Autonomous Indoor Robot Navigation Using a Sketch Interface for Drawing Maps and Routes

Federico Boniardi Abhinav Valada Wolfram Burgard Gian Diego Tipaldi

***Abstract*— Hand-Drawn sketches are natural means by which abstract descriptions of environments can be provided. They represent weak prior information about the scene, thereby enabling a robot to perform autonomous navigation and exploration when a full metrical description of the environment is not available beforehand. In this paper, we present an extensive evaluation of our navigation system that uses a sketch interface to allow the operator of a robot to draw a rough map of an indoor environment as well as a desired trajectory for the robot to follow. We employ a theoretical framework for sketch interpretation, in which associations between the sketch and the real world are modeled as local deformations of a suitable metric manifold. We investigate the effectiveness of our system and present empirical results from a set of experiments in real-world scenarios, focusing both on the navigation capabilities and the usability of the interface.**

I. Introduction

The design and implementation of intuitive methods of communication between robots and humans has attracted considerable attention from both the robotics and AI communities thus far [18]. In the context of robot navigation, significant amount of research has been devoted to develop natural and human-friendly means for transferring spatial information from users to robots as well as to enhance the robot's cognition about the surrounding environment [21], [20], [6], [11]. However, those approaches require the existence of a geometric map to allow the robot to perform the navigation task. Those maps are usually retrieved upfront by means of human teleoperation or autonomous exploration. Although many popular methods for simultaneous localization and mapping (SLAM) have proven to be extremely efficient as well as accurate [4], [5], they all require preliminary operations that could be tediously time-consuming or sometimes even unfeasible. Rescue scenarios, for instance, are common examples where remotely controlling a robot could be impossible for an external operator. Furthermore, new service applications require robots to be employed even by naïve users, such as older people or children, which would be overburdened by onerous or excessively complex operations. To overcome these difficulties, researchers have investigated the use of hand-drawn maps and sketches to provide a rough descriptions of the environment [8], [7], [16].

In this work we propose the use of hand-drawn sketch maps as a mean for interaction between humans and robots, when no prior geometric map of the environment is available.

The authors are with the Autonomous Intelligent Systems Lab at the University of Freiburg, Institute for Computer Science, 79110, Freiburg i. Br., Germany. This work has been partly supported by the European Commission under contract numbers FP7–610532–SQUIRREL, FP7–610603–EUROPA2, Horizon 2020–645403–RobDREAM.

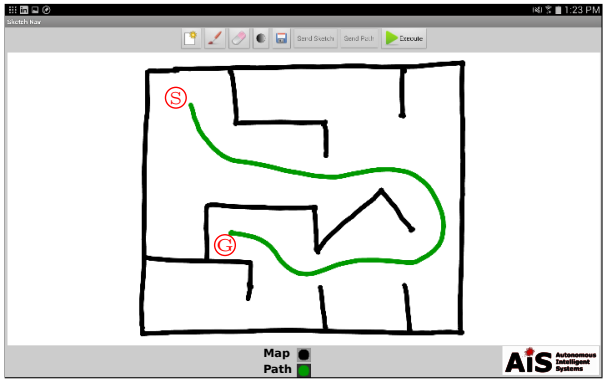

Fig. 1. Top: Snapshot of the sketch interface. Figure shows the drawn map and the path (green). The starting (S) position and the goal (G) position are annotated.

We envision a scenario in which a user can immediately operate her newly-bought robot with just her tablet. We designed and implemented an interface that allows the user to sketch a map of the environment and a path that the robot should follow for simple navigation and exploration tasks. A suitable planner autonomously handles small inconsistencies in the sketch as well as avoidance of unmapped obstacles, therefore the user is only required to provide a high level description of the scenario. For the navigation, we employ the theoretical framework introduced in our previous works [1], [2], where the sketch is interpreted as a Riemannian manifold whose metric tensor is unknown. Consequently, we estimate the metric together with the current robot pose using a Monte Carlo Localization algorithm [19].

The remainder of the paper is organized as follows. In Section III we quickly summarize the theory behind the manifold formalism for sketch interpretation as proposed in [2] and describe the navigation stack employed to perform the autonomous navigation. Section IV outlines the design and core components of the sketch interface. Finally, in Section V, we present the results from our experimental evaluation, both in terms of the autonomous navigation capability of the robot and from the perspective of the interface usability.

II. Related Work

An early attempt to perform simple navigation tasks only relying upon sketched maps was suggested in [8]. In this research, the authors proposed a POMDP based approach to learn a metrical conversion between a sketch, encoded as a topological map, and the real world. More recent

approaches have tackled the problem of providing a quantitative interpretation of a hand-drawn sketch via landmarks' matching, mimicking human-like navigation. Kawamura *et al.* [7] developed a full navigational system in which a robot is instructed to track a trajectory in a sketch. The robot navigates heading towards the waypoints that best match the predicted scenario perceived by the robot's sensors and the landscape observable by the waypoints. The current robot pose is meanwhile tracked by triangulating the relative positions of the predicted landmarks.

A wide and deep investigation into sketch-based navigation has been proposed by Skubic *et al.* [18], [17], [15], [16], [3]. In their works, the authors focused on designing and testing navigation systems that use a sketch of the environment together with a feasible path to navigate through it. A fuzzy state controller is then responsible for outputting suitable motion commands based on the qualitative state of the robot inferred from local sensor readings. The state is retrieved from the spatial relations between landmarks, modeled using histogram of forces, and later converted in a linguistic description by means of fuzzy rules. Shah and Campbell [13] have proposed an extension to this approach. The authors used techniques inspired from landmark-based SLAM to track uncertain landmarks and plan trajectories accordingly. Paths are therefore encoded as a set of waypoints output by a quadratic optimizer that accounts for the mutual position of the robot and estimated landmarks. Other approaches for matching the sketched scene with the real world have been suggested in [10] where Particle Swarm Optimization techniques are used to fit a hand-drawn sketch to an occupancy grid build using the current sensor data.

Along with the sketch-based navigation systems, suitable interfaces were designed and evaluated their usability. Chronis *et al.* [3] proposed an interface implemented on a PDA that interprets the sketch in terms of extracting landmarks suitable for navigation. The users are required to draw regions where objects such as desks, baskets etc. are located and a path for the robot. A similar interface was used in [16], but improved to interactively control a team of robots. Although the system seems to be effective, still it is not clear how detailed the map should be as the authors tested it only when just one object was missing [3]. In fact, such an approach could scale badly in more complex environments such as apartments with many rooms and clutter, which might result in wrong landmark associations. Furthermore, the need of drawing landmarks and understanding the relative displacement of the objects in a complex scene could be a nuisance for a user. Finally, nothing guarantees that the human actually draws those landmarks that are actually useful to the robot navigation. Conversely, our approach only requires a rough map, drawn as a floor plan that only has to be topologically consistent with the real environment. This makes our interface suitable for more complex environments.

Other approaches for sketch navigation, sometimes called *stroke-based*, employ the sketch interface as a mean to specify motion commands and behaviors. In [14] a sketch device uses a Hidden Markov Model to interpret a sequence of patterns drawn on the interface. Such patterns, also called *gestures*, encode robot behaviors: an arrow is used to adjust the robot orientation or a spiral to pinpoint a landmark are just few examples. Similarly, Sakamoto *et al.* [12] proposed an interface to control a vacuuming robot were commands such as *move* or *vacuum* can be sent drawing predefined gestures on the sketch.

III. Navigation in Hand-Drawn maps

In order to infer a metrical description of the sketch as well as to localize the robot in the hand-drawn map we employed the manifold formalism and the extended Monte Carlo Localization algorithm introduced respectively in [1] and extended in [2]. Here we just recapitulate the approach at a high level, for a full mathematical description the reader should refer to [2].

As core assumption in the framework, we suppose the sketch $\Omega_{\mathscr{S}} \subset \mathbb{R}^2$ to be the result of the action of a diffeomorphism $\Phi : \Omega_{\mathscr{W}} \longrightarrow \Omega_{\mathscr{S}}$ that applies local deformations on an underlying unknown map $\Omega_{\mathscr{W}} \subset \mathbb{R}^2$, which is metrically consistent with the robot workspace and sensors. Accordingly, the transformation Φ defines a metric tensor $\mathfrak{g}_{x,y} := [\nabla\Phi(x,y)]^\intercal[\nabla\Phi(x,y)]$, *i.e.* a local metric on the sketch manifold that provides a metrical conversion between the real world and the sketch map.

Under the assumption that a person is able to perceive orthogonality and parallelism of walls in a indoor environment, we can suppose the deformation to be shearing free or, mathematically, the tensor $\mathfrak{g}_{x,y}$ can be represented by a positive diagonal matrix. As a consequence, the Jacobian operator simplifies uniquely, into

$$\nabla\Phi(x,y) = R(\omega)\begin{bmatrix} a(x,y) & 0 \\ 0 & b(x,y) \end{bmatrix}, \tag{1}$$

where $R(\omega)$ is a rotation matrix of angle $\omega \in [0, 2\pi)$, which is supposed to be constant. Here, $a(x,y), b(x,y) \in \mathbb{R}_+$ are two independent scaling factors that account for the local deformation of the map along the direction of a suitable reference frame on the sketch. Such reference absorbs the rotational term $R(\omega)$. More formally, $[\mathrm{T}_{\mathscr{W}\to\mathscr{S}}]^{rot} \equiv R(\omega)$. Such transformation can be computed by preprocessing the sketch as described in [2].

To track the pose of the robot in the sketch the robot state $\mathbf{x}_t^{\mathscr{S}} \in \Omega_{\mathscr{S}} \times [0, 2\pi)$ is extended with the two scaling factors $a_t := a(x_t^{\mathscr{W}}, y_t^{\mathscr{W}})$ and $b_t := b(x_t^{\mathscr{W}}, y_t^{\mathscr{W}})$. The resulting extended state ξ_t is then updated via Monte Carlo Localization by setting proposal distribution and sensor model for the readings $\mathbf{z}_t$ as follows:

- *Proposal distribution*:

$$\begin{bmatrix} \mathbf{x}_{t+1}^{\mathscr{S}} \\ a_{t+1} \\ b_{t+1} \end{bmatrix} := \begin{bmatrix} \mathbf{x}_t^{\mathscr{S}} \oplus ([\mathrm{diag}\{a_t, b_t, 1\}][\mathbf{u}_t \oplus \boldsymbol{\varepsilon}_t]) \\ a_t\gamma_t^{(a)} \\ b_t\gamma_t^{(b)} \end{bmatrix}, \tag{2}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}_{0,\Sigma}$, $\gamma_t^{(i)} \sim \Gamma_{\sigma_i^{-2},\sigma_i}$ $(i = a, b)$.

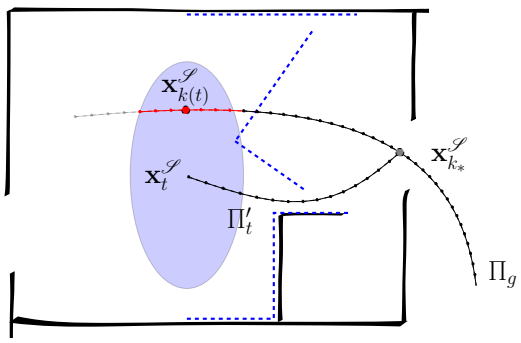

Fig. 2. Sketched representation of the robot avoiding an unmapped obstacle in the during the navigation. In blue the local window $W_L(\mathbf{x}^{\mathscr{S}}_t, r_L)$ to track the position of the robot on the drawn path. The dashed blue line represents the scaled scan $(z'_{i,t})_{i=1}^N$.

- *Likelihood fields model*:

$$p(\mathbf{z}_t|\xi_t) \approx \mathcal{L}_{\lambda,a_t}(a'_t)\mathcal{L}_{\nu,b_t}(b'_t)\prod_{i=1}^{N}\mathcal{N}_{o'_{i,t},\sigma}(z'_{i,t}), \quad (3)$$

where $\mathbf{z}'_t = (z'_{i,t})_{i=1}^N$ are the measurements transformed with respect of $\mathfrak{g}_{x,y}$ and a'_t, b'_t are virtual measurements obtained raytracing the sketch from the predicted robot pose.

A. Trajectory Tracking and Local Planning

In order to set up a navigation system that is able to track and execute the path drawn by a user on the sketched map, we designed the robot's controller to have two different layers, namely:

- A *local planner* responsible for outputting collision free trajectories from the current robot position to the target waypoint on the drawn path. As in [2], we use a Dijkstra planner on the local occupancy grid defined by the scaled readings $(z'_{i,t})_{i=1}^N$ defined above. A local planner that computes collision free trajectories is needed as the sketch should provide a high level description of the indoor environment without accounting for all the possible obstacles in the scene.
- A *trajectory tracker* that matches the current robot position with an approximate position on the desired path, with the aim of coordinating the two planners. It is apparent that, due to the presence of obstacles and inaccuracies in the sketch, only the local path is safe and consequently actuated by the robot. Thus the robot's trajectory can result in significant displacement from the desired sketched path, therefore a trajectory tracker is required.

Assuming that the avoidance of unmapped obstacles results in small detours from the sketched path, in order to match the current position of the robot with a waypoint on the drawn path, we apply the strategy depicted in Fig. 2. That is, given a sketched path $\Pi_g := \{\mathbf{x}^{\mathscr{S}}_k\}_{k=1}^K$ and a current robot pose on the sketch $\mathbf{x}^{\mathscr{S}}_t$, we define the *local window* $W(\mathbf{x}^{\mathscr{S}}_t, r_L)$ to be the set of all poses $\mathbf{x}^{\mathscr{S}}$ so that $\|\mathbf{x}^{\mathscr{S}}_t - \mathbf{x}^{\mathscr{S}}\|_{\mathfrak{g}} < r_L$, where the norm applies only to the positional components. Consequently, we select the subpath $\Pi'_t := W_L(\mathbf{x}^{\mathscr{S}}_t, r_L) \cap \Pi_g$ and consider the current position of the robot on Π_g to be the waypoint $\mathbf{x}^{\mathscr{S}}_{k(t)}$ that best approximates half of the arc length of Π'_t. In general Π'_t is not connected if a user has drawn a convoluted path. However, it is easy to discriminate which connected component of Π'_t should be chosen by following the ordering of the waypoints on Π_g and marking those that have already been visited.

Similarly to [2], to coordinate the local planner with the sketched path, we select a *lookahead window* $W_H(\mathbf{x}^{\mathscr{S}}_{k(t)}, r_H)$ depending on a parameter $r_H > 0$ as above and define the waypoint $\mathbf{x}^{\mathscr{S}}_{k_\#} \in \Pi_g \cap W_H(\mathbf{x}^{\mathscr{S}}_{k(t)}, r_H)^{\complement}$ $(k(t) < k_\#)$ to be the first waypoint in the path that lies outside the lookahead window. Finally, we plan a path in the sketch from $\mathbf{x}^{\mathscr{S}}_t$ to $\mathbf{x}^{\mathscr{S}}_{k_\#}$ with respect of the scaled readings as discussed above. The reader should notice that, since $\mathbf{x}^{\mathscr{S}}_{k_\#}$ lies on the path drawn by the user, there is no guarantee that it is part of the free space. In such case, the problem can be easily overcome by searching along the remainder of the path for the first free waypoint $\mathbf{x}^{\mathscr{S}}_{k_*}$ $(k_* \geq k_\#)$.

IV. Human–Robot Interface

In this Section, we describe the human robot interface that we propose for robot navigation tasks. Figure 4 shows the interaction diagram between the human and the robot, together with the tasks that a user can perform using our proposed human robot interface. The user is first presented with a canvas of size 2540 x 1252 pixels, in which she can sketch a map of the environment and draw polygons for obstacles. The sketched map is then sent to the robot when the user presses the *Send Sketch* button. The user then has the ability to draw the trajectory that the robot should take in the sketched environment. It is assumed that the user starts to draw the trajectory from the current position and orientation of the robot. There were no actual constraints set for the path to be drawn. The user can then send the sketched trajectory to the navigation system by pressing the *Send Path* button. The button is only activated and available to the user after the map is successfully sent.

The sketched map is encoded in the robot as a grid map, while the path is stored as a set of waypoints obtained by listening to touch events on the tablet. We interpret the

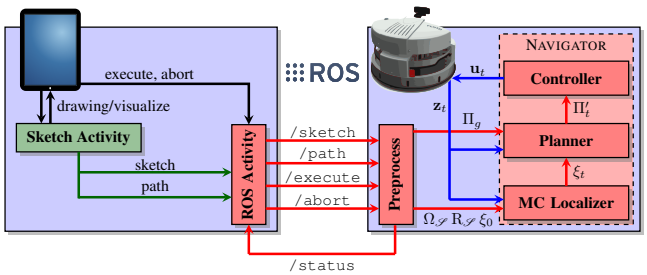

Fig. 3. System architecture showing the software components on the tablet and in the robot. As described in Sec III, $\xi_t := (\mathbf{x}^{\mathscr{S}}_t, a_t, b_t)$ is the robot's extended state [2]. Π_g and Π'_t are respectively the global and local path, $\mathbf{u}_t$ is the control and $\mathbf{z}_t$ the measurements. $(\Omega_{\mathscr{S}}, \mathrm{R}_{\mathscr{S}})$ is the sketched map provided with it's own reference frame.

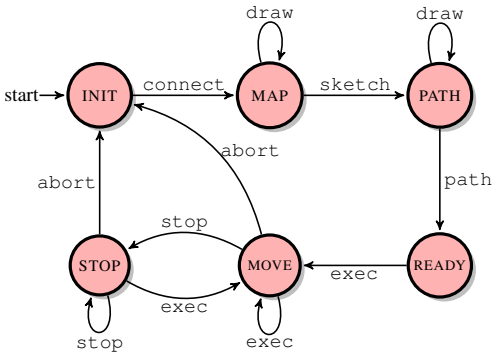

Fig. 4. A finite state machine depicting the tasks that a user can perform using the tablet interface.

initial position of the robot as the starting point of the path and set the initial orientation by estimating the direction of vector from the starting point to the next consecutive point beyond a preset threshold distance. This was done to avoid small squiggles in the beginning that affect the direction computation.

During both the map sketching phase, the user has the ability to redraw or erase parts of the sketch. This gives the user a very similar experience as drawing with a pencil and paper. The robot can then be instructed to navigate the sketched path by pressing the *Execute* button. The button is only activated and available to the user, after the navigational systems on the robot have been successfully initialized. This is notified to the interface using the `/status` command. She can also abort and restore the mission at any point of time during the execution. A feedback message is displayed once the sketch and path are successfully sent and once the task is executing or is aborted.

The sketch interface was designed to run on a tablet or a mobile phone with a stylus or a touch interface. The overall system architecture shown in Fig. 3, was implemented using the Robot Operating System (ROS) framework and the interface components were implemented on the Android operating system. ROSJava, a Java based distribution of ROS was used in the Android application to publish and subscribe to topics to the ROS core running on the robot. The tablet and the robot communicate through WiFi.

V. Experimental Evaluation

We evaluated our system in two indoor environments at the University of Freiburg, built using panels to simulate the walls. To remove any experimental bias, we placed obstacles of different sizes and shapes at random locations inside the test scenarios. The participants were first briefed about the task they had to perform and were shown the environment where the experiment was to be conducted. They did not have any technical knowledge on how the system worked or had seen the environment beforehand. Participants have been chosen randomly among the students in our lab with ages ranging from 22 to 32 years old. In order to maintain consistency in the evaluations, we did not alter the environment in any way between each experiment cycle. For carrying out the experiments, we used the Festo Robotino, an ominidirectional mobile platform equipped with a Hokuyo URG-04LX laser rangefinder. A picture of the test environment is shown in Fig. 1.

The task for the participants was to sketch a map of the environment and draw a path that they want the robot to follow in the sketched map. Most of the participants were not very familiar with drawing on a tablet so we allowed them to draw some sketches for a few trials to get acquainted with the interface. The participants were not specifically instructed whether they should also draw the obstacles in the environment, this was intentionally done in order to evaluate different scenarios.

There were a total of thirteen participants and they were split into two groups. The first group used the tablet in the landscape mode and the second group used the tablet in the portrait mode. We decided to conduct experiments using the tablet in different orientations because we noticed that users feel the urge to use the entire canvas to sketch the map, even if the proportions of the walls that they drew were very different from the real environment. Interestingly, this lead to different results in either cases. We sill discuss those results

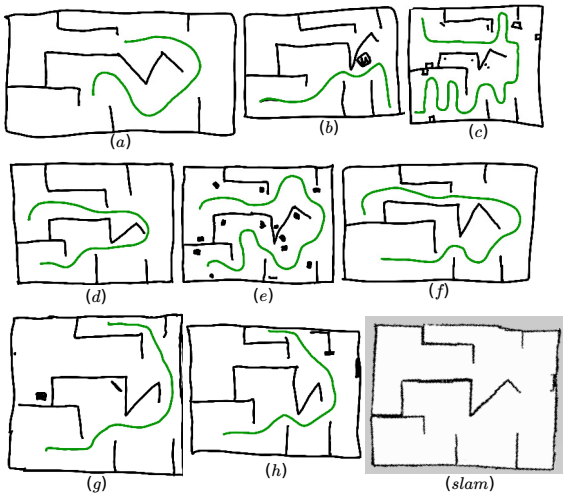

Fig. 5. Example sketches of the first scenario drawn by participants during the experiments. Some participants also sketch the obstacles. Bottom right, a map of the area obtained using a SLAM algorithm.

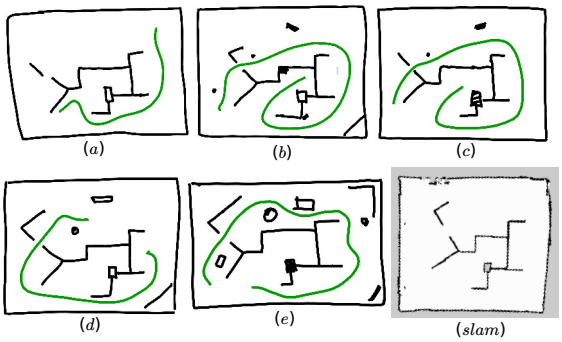

Fig. 6. Example sketches of the second scenario. Again, a consistent occupancy grid map is reported at bottom right.

in the following sections. At the end of the experiment, we asked the participants to fill out a questionnaire and provide suggestions to incorporate more intuitiveness into the interface.

A. Usability Tests

The sketches drawn by the participants show significant variations. A few examples are shown in Fig. 5. Some participants were concerned about drawing extremely straight lines for the walls but ignored drawing the obstacles (Fig 5(d), Fig 5(f)), whereas others were particular about drawing most of the obstacles in the environment (Fig. 5(c), Fig. 5(e)) but did not pay attention to the relative scales and positions of the walls (Fig. 5(a), Fig. 5(f)). Fig. 5(d) and Fig. 5(e) are some of the accurate sketches sufficiently depicting the environment.

The time that the participants spent on drawing the maps varied from 27 seconds to over 6 minutes, with the average being 2.38 ± 1.42 minutes. We did not observe any significant correlation between the time spent on sketching and the success rate for the robot to complete the task, as each participant paid attention to different parts of sketching and some spent considerable amount of time erasing and redrawing the map.

As mentioned in the description of the experiments, the participants performed the experiments using the tablet in two different orientations. We found that in the portrait orientation, the sketches drawn by the participants were more proportionally scaled, resulting in higher navigation success rate. As the screen real estate is smaller on the horizontal direction, it prevented them from drawing disproportionately rectangular sketches.

The questionnaire given to the participants was designed to get an insight on whether they felt at ease using the interface to complete the task at hand. We adopted the Likert scale [9] to rate the questions, with 5 (Strongly agree) being most satisfied and 1 (Strongly disagree) being the least. The survey revealed that using sketches to describe the environment was very intuitive for the participants, as they scored an average of 4.09. The users reported that having a small number of steps to perform in the interface to get the task done, the ability to edit the sketch and having multiple colors to sketch with, were all commendable.

Although only 30% of the participants strongly agree that the sketch is entirely representative of the environment and is easier to sketch than on paper, their comments revealed that this was because free-hand sketching on a tablet requires some practice and most participants had not sketched on a tablet before. This could be improved by providing the option of using predefined geometries for drawing. Almost no participant strongly agreed that the system is sufficient to complete the task, though 53% agreed. The users commented that this was because there was not enough feedback from the robot after the execute command is sent. Timely position updates and warnings or alerts can help provide more feedback to the user.

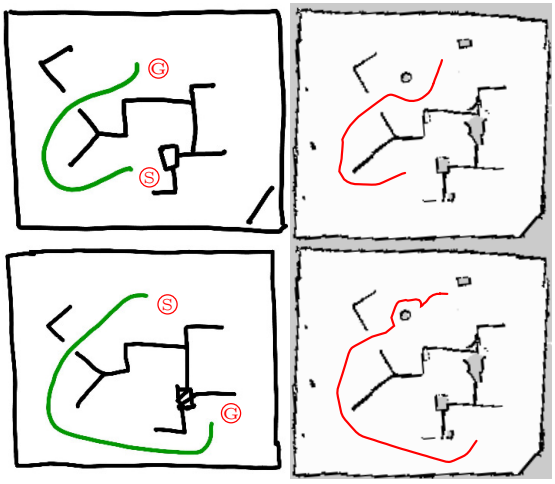

Fig. 8. Comparison between the drawn path and the actual trajectory of the robot during navigation task, unmapped obstacles are avoided. The tracked path is obtained using an external motion capture system (no metrical maps are available to the robot during the navigation). Start and goal positions are annotated in red.

B. Navigational Autonomy

Together with the thirteen experiments described above, other 12 were performed in an another environment (see Fig. 6 and Fig. 8) with an overall success rate of 68.0%, *i.e.* 17 successful runs of 25 sketches. Similarly to [2], a navigation task is considered successful when the actual trajectory of the robot on the sketch $(\mathbf{x}_t^{\mathscr{S}})_{t\in[0,T]}$ and the path drawn by the user are homotopically equivalent with respect to the topology induced by the sketch and the distance between the final robot position and the goal is lower than a threshold. We point out, however, that the reliability of the entire system is dramatically affected by the quality of the sketch.

The parameters in the navigation stack were initially calibrated and kept constant during the experiments. We tuned the parameters for the odometry and sensor model exploiting the results of running Monte Carlo Localization on metrically consistent maps. We chose the variances for the scales' model trading off the capability of adapting to the deformation of the sketch and the risk of increasing false detection. Similarly, the radius of the local lookahead window r_H affects the way the robot tracks the desired trajectory. If the radius is big, the robot is forced to track the locally optimal trajectory output by the Dijkstra planner. This results in considerable displacement from the drawn path if it is significantly suboptimal. However, if the parameter is chosen to be too small, the planner is not able to react quickly enough to unmapped obstacles and the safety of the navigation is severely affected.

We observed that some participants drew sketches with different levels of details, but we observed that a navigation task was successful independent of the amount of clutter drawn in the sketch, for instance Fig. 5(b), Fig. 5(d), Fig. 5(e) were successful, while Fig. 5(c) was not. The experiments reported in Fig. 6 showed a similar behavior.

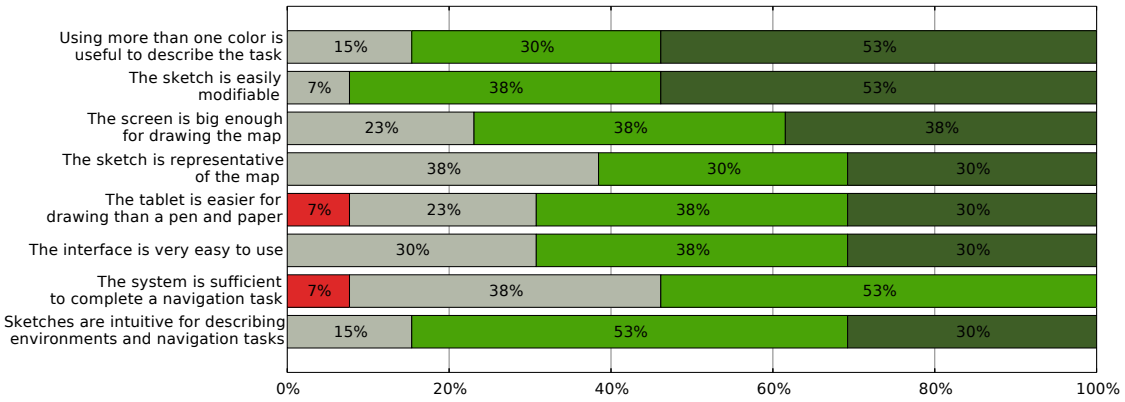

Fig. 7. Plot showing results from the post experiment survey. Strongly disagree, Disagree, Neutral, Agree, Strongly agree.

VI. Conclusions

In this paper we addressed the problem of equipping a robot user with an interactive tool by means of which spatial information about the environment can be communicated to the robot. To accomplish this, we designed and implemented a tablet interface that allows a user to sketch a map of an indoor environment and specify a desired trajectory that the robot should follow. We employed a theoretical framework that enables the robot to localize itself with respect to the hand-drawn map, which is achieved by tracking the pose of the robot together with a local metric of the sketch. We further use this metrical description to convert the sensor's readings into the sketched environment and use these virtual measurements to perform avoidance of unmapped obstacles as well as to overcome small inconsistencies in the drawing.

We performed a usability study of our interface to determine how practical it is to sketch a map of the environment that sufficiently describes the real-world, in order to successfully carry out a navigation task. We found that each user has a very different style and focus while sketching a map and the system has to be robust to all the variations of the sketch. Nevertheless, the system is able to perform navigation tasks in about 65% of the times and only a small percentage of the participants believed that that the minimal representation provided by a sketch is inadequate for successfully navigating a cluttered environment.

References

[1] B. Behzadian, P. Agarwal, W. Burgard, and G. D. Tipaldi. Monte Carlo localization in hand-drawn maps. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015. To appear.

[2] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi. Robot navigation in hand-drawn sketched maps. In *Proc. of the IEEE European Conference on Mobile Robotics (ECMR)*, Lincoln, UK, 2015.

[3] G. Chronis and M. Skubic. Robot navigation using qualitative landmark states from sketched route maps. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1530–1535. IEEE, 2004.

[4] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[5] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and accurate SLAM with rao-blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.

[6] S. Hemachandra, M. R. Walter, S. Tellex, and S. Teller. Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2623–2630. IEEE, 2014.

[7] K. Kawamura, A.B. Koku, D. M. Wilkes, R. A. Peters, and A. Sekmen. Toward egocentric navigation. *International Journal of Robotics and Automation*, 17(4):135–145, 2002.

[8] S. Koenig and R. G. Simmons. Passive distance learning for robot navigation. In *ICML*, pages 266–274, 1996.

[9] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[10] G. Parekh, M. Skubic, O. Sjahputera, and J. M. Keller. Scene matching between a map and a hand drawn sketch using spatial relations. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4007–4012. IEEE, 2007.

[11] A. Pronobis and P. Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3515–3522. IEEE, 2012.

[12] D. Sakamoto, K. Honda, M. Inami, and T. Igarashi. Sketch and run: A stroke-based interface for home robots. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 197–200, New York, NY, USA, 2009. ACM.

[13] D. Shah and M. Campbell. A robust qualitative planner for mobile robot navigation using human-provided maps. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2580–2585. IEEE, 2011.

[14] D. Shah, J. Schneider, and M. Campbell. A robust sketch interface for natural robot control. In *Proc. of the IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*, pages 4458–4463. IEEE, 2010.

[15] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, W. Adams, J. G. Trafton, and A. C. Schultz. Using a sketch pad interface for interacting with a robot team. In *Proc. of AAAI*, volume 5, pages 1739–1740, 2005.

[16] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, and A. Schultz. Using a hand-drawn sketch to control a team of robots. *Autonomous Robots*, 22(4):399–410, 2007.

[17] M. Skubic, C. Bailey, and G. Chronis. A Sketch interface for mobile robots. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, volume 1, pages 919–924. IEEE, 2003.

[18] M. Skubic, P. Matsakis, B. Forrester, and G. Chronis. Extracting navigation states from a hand-drawn map. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 259–264. IEEE, 2001.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[20] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. Learning semantic maps from natural language descriptions. In *Proceedings of Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

[21] H. Zender, O. Martiínez Mozos, P. Jensfelt, G-J.M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.