Natural User Interface for Roombots

Ayberk Özgür, Stéphane Bonardi, Massimo Vespignani, Rico Möckel, Auke J. Ijspeert
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
{ayberk.ozgur, stephane.bonardi, massimo.vespignani, rico.moeckel, auke.ijspeert}@epfl.ch

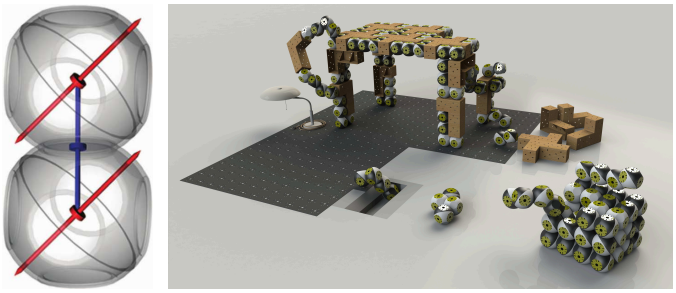
***Abstract*—Roombots (RB) are self-reconfigurable modular robots designed to study robotic reconfiguration on a structured grid and adaptive locomotion off grid. One of the main goals of this platform is to create adaptive furniture inside living spaces such as homes or offices. To ease the control of RB modules in these environments, we propose a novel and more natural way of interaction with the RB modules on a RB grid, called the Natural Roombots User Interface. In our method, the user commands the RB modules using pointing gestures. The user's body is tracked using multiple Kinects. The user is also given real-time visual feedback of their physical actions and the state of the system via LED illumination electronics installed on both RB modules and the grid. We demonstrate how our interface can be used to efficiently control RB modules on simple point-to-point grid locomotion and conclude by discussing future extensions.**

I. Introduction & Motivation

Roombots (RB) are a self-reconfigurable modular robotic platform designed to study both robotic reconfiguration and adaptive locomotion [16]. Each RB module is composed of two connected "sphere-like" structures. It has 3 rotational Degrees of Freedom (DOF) as seen in Fig. 1a, capable of continuous rotation. Up to 10 Active Connection Mechanisms (ACMs) can be installed on each RB module, letting it connect to structured grid tiles and to other modules [17]. The structured grid is designed such that one sphere-like structure of a module fits exactly on a grid tile. A single module can move anywhere on this grid (except on convex edges) in a structured manner by moving one tile at a time, latching onto the next tile before releasing the previous one using ACMs. Additionally, free locomotion and motion on a structured grid using multiple connected modules are also possible.

One of the main goals of the platform is to create adaptive furniture, where modules move both inside and outside of the structured grid environment, as depicted in Fig. 1b. We envision that these pieces of furniture will adapt to the user's needs in order to create an assistive living space, e.g. transforming into different furnitures or changing arrangement according to different hours in the day, or to the user's choice. However, controlling these modular robots can be challenging, especially for a non-expert user.

Up to now, RB modules were mainly controlled by (1) sending hardcoded motion command sequences, (2) using a classical Graphical User Interface (GUI), or (3) through an augmented display on a mobile device [7]. The first method is restricted to developers only. The second method requires the user to focus on a computer, most often using a monitor and mouse/keyboard pair, to command the RB modules that are spatially located elsewhere. The last method requires the user to carry an additional device. To construct a user interface that enables easier control of these robots coexisting with humans, we chose to consider a more natural interface paradigm. An instance of such an interface would be where the user controls the robots using physical gestures and receives sensory feedback in the very same space that the robots are located without carrying additional devices.



(a) (b)

Fig. 1. (a) Degrees of freedom on a Roombots module. (b) Roombots modules moving inside and outside the grid to form adaptive furniture. Modules locomote from outside of the grid (gray area on the floor) to inside to form a table using passive structural elements (in brown).

A study by Hornecker and Buur examines the human interaction with augmented environments [10]. They emphasize significant concepts such as the use of the user's body as an input device exploiting the richness of bodily movement and physical objects embodying aspects of the system state so that it is legible for users. We are strongly motivated to investigate whether these concepts would ease the control of RB modules.

Lichtenstern et al. studied a control interface where a subset of multiple quadrotors can be selected by pointing gestures and moved in 3D space via the user's hand position, detected by a depth sensor [13]. Pourmehr et al. applied pointing gestures similarly to select one of multiple wheeled robots [15]. Couture-Beil et al. implemented an interface where the user selects and commands one of multiple wheeled robots by face engagement and hand gestures detected by an RGB camera [9]. Studies of Jojic et al. [11] and Kortenkamp et al. [12] feature pointing gestures and discuss methods for their recognition. These studies contain useful methods for our application such as agent selection in a multi-agent setting and tracking the user's gestures with depth sensors.

The idea of physical embodiment of the system state and providing sensory feedback to the user is applied by Bellmore

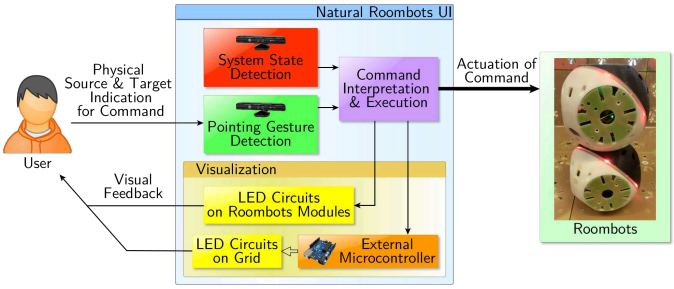


Fig. 2. Overview of our user interface, denoting key components. Pointing gestures of the user and the grid state are recognized by depth sensors (in green and in red respectively). Visual feedback using LEDs is given to the pointing gestures and to indicate the system state (in yellow and in orange). Finally, the module selected by pointing gestures is moved to its target tile, also selected by pointing gestures (in purple).

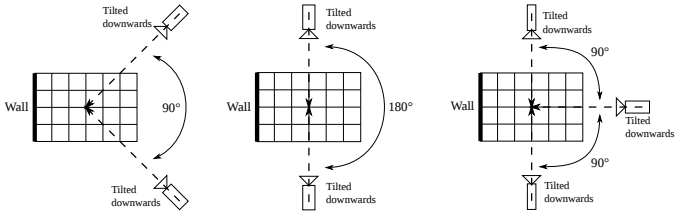


Fig. 3. Initial unsuccessful Kinect setups (top-down view).

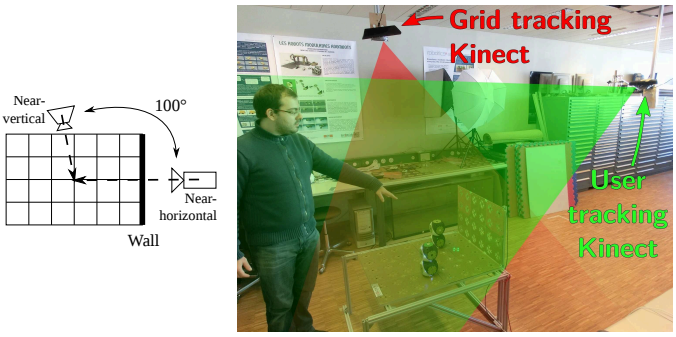


Fig. 4. The final dual Kinect setup. The overhead Kinect (approximate field of vision in red) is used mainly to detect the grid state while the horizontal Kinect (approximate field of vision in green) is used to track the user skeleton.

et al. [4] in their interactive display where the user is informed of the results of their actions by displaying animations on the target area of the action. A more directly related example is the study made by McLurkin et al. [14] where individual swarm robots are equipped with LEDs to inform the user of their state. These methods are good candidates for giving feedback to the user and enhancing system usability in our application.

Motivated by the existing natural control interface studies, we implemented and studied such an interface in order to enable easier and more natural control of RB modules. We limited our study to the structured grid to exploit the ability of a single RB module to move to any position on it. The user is able to select individual RB modules and move them to target locations by pointing at the modules and at the targets. The pointing gestures and the grid state are recognized by a dual depth sensor setup. The user's pointing gestures cause the emission of visual feedback on LED setups on both the grid tiles and the RB modules, indicating where the user is pointing and which object is selected. In addition to this, the system state is also exhibited on the same feedback setup, further enhancing the user interface experience. This paper is organized as follows: Hardware and software design of all components are described in detail in Section II, experiments and results are examined in Section III and our outlook on the study along with future extension plans is given in Section IV.

II. System Design

Our interface, whose structure is summarized in Fig. 2, is composed of a depth sensor setup to perceive the grid state and the user's body, a visualization setup to provide the user with feedback of their actions and the grid state, and finally the motion controller to move the RB modules simultaneously. They will be explained in sections II-A, II-B and II-C respectively.

A. Depth Sensor Setup

The goal of our depth sensor setup is to capture the user's body and the grid state as noiselessly and as completely as possible. Microsoft Kinect for Xbox was selected as the depth sensor for its low price and the abundance of compatible open source software. The problems associated with the setup and their solutions are discussed in the following sections.

1) Number of Sensors & Placement: The number of sensors, their placements and their orientations were first determined considering 3 main factors: The user skeleton tracking (discussed in section II-A3) quality, the depth sensor interference and the area coverage/overlap.

A number of different setups were considered and tested in which the sensors were placed with as large angles between as possible. This tends to reduce the interference, as remarked by Susanto et al. [18] and analyzed in detail by Berger et al. [5]. The initial unsatisfactory configurations can be seen in Fig. 3, where the configurations in Fig. 3a and 3b failed to provide adequate skeleton tracking quality from certain user positions and the configuration in Fig. 3c was not preferred since adding the third sensor did not contribute significantly.

The final configuration, seen in Fig. 4, is formed of two Kinects. One is mainly used for user tracking while the other is mainly used for grid state tracking. The amount of sensor interference and the skeleton tracking quality in this setup was such that we were able to track the user's upper body anywhere within 50 centimeters of the grid periphery, as long as the user's crucial body parts (such as hands) are not hidden behind objects (e.g the rest of their body or the grid wall) with respect to the user tracking Kinect. Also, the amount of overlap in 3D surfaces viewed by both sensors enabled the extrinsic calibration (detailed in the next section) to converge.

2) Extrinsic Calibration: Once the Kinects are placed, their 3D transformations with respect to each other and to the grid

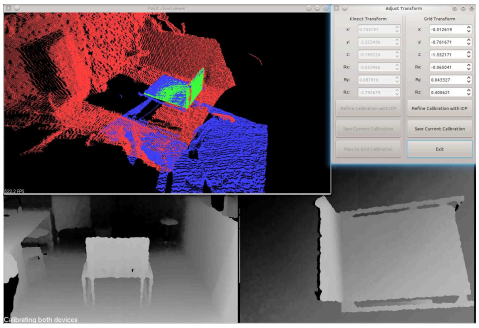


Fig. 5. The extrinsic calibration GUI. Top-left: PCL Visualizer window displaying point clouds seen by Kinects (red and blue) and the artificial point cloud representing the grid's 3D shape (green). Top-right: The transform adjustment window where the user roughly aligns the clouds by adjusting the X, Y, Z, Roll, Pitch and Yaw of the transforms and refines them using the Iterative Closest Point algorithm. Bottom: Filtered depth maps of the Kinects.

must be measured; this allows the untransformed coordinates measured by the Kinects in the camera frame to be placed in the same frame as the objects in the grid. This procedure, called extrinsic calibration, is performed by our auxiliary GUI application seen in Fig. 5. It is performed only once before the usage of the actual user interface.

The two Kinects are calibrated with respect to each other first. 100 depth frames from each Kinect are first recorded. Depth maps coming from a Kinect tend to be very noisy; for this reason, we take the median of all 100-frames and then apply a 5x5 spatial median filter exclusively to unknown pixels, similar to [18]. Our choice of temporal median filtering over mean filtering is to avoid creating spurious depth values. For instance, a pixel directly on the border of two objects with different depths may be averaged to the midway between these depths due to noise, where neither object exists.

At this point, the depth maps are converted into a point cloud format, provided by the Point Cloud Library (PCL) [3]. Next, these point clouds are transformed so that the "upwards" direction approximately corresponds to the Z axis of their frame, using the accelerometer data collected from each of the Kinects. The point clouds are presented to the user in a PCL Visualizer where they can zoom in/out, rotate and move the view in 3D. The user must do a rough alignment of the two point clouds manually by changing one of the clouds' X, Y, Z, Roll, Pitch and Yaw transform values with respect to the other. Since the clouds are approximately "up" towards the Z axis, Roll and Pitch are not required to be manually modified.

Once the rough alignment is done, the clouds are fully aligned with a variant of the Iterative Closest Point algorithm by Besl and McKay [6] implemented in PCL. The number of maximum iterations is kept low as to allow the user to rapidly see if the calibration diverges due to bad initial alignment or lack of sufficient overlapping surfaces seen by both sensors. The algorithm can be applied iteratively to converge to better results. Once the two clouds are calibrated, an artificial point cloud representation of the RB grid is created and calibrated against the union of the two previous clouds, with manual and automated calibrations similar as before.

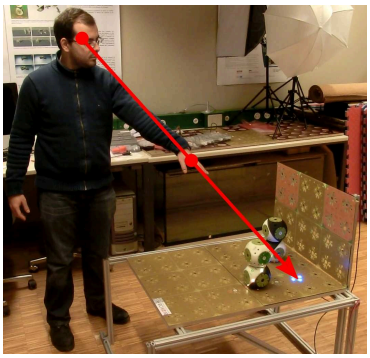


Fig. 6. Head-hand vector extension (in red) being used as pointing direction, the first object that the vector hits is considered "pointed at".

3) Pointing Gesture Detection: In order to detect where the user is pointing at, we use the skeleton tracker of the NiTE "middleware" in the OpenNI framework [2]; this method was successfully used by numerous authors (e.g. [15, 4]). NiTE skeleton tracker requires a depth map as input; therefore, we filter the depth maps before skeleton tracking in real time by a temporal running median filter with window size 3 (to accommodate the real-time constraint) and then a 5x5 spatial median filter applied to unknown pixels. The window size is determined empirically and provides adequate smoothing while not degrading the dynamism of moving entities compared to larger and smaller window sizes.

Next, we patch the missing pixels in the user tracker depth map with the information coming from the grid tracker's readings, if possible. Using this final depth map, the skeleton tracker returns the 3D locations of the 15 "joints" including head and left/right hands along with their confidence values. These locations are observed to be noisy as well, particularly when the user's body parts such as hands are at narrow angles with respect to the Kinect. For this reason, the joint locations are subjected to a weighted running average filter of 4 frame window size, a value determined again empirically to not degrade the inherent body motion. The 4 weights for each joint are confidence values, allowing us to decrease the value of more noisy readings compared to clean ones.

In accordance with Jojic et al.'s [11] and Pourmehr et al.'s [15] methodology, the head-hand vector is used as the pointing direction. It is observed to be more robust compared to other vectors such as shoulder-hand and elbow-hand in terms of user body orientation with respect to the sensor. Once detected, the head-hand vectors (for both hands) are extended and the first objects that these rays hit (if any) are considered being pointed at, as depicted in Fig. 6. For computational efficiency when checking intersections with these rays, the RB modules are approximated by two spheres corresponding to the two "sphere-like" segments, seen in Fig. 1a.

4) Grid State Detection: Using the union of the point clouds coming from the two depth sensors, the grid state perception system should detect the state of the grid and all objects in it. We currently lack this perception system and set the grid state manually; its design is among our future goals.

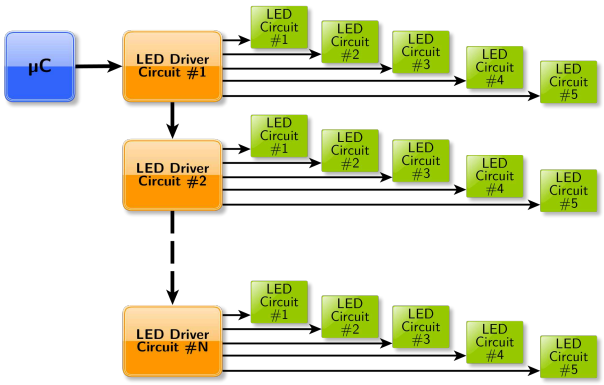


Fig. 7. Roombots grid LED illumination topology. The microcontroller (in blue) sends the state for each LED color on each tile serially, which translate to LEDs (in green) being turned on/off by the LED drivers (in orange).

(a) Grid colors in full intensity. Top: red, green, blue, yellow, cyan, violet tiles. Bottom: 6 white tiles.

(b) Module in green

(c) Module in blue

Fig. 8. LED illumination. (a): Grid tiles. (b), (c): Roombots modules.

B. Visual Feedback Setup

In order to enhance the usability of the interface, we designed LED-based electronics to be integrated to the RB grid tiles and the RB modules, discussed in the following sections.

1) Roombots Grid: During the usage of the interface, the user may point to and select grid tiles. Our goal is to give visual feedback concerning the physical gesture being performed and the current system state. For this, we designed an LED-based modular illumination system that enables us to illuminate each grid tile independently; it is thus compatible with all grid shapes and sizes that are composed of these tiles.

The two electronic circuits in our design, namely the LED circuit and the driver circuit, are connected according to the topology seen in Fig. 7. Each LED circuit is mounted under a single grid tile and can be lit in a desired color. Each driver circuit can drive up to 5 LED circuits, i.e. illuminate 5 grid tiles. The driver circuits are cascaded to form a communication chain, at the beginning of which is an Arduino Uno [1]. It was chosen for its low cost and ease of firmware development instead of integrating a microcontroller into our circuits directly. The Arduino board is connected to the host computer through USB, receiving the desired color for each tile. The mounted system can be seen in Fig. 8a.

2) Roombots Modules: Similar to the grid illumination system, our goal is to give visual feedback to the user about the current system state (such as the associated module's state) and about the current physical gesture being performed (such as pointing at or selecting the module). The design mainly follows the idea of illuminating the two outer diametrical DOFs instead of central points on the faces or corners of the module's cubic structures. This increases illumination visibility from various angles due to simple geometry and enables us to indicate the turning motion of the associated DOF via successively lighting LEDs. Therefore, we designed a circuit with 6 RGB LEDs on the rim and a dedicated microcontroller to illuminate a single DOF. Each RB module has two such circuits illuminating only the outer DOFs through semi-transparent rings as seen in Fig. 8b and 8c. Designing a similar circuit for the middle DOF is among our future works.

The firmware on the microcontroller is designed to produce various effects on the LEDs; these effects are used to convey different states of a module to the user:

- **Constant:** All LEDs are lit in a single color.
- **Breathe:** All LEDs are lit in a single color, intensity decreasing and increasing (similar to [14]) with linear interpolation.
- **Turn:** LEDs are switched on and off sequentially and in a circular fashion in a single color, fading from one to the next with linear interpolation.

The usage of these effects and their intended impressions on the user are explained in the next section. In addition to these, cross-fading during the passage from one effect/color to another is implemented for aesthetic reasons. The host computer, connected to the module via Bluetooth, commands the LEDs of a DOF by sending an effect and a color. We limit the colors to red, green, blue, yellow, cyan, violet and white, reducing the communication overhead and increasing visual distinguishability of one color from another.

C. Moving the Modules & User Interaction

The gesture detection and visual feedback systems described in the previous sections (II-A and II-B respectively) are combined to design a natural user interface in order to control the RB modules. In this stage, our previously proposed movement primitives and paradigms [8] are utilized. For simplicity, we limit ourselves to the structured motion of single RB modules on a planar grid, moving one tile at a time.

In order to command the individual RB modules from a host computer, a multithreaded software architecture is designed to allow concurrent motion. Modules may only perform a primitive after virtually allocating (if possible) all the physical space necessary for that primitive in a thread-safe manner: This allows collision-free motion and is elegantly compatible with our proposed D* algorithm for path planning [8].

The user interaction begins with all RB modules illuminated with the breathing effect in white color denoting their idle state. The grid tiles are not illuminated when idle, differentiating them from the "living and active" RB modules, which is an impression we expect to create on the user by the breathing effect. The user interacts with the modules and grid tiles by pointing at them with either hand; any tile or RB module

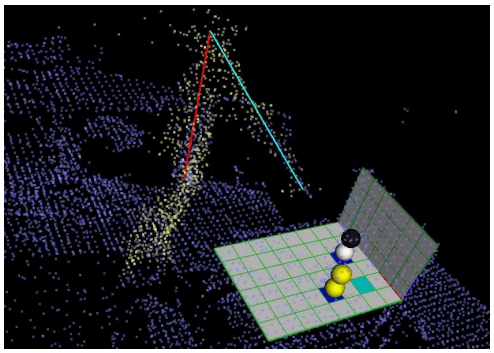


Fig. 9. Virtual system visualizer. Roombots modules (black & white spheres), Kinect point clouds (blue points; yellow points correspond to user's body) and the user pointing vectors (red and cyan lines) are seen. Here, the bottom module is selected (yellow spheres) and the cyan tile is being pointed at.

that is being pointed at is illuminated with a cyan color. Tiles that are being pointed at are lit in constant brightness while RB modules that are pointed at still display the breathing effect, emphasizing the "living" state. Besides creating these impressions, this also allows the user to know where they are pointing at, and forms a closed-loop control mechanism to help the user correct the error in their pointing direction. Since we are tracking the head-hand vector, the user must try to cover the object with their hand in their field of vision to point at it.

To select a tile or module, the user points at it for 2 seconds; with this, we avoid detecting other gestures that could possibly be used to indicate selection, which may be computationally costly. When an object is selected, it is illuminated with a constant yellow color. An already selected object is illuminated with a constant green color instead when being pointed at. When a RB module is selected, the previously selected module is deselected if there is any; tile selection behaves similarly. An object can also be deselected by pointing at it for 2.5 seconds.

As soon as a module and a grid tile are selected, the module is commanded to move to that tile provided that there is a collision-free path. While moving, the module is illuminated with a violet color. The rotating joints are indicated via illuminating their LEDs in the turning effect, indicating the direction towards which the module is moving. With this, we aim to let the user better perceive the motion of a RB module. In all of the above phases, our color selection is arbitrary and is not aimed towards giving an impression to the user.

During the operation of the interface, the user is presented with a simple representation of the system in a PCL Visualizer window in the host computer, seen in Fig. 9. Here, the point clouds (downsampled for better visibility) and pointing vectors of the user (when available) are presented. This extension is particularly useful for development purposes.

III. Results & Discussion

In our experiments, we equipped 12 grid tiles and one RB module with prototype LED illumination boards. We used two RB modules to test selection and simultaneous path execution with multiple modules. Instead of replanning a path, we simply stop the module when it cannot move due to obstruction from another module. When the obstruction clears, the user can reselect the module and reissue the command, allowing us to additionally test our interface.

In order to demonstrate our visual feedback setup better, the modules are not placed and moved far away from the tiles that are equipped with LEDs. Since there is one module and some tiles unequipped with LED boards, the virtual visualizer was used during testing. An example run of our interface can be seen in Fig. 10 and in the multimedia attachment.

IV. Conclusion & Future Work

In this study, we have presented a natural user interface to control the RB modules on the RB grid. We introduced our design for a dual depth sensor setup to track the user's skeleton in order to use the pointing direction to select target RB modules and grid tiles. We described our designs for LED based visual feedback hardware installed on RB modules and grid tiles. Finally, we combined the two aspects with a multithreaded motion controller to execute the user's commands and presented the use of our interface through an experiment.

The next step in our study is to design and implement the grid tracker. Selecting and commanding multiple RB modules is also an idea that we plan to study, as well as enabling locomotion on multiple connected surfaces, such as multiple floors, walls, ceilings or other modules already on the grid. Additionally, we plan to enable controlling off-grid locomotion of RB modules by using a laser projector to give the user feedback, analogous to our LED illumination design for grid tiles. All of these will allow us to realize our initial objective, enabling us to build structures out of RB modules, e.g pieces of furniture. Finally, a user study will be conducted in order to justify our interface paradigms against the state-of-the-art interfaces. As a final thought, our long term plans include applying our interface to other multi-robot platforms thanks to its compatibility. A future research perspective is to use our interface to control multiple wheeled agents.

Acknowledgements

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics. The authors would also like to express their thanks to André Badertscher for his technical aid and to Dr. Alessandro Crespi for his technical aid and advice.

References

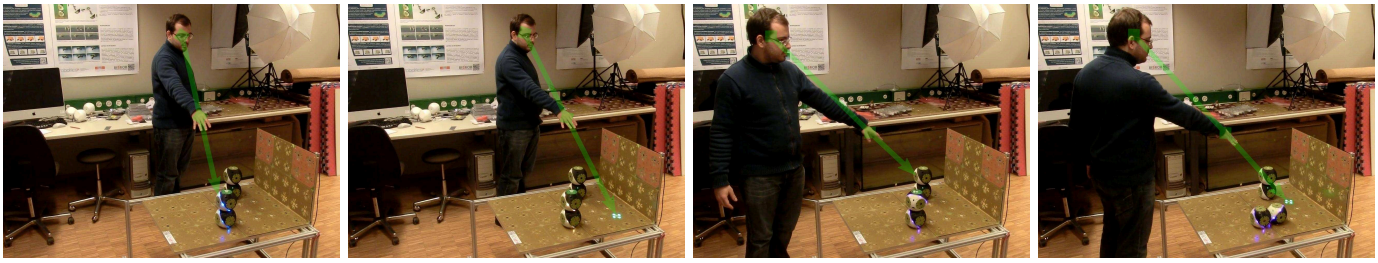
[1] Arduino Uno, 2014. URL http://arduino.cc.

[2] OpenNI, 2014. URL http://www.openni.org/.

[3] Point Cloud Library, 2014. URL http://pointclouds.org/.

[4] C. Bellmore, R. Ptucha, and A. Savakis. Interactive Display Using Depth and RGB Sensors for Face and Gesture Control. In *Image Processing Workshop (WNYIPW), 2011 IEEE Western New York*, pages 1–4. IEEE, 2011.

[5] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor. Markerless Motion Capture using Multiple Color-Depth Sensors. In *Vision, Modeling and Visualization*, pages 317–324, 2011.

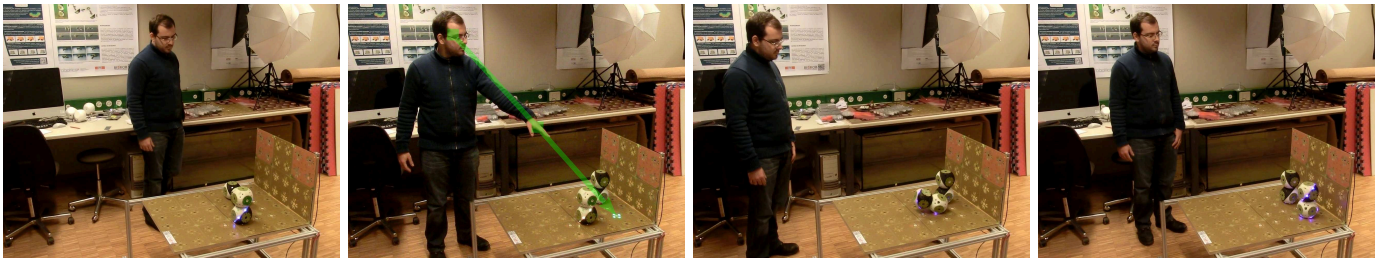


(a) Module 1 (bottom, with LEDs) is being pointed at in order to select it. It is lit cyan while being pointed at.

(b) Target is selected (lit green, i.e. both selected and pointed at), module 1 begins to move towards it.

(c) Module 2 (top, without LEDs) is being selected.

(d) Target for module 2 is selected, it begins to move towards it. Module 1 is still moving.



(e) Module 1 halts since the module 2 obstructs its path.

(f) Module 1 and its target are selected again after module 2 reaches its target.

(g) Module 1 moves along its path.

(h) Both modules reach their targets.

Fig. 10. Our interface being used; scenes are sequential in time. There are two modules and the user commands them to move to different targets. Only one module is equipped with LEDs. Pointing directions indicated in green in scenes (a), (b), (c), (d) and (f). This experiment can be seen in the multimedia attachment.

[6] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1992.

[7] S. Bonardi, J. Blatter, J. Fink, R. Moeckel, P. Jermann, P. Dillenbourg, and A. J. Ijspeert. Design and Evaluation of a Graphical iPad Application for Arranging Adaptive Furniture. In *RO-MAN*, pages 290–297, 2012.

[8] S. Bonardi, R. Moeckel, A. Sproewitz, M. Vespignani, and A. J. Ijspeert. Locomotion through Reconfiguration based on Motor Primitives for Roombots Self-Reconfigurable Modular Robots. In *ROBOTIK*, 2012.

[9] A. Couture-Beil, R. T. Vaughan, and G. Mori. Selecting and Commanding Individual Robots in a Multi-Robot System. In *Computer and Robot Vision (CRV), 2010 Canadian Conference on*, pages 159–166. IEEE, 2010.

[10] E. Hornecker and J. Buur. Getting a Grip on Tangible Interaction: A Framework on Physical Space and Social Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 437–446, 2006.

[11] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang. Detection and Estimation of Pointing Gestures in Dense Disparity Maps. In *Automatic Face and Gesture Recognition, Proceedings of Fourth IEEE International Conference on*, pages 468–475, 2000.

[12] D. Kortenkamp, E. Huber, R. P. Bonasso, and M. Inc. Recognizing and Interpreting Gestures on a Mobile Robot. In *In Proceedings of AAAI-96*, pages 915–921. AAAI Press/The MIT Press, 1996.

[13] M. Lichtenstern, M. Frassl, B. Perun, and M. Angermann. A Prototyping Environment for Interaction Between a Human and a Robotic Multi-Agent System. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 185–186, 2012.

[14] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt. Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75, 2006.

[15] S. Pourmehr, V. Monajjemi, J. Wawerla, R. T. Vaughan, and G. Mori. A Robust Integrated System for Selecting and Commanding Multiple Mobile Robots. In *ICRA*, pages 2874–2879. IEEE, 2013.

[16] A. Spröwitz, S. Pouya, S. Bonardi, J. van den Kieboom, R. Möckel, A. Billard, P. Dillenbourg, and A.J. Ijspeert. Roombots: Reconfigurable Robots for Adaptive Furniture. *Computational Intelligence Magazine, IEEE*, 5(3): 20–32, 2010.

[17] A. Spröwitz, R. Möckel, M. Vespignani, S. Bonardi, and A. Ijspeert. Roombots: A Hardware Perspective on 3D Self-Reconfiguration and Locomotion with a Homogeneous Modular Robot. *Robotics and Autonomous Systems*, 2013.

[18] W. Susanto, M. Rohrbach, and B. Schiele. 3D Object Detection with Multiple Kinects. In *Computer Vision ECCV 2012. Workshops and Demonstrations*, volume 7584, pages 93–102. 2012.