Towards the Evolution of Indirect Communication for Social Robots

Boris Mocialov, Patricia A. Vargas
Robotics Lab at School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, UK
Email: bm4@hw.ac.uk, P.A.Vargas@hw.ac.uk

Micael S. Couceiro
Ingeniarius Ltd.
and University of Coimbra
Mealhada, Portugal
Email: micael@ingeniarius.pt

Abstract—This paper presents preliminary investigations on the evolution of indirect communication between two agents. 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, for instance on a RoboCup1 scenario. 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. This paper summarises studied literature on the topic and the design of a self–organised system for gesture recognition. Although preliminary results show that the proposed system requires further improvements and evaluations, further research is required to fully investigate potential extensions to the system to support real indirect communication in human–robot interaction scenarios as well as between robots.

I. INTRODUCTION

Understanding actions and behaviours without any explicit communication is essential in most human-related tasks, such as autonomous surveillance, medical diagnosis, human–machine interaction, and even sports analysis [1]. 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 [2]. 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 [3], only few works in the field of robotic soccer have been trying to converge towards a more realistic approach by modelling this sort of knowledge [4], [5]. In fact, most of the robotic football strategies and tactics available in the literature, namely the ones related to the RoboCup competition [6], are either hard-coded or present rather simplistic behavioural frameworks that are far from being inspired by the behaviour of real football players [7], [8]. The aim of this study is to bridge the gap between robotic football and real (human) football, so as to pursue the ultimate goal aimed by researchers working in the field: to develop a team of autonomous robots capable of defeating a team of human players [6]. An important step towards this direction is to reproduce the way players understand others’ actions and (re-)act accordingly.

Existing coaching systems currently require extensive calibrations prior execution to somewhat exhibit autonomy on the playground, while displaying very restricted human-like behaviour [5].

Despite the general availability of immediate communication devices on boards of autonomous robotic agents, a range of internal or external factors can compromise their reliability. Failure to establish a communication link between agents’ results in temporal communication impairment, while physical damage of a communication device usually leads to a permanent loss of communication [9].

The results of this study led to development of a novel gesture recognition system using evolutionary approach [10], [11]. The system is intended to be uploaded to a robot and updated in real time as the robot learns new gestures from a teacher. The system had been tested on a PC using standard web camera with a human teacher, sitting in front of the camera. PC-based implementation had been used at this stage for convenient testing of key functionality on video data instead of intricate application directly on board of a robotic platform. The functionality is believed to be platform–independent. It can be implemented on any robotic platform that uses either PC-based or vision-based video data. Ultimately, the system is expected to be used on a robotic platform with vision-based video data, which will pose additional challenges for the system, such as change of orientation, varying illumination, motion blur, etc. This work represents the first steps towards the creation of a truly self-organised system that applies evolution to facilitate indirect communication between agents.

This paper is organised as follows. Section II reviews the related literature on subjects of gesture recognition and feature extraction, Section III describes the developed system, its layers and their functionality, providing some implementation details, Section IV describes the backbone of the system, its sub-components and interaction between; Section V presents conducted experiment setup, while Section VI shows the results obtained from the experiment; Section VII discusses the results obtained and their implications; the project is summed in Section VIII and the future work is proposed.

1RoboCup competition http://www.robocup.org/
II. RELATED WORK

Garg, Aggarwal, and Sofat separate out two main approaches to gesture interaction: glove-based and vision-based [12]. LaViola distinguishes an additional hybrid approach that uses sensor fusion of the two approaches [13]. Despite the ease of data acquisition, devices, used in glove-based approaches, 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.

In the RoboCup environment, [14] propose a vision-based framework. 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 segmentation.

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 [15] or more straightforward approach, where sequence of frames is chained together as in [16].

‘String of feature graphs’ employs graph representation of kinematic (local) features that are further concatenated to make a sequence of feature graphs to encode an action, where every distinct feature graph corresponds to a distinct frame at a particular time. Concatenated sequence of feature graphs represents spatio-temporal features and their relations in a sequence of frames [17].

Previous studies had explored other alternatives for feature extraction. For instance, [18] uses centroid position of blobs, where each blob represents a hand, plotted on a Cartesian plane, to find relationship between corresponding blobs. Willems, Tuytelaars, and Gool propose Hessian Matrices of any changes in frames, arguing that the nature of a change is not important and should not be categorised as, for example, a body part [19]. Bregonzio, Gong, and Xiang focus on clouds of interest points, where a cloud consists of a shape, speed, and density [20]. Alternative approaches extract global features that contain noise, which is a drawback for many applications.

III. SYSTEM DESIGN

‘String of feature graphs’ (SFG) approach is partly used for gesture representation in this system [17].

FGs capture local kinematic features that contain the most relevant information needed for gesture representation. In theory, capturing kinematic features is less noisy than capturing more complex features, such as blobs or salient spots that contain a collection of features, as the amount of noise is proportionate to the amount of data collected [21]. Another advantage when using graphs as underlying data structure is that graphs can be scaled easily by adding extra nodes (features).

The overall system is further explained layer-by-layer.

A. System Design: Feature Extraction

Feature extraction corresponds to layer a) from Figure 1. The implementation utilises functionality provided by the open source OpenCV library version 2.4.10.1. The reason for choosing the library is because it provides useful image processing functions.

Prior to region of interest (ROI) detection, every frame is pre-processed in the following way:

- Background is detected for future background-foreground subtraction
- Illumination reduction by converting every frame to YUV colour space and normalising Y channel which represents the brightness of the frame.
- Edge enhancement to the foreground by applying erosion and dilation

Fig. 1. Overall view of the system
a) sequence of frames in a video stream b) SFG for the video stream c) affinity matrix for the SFG d) feature detectors neural networks e) classifier artificial neural network f) resulting classification of the video stream in a)

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 standard OpenCV Haar feature-based cascade classifier, while moving object detection is a

Fig. 2. ROI Detection (Segmentation)
a) Hierarchy 1 (body frame) b) Hierarchy 2 (body frame and limbs) c) Hierarchy 3 (body frame, limbs, and limb details)
result of background and foreground subtraction. Contour of a moving object is detected using OpenCV findContours function. A moving object is considered as a part of an overall body only if it originates from the upper body.

Potential limbs are analysed by looking at hull and convex defects to find break points (elbows) and smaller details (e.g. fingers). Therefore, the detection is hierarchical, and 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 (e.g. sign language).

2) Feature Extraction: Extracted features are the detected joint positions, encoded as nodes of a graph, and their relations are the Euclidean distances 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

B. System Design: Feature Encoding

1) SFG Representation: In layer b) from Figure 1, extracted features are encoded as FGs and implemented as an adjacent linked list. 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.

C. Affinity Matrix Calculation

In layer c) from Figure 1, SFGs are encoded as affinity matrices that hold similarity information between every frame in a single matrix. 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 [17].

\[
M(a, a) = \begin{cases} 
\tau_1 - d(t_{1}, t_{2}) & \text{if } d(t_{1}, t_{2}) \leq \tau_1 \\
0 & \text{otherwise}
\end{cases}
\]

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

D. System Design: Detectors

Instead of using spectral clustering on the affinity matrix as described in [17], this implementation, as shown in layer d) from Figure 1, utilises artificial neural networks. The rationale behind choosing this approach over clustering, described in SFGs method, is to make a system that will be able to categorise learned gestures from video stream directly.

Kocmìnek describes a novel method for digit recognition from images that employs HyperNEAT [22] algorithm for evolution of detectors together with trained neural network used as the final classifier [23]. The system, presented in this paper, uses the same approach only on the spatio-temporal data encoded as an affinity matrix.

Python-based HyperNEAT implementation, developed in [24], is used for evolution of novel detectors together with the package, developed by [25], for training the classifier network and neural network processing. HyperNEAT exploits indirect encoding to reduce the solution search space by encoding solutions as sets of various activation functions for every node in a neural network and connections between the nodes [24].

In this prototype, maximum of 10 simple detectors are evolved due to the limitations of processing power of the host 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. It is assumed that the novelty search, driven by the fitness function, is capable of producing unique detectors.

Every detector has 441 inputs and a single output without any hidden layers. Inputs are arranged in 21 × 21 square to map every single cell of the resized affinity matrix. The topology of the final neural network produced by the HyperNEAT algorithm is defined in advance and, as a result, the algorithm 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. 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.

1) Detectors Evolution: Detectors are evolved using the HyperNEAT algorithm. Every detector is evolved by a separate instance of peas algorithm using parameters, given in Table I.
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 adding a new connection, adding new node, etc. Objective of the evaluation function is to maximise the Manhattan distance between detectors.

In k-NN, the $k$-nearest neighbour, $k = 10$ (all other detectors are considered). The problem is 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.

A single vector is associated with every detector with as many items as there are gestures to be learned by the system. The vector is used to accumulate rounded output neuron values. The vector describes how many times the detector detected something in the affinity matrix.

E. System Design: Classifier

As can be seen in layer e) from Figure 1, the classifier has same amount of inputs as there are detectors in the system. In this setup, 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 library uses backpropagation [26] 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.

1) Classifier Training: Classifier is trained using library that is invoked directly from the gesture recognition algorithm without any separate process execution. The same library is used for detectors processing.

IV. SUB-SYSTEMS INTERACTION

The system consists of 3 parts and is presented in Figure 4. Communication between peas and gesture recognition algorithm is done via file system, which resembles fifo–queues.

HyperNEAT instance is launched on demand from the gesture recognition algorithm to begin the evolution, while the gesture recognition algorithm is waiting for generated neural networks (genotypes) from HyperNEAT. Once a genotype is generated, it is written in the queue and the HyperNEAT 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.

Neural network processing library (hnn) is launched on demand when genotype is available and affinity matrix is ready to be processed. Two functions are exported from hnn — one for processing genotypes and another for training a classifier using provided training data.

V. EXPERIMENT SETUP

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. 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.

Every video for a single gesture class is passed through evolved detectors and the outputs are collected and used for training the classifier. 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.

Eventually, the classifier is trained using pre-recorded video
streams of gestures. After the training of the classifier, the system is ready to classify gestures in unseen video streams.

After the classifier is trained off-line on a set of gestures, the system is used to classify gestures from PC web camera in real-time. The whole process of 1) ROI detection 2) feature extraction 3) SFG concatenation and 4) affinity matrix construction is repeated continuously for a certain amount of seconds of live video, which was chosen to be 2-5 seconds. No explicit segmentation process was in place.

VI. EXPERIMENT RESULTS

One of the objectives of the experiment was to successfully classify live gestures based on learned examples.

Figure 5 shows evolutionary process for every detector for 15 generations. It can be seen from the graph 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.

Evaluation is performed in empirical fashion using a PC camera. First, detectors are evolved, where fitness function results serve as the evaluation criteria. Second, the 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 length (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 and the exact gesture that had been used during the training correctly.

Failure to classify correctly presented gesture can be due to (i) small number of detectors (ii) too few generations used for evolution of detectors ( [23] reports hundreds of generations used for detectors with hundreds of population members) (iii) due to the values in the affinity matrix, the activation function produces very little variation (between 0.5 and 0.6) (iv) too little training data (40 videos).

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. 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.

VII. DISCUSSION

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

1) Construct affinity matrices of SFGs
2) Evolve detectors
3) Train classifier
4) Combine evolved detectors and classifier to process affinity matrices and perform gesture classification

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.

For a greater picture related to the context and motivation of the work, it is important to find appropriate means for capturing the ever-changing environment. 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. Future research should focus on performing additional tests to determine the nature of misclassifications. In particular, we aim at focusing 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.

VIII. CONCLUSION

The project had been set out to conduct a research on indirect communication to support cooperation between two agents. As a result of the study, a real-time gesture recognition system had been produced that is developed with the use of the evolutionary techniques. 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 preliminary results of this project show that further improvements and investigations are required, this work should be seen as the first step towards the creation of real self-organised systems based on evolution that can be applied to social robots and thus facilitate human-robot-interaction.

Future work lies in further testing of the system and its components on hand-written digit recognition problem. The digit recognition problem is simpler than the gesture recognition problem and the encoding step of the system will be ignored, because the affinity matrix is given by the freely available databases for digit recognition. Further on, potential extensions to the system may include additional feature extraction to accommodate the algorithm for the sign language recognition and processing.

REFERENCES


