Grant agreement No: 101017008

Harmony

Assistive robots for healthcare

Enhancing Healthcare with Assistive Robotic Mobile Manipulation

(HARMONY) | H2020-ICT-2018-20| RIA

Start of the project: 01.01.2021
Duration: 42 months

Deliverable Number	D10
Deliverable Name	Sensor suite first prototype
WP Number	3
Lead Beneficiary	ETHZ
Dissemination Level	Public
Internal Reviewer	BONN
Due Date	31.12.2021
Date of Submission	
Version	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008

Revision History

Version	Date	Author(s)	Comments
0.1	17.12.2021	Paula Wulkop, Francesco Milano, Lionel Ott, Jen Jen Chung	First draft
1.0	11.1.2022	Jens Behley, Cyrill Stachniss, Paula Wulkop, Francesco Milano, Lionel Ott, Jen Jen Chung	Internal review and revision

Table of Contents

Revision History **2**

Table of Contents **3**

Summary **4**

Acronyms **5**

Introduction **6**

Sensor Suite **6**

Sensors 6

Configuration 8

Computation 10

Evaluation 11

Mapping scenario - Reconstruction of hospital corridor 12

Manipulation scenario 15

Sensor Drivers and Calibration **19**

Drivers 19

Calibration 19

Conclusion **21**

References **22**

Summary

In the following document, we introduce the first prototype of a sensor suite that will be used in the Harmony project. Building on the proposal from Deliverable D3.1, the configuration consists of two Azure Kinect RGB-D sensors, an Intel Realsense T265 tracking camera, two SICK 2-D LiDARs, three FLIR Firefly RGB cameras, and a ZOTAC Magnus computer for data processing; optional peripheral sensors proposed by the partners will be installed if required. Additionally, we present a preliminary evaluation of the sensor suite based on recorded test data. Finally, we provide information about the software installation and sensor calibration which is required to reproduce the proposed sensor setup.

Acronyms

CAD	Computer-Aided Design
DOF	Degrees Of Freedom
FOV	Field Of View
FPS	Frames Per Second
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
RAM	Random Access Memory
ROS	Robot Operating System
SDK	Software Development Kit
SLAM	Simultaneous Localization And Mapping
SSD	Solid State Drive
URDF	Unified Robot Description Format
V-SLAM	Visual SLAM
WP	Work Package

ABB	ABB AB
BONN	Rheinische Friedrich-Wilhelms-Universität Bonn
CREATE	Consorzio di Ricerca per l'Energia, l'Automazione e le Tecnologie dell'Elettromagnetismo
ETHZ	Eidgenössische Technische Hochschule Zürich
IDM	IDMind
KUH	Karolinska University Hospital
TUD	Delft University of Technology
UEDIN	University of Edinburgh
USZ	Universitätsspital Zürich
UT	University of Twente

1. Introduction

To operate autonomously and safely in dynamic, human-populated environments, the Harmony robot requires a rich variety of sensor inputs. The required core sensors were identified in Deliverable D3.1 and consist of two Azure Kinect RGB-D sensors, an Intel Realsense T265 tracking camera, two SICK 2-D LiDARs. Additionally, during internal discussions among the partners we identified the need to provide additional sensor modalities to perform 360° tracking of humans and vehicles moving around the objects. For this purpose, three FLIR Firefly RGB cameras are also included in the proposed setup. Further modalities for safety ("cliff detector") and manipulation ("wrist-mounted camera") will be individually integrated at a later stage by the partners upon necessity. Streaming and processing of the sensor data are performed on a ZOTAC Magnus computer. We conducted preliminary evaluation of the sensor suite by analysing the reconstruction accuracy of the RGB-D sensor as well as the tracking accuracy of the T265. We tested the concurrent usage of all the sensors from the suite for data streaming and recording and verified that the proposed setup allows handling the data bandwidth at the desired frame rate. Finally, we open-sourced to all project partners the software required for running and calibrating the sensors and provided instructions to reproduce our sensor setup.

2. Sensor Suite

This section presents the proposed Harmony sensor setup. First, the sensor specifications and their proposed configuration on the robot are described. Second, the hardware required for computation is presented. Finally, an evaluation of the entire setup is provided.

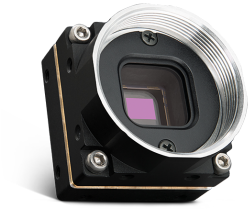
2.1. Sensors

The sensing modalities were selected based on the requirements identified in Deliverable D3.1. Internal discussions among the partners highlighted that the 2-D LiDARs might not provide a sufficiently robust solution for detection and tracking of people and objects moving around the robot. Therefore, three additional RGB camera sensors are included in the setup. Table 1 shows a summary of all the proposed core sensors, consisting of RGB-D and RGB cameras for scene perception, as well as 2-D LIDARs, wheel odometry and a tracking camera for mapping and navigation.

Table 1: Harmony core sensor suite.

Sensor	Modality	Description
Intel® RealSense™ Tracking Camera T265 (1x) 	Visual-inertial odometry	- **Algorithm:** Runs V-SLAM to provide the 6DOF pose - **Interface**: USB - **Cost**: $200 - **Note**: The product will be discontinued in February 2022 ([1]). The partners agreed that the sensor should still be included, since the partners who will rely on it have sufficient availability.

Microsoft Azure Kinect (2x)
 
	RGB-D	- **Sensor**: RGB
		  - **FOV**: 90°x60°
		  - **Resolution**: 1280x720px
		- **Sensor**: Depth
		  - **FOV (wide/narrow)**: 120°x120° / 75°x65°
		  - **Resolution**: 1024x1024px / 640x576px
		  - **Range (wide FOV/narrow FOV)**: 0.25 - 2.21m / 0.5 - 3.86 m
		- **FPS**: 5 / 15 / 30 Hz
		- **Interface:** USB, Power, Optional synchronization cable between cameras
		- **Cost**: $400/camera

FLIR Firefly S
FFY-U3-16S2C-CS (3x)
 
	RGB	- **Resolution**: 1.6 MP
		- **Maximum frame rate:** 60 FPS
		- **Sensor**: Sony IMX296, Color
		- **Interface**: USB
		- **Lenses:**
		  - **Model**: ZWO lens [2]
		  - **Sensor size**: 1/3”
		  - **Focal length**: 2.1mm
		  - **FOV**: 150 degrees
		  - **Cost**: $22/lens

SICK TIM781S-2174104 (2x) 	2-D LiDAR	- **Technology**: Infrared 2-D LiDAR scanning at 15 Hz - **Coverage**: 270°/LiDAR; combine two for 360° coverage - **Range**: 5cm - 25m - **Interface**: USB, Ethernet, Power - **Cost**: $2500/sensor
*Platform specific	Wheel odometry	Sensor integrated in the mobile platform

Aside from the core sensors listed above, additional sensing modalities have been proposed by individual partners. These peripheral sensors, as described in Tab. 2, can be modularly added to the platform as required.

Table 2: Harmony peripheral sensors.

Sensor	Modality	Description
In-hand camera[1]	RGB(-D)	To provide multiple viewpoints
Force/torque sensor at end effector[2]	Tactile/force	To improve manipulation
Microphone[3]	Audio	To interact with human operators. The Microsoft Azure Kinect sensors also feature microphone arrays, which could be used for this purpose.
Sonar	"Cliff-detection"	To detect stairs

[1] proposed by ETHZ/UEDIN
[2] proposed by CREATE
[3] proposed by UT

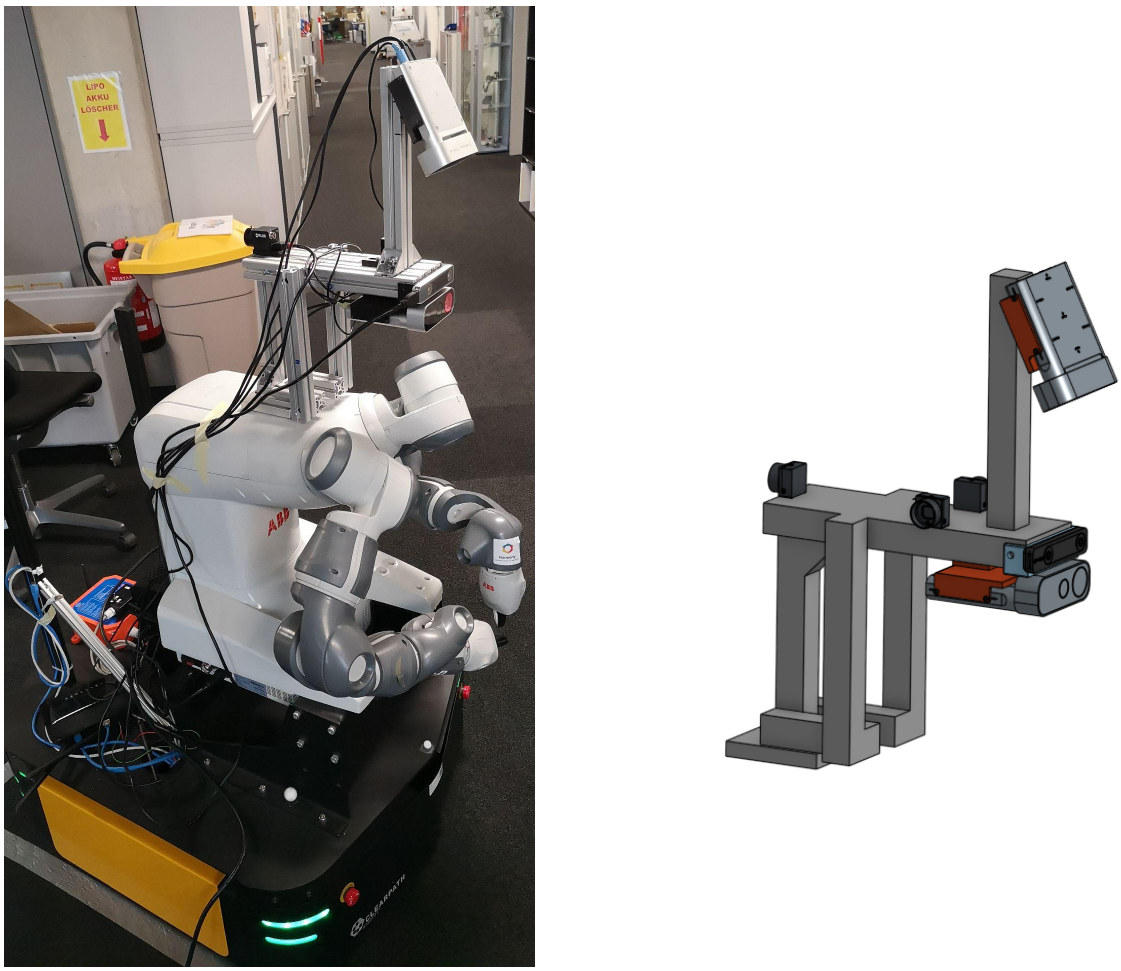
2.2. Configuration

The placement of the sensors on the robotic platform was evaluated so that the requirements of field of view are fulfilled for all the use cases. The final sensor setup consists of one downward-facing Azure Kinect for manipulation and one front-facing Azure Kinect for mapping. Example frames showing the field of view of the two RGB-D cameras are presented in Fig. 1 . Additionally, three RGB cameras provide a 360° view around the robot. For localization, the T265 is mounted facing forward at a height of 1.25m; for specific use cases, if required by partners, it could also be mounted at a lower height. The 2-D LiDARs are integrated into the mobile platform, one in the front and one in the back, at a height of 25cm. For preliminary tests and integration until the final Harmony robot platform becomes available, the sensor setup was mounted on a platform available at ETHZ. This consists of a Ridgeback Clearpath omnidirectional mobile platform and an ABB Yumi dual-arm manipulator, as shown in Fig. 2. To mount the sensors on this platform, we designed and built a frame made of aluminium profiles and 3-D printed parts. All relevant CAD files to reproduce this setup can be found at [3].

Due to supplier delivery delays, only one FLIR RGB camera is currently mounted on the ETHZ setup, as seen in Fig. 2a. Nevertheless, the setup was fully tested and evaluated using additional, borrowed cameras of the same type.

(a) Manipulation view *(b) Mapping view*

Figure 1: Example views from the Azure Kinect cameras.



(a) Picture of ETHZ robot setup *(b) CAD model of sensor mount*

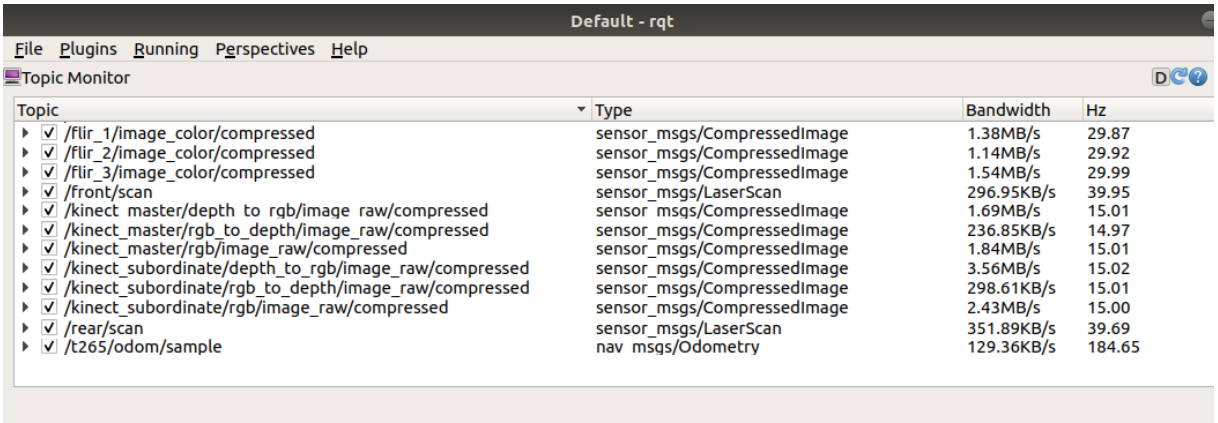
Figure 2: Proposed sensor setup, mounted on the ETHZ robot and displayed as a CAD model.

2.3. Computation

To extract and process the data from the sensors described above, we propose to use the ZOTAC Magnus One Computer (cf. Fig. 3).

Figure 3: ZOTAC Magnus One ECM73070C. Source: [4].

The device is not equipped with an SSD or with RAM, therefore these were separately ordered. The models purchased are shown in Tab. 3. To test the proposed setup, all sensors were plugged into the computer and a rosbag containing all the relevant ROS topics was recorded. Figure 4 shows that the desired frame rates for all topics could be achieved (Kinect: 15 Hz, RGB: 30 Hz, LIDAR: 40 Hz). Additionally, the power consumption of the computer was measured during this test and found to range between 170 and 190 W. This is the power required to stream and record all the sensor data, but it does not include any additional data operation which will be performed on top, e.g., processing of the data in a deep-learning pipeline.



Topic	Type	Bandwidth	Hz
/flir_1/image_color/compressed	sensor_msgs/CompressedImage	1.38MB/s	29.87
/flir_2/image_color/compressed	sensor_msgs/CompressedImage	1.14MB/s	29.92
/flir_3/image_color/compressed	sensor_msgs/CompressedImage	1.54MB/s	29.99
/front/scan	sensor_msgs/LaserScan	296.95KB/s	39.95
/kinect_master/depth_to_rgb/image_raw/compressed	sensor_msgs/CompressedImage	1.69MB/s	15.01
/kinect_master/rgb_to_depth/image_raw/compressed	sensor_msgs/CompressedImage	236.85KB/s	14.97
/kinect_master/rgb/image_raw/compressed	sensor_msgs/CompressedImage	1.84MB/s	15.01
/kinect_subordinate/depth_to_rgb/image_raw/compressed	sensor_msgs/CompressedImage	3.56MB/s	15.02
/kinect_subordinate/rgb_to_depth/image_raw/compressed	sensor_msgs/CompressedImage	298.61KB/s	15.01
/kinect_subordinate/rgb/image_raw/compressed	sensor_msgs/CompressedImage	2.43MB/s	15.00
/rear/scan	sensor_msgs/LaserScan	351.89KB/s	39.69
/t265/odom/sample	nav_msgs/Odometry	129.36KB/s	184.65

Figure 4: Screenshot with an overview of the main ROS topics streamed by the sensors in the suite, along with the associated frame rate and bandwidth. The sensors were running concurrently and being processed on the ZOTAC Magnus computer.

Table 3: Proposed computing hardware.

ZOTAC Magnus One ECM73070C	
Processor type	i7-10700, 8 core
Clock frequency	2.90 GHz
GPU	NVIDIA GeForce RTX 3070
Price	CHF 1581 [4]
Max. Power	500 W
Ports	4x USB 3.1 Type A 3x USB 3.0 Type A 1x USB 3.2 Type C 2x HDMI 3x DisplayPort
RAM: Crucial 32GB DDR4-2666 UDIMM	
Memory	32 GB
Price	€145.13 [5]
SSD: Crucial P5 Plus 1TB PCIe M.2 2280SS	
Capacity	1 TB
Price	€160.50 [6]

2.4. Evaluation

In the following section, we present qualitative and quantitative evaluations of the proposed sensor setup. As described in Deliverable D3.1, the sensors — combined with appropriate software solutions — should provide a high level of reconstruction accuracy, required to carry out the tasks specified in the other work packages, e.g., manipulation, and mapping. We captured data using the sensor mount described in §2.2, and recorded in all the evaluations the RGB and depth images from a single Azure Kinect and the pose estimates from the Realsense T265.

2.4.1. Mapping scenario - Reconstruction of hospital corridor

During the on-site visit at USZ, we recorded a dataset in one of the underground corridors of the hospital. The sensor mount was hand-held, and the front-facing Azure Kinect (mapping camera, cf. §2.2) was used to extract RGB-D information. In a post-processing stage, we integrated the per-frame point clouds using the open-source volumetric mapping framework Voxblox [7] and the poses extracted from the Realsense T265. Voxblox allows setting different sizes (resolution) for the voxels used to build a reconstruction of the environment, and can further perform ICP registration to improve the alignment of the integrated measurements.

Figures 5-7 show the result of the reconstruction in the following two settings: (a) 5cm voxel resolution, without using ICP; (b) 2cm voxel resolution, using ICP. Overall, in both settings the combination of sensors achieve a detailed reconstruction of the scene appearance and geometry, capturing even fine texture details when a smaller voxel size is used (cf. Fig. 6) while being robust to the low-light conditions of the underground corridor.

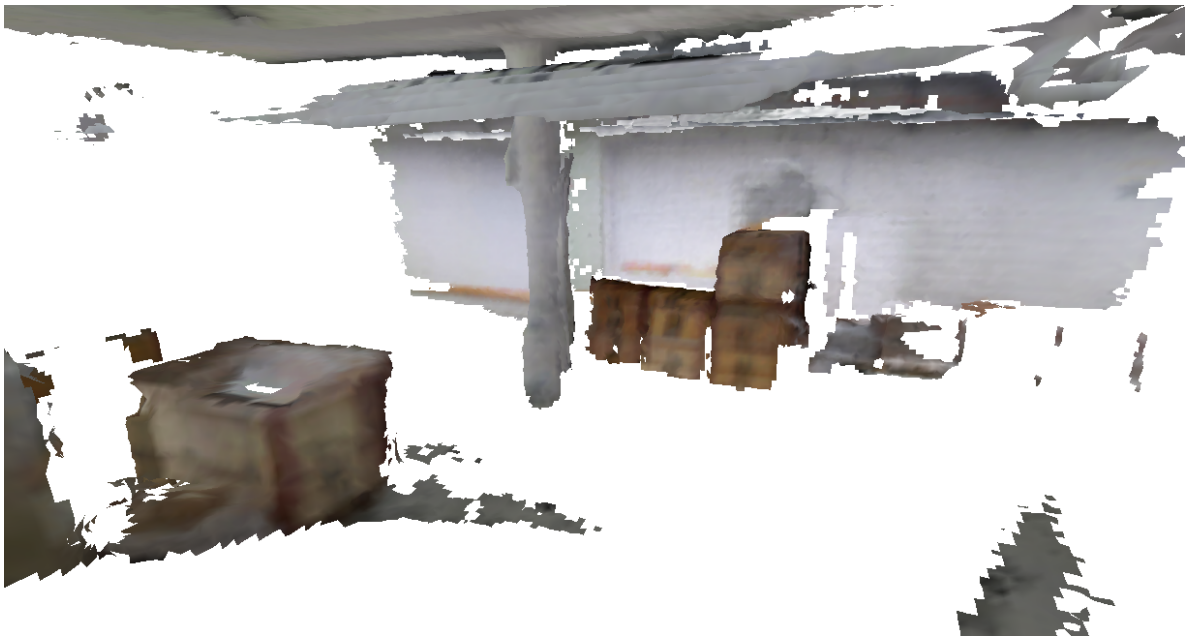
As highlighted by Fig. 7, open-loop usage of the pose estimated by the Realsense T265 without global optimization yields odometry drifts, visible in the curved shape of the corridor; this is in accordance with the results shown by our early evaluation of the tracking accuracy of the Realsense T265, which we reported in Deliverable D3.1. These limitations can partially be addressed using software techniques, e.g., via ICP registration. Furthermore, while a dense mapping setting as the one shown in this evaluation is outside the scope of the project, future deliverables (e.g. Deliverable D4.1) will focus on the development of a mapping and localization framework; in this framework, the measurements from the Realsense T265 will be used as prior estimates, and the final estimate of the pose will be computed through more robust algorithms.

Overall, we conclude that the proposed sensor setup is a suitable solution for the mapping scenario and allows achieving the desired requirements in terms of reconstruction.

(a) 5cm, without ICP

(b) 2cm, with ICP

Figure 5: Evaluation of the reconstruction of the hospital corridor using Voxblox - View from the start of the corridor.



(a) 5cm, without ICP

(b) 2cm, with ICP

Figure 6: Evaluation of the reconstruction of the hospital corridor using Voxblox - View from the center of the scene. A finer voxel resolution allows capturing fine details of the texture, as shown, e.g., by the boxes and by the clock on the background wall in (b).

Figure 7: Evaluation of the reconstruction of the hospital corridor using Voxblox - Top view of the corridor. Different colors denote the two different settings. Orange: 2cm voxel size, using ICP. Blue: 5cm voxel size, without using ICP.

2.4.2. Manipulation scenario

To evaluate the accuracy of the Azure Kinect, we scanned multiple objects and evaluated the resulting point clouds. Figure 8b shows the point cloud obtained from a single frame of the manipulation camera facing a hospital box. It can be observed that while the geometry of the lid is well reconstructed, the side walls of the box are slightly contracted, such that the "*citytrans*" writing on the side wall appears to be lower than its actual position on the reconstructed object. Another issue that becomes evident in Fig. 8b is the transfer of the color of a foreground object (box) to the background object (table), also known as *color spill* [8]. This is caused by the fact that the Azure Kinect consists of two physically separated cameras (one RGB camera and one depth camera); for this reason, in order to obtain a 3-D point cloud the depth image needs to be registered into the frame of the RGB camera or vice versa. This transformation leads to interpolation, which can result in artifacts — particularly at points where the depth map has a discontinuity — if the calibration is not perfectly accurate. In the case of the Azure Kinect, a factory calibration is provided to perform the transformation between the RGB and the depth camera frames, but different users suggest that a manual calibration could be required [9]; additionally, incompatibilities between the provided intrinsics distortion model and the distortion models supported by the ROS drivers give rise to further approximations [10]. To alleviate this issue, a manual calibration of the RGB and depth cameras of the Kinect could be performed, and an alternative interpolation scheme, discarding values at the discontinuities of the depth map, could be implemented. A more thorough investigation of this problem will be carried out in case this issue was deemed to be a cause for errors in the perception algorithms that will be developed.

(a) RGB frame *(b) Point cloud*

Figure 8: Single-view recording of the hospital box. Color spilling and perspective shift can be observed in the right picture (e.g., "citytrans" writing).

Figure 9 shows recordings of the same hospital box, but with accumulated point clouds from multiple viewpoints. One frame every 6.7s was extracted from a video recording and these frames were merged into one point cloud using the T265 tracking camera for pose estimation. Figure 9a shows the accumulated point cloud obtained from a camera mounted on the robot, while the point cloud in Fig. 9b was recorded with a hand-held camera. It is clearly visible that the reconstruction obtained by the hand-held camera is more accurate. In particular, the setup with a robot-mounted camera gives rise to perspective distortion effects, caused by the fixed angle from which the camera observes the scene; on the other side, in the hand-held setup the side walls of the hospital box are more faithfully reconstructed, due to the fact the object could be observed from different heights and angles. Should the perception algorithms developed for the upcoming deliverables require more flexible view points, the option of mounting an additional camera on the robot arm could be evaluated.

It should also be noted that due to slight inaccuracies of the T265 pose, overlapping frames can appear shifted in the accumulated view (Fig. 9a). To mitigate this issue, the T265 pose could be refined — for example using ICP registration between consecutive frames — or more robust solutions for pose estimation could be developed if needed.

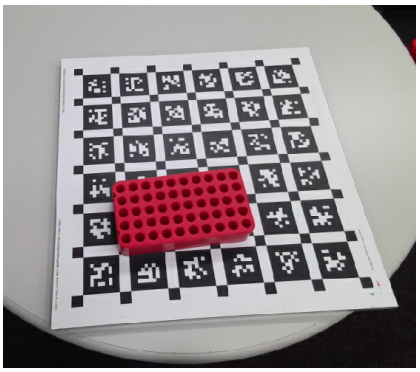
(a) Robot-mounted camera

(b) Hand-held camera

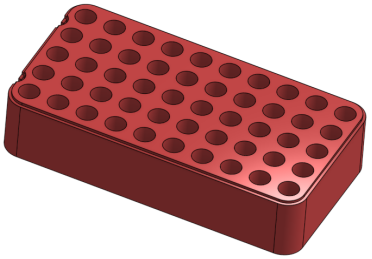
Figure 9: Point clouds of the hospital box with accumulated viewpoints 6.7s apart using the T265 pose.

A second evaluation was performed using a test tube rack provided by KUH (cf. Fig. 10). Figure 10c shows that the point cloud obtained from the manipulation camera substantially differs from the ground truth CAD model shown in Fig. 10b. The object height was strongly underestimated by the Azure Kinect, with a ground truth height of 4.5 cm and a point cloud object height of approximately 2.4 - 3.0 cm. This could be caused by a combination of several issues: First, for the depth estimated by the sensor, Microsoft reports a random error with standard deviation lower or equal than 17 mm and a typical systematic error lower than 11 mm + 0.1% of distance to the surface [11]. Second, from the chosen view point the object can mostly be observed from the top, therefore the side of the test tube rack is not well

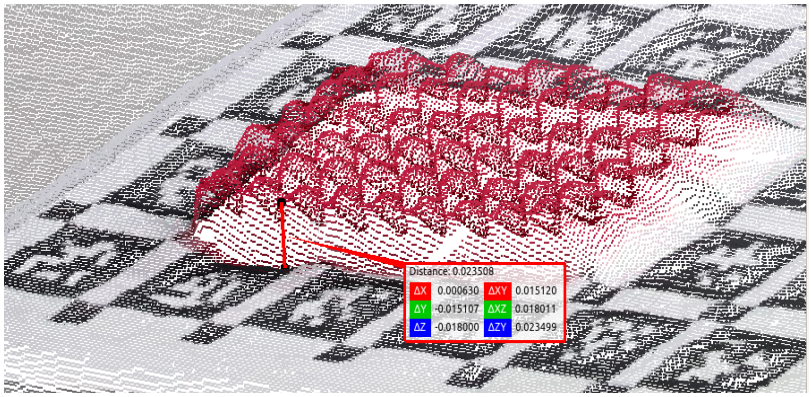
reconstructed, similarly to the hospital box recording in Fig. 8. Finally, the geometric shape of the object with the hole pattern might cause the rays to bounce and scatter, therefore deteriorating the quality of the measurements. To alleviate the issue of inaccurate object reconstruction, we propose to take advantage of the object database described in Task 3.3. This database will consist of known objects that are commonly found in the hospital use cases and are scanned offline; during operation, the pre-scanned objects could be matched to the detected objects to improve the reconstruction. Algorithmic solutions based on this idea will be implemented as part of the upcoming deliverables of WP3.



(a) Azure Kinect RGB frame



(b) CAD Model; object height: 4.5cm



(c) Azure Kinect point cloud; measured object height: 2.4 - 3.0 cm

Figure 10: Azure Kinect point cloud of test tube rack from single frame.

3. Sensor Drivers and Calibration

3.1. Drivers

In the following, we describe the software setup on the ZOTAC computer.

The machine runs Ubuntu 18.04 as an operating system and uses ROS Melodic for data transmission. This choice was made due to the unavailability of some sensor drivers for ROS2: While the T265 already provides a ROS2 driver, the Azure Kinect ROS Foxy branch is still under development and no FLIR ROS2 driver is available yet. Once all the necessary drivers become available, we will migrate the system to ROS2. The drivers, the SDKs, and the ROS interfaces provided by the manufacturers for the Azure Kinect [12, 13], the T265 [14] and the FLIR cameras [15] have to be installed in order to use the sensors. We provide adapted versions of this software, as well as instructions on how to install and use them in the code repository that can be found at [16-19].

3.2. Calibration

The calibration of the sensors in the suite is performed using the open-source software Kalibr [20]. We developed a software interface which wraps around Kalibr and launches the dedicated software drivers. In the following, we provide a summary of the main calibration procedures that we carried out; a more detailed description, along with the instructions to reproduce the process, can be found in the repository containing the software interface, which we make available at [21].

After rigidly mounting the sensors on the sensor rig as described in §2.2, we perform the following extrinsic calibrations:

- Calibration of the (fisheye) cameras of the Realsense T265 to the IMUs of the two Azure Kinect. Alternatively, calibration between the RGB cameras of the two Azure Kinect and the IMU of the Realsense T265 could be performed. However, we found the latter choice to yield suboptimal results; this was due to both the distortion model from the RGB camera of the Azure Kinect and the data type of the IMU provided by the Realsense T265 ROS interface not being fully supported by Kalibr (cf. [22-24]).
  To perform the calibration, the following steps are executed: intrinsic calibration of the T265 (fisheye) cameras, extrinsic calibration from the cameras of the T265 to the IMUs of the two Azure Kinect.
- Calibration of the FLIR RGB cameras to the IMUs of the two Azure Kinect. Due to a delay in the delivery due to world-wide availability issues (cf. §2.2), we perform the calibration on the single FLIR camera that we have currently available for the Harmony Project; the steps remain valid for the additional FLIR cameras that will be delivered at a later time.
  To carry out the calibration, the following steps are executed: intrinsic calibration of

the FLIR camera, extrinsic calibration of the FLIR camera to the IMUs of the two Azure Kinect.

When the sensor rig is mounted on top of the YUMI manipulator fixed on the Ridgeback platform, the transformation between the base frame of the robot and the frame at the base of the sensor rig is retrieved using CAD models provided by the manufacturers. The URDF files to visualize the entire ETHZ robot setup, including the calibrated position of all sensors, is open-sourced on the Harmony Github at [25].

4. Conclusion

In conclusion, we presented the first prototype of the sensor suite for the Harmony project and evaluated the data that can be obtained under different settings with these sensors. While there exists a gap — due to intrinsic hardware limitations — between the desired reconstruction accuracy and the one provided by the sensors, overall we found the proposed setup to provide a suitable solution for addressing the requirements both in the mapping and in the manipulation settings identified in Deliverable D1.1. The gap in the reconstruction accuracy will be addressed through algorithms and solutions which will be provided in the upcoming Deliverables D3.4 and D3.5. The accuracy of the pose estimates returned by the T265 is sufficient and will be further improved by integrating them with the localization pipeline (see Task 4.1).

Finally, we presented and open-sourced code and CAD files to allow all the partners to start using the sensors and replicate the proposed setup.

References

[1] “Intel® RealSense™ Technology focus on Stereo Depth” - Copy from Internet Archive Wayback Machine http://web.archive.org/web/20211109235116/https://www.intelrealsense.com/message-to-customers/. Accessed: 10.12.2021.
[2] ZWO 1/3″ 2.1mm 150 degree lens: https://astronomy-imaging-camera.com/product/zwo-13-2-1mm-150-degree-lens. Accessed: 14.12.2021.
[3] "CAD models sensors mount" in “Project Harmony -> Work Packages -> WP3 -> Data: https://drive.google.com/drive/folders/1RkSECKpsZ_J4O2YcC2Ctlv07cXbBI0nw?usp=sharing. Accessed: 14.12.2021.
[4] ZOTAC Magnus One ECM73070C, on digitec.ch. https://www.digitec.ch/en/s1/product/zotac-magnus-one-ecm73070c-intel-core-i7-10700-2-x-hdmi-3-x-usb-30-type-a-barebones-14643021?gclid=CjwKCAjwxo6IBhBKEiwAXSYBsxrojBM6Lmsffy8m-VjXQ1ggc6JcWGnzkgclkJ7Nhcx7s8SS-zSKMxoCRoIQAvD_BwE&gclsrc=aw.ds. Accessed: 14.12.2021.
[5] Crucial 32GB DDR4-2666 UDIMM: https://eu.crucial.com/memory/ddr4/ct32g4dfd8266/ct19894277. Accessed: 14.12.2021.
[6] Crucial P5 Plus 1TB PCIe M.2 2280SS SSD: https://eu.crucial.com/ssd/p5-plus/ct1000p5pssd8/ct20119973. Accessed: 14.12.2021.
[7] Oleynikova, Helen and Taylor, Zachary and Fehr, Marius and Siegwart, Roland and Nieto, Juan, “Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning”, in IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 2017.
[8] Azure Kinect ROS Driver, “Object Boundary "Leak" in RGB-D Image and PointCloud”: https://github.com/microsoft/Azure_Kinect_ROS_Driver/issues/63#issuecomment-524957010. Accessed: 15.12.2021.
[9] Azure Kinect ROS Driver, “camera intrinsics calibration”: https://github.com/microsoft/Azure_Kinect_ROS_Driver/pull/200. Accessed: 15.12.2021.
[10] Azure Kinect ROS Driver, “Issue with Camera Extrinsic Calibration”: https://github.com/microsoft/Azure_Kinect_ROS_Driver/issues/83#issuecomment-651301454. Accessed: 15.12.2021.
[11] Azure Kinect DK hardware specifications - Depth camera supported operating modes: https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification#depth-camera-supported-operating-modes. Accessed: 15.12.2021.
[12] Azure Kinect ROS Driver: https://github.com/microsoft/Azure_Kinect_ROS_Driver. Accessed: 14.12.2021.
[13] Azure Kinect SDK: https://github.com/microsoft/Azure-Kinect-Sensor-SDK. Accessed: 15.12.2021.
[14] T265 ROS Driver: https://github.com/IntelRealSense/realsense-ros. Accessed: 14.12.2021.

[15] FLIR ROS Driver:
https://github.com/ros-drivers/flir_camera_driver. Accessed: 14.12.2021.
[16] Github repository for the Realsense T265 drivers software:
https://github.com/harmony-eu/t265_tools.
[17] Github repository containing wrappers around Azure Kinect ROS interface:
https://github.com/harmony-eu/azure_setup.
[18] Github repository containing an adapted version of the FLIR drivers and ROS interfaces:
https://github.com/harmony-eu/flir_camera_driver.
[19] Github repository containing an adapted version of the Azure Kinect ROS interface:
https://github.com/harmony-eu/Azure_Kinect_ROS_Driver.
[20] Kalibr: https://github.com/ethz-asl/kalibr. Accessed: 14.12.2021.
[21] Github repository for the Harmony Sensor calibration:
https://github.com/harmony-eu/harmony_sensor_calibration.
[22] Distortion models supported by Kalibr:
https://github.com/ethz-asl/kalibr/wiki/supported-models#distortion-models. Accessed: 14.12.2021.
[23] Intrinsics parameters of the Azure Kinect:
https://microsoft.github.io/Azure-Kinect-Sensor-SDK/master/structk4a__calibration__intrinsic__parameters__t_1_1__param.html. Accessed: 14.12.2021.
[24] Launch parameters in ROS driver for Realsense T265 (cf. `unite_imu_method`):
https://github.com/IntelRealSense/realsense-ros#launch-parameters. Accessed: 14.12.2021.
[25] Github repository for the Harmony sensor description:
https://github.com/harmony-eu/harmony_sensor_description.