On-line Learning of Dynamic Gestures for Human-Robot Interaction

Tobias Rehrl*, Jürgen Blume*, Alexander Bannat*, Gerhard Rigoll*, and Frank Wallhoff°

Technische Universität München, Munich, Germany*
{rehrl,blume,bannat,rigoll}@mmk.ei.tum.de
Jade University of Applied Sciences, Oldenburg, Germany°
frank.wallhoff@jade-hs.de

Abstract. In this paper we present a system, which is capable of learning on-line dynamic gestures for a gaming scenario on a mobile robotic platform. The recognition of the gestures relies on a skeleton tracking based feature extraction method as well as a Graphical Model-based classification. The robotic platform deals as a research platform for Ambient Assisted Living environments, where the interaction between humans and robots should be improved by providing a more natural and intuitive kind of communication. The realized system is a first step heading into the direction of a more pleasant human-robot interaction, since the learning of the gestures is user-dependent.

1 Introduction

European societies are affected by a dramatic demographic change taking place in the years to come, therefore, Ambient Assisted Living (AAL) [1] tries to compensate the drawbacks of the aging society by applying modern information and communication technologies (ICTs). Several programs have been established (e.g. European AAL Joint Programme [2]) to provide the usage of modern ICT solutions to elderly people helping them to stay healthy, socially connected and able to live in their familiar surroundings as long as possible. The usage of robots will help to realize these goals. For providing a natural and intuitive control and interaction with the robot, gestures are pleasant means of communication in addition to speech- and keyboard-based controlling. In general, the area of robotics covers a lot of topics ranging from navigation, manipulation, perception, reasoning, etc. We have focused on activity recognition as well as human-machine dialogs for robotic platforms.

The remainder of this paper is organized as follows: A brief overview of related work applying gesture recognition on robotic platforms is given in Section 2. In Section 3, we introduce the robotic platform. The procedure and the involved modules enabling the on-line learning of new gestures on the robotic platform is presented in Section 4. The entire gesture processing is delineated in Section 5. In Section 6 the scenario is explained, in which the user trains the robot a set of gestures to play a game. The paper ends with a short conclusion and outlook.

2 Related Work

Several different approaches can be found in which the integration of robotic platforms in household environments is a major research objective. These efforts are additionally driven by the AAL initiatives. A system is developed in [3], where both a robotic platform as well as a smart home environment provide assistance and help for elderly people, thus that they can longer stay at home on their own. In [4] the robot platform – Care-O-bot® 3 – is presented, which is applied as a platform for several different projects. The Care-O-bot® 3 platform is applied to support elderly people in their domestic environments in [5]. Via tele-operating it is possible for external people to assist the elderly person in her/his daily tasks (e.g. getting up, etc.). In [6], the so-called ARMAR-III humanoid robot is presented, which should serve in a household scenario. Two robotic platforms are used for providing either cognitive or physical support to an elderly person in [7].

The classical means of human-machine communication are constituted by keyboards and mice for input operations as well as screens and speakers for giving the user feedback from the system. In general, these means are not natural and intuitive like the control via speech or gestures. Several approaches, for instance [8, 9], apply gestures in the human-robot interaction (HRI). In [10] head gestures were applied for document browsing and dialog box confirmation. A system capable of recognizing hand gestures for HRI was presented in [11]. A particular type of hand gestures are pointing gestures, see [12] for instance.

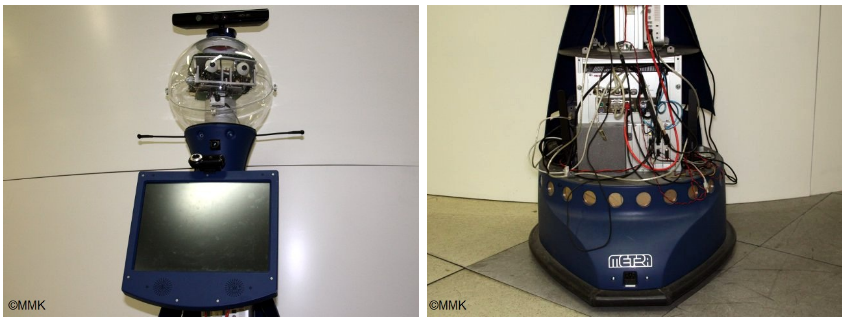
The approach presented in this paper deals with an on-line capable learning of new dynamic gestures for HRI. The robot can learn a new set of gestures, the learning procedure is coupled via the dialog system. Afterwards it is possible to control the robot either via speech commands or the trained gesture set during a game.

3 Robotic Platform

An Ambient Assisted Living surrounding deals as a scenario for the HRI. A user can interact with a robotic platform named ELIAS (Enhanced LIving ASsistant). ELIAS is able to support different kinds of services: web access via touchscreen, controlling via speech, etc. The main research goal of the ELIAS robotic platform is to make the interaction between robots and humans more natural and intuitive by utilizing different verbal and non-verbal communication channels: speech, gaze, facial expressions, head and hand gestures.

The robotic platform is a commercial one having a height of 1.55m and a weight of about 75kg. The platform consists of two different units: an interaction unit (see Fig. 1(a)) and a driving unit (see Fig. 1(b)).

The ELIAS interaction unit is equipped with a mac mini having the operating system Windows. The Windows system features the dialog system (see Section 4.2 for more details). The ELIAS interaction unit constitutes the interface for the communication with the user.



(a) ELIAS interaction unit. (b) ELIAS driving unit.

Fig. 1. The two subunits of the ELIAS robotic platform.

The ELIAS driving unit is composed of three wheels powered by electric motors featuring a differential drive. The batteries as well as the electric motors are located at the bottom of the driving unit resulting in a low center of gravity of the platform ensuring that the robot is hard to tilt. For enabling a collision-free navigation, the platform is equipped with a ring of 24 sonar sensors as well as one laser scanner heading into the front. The components required for driving (sonar senors, laser scanner, etc.) are connected to an industrial PC running a Linux operation system. In addition to the components of the driving unit, several other components (robotic head with LEDs, robotic eyes, Microsoft Kinect sensor, etc.) are connected to the industrial PC as well. On the industrial PC a shared memory data processing framework is running guaranteeing a smooth processing of the different modules.

In general, there are several different features currently available on the robotic platform: knowledge-based system controller, speech input/ output processing, head gestures recognition, user identification via face recognition applying an Eigenface approach, and many more.

4 System Overview

The system is capable of learning dynamic gestures captured via the Microsoft Kinect sensor, therefore a communication infrastructure on the robotic platform has to be set up capable of orchestrating all required processes and modules. For this reason, a shared memory data processing, the so-called Real-Time Data Base (RTDB) is applied to handle the processing of the data. The processed data can comprise either low-level data, e.g. 2D images and 3D depth information, etc., as well as high-level data, e.g. features representing semantic information (position of body parts, etc.).

For controlling the entire gaming scenario (see Section 6) a dialog system is used, which features speech recognition as well as speech synthesis and handles the human-robot interaction procedure within the game. The dialog system relies on a rule-based expert system with a bi-directional interface to the RTDB.

4.1 RTDB

In the following a deeper insight into the underlying framework providing the gesture processing with the Microsoft Kinect is presented. The original area of application of the RTDB was for operations in so-called cognitive autonomous vehicles, however, the presented framework can also be applied in our human-robot interaction scenario as well. The RTDB is capable of dealing with large amounts of data input streams, having diverse kind of features (i.e. data rate, packet losses, etc.). More details about the RTDB can be found in [13, 14].

Instead of the fact, that the name RTDB implies an underlying database, the RTDB acts more like a shared-memory access making data available for a defined period of time, however, it is also possible to record data as well. The RTDB features two important characteristics: First, several different software-modules can access data coming from one input source in parallel without blocking effects. Second, the data originating from different sources having different update rates can be processed in quasi-synchronized manner, since the internal time-stamp handling of the RTDB allows this kind of processing.

RTDB-Writers - making data available in the RTDB: The so-called RTDB-Writers are a framework around existing software modules making the data from different kind of sources accessible in the RTDB. The data can directly originate from a sensor (e.g. cameras, microphones, etc.), or it can be further processed by other software modules. In our case, we apply both writing of sensor-based as well as processed data. For the sensor-based case, we interface the output of the Microsoft Kinect with the RTDB, making the position data of the skeleton tracking available in the RTDB. For the processed data case, we write gained classification results into the RTDB, where the dialog system can access them.

RTDB-Readers - accessing data stored in the RTDB: As counterpart to the RTDB-Writers the so-called RTDB-Readers deliver data from the RTDB into different software modules. In our case, there are two different RTDB-readers. One is coupled to the classification tool, reading in the data of the skeleton tracking. The second RTDB-reader accesses the obtained classification results and hands it over the dialog system.

The kind of data accessed by writers and readers must be defined for the RTDB. This definition provides information about data type, size, etc. In general, the concept of writers and readers supports a modular software design. Thus, different software components can be exchanged and replaced by better software modules, where only the interfaces as well as the defined data types have to remain unchanged.

4.2 Dialog System

The dialog system is the core component where all different software modules come together and have to be orchestrated in a proper manner. The dialog system features interfaces to the gesture processing (see more details in Section 5), the gaming platform *akinator* [15, 16], speech processing (synthesis as well as recognition), the robotic head and the sensors on the platform. The dialog system is implemented in the programming language JAVA using JESS, which is a rule-based engine for JAVA delivering the automation of the expert system. This expert system relies on rules and facts to provide inference in different dialog situations. The facts for the inference are provided as events via the middle ware ICE (Internet Communications Engine) from the connected modules.

Different rules are defined to handle the different acts in the dialog scenario. More details about the course of the dialog within the scenario can be found in Section 6. Speech input and output are provided by a commercial solution. The speech input is based on a context dependent grammar adapted to the different acts – for more details see Section 6. The switching of the correct grammar for the recognition unit is managed by the dialog system. The recognizer as well as the text-to-speech module can be switched between English and German language.

5 Gesture Processing

In addition to the speech recognition incorporated into the dialog system, gestures can be further means of communication. In our case, we focus on upper body motion, which are tracked by applying the Microsoft Kinect sensor. The applied gestures motions of the upper body region can bring in non-verbal communication channels into the HRI. In general, there is a big variety of different gestures, which are applied in a natural manner in everyday human-human interaction. Until now it is not feasible to make these gestures available on a robotic platform applying machine learning techniques. Due to that fact, we focused on a first approach towards on-line learning of gestures, thus, the system can learn a user-dependent set of gestures. A user, who wants to interact with the robot in a non-verbal way can train the robotic platform with a certain set of gestures. This learning procedure is conducted via the dialog system. In our case, we mapped the five possible *akinator* answers $\{(Yes)$, $(Probably$, $Partially)$, $(I$ $don't$ $know)$, $(Probably$ not, Not $really)$, $(No)\}$ to five gesture classes.

5.1 Feature Extraction

For the feature extraction we rely on a commercial system. Microsoft launched in November 2010 the Kinect sensor, which was intended to be a gesture-based control device for the Microsoft XBox 360, however, two open source libraries were launched soon (OpenKinect [17], OpenNI [18]) making the Kinect available for computers. In our scenario, we applied the skeleton tracking for the upper body region which is implemented in the middle ware NITE from Prime Sense [19]. For the entire upper body tracking we used seven points, see Fig. 2. Since, we were only interested in dynamic gestures we neglected the absolute pose information (x,y,z position of the seven body points in space) and we chose one point as reference point. For the reference point – in our case the origin of the user-centered coordinate system – we chose the skeleton point located on the user's chest. Our feature vector Θ_t for time instance t now comprises six point entries, thus we have eighteen values by regarding all three dimensions:

$$\Theta_t = (\delta \underline{p}_t^1, \delta \underline{p}_t^2, \delta \underline{p}_t^3, \delta \underline{p}_t^4, \delta \underline{p}_t^5, \delta \underline{p}_t^6)^T, \tag{1}$$

with $\delta \underline{p}_t^i = \underline{p}_t^i - \underline{p}_t^0$, and $\underline{p}_t^i$ is the vector of point i at time instance t in x, y, z coordinates.

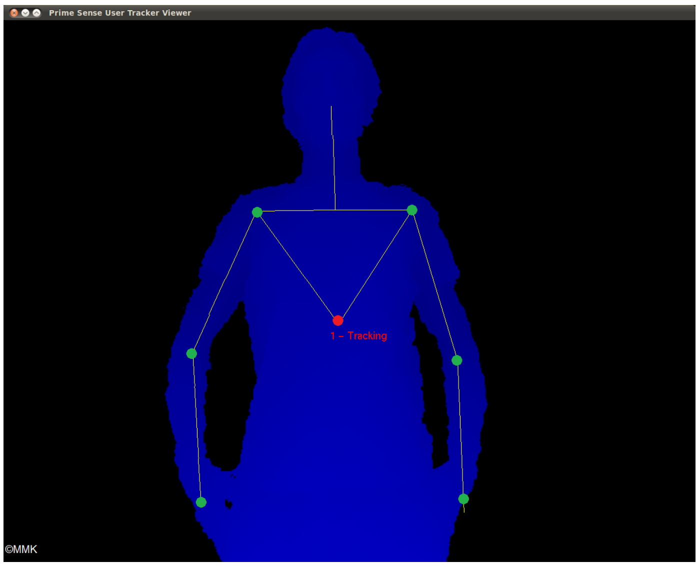


Fig. 2. In this picture the seven body points can be seen. The reference point $\underline{p}_t^0$ of the chest is indicated in red color, the other points $\underline{p}_t^i$ are green.

5.2 Classification

Several machine learning techniques are imaginable for classifying dynamic gestures. We apply Graphical Models (GMs) [20] for this task, because of their ability to capture and emulate the dynamic sequences of the gestures in a generative model. When directed GMs model time series, they are called Dynamic Bayesian Networks (DBNs). Within the DBNs the Hidden Markov Models (HMMs) are a sub-class, where observations are dependent on an unobservable variable, referred as *hidden state*. In addition, the inference process can be realized via the *Viterbi-algorithm* [21], which provides a quick finding of the most probable gesture class. The Graphical Model Toolkit [22] is applied for analyzing the upper body motions.

GM-Learning: Learning in the case of GMs can head into two different directions: first, finding the topological structure of the GM defined by a set of random variables. Second, finding the parameters of the GM describing the probabilistic interrelation between the random variables. A further important aspect in that case is, if the variables are fully observable or some of them are hidden. For our problem, we need not to find the topological form of the GM, in this case we apply the common HMM topology (see Fig. 3), however, with GMs it is imaginable to emulate the structure of the skeleton in the topological order of the random variables. For the parameter finding we apply the *Expectation Maximization Algorithm* on the obtained *Junction Tree* from the applied GM representing a HMM.

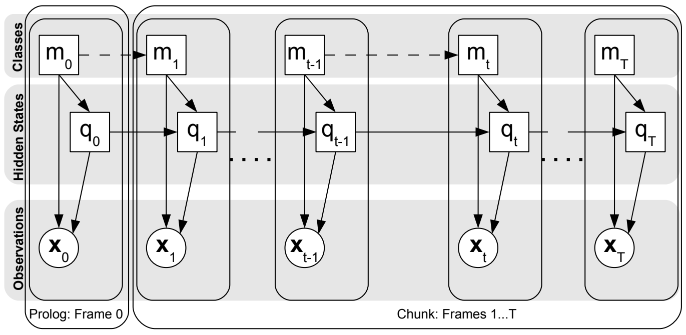


Fig. 3. The applied GM with a HMM topology. m_i models the different motion classes. q_t represents the hidden states at time instance t, whereas x_t are the observations for time instance t.

The general problem for the maximum-likelihood estimation problem is given by the fact that a certain set of parameters θ defines a density function $p(x|\theta)$. In general, there are N i.i.d. x_i in the data set $\mathcal{X} = (x_1, \ldots, x_n)$ forming the observed *incomplete data*. The *complete data* is given by having additional random variables $\mathcal{Y} = (y_1, \ldots, y_n)$, thus forming the likelihood function

$$\mathcal{L}(\theta|\mathcal{X}, \mathcal{Y}), \tag{2}$$

describing the likelihood of the parameters given the data. The likelihood is a function of θ, whereas the data $\mathcal{X}$ is observed and thus fixed and the random variables $\mathcal{Y}$ are introduced for optimization or are hidden variables in the model. For easing the computations the log is applied, thus resulting in:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log \mathcal{L}(\theta|\mathcal{X}, \mathcal{Y}), \tag{3}$$

trying to find the best parameter set θ describing the *complete data*.

GM-Inference:

For inference in GMs there are two different ways of reasoning: bottom-up reasoning and top-down reasoning. In general, a GM describes a generative model, where certain configurations of different nodes emulate a specific system. For the top-down reasoning, the interest of inference is concerned to the leave nodes of the GM trying to make a prediction of their configuration applying a certain configuration from nodes located in the path above of these leaves nodes. In contrast to that, the bottom-up reasoning, applies the observations from the leave nodes to infer the state of the nodes lying in the path above of them. The bottom-up reasoning is applied in our recognition process to find the most probable gesture class m_i.

The *Viterbi-algorithm* [21] is applied in the following way:

$$m^* = \underset{\boldsymbol{q},m}{\operatorname{argmax}}\, p(m, \boldsymbol{q}, \boldsymbol{x}|\theta), \tag{4}$$

to obtain the most probable configuration of all hidden states $\boldsymbol{q}$ for a motion class m_i in a GM defined by the parameter set θ and a given observation sequence $\boldsymbol{x}$.

6 Scenario

The application for the on-line learning of upper body motions is situated in a gaming scenario, however this scenario deals only as an exemplary application of the proposed system. The system can be easily adopted to other applications as well. The user can add a further non-verbal interaction modality to the existing speech-based solution for playing *akinator*. *Akinator* is a game, where a person thinks of another person and the system tries to resolve of whom the person was thinking of. This resolving process is done by asking questions, where five different possible answers are allowed: $\{(Yes)$, $(Probably$, $Partially)$, $(I\ don't\ know)$, $(Probably\ not$, $Not\ really)$, $(No)\}$. The gaming scenario can be subdivided into four acts: Introduction, Gesture Learning Procedure, Gaming, Finishing.

Introduction: In the Introduction act, the robotic platform recognizes the presence of a person via the installed sensors and invites the person to play a session *akinator*. While in this phase, the dialog system adapts the vocabulary of the speech recognition to recognize only statements of agreement (e.g. yes, okay, etc.) or disagreement (e.g. no, not, etc.), this step enhances the speech recognition process.

Gesture Learning Procedure: The dialog system now enables the second phase. Here the vocabulary is updated towards the gesture training phase. The dialog system instructs the user to perform the first gesture corresponding to the (Yes)-Class. The repetition counter for one gesture is currently set to ten (which has proven as a good trade-off between robustness and time efforts). The dialog system provides in addition to the speech-based command instructions also visual cues via the LEDs of the robotic head. The LED blinking encodes the following two states:

- Blinking LED ring: preparing for the next gesture.
- Running LED ring: recording a gesture.

After the first gesture class – (Yes) – was successfully recorded, the remaining four classes ($Probably$, $Partially$), (I *don't know*), ($Probably$ *not*, Not *really*), (No)} are captured in the same manner.

The recording of the gestures is followed by a cross-validation procedure, indicated to the user via speech output. Afterwards, the recognition rate of the cross-validation is compared to a threshold γ. If the recognition rate is above the threshold, the current training configuration will be used for the training procedure on the entire gesture set. Otherwise, the parameters of the GMs (i.e. number of hidden states is varied) are adapted. If a parameter correction fails twice, the user is instructed to either record a new gesture set or play the game only using the speech-based modality. A correct completion of the training of the gestures is also indicated to the user via a speech-based statement.

Gaming: In this third act, the user thinks of a person and the *akinator* system tries to guess the identity of the person. The answer can be delivered via verbal and non-verbal communication means (if gesture recognition is enabled). To the existing speech-based answers for the five classes, each class got an extended set of possible words for instance: Class (Yes) was extended by several other confirming statements like yeah, okay, etc.

A further improvement of the gaming procedure is that the system is able to repeat the current question by giving a command.

Finishing: After the *akinator* system has reached a certain level of confidence, the most probable guess is announced. The user has finally the possibility to confirm the correctness of the guessed character.

7 Conclusion and Outlook

In this paper, we presented a system which is capable of learning dynamic gestures in an on-line manner. The system works quite well and the processing time is adequate for the gaming use case. However, there can be improvements made with regard to either the classification process by applying GMs emulating the structure of the skeleton for enhancing the recognition process as well as reducing the training samples per gesture. Another possible further extension could be setting up a database featuring more than one person and recalling the gesture set by identifying the person (via face or voice).

8 Acknowledgments

This work was supported by the Ambient Assisted Living (AAL) Joint Programme and the German BMBF within the ALIAS project, the European Union within FP7 project CustomPacker and by the DFG excellence initiative research cluster Cognition for Technical Systems CoTeSys.

A very special thanks goes to the people from *akinator* for providing the interface to their web-service for the game-engine and the documentation as well.

References

1. M. C. Katrin Ganer, “Ict enabled independent living for elderly – a status-quo analysis on products and the research landscape in the field of ambient assisted living (aal) in eu-27,” Institute for Innovation and Technology, Tech. Rep., 2010.
2. “Ambient assisted living joint programme,” online http://www.aal-europe.eu/.
3. A. Badii, “Companionable - integrated cognitive assistive & domotic companion robotic systems for ability & security,” 2009.
4. F. Weisshardt, U. Reiser, C. Parlitz, and A. Verl, “Making High-Tech Service Robot Platforms Available,” *ISR/ROBOTIK 2010*, 2010.
5. “Srs – multi-role shadow robotic system for independent living,” online http://srs-project.eu/.
6. T. Asfour, K. Regenstein, P. Azad, J. Schrder, N. Vahrenkamp, and R. Dillmann, “Armar-iii: An integrated humanoid platform for sensory-motor control,” in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2006, pp. 169–175.
7. “Domeo – domestic robot for elderly assistance,” online http://www.aal-domeo.org/.
8. M. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno, “Knowledge-based person-centric human-robot interaction using facial and hand gestures,” vol. 3, oct. 2005, pp. 2121 – 2127 Vol. 3.
9. T. Hashiyama, K. Sada, M. Iwata, and S. Tano, “Controlling an entertainment robot through intuitive gestures,” vol. 3, oct. 2006, pp. 1909 –1914.
10. L.-P. Morency and T. Darrell, “Head gesture recognition in intelligent interfaces: the role of context in improving recognition,” in *Proceedings of the 11th international conference on Intelligent user interfaces*, 2006, pp. 32–38.
11. M. Hasanuzzaman, T. Zhang, V. Ampornaramveth, H. Gotoda, Y. Shirai, and H. Ueno, “Adaptive visual gesture recognition for human-robot interaction using a knowledge-based software platform,” *Robot. Auton. Syst.*, 2007.
12. K. Nickel and R. Stiefelhagen, “Visual recognition of pointing gestures for human-robot interaction.” Elsevier, 2007, pp. 1875–1884.
13. M. Goebl and G. Färber, “A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles,” *Intelligent Vehicles Symposium*, pp. 737 – 740, June 2007.
14. C. Stiller, G. Färber, and S. Kammel, “Cooperative cognitive automobiles,” in *Intelligent Vehicles Symposium, 2007 IEEE*, June 2007, pp. 215–220.
15. “Akinator.com,” online http://en.akinator.com/.
16. “Eloquence,” online http://elokence.com/.
17. “Openkinect,” online http://openkinect.org/wiki/.
18. “Openni,” online http://www.openni.org/.
19. “Prime sense,” online http://www.primesense.com.
20. M. I. Jordan, Ed., *Learning in graphical models.* Cambridge, MA, USA: MIT Press, 1999.
21. A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260 – 269, apr. 1967.
22. J. Bilmes, “GMTK: The graphical models toolkit,” 2002.