MOVING: A MOdular and Flexible Platform for Embodied VIsual NaviGation

Marco Rosano[1](✉), Francesco Ragusa[1,2], Antonino Furnari[1,2], and Giovanni Maria Farinella[1,2,3]

[1] FPV@IPLAB - Department of Mathematics and Computer Science, University of Catania, Catania, Italy
marco.rosano@unict.it

[2] Next Vision s.r.l., Catania, Italy

[3] Cognitive Robotics and Social Sensing Laboratory, ICAR-CNR, Palermo, Italy

Abstract. We present MOVING, a flexible and modular hardware and software platform for visual mapping and navigation in the real world. The platform comprises a flexible sensor configuration consisting of an RGB-D camera, a tracking camera for odometry, and a 2D Lidar, along with a compact processing unit that is equipped with a GPU for running deep learning models. The software is based on ROS, utilizing the RGB-D RTAB-Map SLAM system for mapping and localization and the move base package for path planning and robot movement control. The platform is easily detachable and can be installed on any robot with minimal adaptation required, enabling the reuse of the same robotic software regardless of the robot employed. The effectiveness of the proposed platform was verified through mapping sessions of a large indoor environment, leveraging a Loomo robot. The proposed platform can represent a reasonable solution to speed up the design and testing of new software for autonomous navigation systems, minimizing deployment time in the real world.

Keywords: Robot visual navigation · Visual mapping and navigation · ROS

1 Introduction

Recent years have seen a significant advancement in robotic technology. As a result, increasingly robust and reliable robot platforms have been developed, demonstrating their ability to carry out challenging tasks such as object grasping [11], visual localization [1] and navigation [18], which typically occur in industrial, commercial and even domestic scenarios. Although many hardware components for robotics have been developed for several years now, such as mobile bases with wheels and mechanical arms with gripper, others have been improved only recently. For example, sensors that enable robots to perceive their surroundings, such as cameras and Lidars, have been improved in their performance and

G. L. Foresti et al. (Eds.): ICIAP 2023, LNCS 14234, pp. 75–86, 2023.
https://doi.org/10.1007/978-3-031-43153-1_7

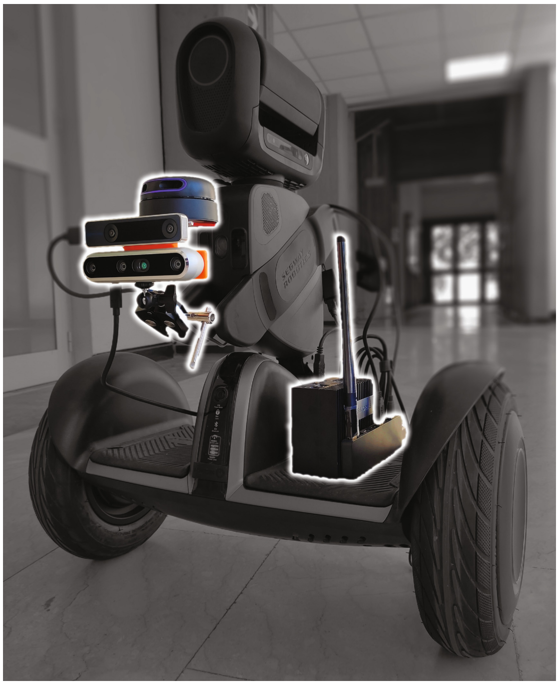

Fig. 1. The proposed MOVING platform, installed on a Loomo robot. The platform consists of a perception module (on the left) formed by an RGB-D camera (bottom-left), a tracking camera (center-left), a 2D Lidar (top-left), and a compact, low-power computational unit (on the right), equipped with a 6-core CPU, a Graphical Processing Unit (GPU) and WiFi connectivity. The sensors are supported by an ad hoc designed 3D printed mount and can be clamped to any robot. The computational unit runs the mapping and navigation software and supports running Deep Learning models on-device.

reliability, and their cost has also decreased. Similarly, the processing units have been improved and compacted to ensure they can be simply installed on any robot and process data directly on-board. On the software side, thanks to the recent progress of Deep Learning algorithms it is now possible to extrapolate richer information from the sensory data, allowing the robot to learn robust control policies and behave more naturally when compared to classic control systems [19]. In Previous works [15,17] several attempts have been made to standardize the software intended for controlling robotic systems. For instance, the ROS project [20] brings together a versatile software stack, developed over the years with the aim of providing the essential tools for the development of robotic control systems. Unfortunately, despite standardization attempts have been made also for robot platforms [3,14], those commercially available do not fully satisfy the different needs of research centers and companies which develop robotic solutions. Additionally, the design of the available robots often lacks modularity and interchangeability of the individual platform components.

In this work we propose a flexible and modular hardware and software platform for SLAM and visual navigation. The platform includes several sensors: an RGB-D camera, a tracking camera which provides odometry and a 2D Lidar. The compact processing unit is equipped with wireless connectivity, including a GPU to run Deep Learning models and can be powered by a simple power bank. The software is based on ROS [20], it uses the RGB-D RTAB-Map SLAM system [12] to map the environment and locate the robot inside it and the move_base package [13] for path planning and robot movement control. The platform, depicted in Fig. 1, can be easily detached and installed on any robot with minimal adaptation required. This allows the reuse of the same SLAM and navigation software regardless of the robot employed, hence reducing deployment time. Moreover, its use encourages to design and test navigation solutions based on the visual platform, rather than on the full stack including sensors, processing units and the robot, which makes the developed solutions more adaptable to different applications. In principle, the same solution could work on a Pepper[1] robot or on a robotic vacuum cleaner. The configuration of the sensors is flexible. It is possible to remove the tracking camera or the Lidar or both according to the precision, size and energy autonomy needed. The sensors' missing information is then estimated at the software level based on the other sensors. The effectiveness of MOVING was verified through several mapping sessions of a large indoor environment, carried out leveraging a Segway Loomo robot[2]. We discuss the flexibility of the proposed platform, which can represent a reasonable solution to speed up the design and test of new software for autonomous navigation systems. Given the remarkable perceptive capacity of the proposed platform and the availability of hardware acceleration (GPU) of the embedded system, MOVING allows to minimize the deployment time of Deep Learning models in the real world.

2 Related Work

Robot Platforms and Applications. Over the years, many robot platforms have been designed with the goal to assist humans in their everyday activities, while working [9], learning [2] or in need of special support [7]. Given the high costs associated with the development of these platforms, the first prototypes were developed by large private companies and large research centers for commercial applications and technological advancement [21]. In the '90s, the work of Hirai et al. [10] represented the first successful attempt to create a working humanoid robot, equipped with a large suite of sensors and able to replicate human activities (e.g. walking, grasping, etc.). The first studies on quadrupedal robots date back to the same period [4]. Regarding wheeled robots, many platforms were developed for research and educational purposes [3]. To foster the development of robotic technologies, many other platforms have been released as

[1] https://www.aldebaran.com/en/pepper.
[2] https://www.segway.com/loomo.

open-source [14]. Each of these robot platforms has a heterogeneous set of sensors with different perceptive capabilities and specific software for robot mapping, navigation and control, which support only a limited set of peripherals.

Our platform for SLAM and navigation contrasts with these approaches. In fact, the entire hardware setup can be easily mounted on any robot and, exploiting the ROS software framework [20], it can support a large variety of sensors and leverage open-source software for SLAM and navigation. This gives the flexibility to choose the most suitable set of sensors and to re-use the same software while changing the robot.

Visual Mapping and Navigation. Most navigation systems requires a map of the environment to carry out the operations of localization and path planning. The environment map can be built beforehand starting from a set of RGB images using a Structure From Motion method [22] to return a 3D reconstruction in the form of a point cloud, from which a navigable 2D grid map is extracted. In contrast, Simultaneous Localization And Mapping (SLAM) systems [6] can be used to reconstruct the environment in real time while exploring, starting from RGB-D images [16], from Lidar signal [8], or both [12]. The resulting floor plan is then exploited by the navigation stack to carry out path planning and provide the robot with the actions to execute towards the goal. As an alternative to traditional methods, Deep Learning-based mapping and navigation models have emerged, which are able to learn directly from the data provided, often collected from simulated environments [18]. These methods can replicate the SLAM algorithm pipeline and reduce the complexity of some of its components [5].

In this work we adopt a robust and reliable RGB-D SLAM system called RTAB-Map [12] along with the navigation stack called move_base [13], both integrated in the ROS framework [20]. However, the proposed platform has been designed to supports the use of Deep Learning models at both the software and hardware levels.

3 The MOVING Platform

Figure 2 summarizes the architecture of the proposed platform, which consists of hardware and software components and has been designed to ensure support for a large number of input peripherals, robot platforms and mapping/navigation software. MOVING embraces the concept of "Write once, deploy anywhere", which allows to develop a software application only once and deploying it on as many platforms as desired. To achieve this, we designed a compact hardware system (Fig. 1), easily installable on a large number of robots, with a high-performance and power efficient processing unit to support the computation load while ensuring adequate autonomy.

3.1 Hardware Architecture

The hardware architecture of MOVING consists of the following components:

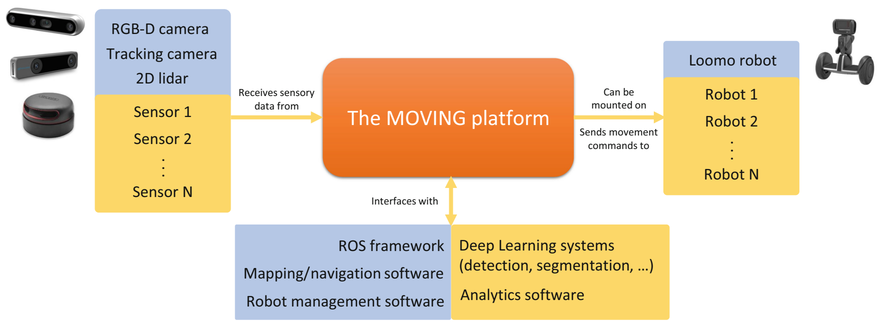

Fig. 2. Architecture of the MOVING platform. In blue we reported the configuration discussed in this paper, whereas in yellow we highlighted the possibility to personalize the platform. With the adoption of the ROS framework [20], MOVING can support a wide range of sensors and can work with classic reliable robotic software as well as custom applications. The platform can be easily installed on a large number of robots, which can be controlled by setting up a simple platform-robot communication interface.

- **RGB-D camera**: the camera acquires RGB images coupled with depth images and sends them to the visual SLAM software to create the environment map. During the navigation, the images can be used to localize the robotic agent within the already acquired map. In the setup discussed in this paper we use a Realsense D455 depth camera[3], which relies on a stereoscopic 3D vision technology to capture a sufficiently accurate depth;
- **Tracking camera**: the tracking camera enabels the robot to use the odometry to track itself in space over time. To avoid pose estimation drift, the system should run at a high frame rate and can integrate visual and inertial information to improve its accuracy. When the device is not available, the odometry can be estimated from RGB or RGB-D images at the expense of less reliability and precision, or from wheel encoders installed on the robot, which however would force to bind the platform to the specific robot. In our setup, we adopt a Realsense T265 tracking camera[4] which combines VO from two fisheye sensors with inertial sensors to compute an highly accurate odometry directly on device;
- **2D Lidar**: the 2D Lidar emits a beam of light (laser) while spinning to measure the distance of obstacles that intersect the scanned virtual horizontal plane. Depending on the model, the 2D Lidar can perceive only the front facing obstacles (180° perception) or all the obstacles around the sensor (360° perception). The perceived range can easily go beyond 10 m even in the case of affordable devices, providing an accurate measurement of the geometry of the environment. In the proposed platform, we adopt the RPLIDAR-A2M12 2D

[3] https://www.intelrealsense.com/depth-camera-d455/.
[4] https://www.intelrealsense.com/tracking-camera-t265/.

Lidar[5], which is capable of a 360° scan, has an angular resolution of 0.225°, a max range of 12 m and a frequency of 10 Hz;
- **Sensors mount**: the considered sensors are installed on a mount, specifically designed and 3D printed. The mount allows to position the tracking camera above the RGB-D camera and is extended by an additional support for the Lidar, which is positioned above the cameras and allows for tilt adjustment. The setup can be then hooked to the robot via a joint equipped at one end with a camera screw and at the other end with a clamp for immediate installation on any robot;
- **Processing unit**: the processing unit processes the incoming stream of data from the sensors, runs the mapping and navigation software and manages the communication between the robot and the external services. We have chosen to adopt a ZED Box[6] given its compact form factor and hardware characteristics. It is equipped with a 6-core processor and a Graphics Processing Unit (GPU) to run Deep Learning-based models and comes with Ubuntu OS for software compatibility. The ZED Box has been equipped with WiFi connectivity, a USB hub, and can be powered by a classic power bank equipped with a Power Delivery (PD) output, given its maximum consumption of 20 Wh.

The platform setup, complete with all hardware components, is shown in Fig. 1.

3.2 Software Architecture

The software system is based on the ROS framework [20]. ROS is an open-source project which consists of a set of software libraries and tools for building robotic applications. ROS offers the advantage of: 1) abstracting the robot platform in order to reuse the code; 2) using highly standardized and stable software libraries; 3) integrating custom software easily into the ecosystem; 4) choosing from numerous supported sensors which are easily to integrated in ROS; 5) move the computational load to a third party machine transparently, if the computational power of the embedded system is limited. The software components used are the following:

- **SLAM software**: RTAB-Map [12] is a robust RGB-D, stereo e Lidar graph-based SLAM approach, integrated in the ROS framework as a ROS package. Thanks to a memory management approach, the system supports the scan of large-scale environments while keeping its ability to work in real time. The clear advantage of using visual SLAM over Lidar-based SLAM is the availability of an image-based loop closure detector, which is able to correct the pose estimation drifts during mapping and navigation and provide a first guess on the initial robot's location. RTAB-Map also offers the possibility to estimate the odometry signal from the RGB-D or the Lidar data stream, when an odometry source is not available.

[5] https://www.slamtec.ai/home/rplidar_a2/.
[6] https://www.stereolabs.com/zed-box/.

- **Navigation software**: the ROS Navigation Stack offers a collection of software packages that can be used to implement a navigation system. Specifically, the move_base node [13] links together a global planner and a local planner which leverage the environment map to generate an optimal navigable path to the destination. The *move_base* node receives as input the sensors' observations, the map of the environment, the robot's position and the odometry source to output the actions to be performed in the form of linear and angular velocities;
- **Sensors ROS interfaces**: thanks to the broad adoption of the ROS framework in the robotic field, most sensor manufacturers provide the required ROS software packages to interface with the framework. The realsense cameras are supported by the *realsense2_camera* ROS package[7], while the RPLIDAR device is supported by the *rplidar* ROS package[8];
- **ROS-robot communication system**: to forward the actions provided by the *move_base* package, we implemented a socket-based communication system where a sender, integrated in the ROS framework, reads the action commands and sends them to the receiver via network connection.
- **Robot management software**: we developed a web-based, cross-platform robot management software, which allows the user to monitor the robot while it operates in the designated environment, track its movements, and specify new destinations to reach and receive updates on the status of the task. Moreover, given the environment map, an annotation tool can be used to store strategic points of interest inside it and associate them with labels for future use. The backend of the web-based software was developed in Python using the Flask library to build the webserver and the rospy library to interface with ROS, while the frontend was developed in HTML and Javascript.

4 Experimental Settings and Results

We validated the proposed platform while performing the mapping and navigation tasks in an environment, by assessing the quality of the resulting floor plans given the full sensor configuration. To show the contribution of each of the sensors used in our setup, we also performed an ablation study by employing a subset of the sensors (i.e. excluding the 2D lidar, the tracking camera or both of them).

4.1 Mapping of the Environment

To carry out the mapping, the proposed platform was mounted on a Segway Loomo robot[9], which received the action commands through the socket-based communication interface discussed in the Sect. 3.2. Given that Loomo comes equipped with the Android OS, the receiver program was developed in Java and

[7] http://wiki.ros.org/realsense2_camera.
[8] http://wiki.ros.org/rplidar.
[9] https://www.segway.com/loomo.

Table 1. List of the performed mapping sessions, considering different sensor configurations. We report if the task has been successfully completed, together with a short comment on the experiment.

Odometry source	**Lidar**	**Sucess**	**Notes**
Tracking camera	2D Lidar	✓	-
Tracking camera	-	✓	Artifacts in the map
Tracking camera	From depth image	✗	Continuous stop and in-place rotations
Lidar - ICP	2D Lidar	✓	-
Lidar - ICP	From depth image	✗	Camera FOV too narrow, unstable lidar signal
RGB-D	-	✓*	*Loss of tracking. Incomplete mapping

used the Loomo SDK to forward the linear and angular velocities to the robot base controller.

To perform experiments in a real scenario, we chose an area of our university department floor[10] that measures approximately 295 square meters. To scan the environment covering the whole area, a path formed by 16 key points has been provided. Points along the track were then sub-sampled at 1.5 m intervals and provided as goals to the autonomous navigation system.

The list of the performed mapping experiments is summarized in Table 1, which reports the sensor configurations used as well as if the mapping was successful or not. Using the complete settings, with both tracking camera and 2D Lidar (first row), the mapping procedure has been complete successfully. When the tracking camera has not been used, the odometry was estimated from the Lidar signal using the Iterative Closest Point (ICP) algorithm (4th and 5th rows) or from visual features extracted from the RGB-D images (last row). In two of the four configurations that do not include the 2D Lidar sensor (3rd and 5th rows), we extracted a Lidar-like signal from the camera's depth images using the *depthimage_to_laserscan* ROS package, which perceives a narrow portion of space and has a shorter perception range (capped to 5 m to avoid noisy measurements), but did not prove to be a useful input signal for the mapping system.

In general, it is possible to observe how four of the six configurations successfully accomplished the mapping task, whereas two experiments, which employed the lidar-like signal, were unsuccessful (3rd and 5th rows). In the first case (3rd

[10] Additional details omitted due to the anonymous submission.

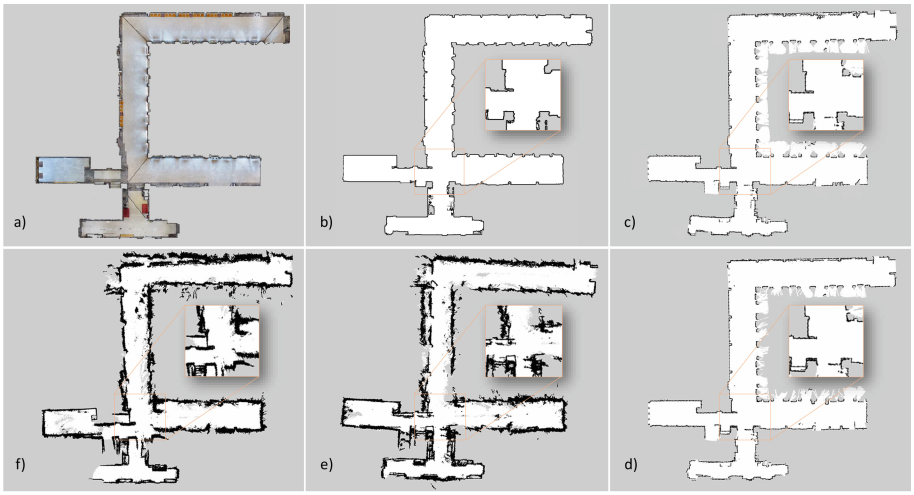

Fig. 3. Floor plans resulting from the mapping sessions performed using different sensor configurations. a) represents the environment acquired with a Matterport 3D scanner, which has been modified (b) in order to be immediately comparable with the maps scanned with the SLAM approach. c) has been acquired using the RGB-D camera+tracking camera+2D Lidar; d) using the RGB-D+ICP odometry+2D Lidar; e) represents the floor plan acquired using the RGB-D+tracking camera; f) represents the floor plan acquired using the RGB-D+RGB-D odometry.

row) the Lidar-like signal was used as a direct replacement of the Lidar for the map construction. In this configuration, the robot was not able to move smoothly, and the navigation was characterized by continuous stops and in-place rotations, probably caused by incorrect measurements. In the second case (5th row) the Lidar-like signal was used only for odometry estimation using the ICP algorithm. Unfortunately, in this configuration, the tracking was immediately lost with continuous odometry resets. In both configurations, it emerged that the Field Of View (FOV) and the perceptive range of the Lidar sensor are essential to benefit from this signal. Concerning the experiment carried out using the odometry estimated from RGB-D images (last row), the system lost tracking of the robot several times after observing textureless walls, resulting in an incomplete map. In fact, it is well known how visual systems based on feature tracking fail in the presence of featureless surfaces.

Figure 3 compares the maps of the environment resulting from the scan sessions performed with the different sensor configurations. For reference, we reported the floor plan of the same environment scanned using a Matterport 3D scanner (Fig. 3a), together with a simplified version (Fig. 3b) which represents a curated optimal mapping. The maps acquired using the RGB-D camera+tracking camera+2D lidar (Fig. 3c) and the RGB-D camera+ICP odometry+2D lidar (Fig. 3d) configurations are both accurate and detailed, presenting a slight distortion in the area of the top corridor. The odometry estimated from

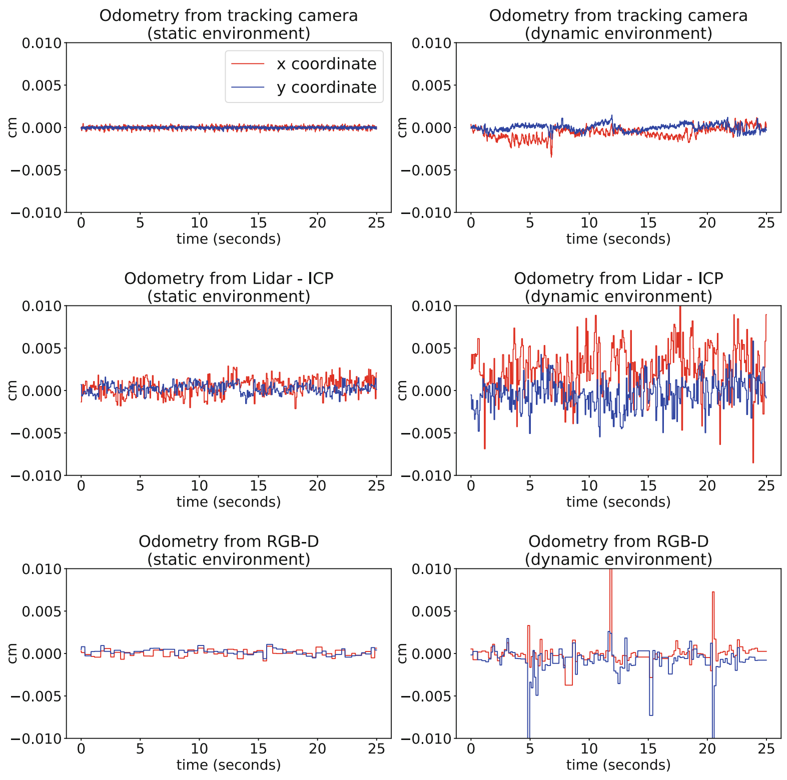

Fig. 4. Influence of the quality of the odometry on the robot tracking system. On the left, we tracked the robot's coordinates when placed in a static environment, for different odometry sources. On the right, the same measurements but in a dynamic environment with people moving around. Since the robot is stationary, the estimated coordinates should be both equal to zero. The chart shows the advantage of using a dedicated odometry device for a reliable localization in challenging scenarios.

the Lidar signal has proven to be reliable in a static scenario as the considered one, but it may exhibit its limitation in a dynamic environment, given the Lidar's perception of motion.

On the contrary, although the maps acquired without Lidar using the RGB-D camera+tracking camera (Fig. 3e) and the RGB-D camera+RGB-D odometry (Fig. 3f) configurations match the geometry of the environment, they have numerous artifacts (irregular and spectral walls), areas incorrectly mapped as obstacles, and unscanned areas. The tracking camera helped limit excessive distortions and misalignments (Fig. 3e), which can instead be observed from the highlighted portion of the map acquired using the odometry estimated from the RGB-D signal (Fig. 3f).

4.2 Odometry and Tracking

To better measure the accuracy and reliability of the odometry signals considered in our experiments, we conducted a further experiment in which the robot was placed in a fixed location of the environment and its position was recorded while observing a static scene (similar to that experienced during mapping) and a dynamic scene (with the presence of people in motion, to replicate a real dynamic environment). As can be observed in Fig. 4, the odometry provided by the tracking camera (first row) is significantly more reliable than the odometry estimated from Lidar (second row) or RGB-D (third row), allowing for a more stable and precise robot tracking even in dynamic environments. While the oscillations of the robot's position derived from the estimated odometry may seem small, it should be noted that, in a scenario where the robot moves around, these inaccuracies can accumulate quickly, resulting in incorrect tracking and localization.

Overall, the results showed how the adoption of different sensor configurations can lead to significantly different results and highlighted the benefits offered by a full sensor configuration.

5 Conclusion

We presented MOVING, a flexible and modular hardware and software platform for SLAM and visual navigation that can be easily installed on any robot requiring a minimal adaptation. The platform allows the reuse of robotic software by encouraging the design and test of navigation solutions based on the visual platform, hence reducing deployment time. The effectiveness of MOVING was verified in a real environment by performing different mapping sessions with several sensor configurations and investigating the reliability of the odometry to track the robot in static and dynamic environments.

Aknowledgment. This research is supported by Next Vision s.r.l. (Next Vision: https://www.nextvisionlab.it/) and by the project Future Artificial Intelligence Research (FAIR) - PNRR MUR Cod. PE0000013 - CUP: E63C22001940006.

References

1. Alkendi, Y., Seneviratne, L., Zweiri, Y.: State of the art in vision-based localization techniques for autonomous navigation systems. IEEE Access **9**, 76847–76874 (2021)
2. Anwar, S., Bascou, N.A., Menekse, M., Kardgar, A.: A systematic review of studies on educational robotics. J. Pre-Coll. Eng. Educ. Res. (J-PEER) **9**(2), 2 (2019)
3. Arvin, F., Espinosa, J., Bird, B., West, A., Watson, S., Lennox, B.: Mona: an affordable open-source mobile robot for education and research. J. Intell. Robot. Syst. **94**, 761–775 (2019)
4. Biswal, P., Mohanty, P.K.: Development of quadruped walking robots: a review. Ain Shams Eng. J. **12**(2), 2017–2031 (2021)

5. Chaplot, D.S., Gandhi, D., Gupta, S., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural slam. In: International Conference on Learning Representations (ICLR) (2020)
6. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. IEEE Robot. Autom. Mag. **13**(2), 99–110 (2006)
7. Feil-Seifer, D., Matarić, M.J.: Socially assistive robotics. IEEE Robot. Autom. Mag. **18**(1), 24–31 (2011)
8. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Robot. **23**(1), 34–46 (2007)
9. Hägele, M., Nilsson, K., Pires, J.N., Bischoff, R.: Industrial robotics. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 963–986. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-540-30301-5_43
10. Hirai, K., Hirose, M., Haikawa, Y., Takenaka, T.: The development of honda humanoid robot. In: Proceedings of 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146). IEEE (1998)
11. Kleeberger, K., Bormann, R., Kraus, W., Huber, M.F.: A survey on learning-based robotic grasping. Curr. Robot. Rep. **1**, 239–249 (2020)
12. Labbé, M., Michaud, F.: RTAB-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. J. Field Robot. **36**(2), 416–446 (2019)
13. Marder-Eppstein, E.: move_base - ros wiki (2016). http://wiki.ros.org/move_base
14. Mondada, F., et al.: Bringing robotics to formal education: the thymio open-source hardware robot. IEEE Robot. Autom. Mag. **24**(1), 77–85 (2017)
15. Montemerlo, M., Roy, N., Thrun, S.: Perspectives on standardization in mobile robot programming: the Carnegie Mellon navigation (carmen) toolkit. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453). IEEE (2003)
16. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
17. Muratore, L., Laurenzi, A., Hoffman, E.M., Rocchi, A., Caldwell, D.G., Tsagarakis, N.G.: Xbotcore: a real-time cross-robot software platform. In: 2017 First IEEE International Conference on Robotic Computing (IRC). IEEE (2017)
18. Möller, R., Furnari, A., Battiato, S., Härmä, A., Farinella, G.M.: A survey on human-aware robot navigation. Robot. Auton. Syst. (RAS) **145**, 103837 (2021)
19. Pérez-D'Arpino, C., Liu, C., Goebel, P., Martín-Martín, R., Savarese, S.: Robot navigation in constrained pedestrian environments using reinforcement learning (2020)
20. Quigley, M., et al.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, Kobe, Japan (2009)
21. Saeedvand, S., Jafari, M., Aghdasi, H.S., Baltes, J.: A comprehensive survey on humanoid robot development. Knowl. Eng. Rev. **34**, e20 (2019)
22. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)