programs and sub-components should be integrated into a single system. A single controller should be evolved to satisfy posed aims for the project. The fitness function should be designed to drive the evolution of the controller and used as evaluation criteria for the controller. Overall system performance should be evaluated by allowing agents run autonomously and empirically testing whether the hypothesis is confirmed.
# Requirements Analysis

## Table 4.1: Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Requirement</th>
<th>Dependency</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Develop gesture representation algorithm</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>1.1</td>
<td>Detect region of interest</td>
<td></td>
<td>1</td>
<td>H</td>
</tr>
<tr>
<td>1.2</td>
<td>Extract features</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Encode extracted features</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Develop gesture matching algorithm that uses encoded features</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Agent training</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>3.1</td>
<td>Investigate different training algorithms</td>
<td></td>
<td>2</td>
<td>M</td>
</tr>
<tr>
<td>3.2</td>
<td>Create training data set</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Train agent</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Create test data set</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Test trained agent</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Show gesture</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>4.1</td>
<td>Create pre-defined set of instructions to perform a certain gesture</td>
<td></td>
<td>3</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>Locate another agent</td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>Track another agent</td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>Detect ball on the field</td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>8</td>
<td>Approach detected ball</td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>9</td>
<td>Kick detected ball</td>
<td></td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Controller evolution</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>10.1</td>
<td>Develop fitness function</td>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>10.2</td>
<td>Investigate evolutionary parameters</td>
<td></td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

O — Objective; P — Priority
All the requirements presented in Table 4.1 are functional requirements. Since the aim is to develop prototypes that will be used to provide evidence of functional operability of the system, non-functional requirements are trivial and mostly concern (1) robustness (2) testability and (3) operation speed of the system. Non-functional requirements related to usability, reliability, privacy and similar user-related issues are of little or no concern due to the aims of the system. Although the optimisation is not the primary concern of the project at this stage of the research, improvements to processing speed and navigation accuracy of robots are possible non-functional requirements that the system can benefit from in the future.

4.2.1 Gesture Representation

Gesture representation algorithm has to ensure that any perceived gesture has a standard encoding format. The encoded gestures are considered as analysable data. The encoded data can be used during gesture matching to determine whether two gestures are similar. The encoded data of a perceived gesture is required to contain all the relevant to that gesture information, while the amount of noise from the environment has to be minimal. Such property of the algorithm will ensure that the captured gesture is reduced down to the atomic components. The property will promote robustness of the algorithm against changes in the environment, as all the unrelated information will be ignored.

The algorithm is expected to perform very well as it will be one of the key components of the total system.

4.2.1.1 Region of Interest

For a gesture to be encoded, it has to be detected and extracted from the sensory data. Therefore, specific regions in the sensory data have to be identified as the stream of the sensory data becomes available. The stream has to be parsed and only relevant to gesture information extracted from the stream. The resulting detected region of interest should be less than the size of the full stream, but also greater than the minimal piece of the information within the
stream. This means that the area, occupied by the information directly related to the gesture, within the stream should be of the significant size, but within some limits. Such constraint will reduce the size of the data and increase the processing speed of the algorithm as a consequence.

### 4.2.1.2 Features

When preliminary region detection is performed and the total data is reduced, the fine-tuned optimised procedure will break down the region of interest into smaller pieces of information of the perceived gesture, leaving more noise behind.

A set of features has to be identified in advance to guide the feature extraction process. The number of features used should be enough to represent a gesture in acceptable details. A simple example of acceptable details of a moving arm used as a gesture would include information about the elbow and fingers, retaining geometrical properties of the arm, such as width and length. Such information as the skin colour of the arm would be considered as excessive.

### 4.2.1.3 Encoding

Extracted from the sensory data features should be encoded using an adequate data structure. Encoding by itself may carry additional information about perceived gesture, such as distances between the extracted features or the size of the features. Chosen data structure should be sparse in order to reduce processing time of the gesture representation algorithm.

### 4.2.2 Gesture Matching

Gesture matching algorithm should use the encoded gesture data from gesture representation algorithm to match two gestures. The matching algorithm should output the similarity probability between the two gestures. Matching process should be fast and perform in real-time as the agent operates in the environment without impeding its operations. This requirements is constrained by the
processing speed of the processor of the robot, therefore algorithm optimisations have to be performed.

During execution, sensory data should be captured in a window–based approach, limiting the sensory stream of data by the length of the window. The length of the window should be decided in advance and it should be tested what length is appropriate for the processing speed of the processor. To avoid splitting information of a single gesture by the size of the window, two overlapping windows have to be used. This is expected to put additional pressure on the processing, because both gesture data encoding and gesture matching will be executed twice during the length of a single window.

### 4.2.3 Training

Gesture matching should be generalised, enabling more than two gestures being matched at a time. Instead of matching, gestures should be classified based on the information obtained during training.

#### 4.2.3.1 Training algorithms

A range of training algorithms has to be explored and tested on the ability to train a robust gesture classification mechanism. Training algorithm is expected to produce classification mechanism that would be able to distinguish classes within the data that will be used for training the classification mechanism. The distinguished classes should be distinctly detached from one another to maximise confidence in every class. The more disjoint a class is, the higher should be the confidence in that class.

This requirement has a medium priority due to the possible time constraints imposed on the project. Evaluation of different training algorithms and selection of the most optimal can be reduced to selection of a single training algorithm that potentially could provide optimal training results. The advantage of evaluating a range of training algorithm could allow investigation of the changes in performance of the trained agent.
4.2.3.2 Training Data Set

To train the classification mechanism, the training data set should contain standardised data that is described in Requirement 1. Training data should consist of entries that would train the classification mechanism to recognise similar gestures in the future. Every entry in the training data set should contain a particular gesture encoded data and a gesture class manually specified that relates encoding to the gesture class. A considerable number of entries per single gesture should be used, so that the classification mechanism would be able to generalise trained data and apply it later for similar gestures. The data set should not contain all possible gesture variations, because the aim is to train the classification mechanism to be able to classify an arbitrary variation of the same gesture. Training set should be shuffled so that the classification mechanism would not focus on a particular class during training.

4.2.3.3 Agent Training

When training algorithm is chosen and training data set is composed, the training can begin.

4.2.3.4 Test Data Set

After training is performed, test data set should be created to test the resulting classification mechanism. Testing data set should be different from the training data set because classification mechanism will be tested on the ability to generalise the data from the training set and apply that generalised knowledge on the test data. Test data entries should be of the same format as entries in the training set. Test data should be shuffled for the same purposes as shuffling is applied on the training data set.

4.2.3.5 Test Classification Mechanism

Test data set can now be applied to test how good the mechanism had been trained. The evaluation criterion for testing is the percentage of correctly classified entries from the training set.
4.2.4 Show Gesture

For an agent to be able to perform a gesture is important for the gesture recognition algorithm since the gesture is an input to the algorithm. A set of gestures that an agent would be able to recognise should be identified and defined.

4.2.4.1 Gestures Definitions

Gesture definition can be accomplished using static pre-defined sequence of movements or dynamically executed joints manipulations based on some external factor. As a mere minimum, a set of gestures should be statically defined and encoded directly into agent’s controller. Gestures should be simple at this stage of the research.

4.2.5 Locate & Track Agent

Locating and tracking another agent on the field is a complex task, as it requires additional sensory input processing and training. Therefore, the system can be constrained such that the agent is manually positioned directly in front of another agent’s sensory field of view. In addition, at this stage of the research, it is assumed that agents are stationary on the field. These requirements have low priority.

The aim with these requirements is to increase the level of autonomy, so that the agents on the field can move freely and the gesture recognition algorithm will only be operational in case if an agent decides to track gestures of another agent.

4.2.6 Detect & Approach Ball

Ball detection on the field and movement towards it is another complex task that can be discarded at this stage. This requirement would require additional sensory data processing and training. The system can be constrained by positioning an agent in front of a ball prior the system execution. Requirements have low priority.
The aim with detection and approach is to increase level of autonomy of the kicker agent.

### 4.2.7 Kick Ball

The kicker agent should be able to perform a kicking action. The decision to kick has to come after gesture recognition process, which would mean that the agent recognised a gesture and acted upon it. The agent should be able to decide to which side of the goals to kick the ball based on the recognised gesture.

### 4.2.8 Controller Evolution

A single system should be constructed of previously developed sub-components, namely, gesture representation program, gesture matching program and (re)action programs.

#### 4.2.8.1 Fitness Function

Fitness function should drive the evolution of the controller for the overall system. The objective of the fitness function should encompass aims of the project. In particular, the fitness should increase as the agent recognises perceived gesture correctly and applies correct (from goalie's perspective) reaction to the recognised gesture.

#### 4.2.8.2 Parameters

Evolutionary parameters should be investigated and the most optimal settings chosen for the evolutionary process. The requirement has low priority due to amount of additional work that has to be performed in order to determine which parameters produce the most stable and fast evolution of the controller. At this stage of research, theoretically optimal evolutionary parameters are acceptable.
4.3 Environment

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>Robots</td>
</tr>
<tr>
<td>Mac</td>
<td>NAO H25</td>
</tr>
<tr>
<td>PC</td>
<td>peas</td>
</tr>
<tr>
<td></td>
<td>hnn</td>
</tr>
<tr>
<td>Packages</td>
<td>Applications</td>
</tr>
<tr>
<td></td>
<td>Redmine</td>
</tr>
</tbody>
</table>

**TABLE 4.2: Environment's Description**

Table 4.2 specifies project's environment as well as the equipment that is provided by the university’s robotics laboratory. The laboratory offers a range of different robot platforms for students to work on. For this particular project, one PC had been provided that is located in the robotics laboratory, different NAO platforms (torso, full body) and a license for the Webots simulator. Open source ‘Redmine’ project management tool had been used to track project development. Peas framework and hnn package had been used to support implementation of prototypes.

4.4 Performance Assessment

<table>
<thead>
<tr>
<th>#</th>
<th>Prototype</th>
<th>Date</th>
<th>Requirements</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>Dynamic scene representation from laser head data</td>
<td>12/06/15</td>
<td>1.1 - 1.2</td>
<td>Dynamic scene extracted features are displayed</td>
</tr>
<tr>
<td>1.</td>
<td>Dynamic scene representation from video data</td>
<td>26/06/15</td>
<td>1.1 - 1.3</td>
<td>Dynamic scene extracted encoded features are displayed</td>
</tr>
<tr>
<td>2.</td>
<td>Dynamic scene recognition using video data</td>
<td>14/07/15</td>
<td>2</td>
<td>Two dynamic scenes matching is performed and results are displayed</td>
</tr>
<tr>
<td>3.</td>
<td>Evolutionary approach to dynamic scene recognition</td>
<td>27/07/15</td>
<td>3.1 - 3.5; 10.1</td>
<td>Training results and fitness development output</td>
</tr>
</tbody>
</table>

**TABLE 4.3: Prototypes Assessment**

Table 4.3 presents specific prototypes that will be developed to satisfy posed requirements from Table 4.1. Prototype 0 is a preliminary prototype; it will be focused on determining whether laser sensor can be used to meet the requirements. Prototype 1 will utilise video camera to extract gesture features and
encode features into a graph. Prototype 2 will concentrate on matching two encoded gestures. Prototype 3 will employ evolutionary approach to gesture recognition using gesture feature encoding from the Prototype 1.

The supervisor and the co-supervisor of the project will assess the prototypes. Due to the exploratory approach of the project, results of assessments will drive the development of the project. Therefore, deviations from the main plan may occur.

4.4.1 Prototype 0 Assessment

Features, extracted from the laser head sensory data, will be displayed for the evaluation and a decision will be made whether the laser sensor can be used for further prototypes. Other alternative sensors will be discussed and the most appropriate one for the requirements stated will be chosen.

4.4.2 Prototype 1 Assessment

Features, extracted from the video data, will be encoded and presented using the chosen data structure. The chosen data structure will persist over the next prototypes, as it will become the main encoding used for the gestures. The advantages of the chosen data structure will be presented and compared to other alternatives.

4.4.3 Prototype 2 Assessment

Two gestures matching will be used to demonstrate that it is possible to determine whether gestures are alike using their encoded features data. Processing time will be measured to determine whether this approach is sufficient for real-time operations. Faster processing speed alternative approaches will be considered for gesture matching purposes.
4.4.4 Prototype 3 Assessment

An alternative approach will be assessed that will be performing gesture classifications real-time based on trained data. Training results and the fitness function will be used to assess evolution of the controller. Processing speed will be measured.
### 4.5 Risk Assessment

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L</th>
<th>I</th>
<th>Resolution Plan</th>
</tr>
</thead>
</table>
| 1. | Chosen sensor cannot satisfy posed requirements or cannot provide quality data | M   | M   | a) Alternative sensor choice should be made.  
b) Sensor fusion should be considered in case if none of the available sensors can provide sufficient data |
| 2. | Sensory data cannot be parsed and encoded for further analysis               | L   | H   | a) Alternative sensors should be considered or b) sensor fusion                                                                                  |
| 3. | Matching of encoded sensory data cannot be performed because two encoded gestures are not compatible and no pattern can be distinguished in the encoding | L   | H   | a) Alternative encoding approach should be considered. Patterns should be visible in the encoded gestures (e.g. saliency regions). Matching algorithm should perform on the same encoding |
| 4. | Communication between developed components cannot be established because used encoding format is not compatible with the functionality required to satisfy requirements | M   | L   | a) Alternative functionality should be considered that would deliver results needed to satisfy requirements.  
b) If no alternative functionality is available, encoding has to be changed to comply with the functions definitions |
| 5. | Communication between framework/package cannot be established               | M   | H   | a) Alternative communication approaches should be reviewed or b) alternative frameworks/packages considered |
| 6. | Training algorithm cannot provide sufficient training for the controller    | L   | M   | a) Increase amount of training data in the training set or b) consider using another training algorithm.  
c) If insufficient training occurs due to inappropriate encoding and the lack of patterns, then another encoding should be chosen |
| 7. | Evolution is taking too long to evolve a controller without fitness improvements | M   | M   | a) Fitness function can be too complicated or b) have contradicting factors that can be revised  
c) Evolutionary parameters may be inadequate and have to be improved |
| 8. | Gestures are not recognisable by the evolved controller                     | M   | H   | a) Fitness function used during evolution is incomplete or b) too complex to be able to evolve a useful controller.  
c) Encoding is inappropriate for recognising any patterns within an encoded gesture |
| 9. | Hardware failure                                                             | L   | H   | a) Replace hardware by requesting it from the department                                                                                         |
| 10.| Difficulty using the environment or performing programming                 | L   | M   | a) Dedicate extra time for learning the environment/programming language                                                                        |

**TABLE 4.4: Enumerated Risks**  
L — Likelihood; I — Impact
4.5.1 Risk 1, Risk 2 & Risk 3 - Sensory Data

Sensors collect information from environment, extracting data that they are designed to see. In example, laser sensor data contains information regarding distances to the objects in the field of view, while camera data contains depthless information about the environment. Therefore, it is necessary to recognise and match abilities of a sensor to posed requirements.

Once agreed on the sensor, the data must be parsed and encoded in some format, so that software can operate on that data. Therefore, the sensor should be able to provide meaningfully structured data. In example, laser sensor should provide a fixed array of distances to the object within the field of view, while camera should provide an array of images that encode colours of the environment within the field of view.

The likelihood of this risk is low as it should be apparent which sensor to use prior the project start. If it is not apparent or not decided, changes in sensors will have high negative impact on the project as the encodings and data structures compatible with one sensor’s data may not be compatible with the other sensor’s data. Nevertheless, an alternative sensor should be used when described above hardware requirements are not satisfied.

4.5.2 Risk 4 - Framework/Package Programming Interface

It is necessary to ensure that the programming interfaces provided by frameworks and packages combined and used during a project can operate on the same data without the need to perform unnecessary conversions during system’s execution. Every conversion consumes additional execution time and may introduce small deviations or even errors in the original data.

The likelihood varies from low to medium depending on whether a right sensor had been chosen from the beginning of the project. The impact of a change in functionality is relatively low as frameworks/packages usually provide a wide range of functionality suitable for most of the data structures and encodings.
4.5.3 Risk 5 - Framework/Package Interaction

When different technologies are combined to solve a task, an explicit communication mechanism has to be constructed to ensure that data is being passed from one technology to another without data losses. When difficulties arise during interaction, alternative communication approaches should be considered or an alternative framework or package chosen to solve the same problem.

4.5.4 Risk 6 - Controller Training

Chosen training algorithm can fail to train the controller using compiled training data set. In such case, training data set should be extended. If adding more training data does not improve training results, another training algorithm should be chosen. It can happen that the chosen gesture encoding would be too random for a controller to extract any pattern from different training samples. In this case the encoding has to be changed.

4.5.5 Risk 7 & Risk 8 - Evolution

Evolution of a controller can be very delicate due to many parameters. Since setting of the parameters is an inevitable part of the evolutionary process, the likelihood of the risk of setting wrong parameters is relatively high.

Another aspect is the fitness function that leads the evolution towards a solution. Too complex fitness function can slow down the evolution process and introduce too many restrictions leading to a sub-optimal solution.

4.5.6 Risk 9 - Hardware

Hardware, such as robots, computers and scanners can fail for many different reasons. The likelihood of such failure is relatively low but the consequences can be devastating to project’s progress due to loss of data or required time to change failed hardware. Backups and additional hardware, such as availability of two or more computers and robots, should support the recovery process.
## 4.6 Project Plan

**Figure 4.1: Project Plan (Gantt Chart)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Begin date</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define project</td>
<td>1/16/15</td>
<td>2/17/15</td>
</tr>
<tr>
<td>Literature Review</td>
<td>1/20/15</td>
<td>4/21/15</td>
</tr>
<tr>
<td>Webots-peas Integration</td>
<td>5/15/15</td>
<td>5/22/15</td>
</tr>
<tr>
<td>Testing integrated model</td>
<td>5/25/15</td>
<td>5/29/15</td>
</tr>
<tr>
<td>Report</td>
<td>6/1/15</td>
<td>6/1/15</td>
</tr>
<tr>
<td>Prototype 0</td>
<td>5/29/15</td>
<td>6/12/15</td>
</tr>
<tr>
<td>Demo &amp; Report</td>
<td>6/15/15</td>
<td>6/15/15</td>
</tr>
<tr>
<td>Prototype 1</td>
<td>6/15/15</td>
<td>6/26/15</td>
</tr>
<tr>
<td>Gesture Encoding</td>
<td>6/15/15</td>
<td>6/19/15</td>
</tr>
<tr>
<td>Gesture Recognition</td>
<td>6/19/15</td>
<td>6/26/15</td>
</tr>
<tr>
<td>Demo &amp; Report</td>
<td>6/29/15</td>
<td>6/29/15</td>
</tr>
<tr>
<td>Prototype 2</td>
<td>6/26/15</td>
<td>7/10/15</td>
</tr>
<tr>
<td>Detectors Evolution</td>
<td>6/25/15</td>
<td>7/3/15</td>
</tr>
<tr>
<td>Neural Network Training</td>
<td>7/3/15</td>
<td>7/10/15</td>
</tr>
<tr>
<td>Report</td>
<td>7/13/15</td>
<td>7/13/15</td>
</tr>
<tr>
<td>Prototype 3</td>
<td>7/15/15</td>
<td>7/27/15</td>
</tr>
<tr>
<td>Report</td>
<td>7/28/15</td>
<td>7/28/15</td>
</tr>
<tr>
<td>Report Writing</td>
<td>7/28/15</td>
<td>8/20/15</td>
</tr>
<tr>
<td>Report Submission</td>
<td>8/21/15</td>
<td>8/21/15</td>
</tr>
</tbody>
</table>
Chapter 5

Methodology

Scene analysis is applicable to a wide range of real-world applications, ranging from static scene content retrieval, presented in surveys by Rosenfeld (1973) and Beghdadi et al. (2013) on picture descriptors, applied to feature extraction, to dynamic real-time video-surveillance activity predictions as presented in Dollar et al. (2005). Such applications represent a scene by following a set of procedures to retrieve most relevant information for further analysis. These procedures are common in both surveys by Poppe (2010) and Aggarwal and Ryoo (2011), which are: (1) detection of regions of interest (ROI), (2) feature extraction and (3) scene representation based on extracted features. Listed procedures can be further specialised or combined based on the nature of the application.

The word ‘dynamic scene’ is used in this chapter to refer to both pre-recorded video and live camera stream, as the aim is to implement a general algorithm for gesture recognition. For the purposes of this project, scene analysis consists of ROI detection, feature extraction and action representation followed by action recognition.
Chapter 4. Methodologies

5.1 Prototype 0: Dynamic Scene Representation from Laser Head Data

During implementation of the prototype 0 (the preliminary prototype), laser head is being put to test to see whether the data obtained from the sensor can be used further in the feature extraction and gesture recognition phases.

5.1.1 Design

When the data from the sensor is acquired, it is clustered using fuzzy clustering algorithm, introduced by Bezdek et al. (1984), to determine which points may belong together in a scene. In this way distinct features may be identified and used for further analysis. No particular features are distinguished, as no specific information about the environment is available at any moment due to the properties of the sensor. Therefore, many assumptions have to be made based on the previous knowledge of the environment to determine what laser data is actually representing.

A simple test is being performed: NAO robot waves its hand to another NAO robot that uses laser sensor to acquire from the environment and detect the waving gesture. Since laser performs scans in 2 dimensions, the following assumptions have to be made: (1) location of both robots must be known prior and during the execution and (2) position of the surroundings must be known in advance and is not to be change during the execution.

5.1.2 Implementation

Laser head data values come in an array of a length equal to the width of the field of view of the laser, which is 240 degrees scanning angle, as described in section 2.2.1.3. Every value in the array represents the distance to an object in the field of view that reflects the laser as it reaches that object. That is why the laser returns 2–dimensional data: first dimension being the field of view and the second — depth (distance from laser to an object).
Array values are then mapped onto a Cartesian plane and clustered first by x-axis (spectrum of the field of view) and later by y-axis (the depth) giving a 2-dimensional clustering. The reason for 2-dimensional clustering is to separate acquired points as much as possible from each other along the horizon (maximum laser range) and by the distance from the robot/laser. Clustering is performed using fuzzy c-means algorithm, which allows an array value (distance measure to an object) belong to at least one virtual cluster. The algorithm calculates centres of each cluster and then makes every point a member of at least one of the clusters on the basis of the distance between the point and the centre of that cluster.

Fuzzy c-means clustering algorithm requires a priori knowledge of the number of clusters that is used during membership assignment. Since this is a preliminary prototype, 5 clusters are chosen for testing purposes. Therefore, 5 clusters are distinguished by x-axis at first and 5 additional clusters are distinguished by y-axis out of the resulting former clusters. As a result, maximum of 25 potential clusters are distinguished during a single moment in a dynamic scene.

Fuzzy c-means clustering algorithm is chosen for testing purposes and no evaluation is conducted with the aims of finding the most optimal unsupervised learning algorithm for the situation at hand. As an alternative to c-means algorithm, a range of other unsupervised learning techniques can be used to achieve similar results. Some of the alternative techniques include hard clustering (MacQueen, 1967) — no more than one cluster per data item (as opposed to the fuzzy clustering) and neural network based clustering (Kohonen, 1982), where a neural network can be evolved or trained. Strnadova-Neeley (2013) provides a summary of an article surveying clustering algorithms. The main difference between existing approaches to clustering mentioned in the summary is their precision of assigning items to clusters that is proportionate to the computational time required for the clustering. Another difference lies in the nature of the data that is to be clustered. In particular, if data is linearly separable, calculation of corresponding agglomerations can be achieved, whereas in cases of non-linear separability, spectral clustering may be necessary as in the example of principal component analysis.
5.1.3 Evaluation

The implementation is evaluated in the Webots environment. A world is created with 2 NAO robots standing opposite to each other. One of the robots has a laser scanner, described in section 2.2.1.3, installed on the top of its head, which points right over the head of another robot and continuously scans the environment, while another robot is performing a hand-waving gesture.

Compliant with the previously mentioned assumptions, the supervisor (Webots specific programming paradigm, where supervisor is an intermediary, judge, command issuer, or all of the above to one or more robots present in the world and performing some tasks) is able to distinguish a moving object (a region of interest), represented by a cluster of points returned by the laser head and assuming that the object is the arm of another robot.

Currently the algorithm exhibits excessive clustering. Some single points are considered as separate clusters although visually they should belong together in a single cluster. This can be avoided by adjusting threshold separation values or by specifying exact number of clusters that are expected to be found in the laser scanner data, which points to the necessity of the domain knowledge prior execution.
Figure 5.1: Prototype 0 - Waving Gesture

a) No gesture  
b) Right hand close to the head (covers up right goal bar)  
c) Right hand away from the head  
d) Left hand away from the head  
e) Left hand close to the head
Figure 5.1 presents 5 scenes in the second column and their corresponding fuzzy (soft) clustering results in the first column. These scenes were chosen as they represent a simple test discussed earlier: NAO robot waves a hand to another NAO robot that uses laser sensor to detect the waving gesture. Points from the same cluster in first column are connected by a line and have the same colour for visualisation purposes. Red rectangles separate out the region of interest - the gesture that is being performed. Clusters that appear static on all scenes correspond to goals bars behind the robot that performs the waving gesture.

To read the figure many assumptions have to be made and the resulting region of interest is limited to a 2-dimensional representation, which is very difficult to analyse further, needless to say, to be used for feature extraction.

5.1.4 Discussion

Due to the need of making the unavoidable assumptions described earlier that act as constraints on the system, laser head is not sufficient for a robust gesture recognition algorithm. One workaround that may overcome the issue of limited 2-dimensional view of the environment is to make a number of laser scans adjusting head angle at every scan. Such approach will unfortunately pose another restriction on the system in form of delays, associated with the head joint readjustments that will lead to hindered execution with interruptions necessary for every scan. Another solution to the emerged constraints is to use sensor fusion, data coming from different sensors, in a pursuit for finding a solution to a single problem, namely, detection of regions of interest (Hall and Llinas, 1997).

Garg et al. (2009) separate out two main approaches to gesture interaction: glove–based and vision–based and LaViola (1999) distinguishes an additional hybrid approach that uses sensor fusion of the two approaches identified by both Pragati Garg and Joseph LaViola. When it comes to glove–based approaches, the author agrees that despite the ease of data acquisition, such
devices make the interaction experience cumbersome. While vision–based approaches are more natural as they resemble biological vision, they tend to introduce additional challenges to gesture recognition tasks, such as lightning sensitivity, camera movements, and depth awareness that impact robustness.

Since this project aims at developing a robust, non-intrusive, and cheap solution to gesture recognition as stated in section 1.1, a vision-based approach will be used from here on with the support of the on-board video camera available on the forehead of the NAO robot. For the future improvements to robustness of the algorithm, data from video camera can be fused together with the data from the laser head sensor resulting in sensor fusion and providing depth awareness to the vision–based approaches.

5.2 Prototype 1: Dynamic Scene Representation from Video Data

Dynamic scene representation consists of ROI detection, feature extraction, and action representation. ROI detection for action recognition means that a scene is analysed to detect where the action is happening, otherwise called as segmentation. Few of the methods for ROI detection using vision-based approach had been studied, some of which include corner detection presented in Laptev and Lindeberg (2003), salient points detection used in Kadir and Brady (2003), subspaces of correlated movements studied by Wong and Cipolla (2007), or Hesian Matrices as proposed by Willems et al. (2008). All approaches achieve some degree of robustness with respect to rotation, translation or noise tolerance. Techniques differ in feature extraction methods (also known as detectors) that dictate further selection of algorithms used on the accumulated features encoding (also known as descriptors) for recognition.

Once scene is segmented, ROI becomes available and is used for feature extraction. The main choice lies between global or local features that are used for scene representation and is solely based on the nature of the application. Global features usually are dense and contain conjunctions of few local features making global features more expensive in terms of required space they occupy (Aggarwal and Ryoo 2011; Poppe 2010). Examples of global features
include silhouettes, edges, optical flows, grid representations, or space-time volumes. Local features, on the other hand, can capture minimal information needed by the application. Examples of local features include kinematic features, or cuboids.

Further, a scene, consisting of extracted features should be represented in a way to allow for faster and efficient operations upon the extracted features by any algorithms that are needed for further analysis. The representation is merely a question of which algorithms will be applied on the representation further. In particular, extracted features may be represented as arrays, matrices, or linked lists.

5.2.1 Design

‘String of feature graphs’ is used for activity representation in this project, which is presented by Gaur et al. (2011). ‘String of feature graphs’ employs graph representation of kinematic (local) features that are further concatenated to make a sequence of feature graphs, where every distinct feature graph corresponds to a distinct scene at a particular time. Concatenated sequence of feature graphs represents spatio-temporal features and their relations in a sequence of scenes.

Feature graphs capture local kinematic features that contain the most relevant information needed for action representation. In theory, capturing kinematic features is less noisy than capturing more complex features, such as blobs or salient spots that contain a collection of features, as the amount of noise is proportionate to the amount of data collected (Lisin et al., 2005). Another advantage when using graphs as underlying data structure is that graphs can be scaled easily by adding extra nodes (features). Yet another advantage of graphs data structure is that graphs allow viewing data in two dimensions — nodes (features) and their relationships.

Representing an action requires to take temporal information into account. Different approaches had been devised to represent an action, some of which include Hidden Markov Models as in Malgireddy et al. (2012) or more straightforward approach, where sequence of scenes is chained together as in Faria et al. (2014). As it had been already mentioned, ‘String of feature graphs’
stores temporal information implicitly by concatenating feature graphs, where every instance of a graph is a character of a string.

### 5.2.2 Implementation

The implementation utilises functionality provided by the open source OpenCV library version 2.4.10.1 ([OpenCV, 2014](#)). The reason for choosing the library is because it provides an array of useful functions that simplify development of applications that process images or video. Another reason for making this choice is because the library can be found on a NAO robot.

Prior to ROI detection, every scene is pre-processed in the following way using OpenCV functions:

- Background is detected for future background-foreground subtraction
- Illumination reduction. Optional, advantageous for cases where illumination is fluctuating
- Edge enhancement to the foreground by applying erosion and dilation

![Prototype 1 - Pre-processed Scene](image)

**Figure 5.2**: Prototype 1 - Pre-processed Scene

a) No illumination reduction  
b) Illumination reduction applied
On Figure 5.2 first column shows result of the pre-processed scene at a particular time and second column shows original (except for illumination reduction) scene at the same time. From the figure it can be seen that illumination reduction in this particular setting, as it can be seen from the image, does not alter the input data noticeably. This method of illumination reduction is expected to be beneficial during modest lightning fluctuations, which will be hard to convey in a single image.

Illumination reduction is performed as follows: frame is converted to YUV colour space, where Y channel (luma) describes every pixel as a value between dark and light and UV represent red and blue colours of a frame (de Campos, 2006). After YUV decomposition, the Y channel is normalised using OpenCV functionality, which results in brightness normalisation after which the frame is reconstructed from the resulting YUV channels. Another approach could be to increase or decrease the Y channel with a specific scalar with regards to the current lightning conditions.

### 5.2.2.1 ROI Detection

ROI consists of the following regions: face, upper body and contour of a moving object. Face and upper body are detected using OpenCV classifier, while moving object detection is a result of background and foreground subtraction. Contour of a moving object is detected using OpenCV function and is later inscribed in a rectangle. A moving object is considered as a part of an overall body only in case if it originates from the upper body.

Further a potential limb is analysed by using OpenCV functions for detection of hull and convex defects to find a break point (elbow) and smaller details, such as fingers. Therefore, the detection is accomplished in hierarchical manner, where hierarchy 1 includes main frame detection, consisting of face and body, hierarchy 2 consists of at least one potential limb detection and hierarchy 3 — potential fingers and other details. It is believed that such decomposition is appropriate for gesture recognition as the gesture detection can be done at any of the three levels of hierarchy — body gesture, limb gesture, or finger gesture (i.e. sign language).
It is worth to note that the algorithm will search for neither hierarchy 2 nor hierarchy 3 unless hierarchy 1 is detected, therefore hierarchy 1, human body detection, is essential for the algorithm.

On Figure 5.3 first column corresponds to original frame before pre-processing while second column illustrates corresponding ROI — face (green), upper body (red), contour of a moving object (black line that is neither part of the face nor the upper body), and details (fingers in this case) of a moving object (blue).

**5.2.2.2 Feature Extraction**

Extracted features are joint positions encoded as nodes of a graph and their relations — Euclidean distance between the nodes. Following features had
been chosen to represent a gesture:

- face and hand with face-hand distance
- first and second hand with hand-hand distance
- first and second shoulder with shoulder-shoulder distance
- first and second elbow with elbow-elbow distance

It is believed by the author that these features are sufficient to represent an upper body gesture of hierarchy 2.

Previous studies had explored other alternatives for feature extraction. For instance, Brand et al. (1997) uses centroid position of blobs, where each blob represents a hand and hand positions are plotted on a Cartesian plane to find relationship between corresponding blobs. Willems et al. (2008) propose Hessian Matrices of whatever is changed in frames, arguing that the nature of a change does not matter and should not be categorised as, for example, a body part. Bregonzio et al. (2009) focus on clouds of interest points, where a cloud consists of a shape, speed, and density. Alternative approaches extract global features that contain noise, which is a drawback for many applications.

5.2.2.3 Feature Graph Representation

Feature graph is implemented as an adjacent linked list, where nodes represent extracted features and edges between nodes correspond to Euclidean distances between the features. The resulting graph is undirected, where every node has its absolute position on a plane, its identification number, and a distance measurement between itself and a connected node.

During feature extraction, a graph is being created and populated with extracted features for each scene. Graphs are further concatenated to create a list of feature graphs.
Figure 5.4: Prototype 1 - Concatenated String of Feature Graphs

Figure 5.4 shows the resulting concatenated String of Feature Graphs, as described by Gaur et al. (2011), for the gesture that is shown on Figure 5.3 a) and b) (that correspond to frames 1 and 6), but not c). Blue points represent shoulders and corresponding Euclidean distance between them as the first feature, while green points correspond to the arm and contribute to the other feature — face-hand distance.

### 5.2.3 Evaluation

Evaluation proceeded in a purely empirical fashion in both Webots environment and using a standard PC camera. The reason for testing in simulated and
real environment had been advised to show that the solution is robust and can undergo a great amount of noise.

In both environments, evaluation consisted of drawing extracted features while random movements/gestures had been executed. At the same time, feature graphs were printed out for manual verification.

As a result, the implementation is robust and can bear a certain amount of noise and lightning fluctuations. Experiments showed that enforced illumination reduction had shown to be useful during the increased lightning (i.e. day time), but harmful during reduced illumination (i.e. night time), which complies with Maddern et al. (2014)

Since OpenCV standard classifiers for body parts detection are trained for humans, it had been a challenge to get the classifier to operate in Webots as NAO face is far from human-like faces. Therefore, it had been decided to add a photo of a human on NAO in the Webots simulator and then classifier began to detect faces. This is rather a workaround and the standard classifier can be trained for NAO face recognition in the future.

Limb detection showed to be quite precise although the details detection, such as elbows or fingers is a challenge, as many threshold values are required to distinguish small variations in geometric figures. Therefore, elbow detection is not as good as limb detection, which leads to a position of detected elbow to ‘jump’ between frames; also finger detection is not as precise.

In summary, the implemented algorithm detects distinct joint locations, which are extracted and stored in a feature graph. Feature graphs are concatenated to create a list (string) of feature graphs.

### 5.3 Prototype 2: Dynamic Scene Recognition

Dynamic scene in this context consists of a sequence of static scenes, where a single static scene represents a scene at a particular time. Dynamic scene recognition includes calculation of affinity matrix, spectral clustering, and dynamic time warping (Aggarwal and Ryoo 2011; Poppe 2010). Affinity matrix holds similarity information between two dynamic scenes in a single matrix,
where one scene is the original sequence of frames and the other one is the query sequence of frames. Spectral clustering is a clustering algorithm, where eigenvectors represent cluster. Dynamic time warping is used to measure distances between two extracted sequences from dynamic scenes.

String of feature graphs will be used further as ‘SFG’ to refer to any dynamic scene since all the further explanations will be in terms of the feature graphs (FG).

5.3.1 Design

Same approach, described in Gaur et al. (2011) is followed to match two SFGs. Output from section 5.2 is used further in the matching process.

Two SFGs are represented in affinity matrix in a way so that the diagonal holds information about similarity between features nodes, while the rest of the matrix represents similarity between edges. Each cell has value that is either zero if the similarity is greater than a threshold value or a difference between threshold and a similarity value if the value is less than the threshold value.
\[ M(a, a) = \begin{cases} \tau_1 - \text{distance}(i_1, i_2) & \text{if } \text{distance}(i_1, i_2) \leq \tau_1 \\ 0 & \text{otherwise} \end{cases} \]

\[ M(a, b) = \begin{cases} \tau_2 - \text{distance}(i_{1j_1}, i_{2j_2}) & \text{if } \text{distance}(i_{1j_1}, i_{2j_2}) \leq \tau_2 \\ 0 & \text{otherwise} \end{cases} \]

\((5.1)\)

where

\(M\) : affinity matrix
\(a, b\) : matrix indices
\(\tau_1, \tau_2\) : threshold values, where \(\tau_1\) is the maximum allowed Euclidean distance between two nodes and \(\tau_2\) is the maximum allowed deviation between edges inclinations
\(i_1, i_2, j_1, j_2\) : nodes, where \(i_1/j_1\) belongs to the original SFGs and \(i_2/j_2\) belongs to the query SFGs
\(i_{1j_1}, i_{2j_2}\) : edge between nodes \((i_1, j_1)\) and \((i_2, j_2)\)
\(\text{distance}(i_1, i_2)\) : distance between two nodes that belong to different SFGs
\(\text{distance}(i_{1j_1}, i_{2j_2})\) : inclination between edges that belong to different SFGs

**EQUATION 5.1: Affinity Matrix Definition**
Affinity matrix is later used for clustering, which results in a single eigenvector per one affinity matrix that represents every cluster in the matrix. Principal eigenvectors (eigenvector associated with the highest eigenvalue) are retrieved from both affinity matrices of the original SFGs and of the query SFGs. Alternatively c-means or k-means could have been used for clustering of the data, but spectral clustering method is superior in cases when the data is not linearly separable. In case of feature graphs, depending on a gesture, some gestures may not be linearly separable so the traditional k-means or c-means algorithms would produce unexpected results (von Luxburg 2007; Kaur et al. 2011).

Having a principal eigenvector of an affinity matrix that has both original SFGs and query SFGs encoded in it, the correspondence has to be established between the two SFGs through binarisation of the extracted eigenvector. Leordeanu and Hebert (2005) describe a greedy algorithm for doing that in detail. In summary, algorithm operates in the following manner. First largest value in the vector is found and is changed to 1, while all other occurrences of the same value in the vector are set to 0. The algorithm is repeated until either the largest value found is 0 or the end of the vector is reached. In this way, the greatest correspondence confidence between the two SFGs is established.

Binarised eigenvectors are used in calculation of the distance between original and query SFGs.
\[ S_1 = M_1 \times x'_1 \times (x'_1)^T \]
\[ S_2 = M_2 \times x'_2 \times (x'_2)^T \]
\[ D = 1 - \frac{S_1}{S_2} \]

(5.2)

where

- \( P \) : original SFGs
- \( Q \) : query SFGs
- \( M_1 \) : affinity matrix from Equation 5.1 for both \( P \) and \( Q \)
- \( M_2 \) : affinity matrix from Equation 5.1 for \( Q \)
- \( x'_1 \) : binarised principal eigenvector for \( M_1 \)
- \( x'_2 \) : binarised principal eigenvector for \( M_2 \)
- \( S_1 \) : similarity matrix between \( P \) and \( Q \)
- \( S_2 \) : similarity matrix between \( Q \) and \( Q \)
- \( D \) : distance matrix between \( P \) and \( Q \)

**Equation 5.2: SFGs Distance Definition**
Resulting distance matrix ($D$) is then used for calculation of the cost matrix that is further used to perform dynamic time warping over the accumulated cost matrix to align SFGs. The alignment is necessary because two dynamic scenes may be of different length or a gesture may begin at any point in a dynamic scene. Ideally, it is desirable to find a gesture of any length in a sequence of actions of any length, where a gesture can begin and end at any time.

\[
C(n, 1) = \sum_{k=1}^{\text{width}(M)} D(k, 1)
\]

\[
C(1, m) = D(1, m)
\]

\[
C(n, m) = \min(C(n - 1, m - 1), C(n - 1, m), C(n, m - 1)) + D(n, m)
\]  

(5.3)

where

- $M$ : affinity matrix from Equation 5.1
- $C$ : accumulated cost matrix
- $n, m$ : matrix indices, where $n = \text{width}(M)$ and $m = \text{height}(M)$ and $M$ is affinity matrix
- $D$ : SFGs distance matrix from Equation 5.2
- $\min(...)$ : function that returns minimum value of surrounding cells in the cost matrix

EQUATION 5.3: Cost Matrix Definition

Dynamic time warping is further performed by traversing the matrix and extracting the cheapest path.
5.3.2 Implementation

5.3.2.1 Affinity Matrix Construction

Symmetric affinity matrix is constructed of size equal to the length of a SFG multiplied by the amount of features present in every feature graph. Both SFGs (the query and the original sequences) are iterated and affinity matrix is filled based on the threshold values — $\tau_1 = 1\%$ of the frame diagonal is allowed as the maximum deviation between location of nodes and $\tau_2 = 1\%$ of $2\pi$ as the maximum deviation between edges.

**Algorithm 5.1:** Affinity Matrix Calculation Algorithm

**Data:** original SFG, query SFG  
**Result:** symmetric affinity matrix

```plaintext
foreach oFG in (original SFG) do  
  foreach qFG in (query SFG) do  
    foreach olist-item in oFG do  
      foreach qlist-item in qFG do  
        if olist-item.destination == qlist-item.destination then  
          calculate Euclidean distance between original linked list item and query linked list item  
          if (distance $\leq \tau_1$) then  
            set affinity matrix value to $\tau_1 - distance$  
          else  
            set affinity matrix value to 0  
        else  
          calculate angle of the edge between olist-item end-points  
          calculate angle of the edge between qlist-item end-points  
          if (deviation between both angles $\leq \tau_2$) then  
            set affinity matrix value to $\tau_2 - deviation$  
          else  
            set affinity matrix value to 0
```

Algorithm 5.1 shows steps taken in construction of affinity matrix for original and query SFGs. For clarification, list-item refers to the linked list that represents a feature graph, where prefix o/q correspond to original/query identifiers.
Figure 5.5 presents a heatmap of the affinity matrix of the gesture partly shown on Figure 5.3 a) and b). Black area means that either the corresponding feature (from features given in section 5.2.2.2) is not present on a feature graph (along the width) or that no similarity is found on the query feature graph (along the height). The brighter the cell, the more similar the corresponding feature graph is to a query feature graph.

This particular example shows an affinity matrix of the same SFG being both the query and the original sequence. The more variations there are in in the query and original SFG, the more varied the matrix becomes.

It is worth noting that in cases when original SFG and query SFG have different lengths, the shortest is being appended with empty FG until the sizes of the both become equal.

### 5.3.2.2 Spectral Clustering

Spectral clustering is performed using OpenCV function for computation of eigenvectors and corresponding eigenvalues of the affinity matrix. Further, eigenvectors are sorted and principal (associated with the highest eigenvalue) vector is extracted. Extracted principal eigenvectors represent separate clusters of spatio-temporal features.
Figure 5.6 shows plotted principal eigenvector, where black circles show values of principal eigenvector of original SFG and red plus-signs show values of principal eigenvector of query SFG. In the presented case, black and red markers are on exactly the same positions since both original and query SFGs are the same gesture.

The figure can point on the number of clusters present on the affinity matrix over the length of the eigenvector, which is the total length of all clusters present on an affinity matrix (in this case, the total length of all clusters is 180). Every continuous interval on the figure corresponds to a single cluster. For visualisation purposes, potential clusters are distinguished using red rectangles, although such distinction is just a speculation.
5.3.2.3 Dynamic Time Warping

Using Equations 5.2 and 5.3, cost matrix is computed. The cost matrix represents the distance of a path from the original SFG to the query SFG.

Figure 5.7 shows heatmap of the cost matrix of the two identical SFGs that encode a gesture that is shown on Figure 5.2 a) and b). The white line represents
the cheapest path between the two SFGs.

**Algorithm 5.2:** Searching for Cheapest Path over DTW–Generated Cost Matrix

**Data:** DTW cost-matrix

**Result:** cheapest-path

initialization cheapest-path[i][j] = 0;

\[ i = \text{cost-matrix.rows} \]

\[ j = \text{cost-matrix.columns} \]

**while** \( i > 0 \) or \( j > 0 \) **do**

\[ \text{if } i > 0 \text{ and } j > 0 \]

\[ \quad \text{diagonal-cost} = \text{cost-matrix}[i-1][j-1] \]

\[ \text{else} \]

\[ \quad \text{diagonal-cost} = \text{INFINITY} \]

\[ \text{if } i > 0 \]

\[ \quad \text{left-cost} = \text{cost-matrix}[i-1][j] \]

\[ \text{else} \]

\[ \quad \text{left-cost} = \text{INFINITY} \]

\[ \text{if } j > 0 \]

\[ \quad \text{down-cost} = \text{cost-matrix}[i][j-1] \]

\[ \text{else} \]

\[ \quad \text{down-cost} = \text{INFINITY} \]

\[ \text{if } \text{diagonal-cost} \leq \text{left-cost} \text{ and } \text{diagonal-cost} \leq \text{down-cost} \]

\[ \quad i = i - 1 \]

\[ \quad j = j - 1 \]

\[ \text{else if } \text{left-cost} < \text{diagonal-cost} \text{ and } \text{left-cost} < \text{down-cost} \]

\[ \quad i = i - 1 \]

\[ \text{else if } \text{down-cost} < \text{diagonal-cost} \text{ and } \text{down-cost} < \text{left-cost} \]

\[ \quad j = j - 1 \]

\[ \text{else if } i \leq j \]

\[ \quad j = j - 1 \]

\[ \text{else} \]

\[ \quad i = i - 1 \]

\[ \text{cheapest-path}[i][j] = 1 \]

Algorithm 5.2 shows the cheapest path between original SFG and query SFG that results in a white line from Figure 5.7. Found cheapest path corresponds to the most optimal mapping between the two SFGs. In particular, the final output of the algorithm shows which FG of the original SFG correspond to which FG of the query SFG. The resulted mapping between FGs is not precise and should not be taken literally. The found cheapest path between two SFGs represent
global mapping, taking into account the overall picture rather than trying to map every individual FG of the both SFGs.

5.3.3 Evaluation

The algorithm had been tested both in Webots environment and using a simple PC camera. The original video had been recorded in advance and stored in the memory. The query video had been recorded later and the algorithm tried to find whether the query video matches the original video — if gestures in both videos are similar. Therefore, it can be said that the testing underwent empirical evaluations.

PC camera had been used the most during intermediate and final evaluations due to the ease of execution. In Webots environment additional steps need to be taken and previously mentioned supervisor model obeyed, which results in additional time being spent on porting the algorithm to the Webots environment. Nevertheless, such steps were taken to ensure that the algorithm operates in Webots environment as well as it does when executed from command line.

No statistical analysis of the resulted algorithm had been conducted, as this is not the task of this project. For statistical results, the reader is referred to the original paper on String of Feature Graphs mentioned earlier. Assuming that the algorithm is reproduced correctly, the results should be alike with the exception of the algorithm being run in the Webots environment, since the noiseless environment, provided by the simulator, should ensure purer gesture recognition results.

5.3.4 Discussion

Although the algorithm offers a stable method for gesture recognition in vision-based approaches, the processing is slow. Moreover, the algorithm becomes slower when the length of the dynamic scene increases. As an example, the processing time of the algorithm for both original and query videos of 1 second in length is equal to 1.5 seconds. When both original and query videos are of 7 seconds in length, the processing time increases to 32 seconds. Therefore,
this method is unacceptable for the purposes of this project, since the objective is to develop an approach that will allow real-time gesture recognition.

5.4 Prototype 3: Evolutionary Approach to Dynamic Scene Recognition

Instead of using spectral clustering on the affinity matrix as described in Section 5.3, approach taken in this prototype considers artificial neural networks as a machine learning method. The rationale behind switching to artificial neural networks over clustering described in previous prototype is to make a system that will be able to categorise a gesture from video/camera stream directly without any explicit conditional statements as was required by the prototype 2.

In prototype 2, in order to categorise a gesture, all the sequence of procedures — affinity matrix construction, clustering, warping, and cheapest path search — would have to be performed for every combination of original and query SFGs. This is time consuming and cannot be achieved in real time with processing time that was mentioned in Section 5.3.4. Therefore, this prototype will describe the usage of artificial neural network to perform classification of gestures in real-time.

5.4.1 Design

Kocmánek (2015) describes a novel method for digit recognition from images that employs HyperNEAT algorithm for evolution of novel detectors together with trained artificial neural network used as the final classifier. This prototype considers the same approach only on the spatio-temporal data.

Peas, a Python-based HyperNEAT implementation, developed in van den Berg and Whiteson (2013), is used for evolution of novel detectors and pure Haskell–based hnn package developed by Mestanogullari and Johnson (2014) for training the classifier network and network processing. Other open-source implementations of the HyperNEAT are available and are listed in D’Ambrosio et al.
(2014) and EPLEX (2015). Peas implementation was selected due to its complete and lightweight source code and a range of examples, which were developed for the purposes of the original research that helped in understanding of the framework as opposed to other implementations that offered many more features of the HyperNEAT algorithm. For training and input processing of an artificial neural network, hnn package provides a very simple and powerful implementation, although many other libraries exist for more sophisticated artificial neural network processing.

Haskell is an appealing programming language for artificial neural network processing due to programming paradigm that allows pure functions and promotes recursion and supports lazy execution. Lazy executions are extremely appealing for network’s input processing especially in cases, where the network is recurrent. Haskell can easily be interfaced with the main programming language of the gesture recognition system (C/C++) via Haskell Foreign Function Interface (FFI).

Python is less interface–friendly with the programming language of the gesture recognition system. Therefore, it had been decided to use the operating system's file system to exchange data between the HyperNEAT and gesture recognition system.

Overall, the approach to gesture recognition using artificial neural networks can be described as follows:

1. Construct affinity matrix of SFG of the query video in the same manner as it is done in 5.3
2. Evolve detectors
3. Train artificial neural network
4. Combine evolved detectors and artificial neural network to process affinity matrix and perform gesture classification
Figure 5.8: Prototype 3 - Overall View of Evolved System for Gesture Recognition

a) sequence of frames in a video b) string of feature graphs for the video c) affinity matrix for the SFGs d) affinity matrices feature detectors e) classifier artificial neural network e) resulting classification for the video in a)

Figure 5.8 shows the overall view of the system that utilises SFG and affinity matrix construction techniques from Section 5.2 and additionally evolved unique feature detectors and a trained neural network that performs the final classification.

5.4.1.1 Detectors Evolution

Evolution of detectors is accomplished with the HyperNEAT evolutionary algorithm by modifying one example task from van den Berg and Whiteson (2013). In the work, performed by Kocmánek (2015), the author describes detectors as processing units that detect some feature in an image. The author begins by evolving 10 simple novel detectors and later increases the amount of detectors and makes them more complicated with rationale that the more detectors are used, the more features will be extracted. In addition, the author is using a separate pool of potentially useful detectors that is later analysed and only the most useful detectors are selected.

In this prototype, maximum of 10 simple detectors are evolved due to the limitations of processing power of the PC and the evolution time. Detectors are evolved using novelty search, which means that the fitness function tries to maximise the difference between detected features of the detectors.

Every detector has 441 inputs and a single output without any hidden layers. Inputs are arranged in $21 \times 21$ square to map every single cell of the reduced
affinity matrix. The topology of the final artificial neural network produced by the HyperNEAT is defined in advance and as a result, the HyperNEAT is robbed of the ability to evolve the size of the network as well as the activation functions. All this is done to reduce the evolution time of the HyperNEAT algorithm. For more sophisticated evolutions of detectors, the austerity of the resulting topology could be loosened, which would exploit the full potential of the HyperNEAT algorithm.

It had been decided to use 4 different gestures for the purposes of this project: (i) left hand wave, (ii) right hand wave, (iii) both hands wave simultaneously, and (iv) no hands waving. Every gesture had been represented by 5 different videos.

Every video for all gestures is fed through all detectors and the output is accumulated in a vector associated with every single detector at a position dedicated for that particular gesture. For example, when video of the first gesture is fed through all detectors, the output value of every detector is rounded to the nearest integer (0 or 1) and is added to the first position of a vector associated with every detector. When video of the second gesture is fed through all detectors, the output values are rounded and added to the second position of a vector associated with every detector. The process continues until all videos are fed through detectors.

The fitness function then calculates the k-nearest neighbour Manhattan distance with \( k = 10 \) (for all detectors), which becomes the fitness of every detector. The objective of the fitness function is to maximise the distance between all detectors, which will mean that detectors are novel and every detector is able to detect different features in an affinity matrix.

5.4.2 Classifier Training

After evolution of the detectors is accomplished, their weights are saved and the turn comes to training the classifier. The classifier has 10 inputs, where every input corresponds to a single detector’s output, 1 hidden layer with 12 nodes and 4 outputs, every output corresponding to a gesture class. The topology of the classifier is taken from Kocmánek (2015), although it is not fixed and can also be evolved using the HyperNEAT algorithm.
Every video for a single gesture class is passed through evolved detectors and the outputs are collected and used for training the classifier manually. In this way, all detectors outputs for all gestures are assembled in 4 collections — one collection per gesture class. Later, a file is created that contains the training data for the classifier together with manually assigned classification. For the last, entries in the training file are shuffled, which is required for the training to be unexpected.

Alternatively, detectors could be evolved to perform classification. Although this approach would make the overall system less complicated, the fitness function will have to incorporate classification issues mentioned in Kocmánek (2015), which would lead to a very complicated fitness function.

5.4.3 Implementation

5.4.3.1 Sub-Systems Communication

The system consists of 2 parts, namely, Python HyperNEAT algorithm (peas), Haskell artificial neural network library (hnn), and C/C++ gesture recognition algorithm.

As mentioned earlier, the communication between peas and gesture recognition algorithm is done via operating system’s file system, which resembles fifo–queues using file system. Alternatively, Python can be directly embedded in C/C++ source code, but in the case of HyperNEAT implementation, the embedding process would be very difficult due to modular nature of the HyperNEAT implementation.

HyperNEAT algorithm instance is launched on demand from the gesture recognition algorithm and is being run alongside. Once the HyperNEAT algorithm is launched with a specific task, the evolution begins and the gesture recognition algorithm is waiting for the first genotype, generated by the HyperNEAT algorithm, to appear in the fifo–queue. Once the genotype is written in the queue, the HyperNEAT algorithm is suspended until the evaluation results are posted in another fifo–queue that is used for communicating the evaluation results between the two algorithms. Having obtained the genotype, gesture recognition algorithm evaluates it, calculates the fitness, records the fitness to the fifo–queue,
and suspends until the next genotype becomes available. At this point, HyperNEAT algorithm is awakened from suspension, it reads the fitness of the previously passed genotype and stores it for further evaluations of the population.

Hnn neural network processing is launched on demand when genotype is available and affinity matrix is ready to be processed by the phenotype. Two functions are exported from hnn — one for neural network processing that accepts a genotype and an affinity matrix together with a parameter that is used to decide whether processing will be done using feed-forward or recurrent model, and another function is training function that trains a neural network and stores weights on disk.
Figure 5.9 visually presents communication between sub-systems that is explained above in text. It is important to notice that what is regarded as phenotype for the HyperNEAT is genotype for the gesture recognition algorithm since HyperNEAT uses Compositional Pattern Producing Networks (CPPNs) that are later translated into artificial neural networks and utilised further by the gesture recognition algorithm.
recognition algorithm. Therefore, CPPN is a genotype for HyperNEAT and artificial neural network is a phenotype, while artificial neural network is a genotype for gesture recognition algorithm and phenotype is the behaviour cause by a particular genotype.

### 5.4.3.2 Detectors Evolution

Detectors are evolved using peas, the HyperNEAT algorithm implementation. Every detector is evolved by a separate instance of peas algorithm. This means that gesture recognition algorithm launches a separate instance of the peas evolutionary algorithm for every single detector, which results in 10 different instances of peas framework. It can theoretically be possible to launch a single instance of peas that would evolve all detectors at the same time, but then additional changes to the algorithm would have to be made to accommodate distinction of genotypes.

Peas framework uses task modules that specify substrate of the final phenotype (topology), number of generation, population size, and number of inputs per individual solution, number of outputs, maximum depth of the phenotype, weight range of the phenotype, probabilities of adding a new connection, probabilities of adding a new node, probabilities of weight mutations, probabilities of resetting a weight, probabilities of disabling a connection, probabilities re-enabling a connection, standard deviations for mutations, range of possible node types, and individual evaluation function. One of the example tasks from van den Berg and Whiteson (2013) had been used to transform it to a task of evolving a detector.
Table 5.1 presents parameters that are used during every detector's evolution. Substrate consists of two fully interconnected layers, where first layer (input layer) has maximum of $21 \times 21$ nodes and second layer (output layer) has maximum of $0 \times 1$ nodes. ‘P’ refers to probability of either adding a new connection, adding new node, etc. Objective of the evaluation function is to maximise the Manhattan distance between all other detector's vectors that were described in Section 5.4.1.1. k-NN corresponds to the k-nearest-neighbour, where $k=10$ (all other detectors). The problem becomes reduced to finding the maximum possible Manhattan distance between a set of arrays. Minimum allowed fitness is the minimal distance between vectors that is considered as a solution and is later selected to the next generations.

After evolution for 15 generations, phenotypes of the best individuals from every generation are stored in a separate file.
Figure 5.10 shows evolutionary process for every detector for 15 generations. The graph is hard to read due to very compact values. The important message that the graph conveys is that the evolution is unstable (fluctuation of fitness) and the fitness is not improving over generations. The improvement in fitness over generations would mean that the distance between detectors is increasing as there is a one-to-one relationship between the fitness and the distances (uniqueness) between detectors. When higher minimum allowed fitness value is selected (e.g. 0.15), the evolution pattern remains unstable and not improving.

5.4.3.3 Classifier Training

Classifier artificial neural network is trained using hnn library, the neural network processing Haskell implementation. As it can be seen from the Figure 5.9, hnn is invoked directly from the gesture recognition algorithm without any separate process execution.

After detectors genotypes are evolved and stored in separate files, they are used to detect features in training data — pre-recorded training videos that were mentioned previously in Section 5.4.1.1. Detectors process training videos and every detector outputs a real number that is collected and stored associated with a certain gesture class from that training video. For example, gesture of
class 1 is represented by a set of 10 real numbers that are obtained from detectors that processed one set of videos, while gesture of class 2 is represented by another set of 10 real numbers that are obtained from detectors that processed another set of videos.

When all training videos are processed through the detectors, a file is created that contains all detectors outputs for all training videos and corresponding gesture class. Currently, the contents of a file are directly copied into the source file that performs the artificial neural network training. In the future, the training procedure should read file contents instead to make the implementation more robust.

The hnn library uses backpropagation as network training method. The backpropagation algorithm is run on the training samples 1000 times with the learning rate of 0.8, which tells weight modification degree on output error. After 1000 iterations, the resulting weights of the network are stored in a separate file that is later used for processing the output of the detectors.

### 5.4.4 Evaluation

Evaluation is performed in empirical fashion using a PC camera. First, detectors are evolved, where fitness function results serve as the evaluation criteria. Second, classifier neural network is trained fully relying on the backpropagation algorithm. Third, using PC camera, a window–based approach is used to obtain a video sequence of certain size (equal to the size of training videos), transform video sequence into SFG and further into an affinity matrix. Finally, resulting affinity matrix is passed through the set of evolved detectors and the trained classifier that returns a real number for every gesture class. The greatest real number should tell the class to which the gesture belongs.

After short testing session it can be seen that the algorithm fails to classify presented gesture. Moreover, the algorithm fails to classify correctly the exact gesture that had been used during the training.

Figure 5.10 is an appropriate evaluation criteria for the detectors. The figure points that the evolution of detectors was not successful.
5.4.5 Discussion

Failure to classify correctly presented gesture can be due to (1) small number of detectors (2) too few generations used for evolution of detectors (Kocmánek (2015) reports hundreds of generations used for detectors with hundreds of population members) (3) due to the values in the affinity matrix, the activation function produces very little variation (between 0.5 and 0.6) (4) too little training data (40 videos). It is important to note that longer evolution will result in days performing the evolution. A single individual detector is generated and evaluated per circa 3-5 minutes.

For testing purposes, evolutionary approach has been applied to classifier training to see whether the problem lies in the backpropagation method. Using only 20 videos for training and 20 for testing, it had been proven that the evolutionary approach evolves a classifier that can classify videos that are used for testing with probability of > 90%. This means that the training of the classifier is not an issue in the misclassification.

The most probable cause for misclassification is too short evolution of detectors. The reason for short evolution during the implementation can be explained by the amount of time and processing power that is required by the evolution — to evolve 10 detectors for 15 generation requires up to 8 hours of execution of the HyperNEAT algorithm, which considerably impacts the implementation process.

Another probable cause for misclassification is the wrong or too strict fitness function. Usually, when fitness function contains too many contradicting factors, the evolution becomes slow and evolutionary algorithms require less steady evolutionary strategies. In order to find whether fitness function is wrong, it should be tested further and even applied to a similar problem of lesser magnitude.
Chapter 6

Discussion

Initially the project took direction towards using low-cost sensor and it had been discovered that, without fusion of few sensors, such approach would not be enough to capture the ever-changing environment. Next, vision-based approach showed to be more satisfactory in meeting the aims of the project. Then, reproduction of a recently proposed algorithm deemed the performance unsuitable for the real-time execution. Last, by carrying initial steps of the reproduced algorithm, a new approach had been devised that partly depended on the evolutionary algorithm. The new approach failed at recognising gestures. No extensive assessments of the new approach have been conducted due to the planned end of the project.

Even though the execution speed of the underlying clustering in prototype 0 can be achieved during the run-time, a great amount of assumptions have to be made prior the use of the sensor. This makes the use of only a single laser range scanner improper as it cannot capture the complexity of the environment by performing scans in two dimensions: width and depth. Allowing scans on multiple heights would introduce unwanted delays on slow joints adjustments. Possibly, by possessing the knowledge about the exact location of desirable region in space, the amount of joints adjustments could have been reduced to a minimum. Unfortunately, the desirable region in space would have been either assumed prior execution or detected using additional sensors.

Prototypes 1 and 2 applied a number of transformations to perceived vision-based data for both the query and the original videos to reduce them to a single
data structure that contained temporal and spatial similarity of the both. Resulting data structure had been used to determine a similarity distance between the two videos along the temporal dimension, where the shortest distance meant the greatest similarity between the two videos on that region. Unfortunately, due to a sophisticated algorithm, the processing speed became intolerable for the real-time processing despite the adequate results. Results showed that the gesture recognition had been achieved on recorded videos of variable size. The results were presented in a form of a low-cost path over the cost–distance matrix. The results carried global solution to video-matching problem, which means that the solution did not represented a matching degree of every frame in a video individually, but rather considered all the frames taken together by performing dynamic time warping.

Prototype 3 substituted complex matching of videos from prototypes 1 and 2. This drew the processing speed down with the use of a neural network that was expected to increase the speed by training the network prior gesture recognitions. The substituted processing consisted of two levels: a set of evolved detectors and a trained neural network. The task of detectors had been to capture unique features from a constructed affinity matrix that contained both spatial and temporal information of a single gesture, while a task of the trained neural network had been to classify the unique features extracted by the detectors.

Training of the second level had given over 90% successful classification results on a set of pre-recorded gestures. Evolution of detectors had been very time consuming as an instance of evolutionary algorithm had to run for every detector and every evaluation had to wait for the other detectors to be generated as the fitness function needed results from all the detectors to determine the fitness for every single one. Such slow evolutionary process had forced to reduce the amount of generations and the size of populations to a minimum.

During testing of prototype 3, the processing speed had increased considerably over that of prototypes 1 and 2. Now the processing had been done near real-time, although giving wrong classification results for a set of live and pre-recorded set of gestures.

Prototypes 0, 1 and 2 had been tested in Webots environment and via the command line interface (except prototype 0 simulation part) on both Windows and
OS X operating systems. Prototype 3 had not been tested in Webots environment nor had been executed on a Windows-based machine due to the project's deadline.

For a greater picture related to the context and motivation of the project, it is important to find appropriate means for capturing the ever-changing environment as it may not always be possible as seen with the laser head scanner trials. Once the environment is captured, the necessary information must be extracted from the captured data in order to reduce complexity of that data and to find patterns within it, thus, creating simpler abstractions. Abstractions help to create higher-order knowledge, although, require a priori knowledge of the domain and may introduce bias in modelling the world around.

It is also necessary to realise the importance of generalisation over the acquired or learned knowledge through self-organisation and adaptation as the new devised or learned knowledge may help to expand already existing abstractions of the world.

Applications of self-organised controllers can contribute to many areas, where artificial intelligence may be applied, including autonomy in sport analysis, human-machine interactions, medical diagnosis or security enforcement. Although the results of this project are doubtful at this stage of research, it is strongly believed that self-organised controllers can support indirect communication between human-being and a robot as well as between two robots.
Chapter 7

Conclusion

The project had been set out to conduct a research on indirect communication to support cooperation between two agents. Different approaches had been investigated that allowed an agent to perceive and interpret intentions of another agent with the help of gestures.

The report begins by setting a context for the project and providing motivation for the research as well as lists aims and objectives and introduces the hypothesis that the project will concentrate upon during the research. Further, concepts that influence the project are identified and researched to a great extent in literature review section, giving the overview of the field and linking key topics to this project. The review of the literature spans a wide range of sub-fields that all tie back to the overall field of artificial intelligence. A particular attention is paid to intelligent robotics and the evolutionary robotics, as the latter has the central focus in this research. Right after the literature review, professional, legal, social and ethical issues for this project are determined and acknowledged together with possible solutions to potential problems. After, requirements analysis chapter reviews previously identified objectives and presents a list of requirements together with assessment criteria. Later, possible risks are identified that can arise due the course of the project and a possible resolution plan. An execution plan at the end of the chapter shows the flow of the execution.

The main part of the report constitutes the methodology chapter that presents implemented prototypes, conducted evaluations and discussions on that part. Finally, discussion chapter summarises and discusses all the practical work.
7.1 Future Work

Very little evaluation had been done for the prototypes developed. Future research should focus on performing additional tests on the prototype 3 to determine the nature of misclassifications. In particular, it is strongly advised to focus on longer evolution of detectors as they play the most important part of the system. In the case of successful evolution of the detectors, more gestures used for training should be recorded or any open database used that can provide gestures in the form of short videos. When a complete functioning system will be developed, it should be tried on a real robot (e.g. NAO) to see whether the processing speed is degraded on the processor available on the robot as the system can be further optimised. In addition, the project can be extended from gesture recognition to sign language recognition by using the same system. Some initial work has already been done that should support detection of sign language gestures (hierarchy 3 features from prototype 2).
References


References


References


Appendix A

Prototype 0

Source code for prototype 0 can be obtained from the following address:
https://github.com/mocialov/MSc/tree/master/prototype0

File ‘worlds/soccer.wbl’ should be opened with Webots simulator. ‘nao_clustering.mov’ shows a short demo of the prototype.
Appendix B

Prototype 1 & 2

Source code for prototype 1 & 2 can be obtained from the following address:
https://github.com/mocialov/MSc/tree/master/prototype1%262

`stable_version_gesture_recognition.cpp` should be compiled using command from the first line inside the file. Not all the libraries are necessary that are listed in the compilation command, but OpenCV libraries must be installed on the PC.

Directory contains demo videos for the prototypes 1 and 2.

Source code for prototype 1 within Webots environment can be obtained from the following address:
https://github.com/mocialov/MSc/tree/master/prototype1%20-%20Webots

File `worlds/soccer.wbt` should be opened with Webots simulator.
Appendix C

Prototype 3

Source code for prototype 3 can be obtained from the following address: https://github.com/mocialov/MSc/tree/master/prototype3

`stable_version_gesture_recognition.cpp` should be compiled using command from the first line inside the file. Not all the libraries are necessary.

`peas/` folder should be merged together with the current version of the peas framework. `data` folder contains test data videos and stored reduced affinity matrices together with training data.

When compiled program is started, user is presented with four choices: (1) evolve detectors (2) pass training data through evolved detectors (3) training classifier neural network (4) testing trained classifier neural network.