MIMIC: A Human motion imitation component for RoboComp

L. V. Calderita[1], P. Bachiller[1], Juan P. Bandera[2], P. Bustos[1], and P. Núñez[1]

[1] Robolab, Universidad de Extremadura, Escuela Politécnica, Cáceres, Spain
[2] Grupo ISIS, Dpto. Tecnología Electrónica,
Universidad de Málaga, Campus de Teatinos s/n 29071-Málaga, Spain
lvcalderita@unex.es, pilarb@unex.es, jpbandera@uma.es, pbustos@unex.es, pnuntru@unex.es

Abstract. Human motion perception and imitation have become a key topic for the robotics research community in the last years. Social robots, that are designed to work in daily life environments, should be able to detect people using natural and intuitive channels. The ability to per- ceive and imitate people is not only required for safety reasons, or to ease human-robot interaction. It also allows the robot to learn from demon- stration, one of the most powerful tools in the social learning processes. In the last decade, many learning by imitation architectures for social robots have been proposed. Most of them require the robot to perceive and imitate human motion. This paper presents one of the first steps towards the implementation of such a robotic assistant. More precisely, it describes the integration of a human arms motion imitation component (MIMIC) into a software framework specifically oriented to robotics, named RoboComp.

1 Introduction

A new generation of robots has arisen in the last years that is designed to work in daily life environments [1]. These robots should relate to people, and cooperate with them, to solve everyday tasks. They have to work in social environments, and they have to behave socially, thus they are usually named *social robots* [2]. The term, however, is usually employed at different contexts, and it is sometimes confused with the term *humanoid* robot, that was firstly used to name this new generation of robots [1]. Following previous proposals, in this paper social robots are understood as *robots that work in social environments, and that are able to perceive, interact with and learn from other individuals, being these individuals people or other social agents* [3].

In order to adapt to dynamic, unpredictable environments, social robots are equipped with many more sensing capabilities than the traditional industrial robots. Among these sensors, and despite the growing importance of audio and tactile data [2, 4], vision still represents the main sensory input for most social robots [3].

1.1 From pure visual perception to multisensory systems

Social robots, that have to perceive the same phenomena that people do, traditionally rely on biologically inspired implementations. Thus, some social robots use a pair of

foveal stereo cameras, that aims to imitate human eyes [5]. More practical solutions use a pair of narrow angle cameras to achieve precise visual perception where the attention is focused (fovea), and one or more wide angle cameras to maintain a certain level of awareness about the environment of the robot [4]. A single pair of stereo cameras can also be an interesting option, specially when hierarchical image processing algorithms are applied that allow to use a software (or firmware) implementation of foveated vision [3].

Despite their advantages and usefulness, the previous visual systems have also several drawbacks: pure foveal cameras are expensive, unusual, and have limited resolution. As pointed out by some researchers, hybrid solutions tend to accumulate all visual processes in a certain type of cameras (i.e. wide angle cameras). This transforms them into an expensive and complex version of a pair of stereo cameras, or even of a single low resolution camera. Stereo cameras themselves have to deal with calibration issues, require textured objects to obtain 3D information, and are very sensitive to noise and light changes [3].

The issues that affect stereo perception have moved some robotic researchers to look for alternative -or complementary- perceptual systems. These systems include laser range finders, TOF cameras or infrarred sensors [6]. They may not be biologically inspired, but they offer good results when put into practice. Among these sensors, the infrarred-based sensor used by Kinect represents one of the most convenient complements for vision systems, in terms of price and precision.

1.2 Perception of human motion

Perceiving human motion is not only convenient to ease human-robot interactions. If the robot is provided with the ability not only of perceiving, but also of imitating human motion, it will be able to use one of the most powerful tools in the social learning processes: Learning by Imitation (LbI). Social robots that learn by imitation can be easily taught by untrained users, as these users only need to demonstrate the task to perform using the same procedures they would use to teach another human.

In the last two decades, many different LbI systems for social robots have been proposed [2, 3]. Given the perceptual limitations of the used robotic platforms, many of these systems avoid addressing the complete problem and focus instead on certain particular tasks. Thus, some proposals rely on kynesthetic teaching, in which the robot read the positions of its motors, set to passive mode, while the human acts as a puppeteer [7]. Other systems focus on task-oriented scenarios, where the particular characteristics of the task lead perception to only certain relevant data (such as the position of a certain object) [2]. But, in order to achieve LbI systems that are able to adapt to unpredicted demonstrations, that are able to achieve generalization effectively, and that allow untrained users to interact with the robot intuitively, the social robot should be equipped with the ability to capture human motion, and to imitate it (i.e. translate perceived human motion to its own motion space) [3].These motion imitation algorithms used in social robots should consider some specific requirements. First of all, they should not impose the human performer to wear specific markers or stay at specific scenarios. On the other hand, social robots should not use algorithms that do not provide an output

at human interaction rates. Finally, detected human poses have to be translated to the robot body.

Many algorithms has been proposed to achieve these requirements. Most of them are focused on visual perception. Among these solutions, model-based approaches have gained a growing popularity [8]. However, the limitations of the visual perception systems that can be mounted in the head of a social robot make it difficult to find fast but precise enough mechanisms. Thus, a standarized method to capture human motion in social robots has not yet been adopted.

In the last year, the apparition of the Kinect sensor, and the open source libraries provided to use it, has opened a new perspective in the field of *on-line*, markerless human motion capture. These libraries only require the human to perform an initialization pose, and after it they are able to track human motion at a high frame rate, using a cheap sensor that can be mounted on the head of a robot. As before, it is important to consider the drawbacks of this solution: it requires initialization, it may produce odd results specially when the arms move close to the body, and the information about used algorithms is limited.

1.3 The inverse process. Humans that imitate robots

As commented above, social robots should be able to imitate human motion. But, in the field of assistive robotics, the inverse process may also be useful. A robot that is able to execute a movement, and evaluate how people imitate it, could help therapists to conduct rehabilitation exercises. This role of social robots as assistants for therapists may benefit from the ability the robot has to store and access on-line information about the patient such as his/her evolution during the therapy, biosignals, etc. The robot has also the possibility to adapt the exercises to different people to a level that is difficult to achieve for a human demonstrator (e.g. the robot may repeat an exercise or wait for a human response for hours).

This paper presents one of the first steps towards the implementation of such a robotic assistant. More precisely, it describes the integration of a human motion imitation component (Model-based Interactive Motion Imitation Component, or MIMIC) and other tracking components into a software framework specifically oriented to robotics, named Robocomp. MIMIC allows the robot to use an internal human model to imitate the motion of the perceived human. The set of joint angles of this model for each pose can then be compared against a set of desired values in order to evaluate the exercise. It is important to consider that imitation, in this paper, is performed in the human motion space (the virtual human model imitates the perceived human). Translations between human and robot motion spaces will have to be incorporated to MIMIC in further works. These works will follow previous contributions of the authors on this research field [3].

The proposed human motion perception component employs both a color and an infrared-based sensor (Kinect) as input. For the experiments described in this paper, different markers have been located in the wrists, elbows and chest, in order to obtain an accurate measurement of her motion. The human motion perception component relies on a novel tracking system to follow the 3D movements of these markers on-line. Then, as commented above, MIMIC uses a virtual model to translate these trajectories to a

valid human motion, using a fast, analytic inverse kinematics algorithm. The core of this model-based algorithm has already been contributed [3], although in this paper an extended version is presented.

The experiments presented in this paper compares the performance of the vision-based tracker against the motion tracker used by the OpenNI library, developed for the Kinect sensor. In order to obtain a quantitative evaluation for these experiments, the test exercise has been executed not by a human, but by a robotic torso available at the Robolab group of the Unex. The sequence of angles adopted by the arm motors during the exercises have been taken as the ground-truth.

The rest of the paper is organized as follows: Section 2 describes the color-based tracker, Section 3 gives a brief description of the employed model-based motion imitation system, emphasizing its improvements respect to previous implementations. Section 4 details the MIMIC component for RoboComp. Section 5 analyzes performed experiments and, finally, Section 6 concludes the paper.

2 RGBD human arm tracking in rehabilitation scenarios

As in other fields of application, robots can offer several key advantages, such as the possibility to perform, after establishing the correct set-up, a consistent and personalized rehabilitation treatment without getting tired. Besides, the capacity of the robot to acquire data could provide an objective quantification of the recovery of the patient. In this regard, the design of physical sensors for acquiring information of the environment has been crucial to achieve success in autonomous robotic field. The use of PrimeSense RGBD sensor (Kinect style) [9] is rising in the last months for several of these applications (e.g. navigation, mapping or human-robot interactions). This sensor allows, simultaneously, to acquire RGB and distance information of the environment, and it is cheaper than other RGBD sensors using laser technology. In fact, RGBD cameras allow the acquisition of reasonably accurate mid-resolution depth information at high data rates. In particular, PrimeSense RGBD sensor, is able to capture 640x480 registered image and depth points at 30 frames per second.

In this paper, a finite-state machine is used to estimate the 3D position of the interest points of each rehabilitation exercise (see diagram in Fig. 1). The rehabilitation exercise has been limited to the particular case of patients that suffer from injuries or diseases on their upper limbs. At this respect, three different trackers have been described in the proposed work. On one hand, elbow and wrist blocks represent two different threads associated to the elbow and wrist tracking, respectively. This visual tracking is implemented using artificial colored landmarks and a modified version of the well-known Continuosly Adaptive Mean Shift (CAMSHIFT) algorithm [10]. The vision color tracking algorithm used in this paper has been modified from [10] in order to include distance information acquired by the RGBD sensor. This distance information allows the tracker to reduce the search windows. On the contrary, the torso is tracked according to ARToolkit [11] and using a particular ART marker.

The new color tracker is composed of three concurrent modules, each one tracking an anatomical part of the body-arm system. The first one tracks an ARToolkit mark placed on a static and know position of the body, usually the chest. The second one

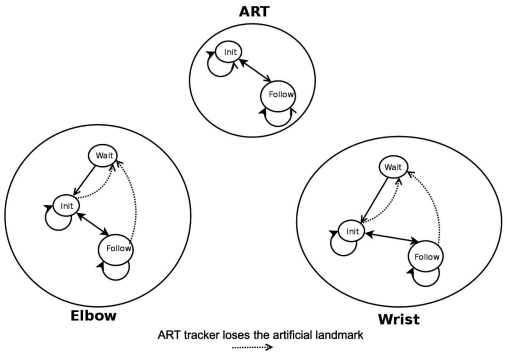

Fig. 1: State machine of the proposed human arm tracking for rehabilitation scenarios.

tracks a much smaller color mark placed on the elbow and the third one tracks a last color mark placed on the wrist. These three modules are connected with signals that transmit the hierarchical physical restrictions of the geometry of the arm. These restrictions define an starting order, and also a restarting order in case any module looses its target. They also are used to limit the search spaces during the tracking process. Fig. 1 describes the state machine for the human arm tracking. Each individual tracker (elbow and wrist) are defined by three different states. On the contrary, ART tracker is described by two states, *init* and *follow*. This ART tracker manages the final system according to the next state machine description:

1. ART tracker. First, during the initial stage, the *init* state is executed while the evaluation of the transition condition, that is, the ART marker is detected by the tracker, is false. The rest of vision color trackers are in the wait state.
2. Vision Color tracker. Once this transition condition is evaluated as true, there is a transition from the wait state to the *init* state in both vision color trackers. During this state, the color search is achieved on a distance range from the ART marker. If D_{ART} is the distance measure from the sensor to the ART marker, the color search is only achieved from D_{ART} - T_f to D_{ART} - T_b, being both T_f and T_b, two thresholds used for limiting the 3D location of the color landmarks front and behind of the ART marker. The transition condition is the detection of the color landmark.
3. Vision Color tracker. If the evaluation of this transition condition is true, the vision color tracker state changes to the *follow* state. Now, according to the distance information of the RGBD sensor, a 3D window is generated to reduce the search space. Similar to the previous state, the size of the windows is selecting according to two thresholds $D_{color} \pm T_w/2$ to $D_{color} \pm T_h/2$, being both T_w and T_h, thresholds associated to the width and height of the window, respectively. The vision color tracking algorithm is implemented according to the work described in [10].
4. If the color tracker loses the 3D position of the landmark, there is a transition to the *init* state. Equally, if the ART tracker loses the ART marker, there is a transition to its *init* state, and the color tracker changes to its respective *wait* state.

3 Model-based interactive motion imitation

While the data provided by the RGBD tracker may be used to obtain an estimation of body links from spatial distances between detected points, they are affected by perception errors, occlusions and noise. In this paper, a virtual model of a person is used to help in the perception and imitation process. Fig. 2 shows this model and the Degrees of Freedom (DOF) of each of its joints. It is scaled to match human height. The objective for this model is to imitate the pose the human is performing, providing a valid human pose for each processed frame. The imitation process is executed through the following steps:

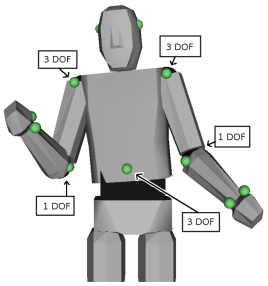

Fig. 2: Human model.

1. The model rotate the torso to match the information provided by the ARToolkit [11] marker.
2. An arm pose is computed that locates the hand of the model at the desired (perceived) position, while the model elbow is located as close as possible to the perceived elbow. This process is achieved using a fast, analytic algorithm based on inverse kinematics [3], that has been modified to consider desired elbow positions (the previous version of the algorithm used only the hand position to compute arm pose).
3. If the obtained arm pose is not valid (due to collisions or joint limit violations), alternative poses are explored to make the hand of the model moves as close as possible to the desired position.
4. Once the human model is imitating the perceived pose, its joint angles are stored as the representation of the perceived pose.

The use of a model provides several advantages over non-model based algorithms. First of all, it guarantees that the resulting pose for each frame is valid. This filters perception errors and noise (e.g. if the tracker provides an erroneous hand pose that is beyond reachable space, the model will just stretch the arm to its possibilities, avoiding odd pose data). On the other hand, a model is useful for generalization and comparison

purposes, as it provides a common space in which motion of different performers can be easily compared.

Finally, the model helps in the perceptual process. If the elbow is not perceived in certain frames (a common issue due to occlusions), the model is still able to adopt an arm pose using the perceived 3D position of the hand, and locating the elbow as close as possible to the position perceived in the previous frame. The resulting arm pose provides an estimation of the elbow position that allows the system to follow the arm motion. This estimated elbow position is also feed back to the tracker, that uses this information to update the elbow search region in the next frame.

4 RoboComp: Building The MIMIC component

RoboComp is a distributed, component-oriented, DSL-based, tool-enhanced, robotics framework in development since 2005 [16, 19, 20]. Components are processes with structured interfaces, typically built using object oriented technologies [13, 14]. Each component encapsulates a functionality and interacts with other components giving raise to more complex functions sustained by the dynamic interaction among them. On the other hand, domain specific languages (DSLs) are languages designed to express concisely and efficiently a particular class of entities [18, 17].

DSLs are used in RoboComp to specify those generic parts of the components that can be derived automatically from the framework semantics, versus those parts written by to the user to express and specific functionality. The generic part contains the logic of interprocess communication, the general structure of the components -main program, principal threads, source directory structure, documentation rules, configuration parameters- and some introspection and self-monitoring capabilities. This generic functionality is implemented with abstract classes that are inherited and extended by the user specific code to achieve the final working component. Any changes in the generic structure of the component implies a re-generation of the generic code, and no change in the code, specific or generic, ever gets to the defining DSLs. Thus, a component is divided by a tight line separating the generic from the specific. In other words, the set of DSLs used to generate a component defines unambiguously what is inside the source code up to the specific part generated by the user [18, 17].

Other DSLs are used in RoboComp to define the syntax of configuration parameters, the deployment of large sets of components with specific initial parameters, the optional state-machines that can control the inner functioning of the component and, finally, the definition of robot kinematics to be used by all components controlling a specific robot.

An special entity in RoboComp is the *interface*. Interfaces represent a piece of computing behaviour. They are definitions of the data structures and procedures that a component has to implement in order to provide that behaviour. A component may implement several interfaces, and thus, a component can provide different simultaneous views of itself. Also, an interface can be implemented by many components facilitating, for example, the integration of similar devices manufactured by different companies.

Currently, RoboComp supports two communications middlewares: Ice by ZeroC [23] and DDS (Data Distribution Service) [21] in its OpenSplice implementation. [22]. Both technologies are complementary in many senses and offer state of the art perfor-

mance and functionality. As a next step in RoboComp development, we are moving towards a communications abstraction layer that will allow the users to choose among different available implementations when designing the components.

The RoboComp repository currently has a special set of components, grouped together with the name of Hardware Abstraction Layer (HAL) that provide access to a wide variety of robotics hardware. Moreover, there are several repositories maintained by research groups and companies, providing dozens components implementing a wide range of functionalities in the fields of robotics and computer vision. To manage the creation, deploy, debugging, simulation and maintenance of these repositories of components, RoboComp counts with several useful tools that eases the development cycle of robotics software. Also RoboComp can communicate with other frameworks taking advantage of a increasing amount of coding made by hundred of researchers around the world. A deeper introduction to RoboComp can be found in [16].

The functionality for human motion imitation in RoboComp has been built using several components. Fig. 3 shows two screen captures of the RoboComp's deployment tool with the components involved. The one on the left shows the components that take part in the color tracking experiment, and the one on the right the OpenNI tracking experiment.

For the color tracking experiment, the components involved are:

- *Dynamixel (Neck)*: Belongs to the HAL group and provides access to the set of servos that move the neck of the robot Ursus.
- *Dynamixel (Left Arm)*: (HAL group) Provides access to the set of servos that move the right arm of the robot Ursus.
- *GestureComp*: Coordinates the movement of the 9 servos (neck+arm) to generate predefined arm and neck gestures.
- *KinectComp*: (HAL group) Provides access to the Kinect device providing a calibrated RGBD image. The calibration takes care of the small, but significant, translation and rotation between the infrared and the RGB camera.
- *ArmTrackerComp*: Color tracking component holding the algorithm presented in Section 2.
- *MimicComp*: Human torso modelling component activated by the 3D positions tracked by the ArmTracker component. It integrates the algorithm presented in Section 3, and provides concurrently the current kinematic configuration of the model to other components.

For the OpenNI tracking experiment, the single different component is:

- *OpenNITrackerComp*: Tracking component using the OpenNI library.

The components written for this work are *ArmTrackerComp*, *OpenNITrackerComp* and *MimicComp*. The others belong to the RoboComp repository.

ArmTrackerComp integrates the color tracker algorithm described in 2. The implementation uses the state machine engine provided by Qt taking advantage of its hierarchical and concurrent capabilities.

OpenNITrackerComp integrates the OpenNI library in the RoboComp system making it usable by the other components in the repository.

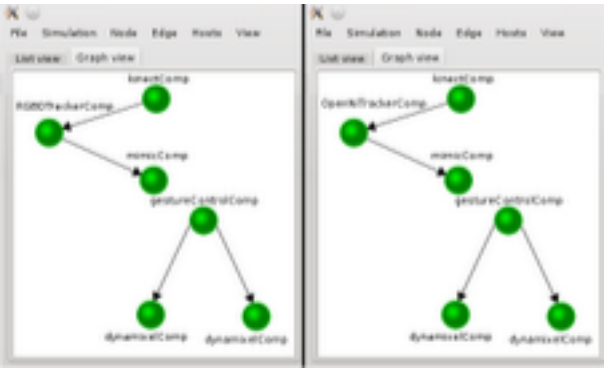

Fig. 3: Graph of processes

MimicComp integrates the algorithms developed in Section 3, that combines the open source 3D graphics engine OSG[3] with an analytic inverse kimenatics algorithm, to provide a realistic model of a human robot torso. We have redefined the application interface of the program, splitting it into two elements: a set of configuration parameters and a public RoboComp interface. The configuration parameters provides the component with a parametrized start, and the interface provides a way for others components to communicate with MIMIC.

5 Experimental results

This Section details the experiments performed to obtain a quantitative evaluation of the human motion imitation system. In this experiments the ground truth is provided by the robot Ursus, designed and built in Robolab-UEx. It consists of a torso placed over a mobile platform. The torso has two 5 DOFs arms, a 4 DOFs neck and a head with an articulated mouth. The motors driving the joints are Dynamixel servos summing a total of 15 units, of different torques. The robot is placed in front of a Kinect sensor at a distance between 1 and 6 meters inside our lab. All the processing is done on a i7 computer running at 2.8 GHz.

For this experiments we have selected an exercise consisting on the repetitive flexion and extension of the elbow. This exercise, being simple, is habitually used in rehabilitation therapy of upper limbs.

The setup for RGBD tracker, includes a ARToolKit marker placed on the chest of the Ursus robot. This mark provides the perceived reference system of the patient - the robot Ursus in the experiment. There are other two colored marks situated on the elbow, and on the wrist respectively. The RGBD tracker estimates the 3D positions of the marks placed on the joints for each acquired frame. The acquisition rate is 20 fps. The tracker sends this information to the MimicComp so it can imitate the perceived movement of the robot. At the same time, the MIMICComp sends the position of all

[3] www.openscenegraph.org

the joints to the RGDBTracker, that uses them to recover from a lost state caused by an occluded joint -typically the elbow when the arm is completely extended.

While the real application requires a person to perform the motion, a ground-truth is required for a quantitative evaluation. In the performed tests the ground-truth is extracted from the joint motor angles of the Ursus robot. As these angles and the perceived ones are not read in the same time stamps, the later are linearly interpolated in the time stamps in which motor joint angles are read.

Two different tests were performed. In one of them, the motion is perceived using the detailed RGBD tracker, reinforced with the MIMIC model-based imitation system. The other test uses the motion tracker provided by the OpenNI library. It is important to consider that this tracker requires the user to perform an initialization pose, thus additional motion has to be programmed for the robot.

Before comparing the results of both tests, the ground-truth values (i.e. motor joint angles) have been compared to guarantee that both motions are similar enough. Table 1 show the results of this comparison. Maximum differences are under 3.5 degrees, and mean differences are below 0.1 degrees, thus both motions can be considered similar enough.

Table 1: Differences between elbow motor angles.

Maximum difference (rad)	<0.06
Mean difference (rad)	0.0015
Standard deviation (rad)	0.0145

Fig. 4 shows the perceived elbow angle and the ground-truth, when the motion is perceived using RGBD tracker plus MIMIC. It can be seen that the perceived motion follows the real one, but there are two main error sources that affect perception:

- The model arm does not bend to its maximum value. This error is produced by the ARToolkit[11] marker, that provides an inaccurate orientation for the torso. This incorrect torso pose affects the reachable space of the arm, and modifies the elbow angles that have to be adopted to reach certain poses.
- There are certain pose outliers that are caused by tracking errors, that move the arm to an erroneous pose within its reachable space.

Fig. 5 and Table 2 show the errors obtained when the elbow motion is perceived using RGBD tracker reinforced with the MIMIC system. The effects of the previously commented error sources are clearly visible. The effects of the torso errors are difficult to reduce without considering different torso pose estimators. On the other hand, as Fig. 5 shows, errors produced by the tracker use to appear as outliers, that could be filtered using RANSAC or other method. The model could also help in filtering these outliers, by imposing a maximum joint angle velocity for the model joints.

As commented above, the tests have involved two executions of the same exercise. In the second one, the motion was captured using the OpenNI library. Fig. 6 show the

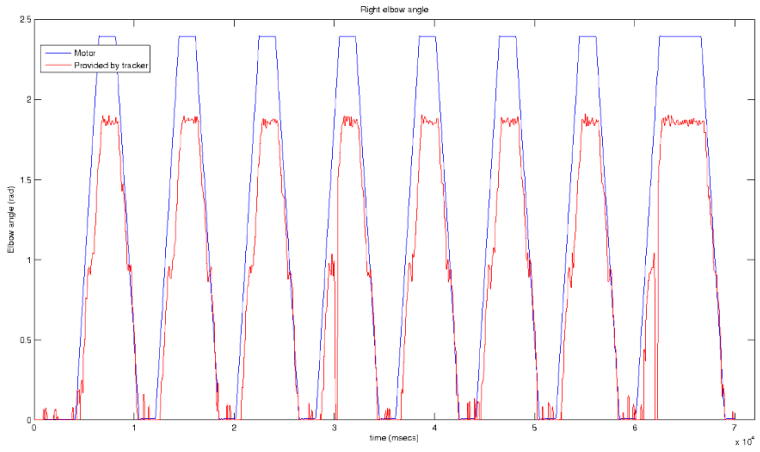

Fig. 4: Elbow angles obtained by MIMIC, using RGBD proposed tracker.

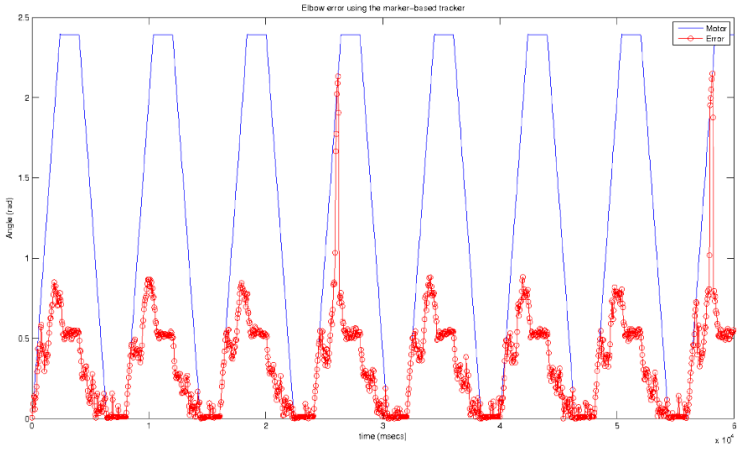

Fig. 5: Tracking errors for RGBD tracker + MIMIC.

results of this test. It can be seen that the exercise requires some prior initialization movements, but after the OpenNI captures the initial pose, it is able to successfully track the motion. While maximum and minimum elbow angles are not reached, the results seem more accurate than with the previous solution.

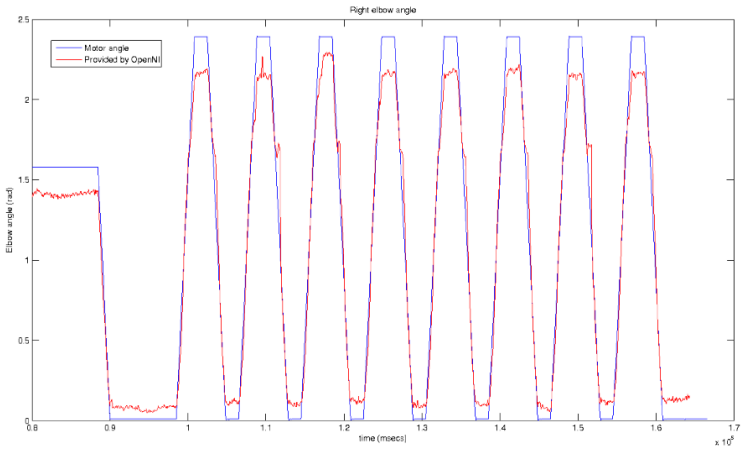

Fig. 6: Elbow angles obtained by OpenNI human motion tracker.

Fig. 7 and Table 2 confirm that OpenNI tracker, that also benefits from a higher frame rate, is more accurate than vision-based RGBD tracker, even although the later is reinforced using MIMIC. The main drawback for OpenNI tracker is the requirement of an initialization phase. It also uses a too unconstrained model, in which link lengths may change dynamically, collisions are not considered, and certain poses may lead to arbitrary changes in the perceived pose (e.g. when the arm is stretched close to the body).

Table 2: Tracking errors.

	RGBD + MIMIC	**OpenNI tracker**
Mean error (rad)	0.3451	0.1261
Standard deviation (rad)	0.2976	0.0880

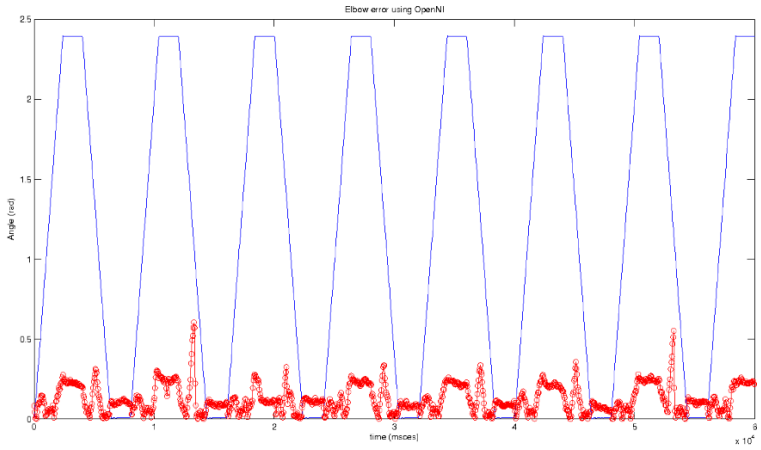

Fig. 7: Tracking errors for OpenNI.

6 Conclusions and Future Work

The main conclusion of this paper is that OpenNI tracker overcomes the RGBD tracker, even when this last one is reinforced using a human model (MIMIC) to help in the perception process. OpenNI has also the advantage that it does not require color patches nor other marks to track human motion. On the other hand, its main disadvantage is the necessity of performing (and maintaining for some seconds) an initialization pose. This drawback is specially critical in rehabilitation scenarios, in which the performers may have different issues regarding the mobility of their limbs. However, OpenNI is in the process of including additional functionalities and capabilities. Personalized initialization poses will most probably be added in next versions of the library.

During the experiments, it could be observed that the OpenNI tracker does not consider collisions between body segments (it does not use models for body segments at all), and modifies dynamically the lengths of these segments. Occlusions of body parts, on the other hand, make the tracker discard the complete limb for the affected frame. The effects of all these issues could be reduced if a model would be used to reinforce the tracking process. Thus, further work will focus on combine the OpenNI tracker with the MIMIC model (i.e. the RGBD tracker is substituted by the OpenNI tracker). Promising prior results have been already obtained. In the short term this integration process will be finished, in the hope that MIMIC becomes a key component to interact with real robots.

Acknowledgements

This work has been partially supported by grants PRI09A037 and PDT09A044, from the Ministry of Economy, Trade and Innovation of the Extremaduran Government, and

by grants TSI-020301-2009-27 and IPT-430000-2010-2, from the Spanish Government and the FEDER funds.

This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación (MICINN) and FEDER funds project n^o. AIB2010PT-00149, TIN2008-06196 and IPT-430000-2010-002, by the Spanish Ministerio de Industria, Turismo y Comercio project n^o TSI-020301-2009-27, by the Junta de Andalucía project n^o. P07-TIC-03106 and by the Junta de Extremadura project n^o. IB10062. We would like to thank Dr. Axel Pinz for providing us different datasets of stereo image sequences [12]. These datasets have been widely used to test the approach until its inclusion in the Human Motion Capture (HMC) framework.

References

1. Inoue, H., Tachi, S., Makamura, Y., Hirai, K., Ohyu, N. Hirai, S., Tanie, K., Yokoi, K. and Hirukawa, H.: Overview of humanoid robotics project of meti. 32nd International Symposium on Robotics. 1478 – 1482 (2001)
2. Breazeal, C.: Towards sociable robots. Robotics and Autonomous Systems. 42 (3-4), 167 – 175 (2003)
3. Bandera, J.P.: Vision-based gesture recognition in a robot learning by imitation framework. PhD Thesis, Dpto. Tecnología Electrónica, University of Málaga (2010)
4. Asfour, T., Regenstein, K., Azad, P., Schr de, J., and Dillmann, R.: Armar-iii: A humanoid platform for perception-action integration. 2nd International Workshop on Human-Centered Robotic Systems (2006)
5. Metta, G., Panerai, F., Manzotti, R., and Sandini, G.: Babybot: an artificial developing robotic agent. Sixth International Conference on the Simulation of Adaptive Behaviors. 42 – 53 (2000)
6. Stanford, S.: PR2 Hardware Modifications and Add-ons (2011) http://www.ros.org/wiki/Robots/PR2/HardwareMods
7. Calinon, S.: Continuous extraction of task constraints in a robot programming by demonstration framework. PhD Thesis, École Polytechnique Fédérale de Lausanne (2007)
8. Agarwal, A. and Triggs, B.: Recovering 3d human pose from monocular images. IEEE Transactions on Pattern Analysis and Machine Intelligence. 28 (1), 44 – 58 (2006)
9. Microsoft Kinect, http://www.xbox.com/en-us/kinect
10. Bradski, G.R.: Computer vision face tracking as a component of a perceptual user interface. Workshop on Applications of Computer Vision. Princeton, NJ. 214 – 219 (1998)
11. Kato H., Billinghurst M., Blanding B., May R.: ARToolKit, Technical Report, Hiroshima City University (1999)
12. P*ö*tsch, K., Pinz, A.: 3D geometric shape modeling by '3D contour cloud' reconstruction from stereo videos. Computer Vision Winter Workshop (2011)
13. Brugali, D., and Scandurra, P.: Component-based Robotic Engineering. Part I: Reusable building blocks. IEEE Robotics and Automation Magazine (2009)
14. Brugali, D., and Shakhimardanov, A.: Component-based Robotic Engineering. Part II: Models and systems. IEEE Robotics and Automation Magazine (2010)
15. Alonso, D., Vicente-Chicote, C.,Ortiz, F., Pastor, J. and Álvarez, B.: V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development. Journal of Software Engineering for Robotics. 3 – 17 (2010)
16. Manso, L.J., Bachiller, P., Bustos, P., Núñez, P., Cintas R. and Calderita, L.: RoboComp: a Tool-based Robotics Framework. Proc. of Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots. 251 – 262 (2010)

17. Schlegel, C., Steck, A., Brugali, D. and Knoll, A.: Design Abstraction and Proccesses in Robotics: From Code-Driven to Model-Driven Engineering. 2nd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots. (2010)
18. Brown, A.W.: Model Driven Arquitecture: Principles and practice. Sofware and systems modeling. 314 – 327 (2004)
19. Martínez, J., Romero-Garcés, A., Manso, L.J. and Bustos, P.: Improving a Robotics Framework with Real-Time and High-Performance Features. 2nd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots. (2010)
20. http://robocomp.sourceforge.net
21. Object Management Group: Data Distribution Service for Real-time Systems(DDS), version 1.2 (2007)
22. http://www.prismtech.com/opensplice
23. http://www.zeroc.com