BendIT – An Interactive Game with two Robots

Tim Niemueller, Stefan Schiffer, Albert Helligrath,
Safoura Rezapour Lakani, and Gerhard Lakemeyer

Knowledge-based Systems Group
RWTH Aachen University, Aachen, Germany
(niemueller,schiffer,gerhard)@kbsg.rwth-aachen.de

Abstract. In this paper we report on an interactive game with two robots and review its components. A human user uses his torso movements to steer a Robotino robot along a pre-defined course. Our domestic service robot Caesar acts as a referee and autonomously follows the Robotino and makes sure that it stays within a corridor along the path. If the user manages to keep the Robotino within the corridor for the whole path he wins. The game can be used, for example, to engage people in physical training such as a rehabilitation after an injury. It was designed and implemented as a student project in winter term 2011/2012.

1 Introduction

Humans and robots interact in various ways, often including speech or gestures. In this paper, we present an interactive game that two robots and a human play with each other. The human can control the movements of a simple robot by his torso movement, virtually acting like a joystick, to steer it along a virtual path by bending and turning his own upper body. A second more powerful robot acts as a referee, employing methods for self-localization, navigation, and 3D perception to overview the game and to judge on the human's performance in the game. The two robots involved in this demo are the Festo Robotino[1] as the robot controlled by the human, and our custom built domestic service robot Caesar as the referee. The game setup with the involved robots is shown in Figure 1.

To realize such a task several challenges need to be addressed. First, a system must be in place to allow the referee robot to detect the Robotino to judge on the state of the game. Here, we employ well-known methods from the Point Cloud Library (PCL [6]) as described in Section 2. Then, to control the movements of the Robotino by a human bending and turning his upper body, our system uses an RGB-D camera mounted beside the playing field. It does so employing a custom body posture estimation approach described in Section 3. Finally, new and existing components must be integrated with the robot base systems like self-localization and navigation, and enriched with user interaction and the behavior to facilitate the game as described in Section 4. We conclude in Section 5.

[1] http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino/

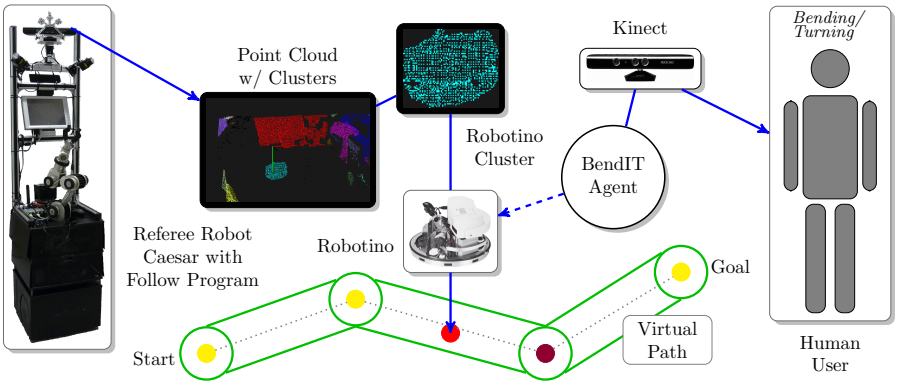

Fig. 1. The game setup and its components

2 Robotino Detection

To detect the Robotino we use the point cloud generated from the RGB-D camera mounted on a pan-tilt unit on the head of the referee robot (cf. Figure 1) combining existing methods and implementations. The overall approach is separated into two phases. First, a model of the Robotino is learned and later this very model is used to recognize the Robotino in a set of candidate point clusters. For both phases point clouds are acquired. First, the point cloud is down-sampled using a voxel grid, meaning that for each volumetric unit in a 3D grid an averaged point is chosen. Afterwards, all planar areas in the point cloud are determined and removed. The remaining points are segmented into clusters.

To create the model a-priori, the cluster known to represent the Robotino is manually selected and a model is generated using Viewpoint Feature Histograms (VFH [7]). The cluster is segmented into patches. In each patch the relative pan, tilt, yaw angles, and the distance between the central point translated to the central viewing direction are calculated, as well as the angle between the central viewing point and the normal of each point. Binning these values produces the desired feature histograms. During the on-line recognition, for each of the clusters, the VFH signatures are determined and compared to the model's feature histograms. The closest matching cluster is chosen as the target Robotino.

The computing power requirements are moderate. Interestingly, one of the most expensive parts is the down-sampling of the point cloud after acquisition. It works by taking the average for a volumetric grid with a edge length of 2 cm. The learned VFH models depend on this parameter, so adjusting and tuning it requires re-learning of the models each time. With this limitation we can currently operate at about 5 Hz.

3 Human Body Posture Estimation

Fast and on-line estimation of the human body posture is done by combining well-known methods with a custom, simple but effective approach. The posture

recognition roughly works as follows. During the game, point clouds are acquired continuously. For each cycle, the point cloud is down-sampled using a voxel grid and the points are vertically constrained depending on the room height to cut off ground and ceiling planes. The algorithm works with small data-sets and can be implemented with readily available Open Source libraries as opposed to [8].

In a first step, we segment clusters in the remaining points. For every cluster we verify if it represents a human using a VFH model of a human that we trained with data collected from different humans. If the human bends towards the robot, this may fail, because then head and torso are segmented as two separate clusters. In such a situation, we look for small clusters close to a larger one and merge them and check again if the VFH model applies now. Then, on the human cluster we start with calibrating a neutral posture of the human by finding the hips. The movement indicators are then computed as deviations of the body posture, for one from the perpendicular (bending vector) and from the angle facing the camera (turning angle). The human is tracked until it is lost for a certain period of time, after which we start the detection on all clusters again.

The first step of the *calibration* is to identify the height of the hips. Therefore, we start looking for the head in the human cluster as the smallest subset at the vertically highest point of the cluster for which we can match a sphere using RANSAC [2]. Then, starting from the centroid of the human cluster, we move down in slices parallel to the ground plane, clustering the points in each of the slices. In those slices we are looking for the cluster with the biggest width. In each slice, the biggest cluster belongs to the torso. We assume the biggest cluster among all slices to be the hips.

After calibration has been completed we switch to a *tracking* mode re-using the estimated data to detect the human more quickly. If multiple human are in the image, we take the cluster within a certain radius around the previous sighting. During tracking, we take the slice at the same height as was found for the hips during calibration. Starting from this slice, we move upwards detecting the shoulders. We first remove the points belonging to the head from the human cluster. Then, we move upwards from the hips centroid of the human-cluster in slices again, looking for the topmost points of the body, the leftmost and rightmost points being the left and the right shoulder, respectively. We compute movement indicators from the posture of the upper body as two vectors as follows. First, the deviation of the vector between the centroid of the hips and the centroid of the shoulders to an upright vector is interpreted as a movement command in the xy plane. Second, we interpret the deviation of the vector between the two shoulders and a horizontal vector in the image plane of the RGB-D camera for turning commands.

There are other approaches to body posture estimation such as the one presented in [1]. However, with some assumptions that follow from the design of the game and the corresponding restrictions in the space of possible postures our method only needs the simple steps sketched above to yield sufficiently accurate results in very little time. We have implemented the described cylinder fitting for arms, but did not need it for the given game.

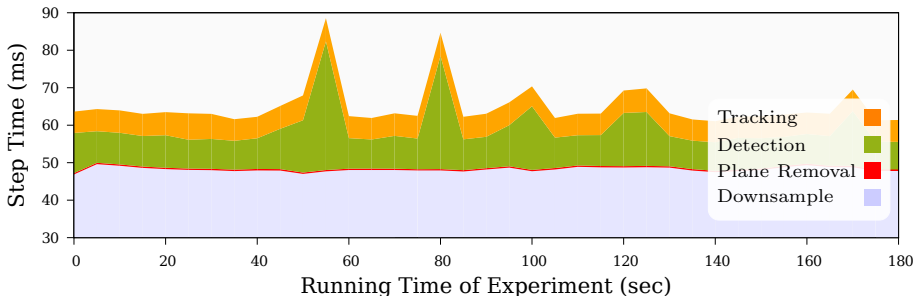

Fig. 2. Processing time of steps of the approach in a game

With the described approach we are able to process about 15 point clouds per second on an Intel E6750 at 2.66 GHz. In Figure 2 you see the performance plotted for a game of three minutes. A considerable amount of time is used for the down-sampling of the point cloud. The plane removal step is very quick and helps to filter out large parts of the point clouds. The detection step takes about 10 ms. But this step can become more costly. For example, the merging steps if torso and head are separate clusters involves running the VFH step more often. Also, if the user is lost we need to restart the more expensive detection on all clusters. The tracking takes about constant time.

4 An Interactive Human-Robot Game

The described perception modules are integrated using the Fawkes robot software framework [3]. For the visualization of the game, we use ROS' rviz [5]. The actual game logic was implemented using our Lua-based Behavior Engine [4].

The human initiates a new game for example by instructing the referee robot using speech. The stationary Kinect is calibrated to detect the human. Afterwards, the forward and sideward bending angles of the human's upper body are converted into holonomic forward and sideward motions. Twisting the torso and hence rotating the shoulders adds a rotation to that movement. The referee robot uses one out of a set of pre-defined paths ("levels"), which differ in length and deviation tolerance and thus in difficulty. It waits until the Robotino has reached the starting position, at which point the referee announces that the game starts.

The output of the human posture recognition is transformed into the exact same data that a 3-axis joystick would produce. Using this data, the human steers the Robotino through the environment. In the visualization, he can observe the referee robot's perception of the situation, including the positions of the referee and the Robotino, and the corridor along which the Robotino is to move. The referee robot will autonomously follow the Robotino at a certain distance to assert that it remains visible to the referee, even for extended games. If the Robotino stays outside of the allowed path margin for a certain amount of time (in the order of a few seconds), the referee declares that the human lost the game. If, however, the human manages to steer the Robotino towards the end

of the path within the allowed tolerance, he wins the game. The referee robot's checking of the Robotino following the path uses a global localization [9] for its own position and the relative position at which it perceives the Robotino.

The overall performance is good enough to introduce new users to the game within a few minutes. At the moment the game is explained by a human. A further step would be to have the robot instruct the player on what to do.

5 Conclusion

We presented an interactive game that a human can play with two robots. The human's objective is to steer the smaller of the two robots along a virtual path to a goal position using his upper body movements as a "joystick"-like control. The second robot acts as a referee overseeing the performance of the human player. We briefly reviewed the components needed to implement such a game ranging from perception of the human and the robot to translating the body movements to control commands. We built on existing methods where possible and implemented a novel approach to human body posture recognition using the Kinect RGB-D camera. We think that such a game could potentially be applied in rehabilitation or, in general, as a motivation to engage in physical training.

References

1. Droeschel, D., Behnke, S.: 3D Body Pose Estimation Using an Adaptive Person Model for Articulated ICP. In: Jeschke, S., Liu, H., Schilberg, D. (eds.) Intelligent Robotics and Applications, LNCS, vol. 7102, pp. 157–167. Springer (2011)
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (Jun 1981)
3. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes. In: Int'l Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR) (2010)
4. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009 (2009)
5. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
6. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE Int'l Conf. on Robotics and Automation (ICRA). Shanghai, China (May 9-13 2011)
7. Rusu, R., Bradski, G., Thibaux, R., Hsu, J.: Fast 3D recognition and pose using the viewpoint feature histogram. In: IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS). pp. 2155–2162. IEEE (2010)
8. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (june 2011)
9. Strack, A., Ferrein, A., Lakemeyer, G.: Laser-Based Localization with Sparse Landmarks. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005: Robot Soccer World Cup IX. LNCS, vol. 4020, pp. 569–576. Springer (2006)