DOI: 10.3217/978-3-85125-528-7-20

Experiences with RGB-D based navigation in real home robotic trials*

P. de la Puente[1], M. Bajones[1], C. Reuther[2], D. Fischinger[1], D. Wolf[1], M. Vincze[1]

[1] ACIN Institute of Automation and Control.
Technical University of Vienna. Austria.

[2] MetraLabs Robotics GmbH Ilmenau. Germany.

Abstract
Autonomous robot navigation is an important and challenging component that is still missing in many real applications. In particular, home environments present open challenges that differ notably from one user apartment to another. Laser sensors cannot perceive objects at all heights commonly found in homes, we investigated the feasibility and suitability of using RGBD sensors for 2D autonomous navigation and a variety of tasks at real user homes. We use the concept of virtual laser scans to integrate RGBD data into mapping and localization methods. For realistic user interaction in actual homes we designed and improved ,over several pilot studies, the robot behavior for tasks such as approaching the user. In this paper, we report the adaptations needed to cope with home-specific challenges using RGBD sensors as a solution to perceive 3D environments.

1. Introduction

In many areas of application of service robots, mobility plays a role of great importance. Non-industrial real environments present increased complexity and in general are very hard to handle [18, 14].

In particular, autonomous navigation in user homes is a challenging aspect of care robotics projects. The SRS (MultiRole Shadow Robotic System for Independent Living) project pointed this out and focused on the development of remotely-controlled, semi-autonomous robotic solutions [11]. In other projects, such as Giraf and Giraff++, the robots were also externally teleoperated [2] and there was no autonomous navigation. This missing autonomous mobility has been identified as a key next aspect that needs to be solved. In the Companionable project, autonomous navigation to fixed predefined places was incorporated, but a controlled test home was used [17] instead of different real home environments.

RGB-D navigation poses additional difficulties [7, 3, 10]. The reduced field of view, the blind detection area and the small maximum range of this kind of sensors provide very limited information about the robot's sorroundings. Noisier points, spurious measurements and scale issues in the depth data also affect the perception capabilities.

This paper presents our developments, adopted solutions and identified issues to overcome the challenges of home environments using RGB-D sensors. We studied different navigation tasks and the adaptive approach to the user and conducted trials in several homes of older adults. The contributions

*This work was partially funded by the European Commission, project HOBBIT (FP7-288146).

to resolve the different tasks are described and specific problems of homes are addressed.

The paper is organized as follows. Section 2. describes the robot platform and sensor setup configuration. Section 3. presents our system architecture and implementation overview focusing on the navigation related tasks and components, which are explained in more detail in Section 4.. In Section 5., observed navigation problems are identified and addressed, while Section 6. provides an initial overview of the different navigation functions usage during the trials. Finally, Section 7. includes conclusions, final remarks and future challenges.

2. Robot platform and sensor setup

The PT2 mobile robot platform (prototype 2) was developed for the Hobbit project by partner Metralabs [12]. Two symmetric drive units and one supportig castor wheel constitute the low level locomotion system. A safety edge bumper protects the platform and blocks the motors while pressed, preventing the robot from moving while it is hitting an obstacle.

The sensor setup is based on two main RGB-D sensors, keeping a configuration similar to the one proposed for a previous prototype of the robot [5, 3]. The bottom camera -used for localization- is placed at a height of 35 cm. The head camera -used for obstacle avoidance, user detection, object and gestures detection and recognition- is mounted inside the robot's head, and can be tilted. 2D virtual lasers are created from each of the sensors, considering the largest measurements for localization with the bottom sensor (since they correspond to obstacles further away, like walls) and the closest measurements for obstacle avoidance with the top sensor. A height interval within the whole 3D point clouds is considered for the generation of the virtual scans.

3. System architecture and implementation

The whole system architecture has high modularity. To facilitate development, code reuse, communications management and integration, the popular Robot Operating System (ROS) [16] framework was used. Metralabs robots run the MIRA (Middleware for Robotic Applications) [4] framework, which manages low level control aspects of the platforms and also provides autonomous navigation functionalities and the Miracenter tool, which runs a complete instance of the framework with a graphical user interface.

In order to integrate MIRA into our ROS based system, several interfaces were required. The basic infrastructure of the new interfaces was based on existing interfaces from the STRANDS[1] project, modified and extended for our choices and needs. In our case, we decided to use MIRA navigation instead of ROS navigation because it was already well tuned for the current prototype platform and for reasons similar to the ones outlined by the Robot-ERA project team, such as enhanced support and robustness [9]. The required interfaces are implemented in different classes and run as a single ROS node.

In the first place, the virtual laser scans generated by ROS nodes had to be read, converted and adapted to be used by MIRA for localization and obstacle avoidance. In the other direction, an interface to provide the current localization pose as a ROS topic was required as well, and the corresponding trasformations are also computed and broadcasted.

[1]STRANDS project, EC 7th Framework Programme. Grant agreement num. 600623

Also, goal poses had to be sent from ROS nodes and SMACH to be processed by MIRA. This function was implemented as a ROS action server from which a MIRA navigation task including position, final orientation, and preferred driving direction subtasks -with their corresponding tolerances- is started. This way, a navigation task to a given goal can be preempted from the behavior side and the necessary feedback about the task status is provided by the actionlib server. The interface to start the action is the same as when sending a goal to ROS MoveBase action server, and the provided feedback is also translated to similar terms. Interfaces for discrete motion commands (distance to move, angle to turn) were also created to run in a separated ROS thread based on the distance mode experimental feature of MIRA.

Another action was created in order to start and interrupt docking on and off from the charging station. These procedures were implemented internally within MIRA.

Fig. 1 shows a simplified overview of the system architecture, including the most important modules and data flows with regard to navigation related tasks. More details about these tasks and methods are included in Section 4..

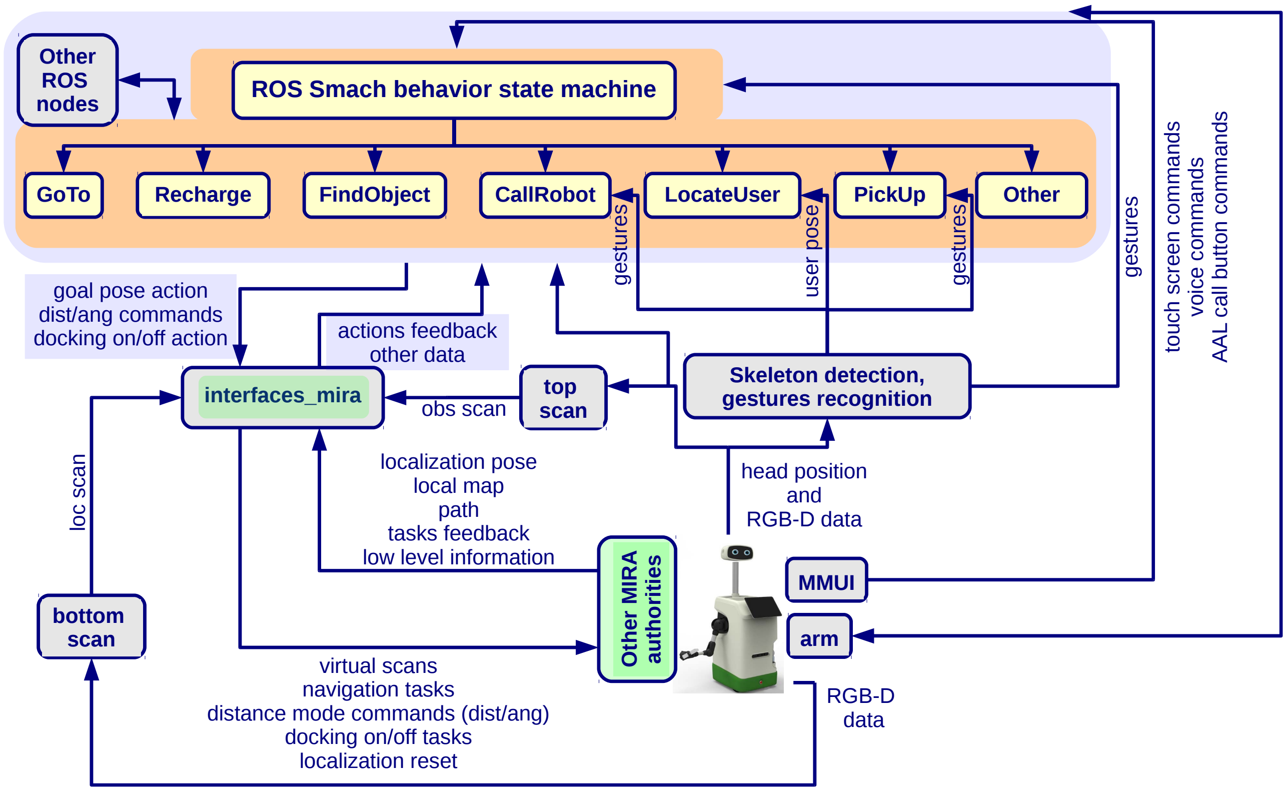

Figure 1. System components overview, focusing on navigation related tasks and modules (other details omitted for the shake of clarity). ROS Smach behavior state machines can interact with MIRA through the interfaces node directly or through other nodes.

The mapping process took place in advance, during the setup phase at each new trial site. For building new maps, we used the ROS implementation of Gmapping [8] and converted the generated maps into the MIRA corresponding format.

4. Navigation-related functions

Several functions desired by the users required navigation capabilities. This section describes these functions. More particular details of navigation between fixed places in real user homes are described in Section 5..

4.1. Go to place

This is the most basic navigation function. When started by the user, the robot should move from the current position to a given place, using predefinid positions and labels. The user can select the desired place/room name from a list displayed on the user interface or can use a voice command.

4.2. Recharge

During the setup phase, the charging station must be placed in a suitable place, which is not always easy to find in real apartments. Enough space for the station itself and its supporting plane that prevents it from moving is needed. There should also be enough space in front of it, so that the robot can detect the station with the bottom RGB-D sensor from a distance (the minimum recommended distance is 50 cm). Obstacles at the sides of the station can also reduce manouverability, increase the risk of false positives in the detection and result in a higher number of failures. Proper wall sockets and satisfactory conditions in the room are required, and visibility should be good, with no direct light coming from nearby windows. Last, but not least, it is very important that localization in front of the station is good so that the template is within the field of view (but the error distance to it is not so critical). Therefore, the station should not be placed along a featureless wall with few references in the orthogonal direction. Places in front of doorways are preferred over positions with a lower degree of geometrical variance in the alignment direction.

This task comprises several actions. In the first place, the robot has to reach a predefined position in front of the charging station. From this position, a docking action is started. The docking action starts the MIRA docking procedure, which first of all applies template matching between the bottom virtual scan and the station shape template (recorded from that point during the setup phase). If the template is found, template-based localization is activated and when the robot approaches the station obstacle avoidance is disabled. This procedure was specifically adapted by Metralabs to work with RGB-D sensors, since during the last part of the movement the robot is blind and open loop commands are applied then. When the docking movement is completed, the state machine checks whether the robot is indeed charging or not. If docking succeeded, the robot looks down and the user is notified, otherwise the robot should dock off and try again. If the template is not found, the action is considered aborted and then the state machine should apply a small rotation to one side and start the action again. If the station template is not detected again, then a rotation towards the other side should be applied and the action is started once more. If detecting the template is still not possible the whole docking task is aborted and the user is notified.

Once the robot detects that it starts recharging (either after autonomous docking or when manually placed into the station), localization is reset to the position of the station recorded and saved during the setup phase. This recovery mechanism was very useful both for testing and during the trials.

4.3. Find object

This function requires navigation to a set of predefined searching positions, usually located in front of tables, chests of drawers and other horizontal surfaces at intermediate height intervals. The searching positions were placed around 60-80 cm away from the surfaces border, trying to cover a large area of the horizontal plane with the head sensor. For this function, the path to all the searching positions was requested by setting a navigation task and muting the navigation until a path is received or a given timeout is reached. This way, it was possible to obtain the path from MIRA, which does not allow arbitrary path planning. If all paths are received, the searching positions are checked in increased length order, otherwise goals to which a path is not provided are visited first.

4.4. Call robot

At first, a fixed predefined place was associated to each call button id, so that the robot would come to that place when the call button was pressed. This simple method was not flexible enough, hence a novel approach was developed for this function.

It was convenient for the users to call the robot while sitting on their favourite chairs, armchairs, sofas, beds, etc. However, since the prototype platform was not able to rotate on the spot and the back side of the robot could collide with the furniture when turning around to drive away, the predefined positions could not be very close to the actual sitting place. It is also important to take into account that there can be different localization errors, plus the allowed error to decide whether the goal has been reached, so the real position is not always exactly the same (the uncertainty accumulates). A compromise was often needed so that the user could reach the touch screen while allowing the robot to detect the obstacle and drive away safely. Furthermore, a more significant limitation was the fact that chairs and armchairs usually can move and the sitting position of the user along a sofa or a bed will most likely vary from time to time. So the desired final position should not be fixed in an improved method.

The new approach we present incorporates user detection and interaction, remembered obstacles and discrete motion commands for coming closer to the user with better, adapted positioning. Discrete motion commands towards the detected user were chosen over a planned path for three main reasons: 1) the movement is more direct and predictable; 2) with our sensor setup, obstacle avoidance while following the path would require the head to look down whereas user detection and a nicer interaction require the head to look straight forward; 3) existing navigation frameworks, to our knowledge, are not directly suitable for planning a path to a goal out of the fov or blocked and inside an occupied area, so defining a proper goal pose in free space would be challenging, depending on the environment characteristics and requiring higher level knowledge about the specific furniture (orientation, feasible approach directions and so forth). Discrete motion commands based solely on the user detection with no obstacle avoidance whatsoever, on the other hand, can be risky.

Our solution works as follows. Predefined positions are defined further away from the sitting zone so that the user's skeleton should be inside or close to the head sensor fov. The maximum distance limit is given by the defined range of the virtual scan for obstacle avoidance or by the maximum distance for a discrete motion command. When a predefined position is reached, the navigation parameters are changed so that occupied areas in the current local map are not degraded or overwritten by new sensor data (user navigation mode). Then the head can move up for user detection[15], and ray-tracing on the local map is performed. The local map's origin is located at the odometry reference system and the origin for ray-tracing is defined at the robot hull front, based on the robot's corrected pose. The local map is used instead of the current measurements alone because it also contains obstacles remembered from before, even if they are already inside the sensor's blind zone. The robot turns to face the user and moves towards the sitting position up to a distance given by the closest projected measurement within an angle around the detected user, considering a safety margin. Fig. 2 illustrates this.

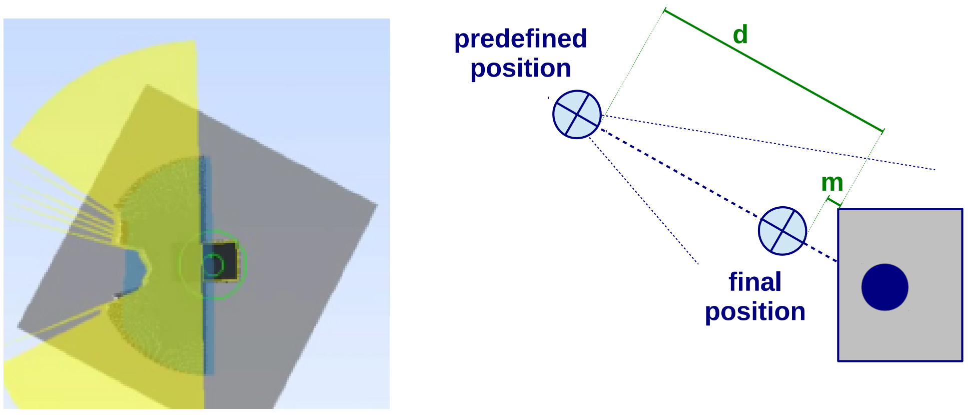

Figure 2. Obtaining the distance to move towards the user. Left: ray tracing on the local map. The current virtual scan when the head is looking up is depicted in blue, while the scan obtained from ray tracing is shown in yellow. The ray tracing scan is considered to be obtained from the frontal part of the robot (excluding the bumper, which is taken into account separatedly). The obstacle in front of the robot is not detected by the head sensor but is present in the remembered local map and therefore in the scan obtained by ray tracing. Right: simplified diagram of how the distance to move is computed. The minimum distance from ray tracing is denoted d and m is the safety margin.

For enhanced flexibility, reliability and better adaptation to different users, after this process is executed the robot asks whether to come even closer. When there is a positive answer the robot moves 15 cm more, completely blind now and trusting the user's input, and then the question and subsequent movement can be repeated up to three times. The user can reply by means of voice, gestures, or the touch screen commands. The robot remembers if it moved closer and if so it moves backwards again before starting a new navigation task.

The described method for approaching the user was developed for the call robot function, but it can also be applied in other tasks and contexts. For instance, we also incorporated it at the end of a fitness function scenario so that the robot comes closer to the user for further interaction after exercising.

4.5. Pick up object

For this function, in the first place, navigation to a goal obtained from the user's pointing gesture is needed. Using autonomous navigation in this way provides a much wider applicability and can be useful for picking up objects not directly inside the robot's field of view. The static map of the environment is also used for checking whether a detected object is too close to walls or funiture, since that means risk of collision and picking up the object is not possible then. Once the precomputed pose is reached, fine positioning with respect to the detected object is performed, based on discrete motion commands with sufficient accuracy. More details about the pick up algorithms can be found in [6].

4.6. Locate user

This function requires navigation to a set of predefined searching positions which in this case were usually located in the middle of the rooms or were the same as the call button places. Since detecting the user while the robot is rotating is hard, several shorter rotations were performed, stopping to call and detect the user in between. Using discrete motion rotations with our settings resulted in a more abrupt movement that affected localization, so a navigation task including only an orientation subtask was preferred. Still, the orientation estimate was sometimes not so good after the short rotations and the uncertainty associated to the orientation measurement had to be adjusted. Also, it was usually better to define this kind of positions in places where distinctive references were present and in relatively open space so that localization could get better before reaching narrower areas that require better accuracy.

5. RGB-D based navigation in home environments

One of the first things to check with the proposed sensor setup, depending on the environment, is that the height interval to be considered for the bottom sensor virtual scan generation may need to be changed. In general, we used a fixed width of 8 cm around the horizontal plane at the sensor's height. In the presence of low sofas and similar furniture, however, this interval had to be lowered since otherwise the virtual scan generated with the largest measurements included irregular surfaces such as cushions, etc. Fig. 3 illustrates this kind of problem.

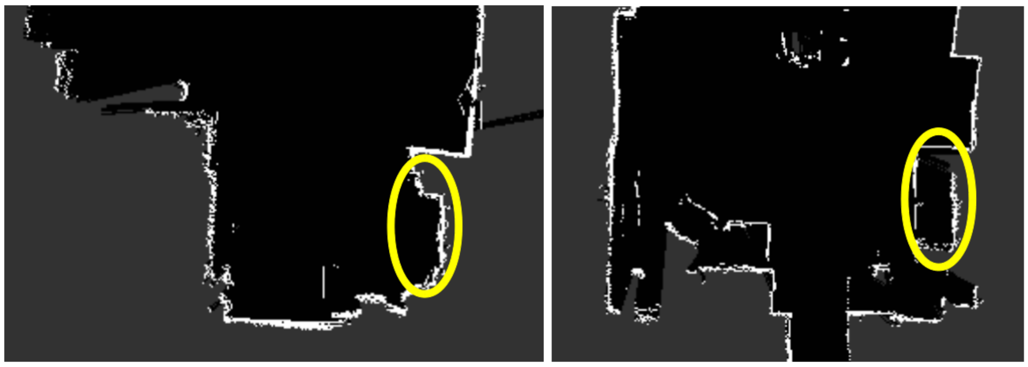

Figure 3. If the height interval considered for the bottom virtual scan is too high in the presence of low sofas, irregular borders and cushions will be present in the generated scan.

Regarding complete map building, the main limiting factors are the small field of view and the range properties of the RGB-D sensors. Since home environments may have narrow areas and excessive rotations during the mapping process can lead to less accurate results and even cause distortion, some walls may be missing in the resulting maps. This fact, together with the possibility that current sensor data overwrite the global map, can bring about problems in global path planning (see Fig. 4 for an example).

One first thing that can be done to prevent this problem when the robot is not facing the obstacle is carefully edit the map manually, to include the complete walls without risking the map building process overall result. This, however, does not help when the current sensor data are used to overwrite and clear the map. In that case, with this feature included within the MIRA based implementation, one option is to add MIRA `nogo` areas in order to avoid undesired plans when the robot is too close to a wall. The problem is that in very narrow areas some margin must be allowed for possible localization

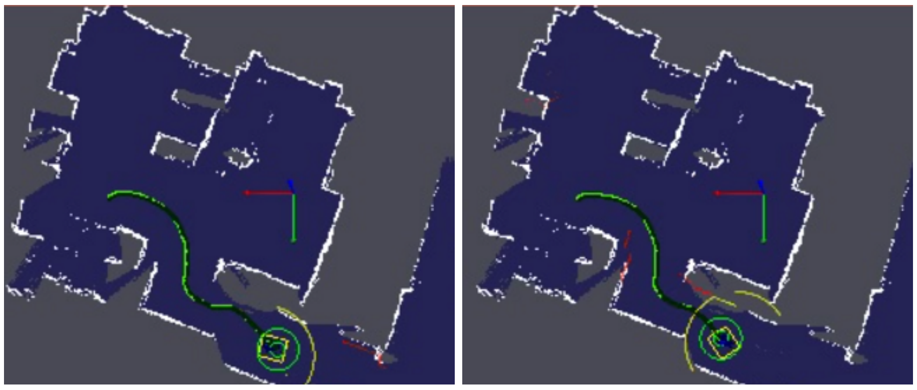

Figure 4. Global path planning in a narrow corridor in a home-like lab. One wall is missing in the map and is not always observed by the obstacles virtual laser (yellow). The initial path may be too close to the wall (left) and only be corrected if the robot gets to face the wall and is not already too close (right).

errors, so a compromise is needed. Fig 5 shows a couple of examples from real user homes. The environment on the left presents more critical localization conditions when entering the corridors that look horizontal in the image, since they are reached after traversing a long featureless corridor. Along this corridor, however, lateral localization is more accurate because the robot should not accumulate so much uncertainty before accessing it from these lateral, horizontal, shorter corridors.

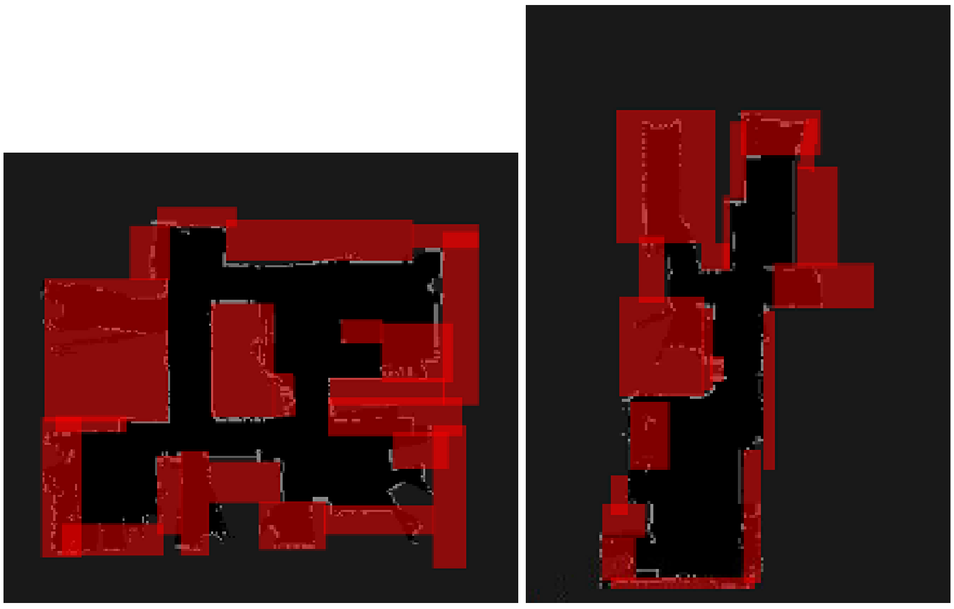

Figure 5. MIRA nogo areas can be added in order to avoid undesired global paths when the static map is cleared. In very narrow corridors, however, some margin must be left to account for posible localization errors.

Today's laser based localization methods assume that features of the environment can be detected. This is not true in long corridors where the robot could be anywhere along the parallel walls unless an end is observed, which is sometimes not feasible if the corridor is long or when the corridor ends in a room and hence a further opening. This problem can be avoided by using laser sensors, with a significantly longer range. Another option is to combine geometric methods with visual methods [13], but if the walls are uniform the same problem remains. A possible addition of a few external references in difficult areas such as corridors, very wide or narrow spaces, and ambiguous places could also help in the mapping and subsequent localization processes. In the previous case (Fig 5, left), we ended up adding a small extra piece of furniture in the long corridor as an additional reference. Visual markers with a fixed pose could help completely correct localization in those areas.

When entering a room, it is important that the robot is correctly localized in the transversal direction to the doorway and that the doorway is approached from the front, so doors located at one side of a corridor may cause problems, while doors located at the beginning or end of a corridor are better. In order to approach doors from the front, avoiding getting too close to the corner sides, a useful strategy to try in wide enough places is adding `nogo` areas at sides of a doorway entrance or at sharp corners (Fig. 6). This way, it is possible to have safer navigation behaviour in wide areas while keeping the capability to go through narrower areas. This provides more flexibility than methods with fixed security margins for the whole operational area.

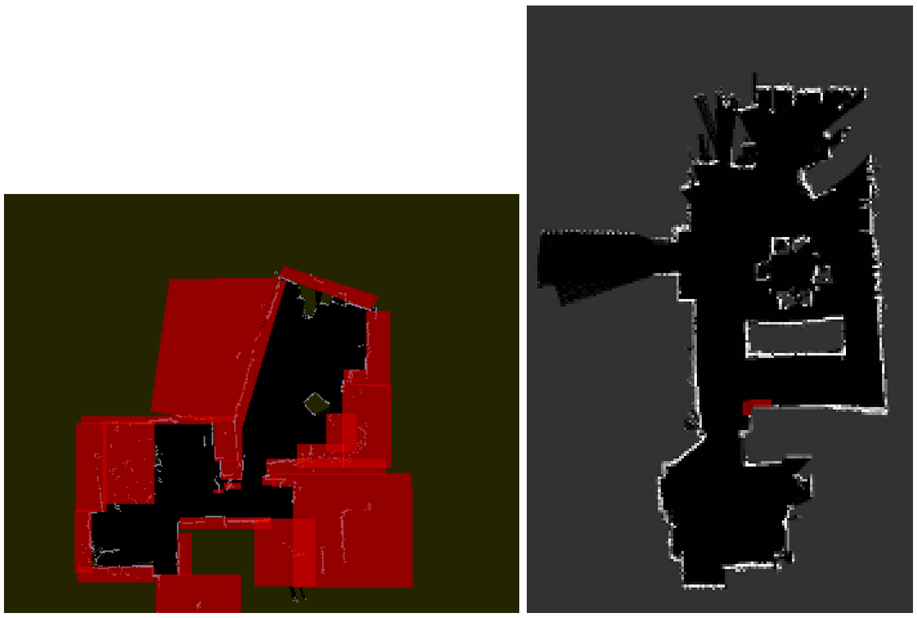

Figure 6. MIRA `nogo` areas can be used to avoid getting too close to corners and door sides. Note that good enough localization in the transversal direction is still very important.

`Nogo` areas were also useful to avoid difficult and not allowed areas and rooms in the environments. A few examples are shown in Fig 7. Areas with cables and thin obstacles on the floor and very narrow rooms (usually kitchens) where this non-omnidirectional robot could not manouver were also avoided. It is worth noting, of course, that `Nogo` areas are only useful if localization is good enough.

Other challenging situations were caused by thresholds, bumps on the floor and carpets. In the case of thresholds, we tested commercial and home made ramps when possible (Fig. 8). After testing different configurations and finding proper steep limits, the robot was usually able to pass thresholds. In a few cases problems were observed with standard planning methods if a new plan caused the robot to turn while driving on a ramp. A direct behaviour control may actually improve over plan-based approaches in narrow passages and in these particular cases, but it needs separate implementation and specific, situation-dependent triggering. Regarding localization, it would be useful to switch between 2D and 3D localization methods, but again the complexity would be increased.

Another important thing to take into account is that dealing with the dynamic nature of real environments is an open research issue to which new projects are dedicating big efforts [19, 1]. Our experience with the previously described approaches showed that minor changes in the position of

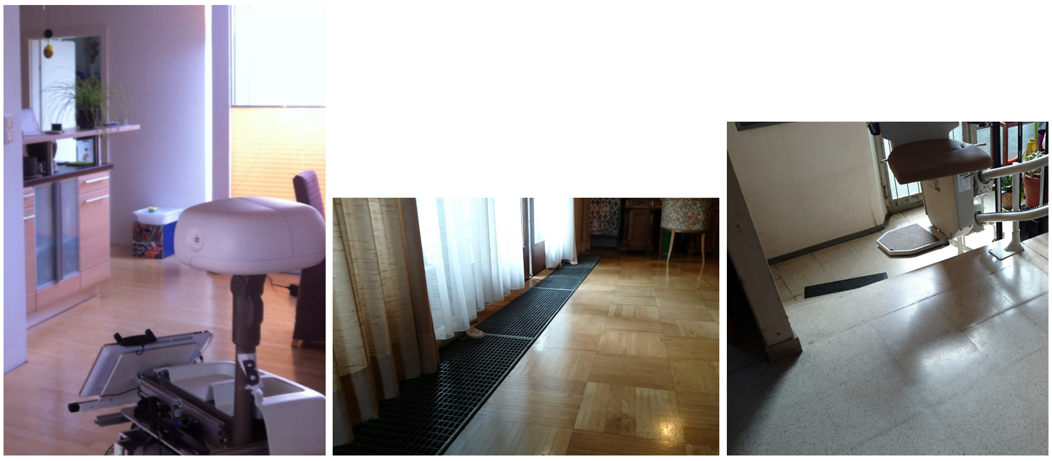

Figure 7. Examples of difficult areas in the environment to be avoided. Left: high outer shelves canot be observed with this sensor setup. Middle: the robot should not go through the uneven ventilation area close to the window. Right: areas with stairs are particluarly dangerous and should not be allowed.

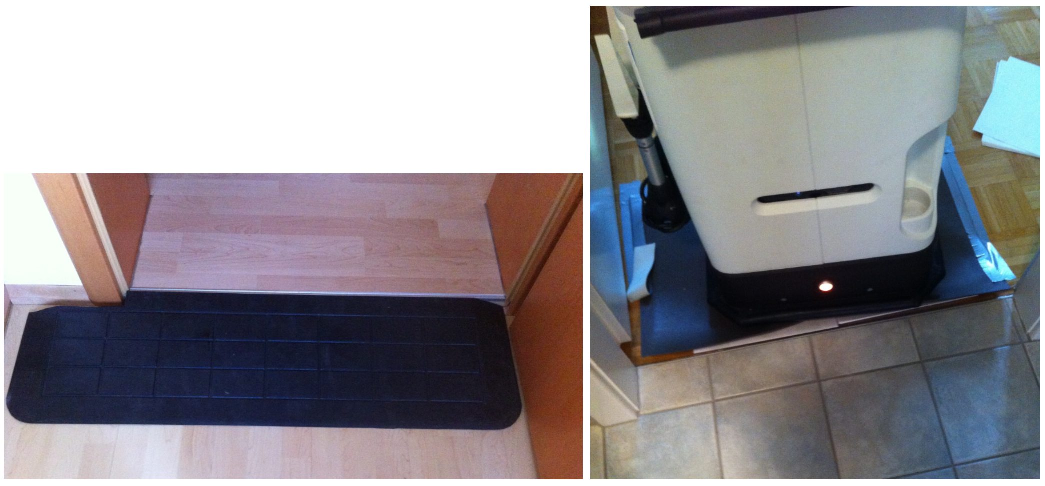

Figure 8. Using ramps to go through thresholds.

chairs and small items did not usually have a significant influence on the localization quality, as long as discrepancies with respect to the static map were limited. Removing or changing furniture that provided relevant geometrical references present in the map, however, led to serious differences, especially in narrow areas and corners with little space to recover while moving. Detecting when the robot is actually lost and trying to apply autonomous recovery methods are very challenging problems that require specific research.

6. Functions usage

Table 1 provides results on how often different functions were used in the user trials conducted in Vienna. Preliminary results show that most of the users evaluated these functions overall as “Good”, but there were also important failures and negative comments. Results in the lab were significantly better and we think that likely reasons are related to the following facts: there was more time for setup, preparation and improvements; the environment is easier and better known; we know the system better than the real users; robot transportation and long term operation can degrade platform performance and calibrations, etc.

Table 1. FUNCTIONS USAGE DURING THE TRIALS

User	`GoTo`		`CallRobot`		`Recharge`
	Used	Cancelled	Used	Cancelled	Used
V1	89	33	340	190	49
V2	29	8	172	81	186
V3	18	5	74	21	126
V4	46	26	97	75	16
V5	117	24	71	42	88
V6	386	85	146	60	312
V7	41	17	349	263	168

7. Conclusions and future work

This paper presented a summary of contributions and findings for navigation tasks of care robots operating in real home environments, using an RGB-D based sensor setup. Increased flexibility and adaptability over existing solutions were provided for the tasks execution, also addressing different limitations of the sensors used.

Our experience indicates that autonomous RGB-D navigation in real home environments is usually feasible and is much appreciated but presents drawbacks, open problems and further challenges. Directions for future work were already highlighted in the paper.

References

[1] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[2] S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, C. Galindo, J. Gonzalez, L. Karlsson, A. Forsberg, S. Frennert, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo, F. Pecora, S. von Rump, A. Stimec, J. Ullberg, and B. ¿Otslund. Giraffplus: A system for monitoring activities and physiological parameters and promoting social interaction for elderly. In *Human-Computer Systems Interaction: Backgrounds and Applications 3*, volume 300, pages 261–271. Springer International Publishing, 2014.

[3] P. de la Puente, M. Bajones, P. Einramhof, D. Wolf, D Fischinger, and M Vincze. RGB-D sensor setup for multiple tasks of home robots and experimental results. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[4] E. Einhorn, T. Langner, R. Stricker, C. Martin, and H. Gross. Mira - middleware for robotic applications. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[5] D. Fischinger, P. Einramhof, W. Wohlkinger, K. Papoutsakis, P. Mayer, P. Panek, T. Koertner, S. Hoffmann, A. Argyros, M. Vincze, A. Weiss, and C. Gisinger. Hobbit - the mutual care robot. In *Workshop on Assistance and Service Robotics in a Human Environment at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[6] D. Fischinger, A. Weiss, and M. Vincze. Learning grasps with topographic features. 2015.

[7] J. González-Jiménez, J. Ruiz-Sarmiento, and C. Galindo. Improving 2D reactive navigators with Kinect. In *10th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2013.

[8] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:2007, 2007.

[9] N. Hendrich, H. Bistry, and Jianwei Z. PEIS, MIRA, and ROS: Three frameworks, one service robot -A tale of integration. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014.

[10] M. Jalobeanu, G. Shirakyan, G. Parent, H. Kikkeri, B. Peasley, and A. Feniello. Reliable kinect-based navigation in large indoor environments. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[11] M. Mast, M. Burmester, E. Berner, D. Facal, L. Pigini, and L. Blasi. Semi-autonomous tele-operated learning in-home service robots for elderly care : A qualitative study on needs and perceptions of elderly people, family caregivers, and professional caregivers. In *Proc. of the 20th International Conference on Robotics and Mechatronics*, 2010.

[12] Metralabs.

[13] P. Panteleris and A. Argyros. Vision-based SLAM and moving objects tracking for the perceptual support of a smart walker platform. In *Computer Vision - ECCV 2014 Workshops*, volume 8927 of *Lecture Notes in Computer Science*, pages 407–423. Springer International Publishing, 2015.

[14] E. Prassler, R. Bischoff, W. Burgard, R. Haschke, M. Hñgele, G. Lawitzky, B. Nebel, P. PlÂger, U. Reiser, and M. ZÂllner, editors. *Towards Service Robots for Everyday Environments. Recent*

Advances in Designing Service Robots for Complex Tasks in Everyday Environments. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2012.

[15] A. Qammaz, N. Kyriazis, and A. A. Argyros. Boosting the performance of model-based 3d tracking by employing low level motion cues. In *Proc. of the British Machine Vision Conference (BMVC).* BMVA Press, 2015.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[17] C. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, C. Huijnen, H. van den Heuvel, A. van Berlo, A. Bley, and H.-M. Gross. Realization and user evaluation of a companion robot for people with mild cognitive impairments. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1153–1159, 2013.

[18] A.J. Soroka, Renxi Qiu, A. Noyvirt, and Ze Ji. Challenges for service robots operating in non-industrial environments. In *Proc. of IEEE International Conference on Industrial Informatics (INDIN)*, 2012.

[19] G.D Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *International Journal of Robotics Research*, 32(14):1662–1678, 2013.