EU Horizon 2020 Research & Innovation Program
Advanced Robot Capabilities & Take-Up
ICT-25-2016-2017
Grant Agreement No. 779942

Safe Robot Navigation in Dense Crowds

http://www.crowdbot.org

Technical Report

D 5.2: First Basic Integrated Robot System Prototype in Experimental Scenario

Work Package 5 (WP 5)
System Architecture & Integration

Task Lead: SoftBank Robotics Europe, France
WP Lead: SoftBank Robotics Europe, France

Table of Contents

Executive Summary

Work package 5 (WP5) is about developing about a coherent theoretical and functional system architecture that can accommodate the targeted scenarios and facilitate the integration of the different work packages across all three robotic platforms. However, this process must be iterative and reviewable after each milestone so that the achievement of such objectives is guaranteed during all project development. In order to reach the previous, coordinating periodic integration activities among partners, updating the platforms and incorporating new developments is a must.

Specifically, we want to ensure that we are incorporating all project experiment needs, both in the robot architecture model and the hardware. At the same time, we must ensure the quality of individual components as well as its robustness is enough for posterior commercial and academic purposes. Then, ensuring that the low level sensor data from the robot and/or additional sensors is correctly managed so that high level situation assessment becomes possible.

The focus of this deliverable is to propose a flexible architecture that can be easily adapted for the three robot platforms that are involved in the CROWDBOT project. However, this document will provide further information in terms of hardware and software specifications limited only to the robotic platform Pepper [8].

Therefore, the present document extends the already delivered *D51 System architecture* document in M4, so that communication through messages exchanges between the different modules are specified in section 3. Then, a deeper view of the system components as well as their function is provided in section 4. Moreover, the first prototype of Pepper with additional sensors and augmented computational power in order to cover the experimental scenarios delivered in *D11 Specification of scenarios requirements* is described in section 6 and 7. As a consequence, several experimental procedures and results are presented in order to back up the decisions in terms of design and architecture, in section 5.

1. Introduction

The software architecture of a robot platform is a critical part to guarantee the correct behavior of the system and a smooth interconnection of messages between modules. In fact, ensuring the right choice, will allow the platform to be modular so that addition, deletion, substitution or expansion of the different components of the system should not require a redesign of the architecture. More specifically, the benefits of designing and implementing a suitable architecture include, but are not limited to:

- Provide flexibility and adaptability in different scenarios.
- Allow for interoperability with other partners in the project.
- Provide leverage of control in a project.
- Help developers to abstract and focus on a specific module within the project.
- Help reduce maintenance cost.
- Assist in task organization and with project oversight and control.
- Establish a common partnership vocabulary.

- Shorten learning curve of the system functionality.

With this objectives mind, the high level architecture base line is presented in section 3 as well as a more detailed proposal in section 4 which sets the functionality as the abstraction criteria to show the connection between modules.

1.1 Research Strategies

Based on the deliverable D11, *Specification of scenarios requirements*, a common strategy for sensor, perception and actuation needs to be proposed so that the platform can maximize the success in the experimental scenarios. In order to do that, limitations in the current platform need to be identified and mitigated in the short term, if possible. If not, a midterm solution is proposed to be developed in order to address it.

Regarding the scientific research lines that are intended to be explored during the project, those can be summarized in the following areas:

Crowd sensing. Pepper is intended to detect people in the close surroundings at the same time it estimates the dense scene flow so that adjusts its motion accordingly. More specifically, the system should: 1) detect people. 2) estimate dense scene flow. 3) generate segmented depth map.

Prediction of crowd motion in short-term. As an extended outcome, in later stages, the platform could anticipate the movement of the tracked crowd and adapt its trajectory.

Safe and efficient navigation on the crowd. Enable navigation, reactive planning and local motion planners based on pedestrian detection and tracking (WP2). In this case, machine learning-based approaches will be investigated within this topic.

Mapping and localization. Closely related with navigation, this area present a big challenge due to the nature of some of the scenarios presented in D1.1 where partial occlusions cover the robot's field of view.

Robot motion control in crowd. Ensure that potential contacts between pedestrians and the platform are safe and compliant for both. The robot should react locally to external forces in a way that preserves stability at the same time that contact against its surface is minimized.

Safety, ethical and legal recommendations for robot navigation. Codesign and evaluate the acceptable and expected robot behavior when operating in public spaces and setting up a theoretical frame.

1.2 Deliverable Outcomes

In this work we present a summary of the outcomes delivered during the research study conducted on the platform in terms of the adequacy of sensors, perception, actuators and computational strategies in order to successfully accomplish the experimental scenarios proposed in D1.1. Specifically, we will focus in the following topics:

High level architecture conceptualization. An extended version of the architecture diagram presented in the deliverable D51 is provided. A deeper level of granularity is achieved by defining the exchanged messages between the different modules.

Libraries and frameworks. The different resources already available such as NAOqi, ROS and its basic packages, that allow to define a baseline with a higher level of abstraction, are presented. In addition, graphical support is provided to better understand the functionality and relation between each other.

Pepper sensor performance. The performance of the different sensors and actuators is tested in order to be able to fulfill the high demanding expectations of a real-time pedestrian tracking, navigation in cluttered environments as well as reactive obstacle avoidance.

Mitigation approaches. Here we present the different approaches that has been used to overcome some of the limited results provided in the previous outcome. Such solutions include good practices, minor software fixes, as well as a sensor augmentation proposal.

Augmented computation solution for Pepper. Due to the increasingly demand of computational power of real-time image processing algorithms for analysis and tracking, navigation approaches and fusion of different sensors, an alternative computation solution is presented. Such upgrade consists of an additional computer attached at the back of the platform as a backpack.

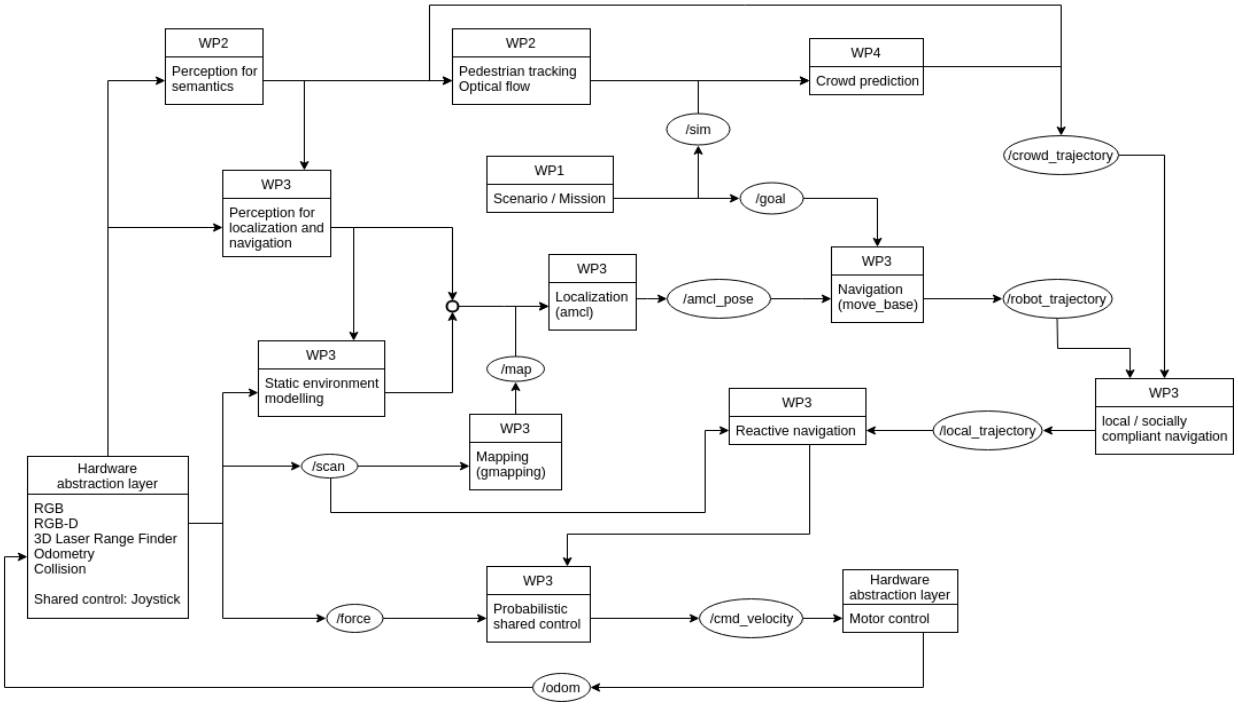


Figure 1: High level architecture considering the project Work Packages (WPX) and the exchange of message type between them in a ROS fashion.

2. CROWDBOT Architecture

In order to understand the basic flow of figure 1, we must start form the *Hardware abstraction layer*, where the low level sensor information is flowing from. For instance, laser scans will be the basic information from which the map representing the world will be composed in *Mapping*. Such representation is then used for approximate the localization of the robot frame within the map

provided using at least two elements: observations and odometry. In this way, the *Localization* module provides estimation so that the robot navigates in the real world knowing its location and the remaining till its goal.

The robot trajectory between point A and point B can be modified due to environmental constraints such as static/dynamic obstacles and potential collisions, interactions with other entities, or constraints provided with the aim to achieve a more socially compliant navigation: *Reactive navigation, Probabilistic shared control* and *Socially compliant navigation*, respectively.

In the case of a *Socially compliant navigation*, additional inputs such as the location of the crowds for a better trajectory assessment are crucial before and during the execution of the *Scenario/Mission*. Then, the *Reactive navigation* allows to minimize the risk of undesired collisions during such execution at the local level using *scan* and *rgb-d* readings. Finally, the *Probabilistic shared control* module will allow us to add an additional constraint during a task so that the robot can adapt to human limitations during the execution such as speed or support.

In order to close the loop, velocity commands are sent to the *Motor control* so that movement is produced keeping odometry (odom) readings up to date.

3. Robot Architecture Overview

In this section, we describe the different frameworks and packages used as baseline for the first prototype, separately. In the following subsections, a consolidated overview of the different components is provided.

3.1 Robot Operating System (ROS)

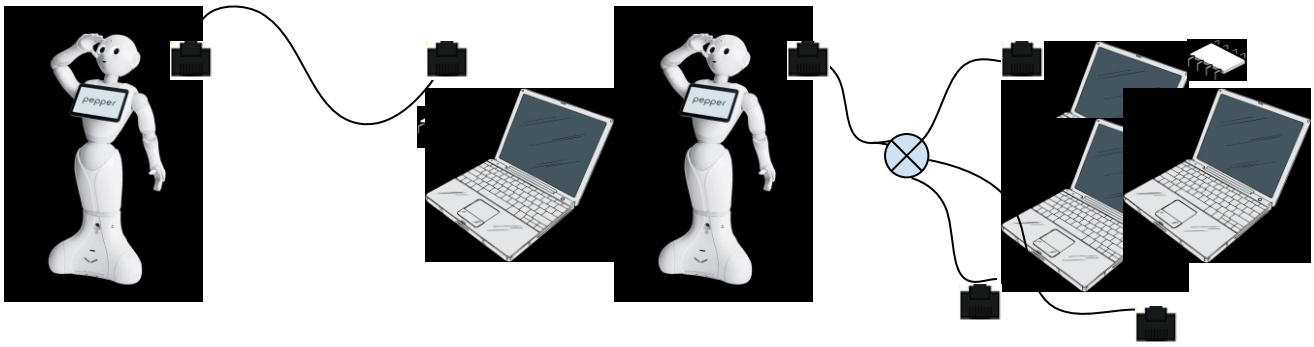
As described in [4]: *The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.*

Moreover, according to [5], *ROS is an open-source, middleware for your robot. It provides the services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.*

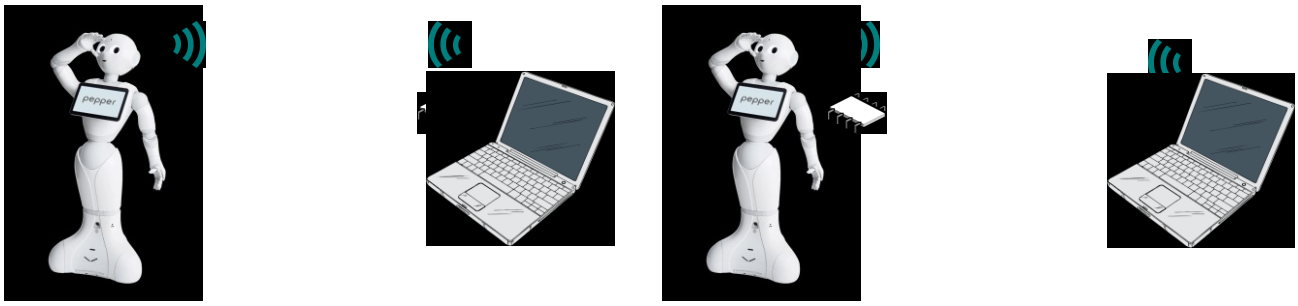
As a result, ROS was built from the ground up to encourage *collaborative* robotics software development. For example, one of our project partners might have experts in mapping indoor environments, and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might work on computer vision approaches recognizing and tracking pedestrians. ROS was designed specifically for groups like these to collaborate and build upon each other's work.

Processing strategies. Perception processing strategies allowed by ROS. (a) The default will consists of using a wired connection from Pepper to the processing laptop with GPU. (b) Potentially, the wiring may by shared by multiple laptops (e.g. to share with other WP processings). (c) If the WiFi connection and bandwidth allows the streaming of images and depth and in general

data at a sufficient frame rate, this option will be used. (d) Ideally, if an embedded GPU is available, the processing can be done all on-board, or with some WiFi connection to the laptop.



(a) Wired (b) Wired with multiple laptops



(c) WiFi connection (d) WiFi with embedded GPU

Figure 2: ROS processing strategies review

3.2 NAOqi APIs

NAOqi[1] is the name of the main software that runs within the robot and it is used to control it. In addition, it is the framework that used to program Softbank Robotics products: Romeo, NAO and Pepper. Generally speaking, it answers to common robotics needs such as parallelism, resource sharing, synchronization and event management. It allows homogeneous communication between the different modules running on the robot and exposed through an API that provides a convenient level of abstraction from the low-level architecture (rotate motors, image acquisition...).

Moreover, it has the following characteristics:

- Cross-platform, which means that it is possible to develop with it on Windows, Linux or Mac.
- Cross-language, with an identical API for both C++ and Python.
- Provides introspection, which means the framework knows which functions are available in the different modules and where.

3.3 ROS naoqi-Bridge

In this project, SBRE provides initial components for perception that are integrated and delivered in NAOqi running on a robot. Among perceptual components, there are detection of people, faces,

[1] http://doc.aldebaran.com/2-5/index_dev_guide.html

gaze analysis and analysis of facial characteristics and expressions. All these perceptual components are running locally on the robot and take their inputs from sensors embedded on Pepper. Both the outcome data from the perceptual components and the raw data from sensors embedded on Pepper can be accessed via NAOqi API. In addition to NAOqi, SBRE provides interfaces to access the most useful data and functions from NAOqi via a ROS environment (on demand of partners) using one of the following ROS Bridge packages:

- Naoqi Driver[2]: A generic ROS bridge that is common for all SBRE robots such as Pepper, Nao or Romeo;
- Pepper bringup[3]: Pepper-specific package that is based on NAOqi Driver and provides additional functionalities for Pepper robot, such as perception functions including data registration for depth and RGB embedded sensors.

The goal of this ROS bridge is to provide an access to the data from the robot (its sensors and motors) and from NAOqi as efficiently as possible. Thus, ROS bridge gets the data straight from the lowest levels of NAOqi hence ensuring low latency and low CPU usage. It publishes all available sensors data to ROS as well as the robot's position trying to be as close as possible to ROS standards by exposing standard topics. The ROS Bridge used in this project is initially an open-source project adapted for the needs of this project to ensure that partners can access all required data from the robot and its software as efficient as possible and can control the robot via ROS.

Figure 3: From left to right. Pepper, NAO and Romeo. The tree robots controlled by NAOqi framework.

3.4 Mapping Strategy and Framework

Mapping will rely on several frameworks operating simultaneously. It is expected that mapping will get increasingly challenging as the density of the crowd in the environment increases. Maps will be created using the various sensors on the platform.

- A 2D Map is generated using high resolution LiDAR sensors, to be used for localization, path planning and navigation. (See Figure 20)

[2] https://github.com/ros-naoqi/naoqi_driver
[3] https://github.com/ros-naoqi/pepper_bringup

- A 3D free-space map can be created using the depth sensor, and allows for additional safety in static obstacle avoidance, and visualization. (See Figure 4)
- In addition, a 3D map of visual features is created in environments which present feature-rich ceiling/sky views, in order to further improve the localization capabilities and compensate for cases where 2D LiDAR-based localization fails. (See Figure 21)

The possibility of merging the above maps into a single reference frame will be evaluated. Feasibility will depend on the quality of the transform data between the various sensors, which includes for example, in the case of Pepper, the encoders within the robot joints. Inaccuracy in these transforms will result in errors between the map reference frames. Mutual information encoded in both maps, if present, could theoretically be used to correct these errors. In the case of 2D LiDAR versus 3D visual features maps for example, mutual information is not readily present, whereas the 3D free-space and 2D LiDAR maps potentially encode geometrical information for overlapping space and could in theory be aligned through this.

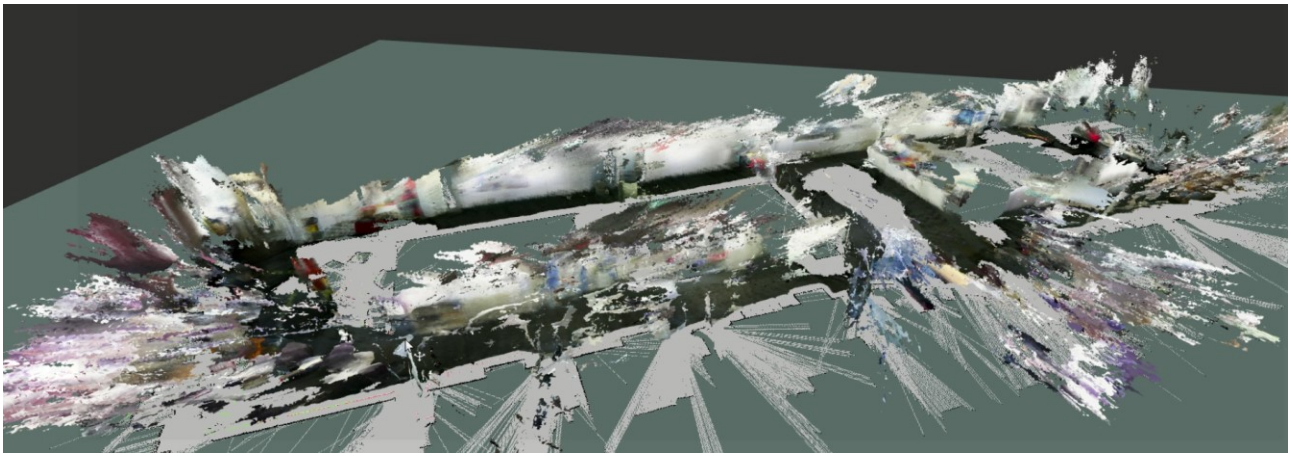


Figure 4: 3D free-space map overlaid on a 2D LiDAR map of the ETHZ ASL Office, all captured by Pepper with external sensors.

Figure 22 shows an example of the 3D Visual Features map and 2D LiDAR map represented in the same reference frame. However, one map is originally in the LiDAR sensor frame (tied to the base frame), whereas the other is in the VI sensor frame (tied to the head frame). As mentioned, inaccuracies in the measured transform between head and base (reported by the Naoqi software) will result in a position error between the two maps when visualized in a single frame.

3.5 Baseline System

The baseline architecture chosen for developing modules such as the crowd detection, reactive navigation, sensor perception and many others, is already an standard for european projects due to its latest state of the art, flexibility and robustness. Specifically, the CROWDBOT framework will be built on top of four main components whose use will vary depending on the robotic platform performing.

From top to bottom, figure 4.5 shows how the different work packages that encode high level behaviors in the first layer communicate to the already well established ROS which contains

different standard modules. This setup allow us, for instance, to define a path from point A to point B using a the ROS navigation stack and encode more complex behaviors that could deal with obstacle avoidance in a WPX within the CROWDBOT framework.

ROS modules allow us to set a common stack between different robotic platforms so messages can be homogeneous. Moreover, this approach benefits the possibility of sensor augmentation, integration and/or modification (see Section 6).

Pepper and other SBR robots run under the already presented commercial operating system NAOqi that provides an open API for operating their platforms. In order to communicate ROS and NAOqi, Ros-naoqi bridge provides a thin layer of message transformation that need to be in between both components. However, both wheelchair and Locomotec's smart cart do not run under the same OS, so integration will be through a different layer.

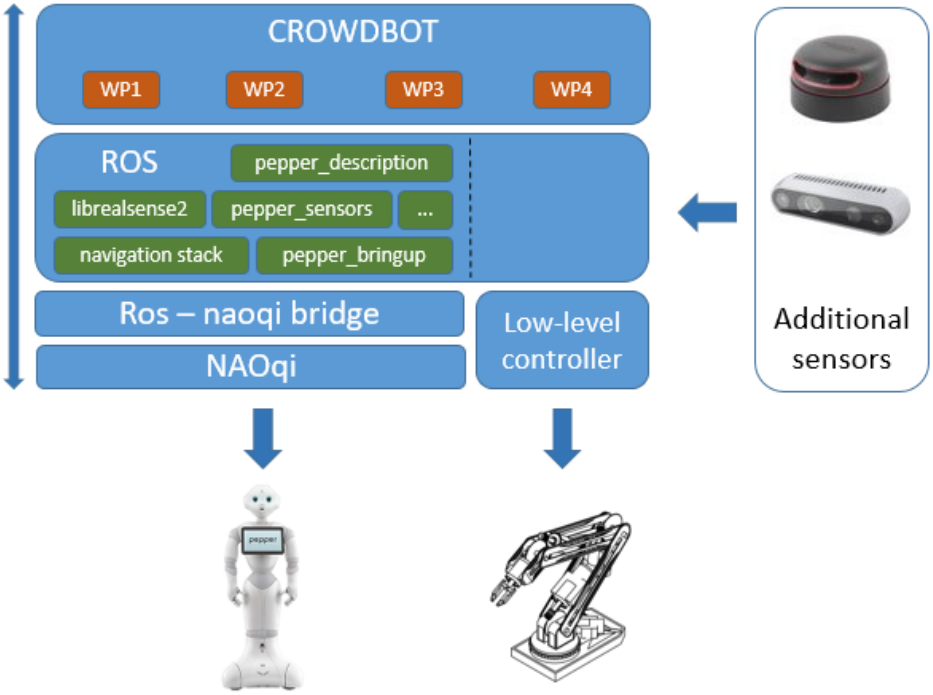


Figure 4.5: System architecture baseline for the first prototype.

4. Pepper Depth Sensing and Video Capabilities

The Mummer project[4] (European Union's Horizon 2020 program) [6] has spent efforts on investigating issues related to the Pepper depth sensing and video capabilities, and eventually collaborating providing feedback and inputs to the manufacturer of Pepper, SBRE. Delivery D2.1[5] proved that depth is an important cue that provides essential information for 3D scene understanding and allows a fast processing. Noticing that the quality of depth information acquired with Pepper is rather low in comparison to the normal ASUS Xtion sensor was a valuable

[4] http://mummer-project.eu/
[5] http://mummer-project.eu/media/media_517323_en.pdf

outcome relevant for CROWDBOT project. In this regard, better results are expected to be achieved using an Intel RealSense D435.

Moreover, the same deliverable shown that QVGA data provided by Pepper actually corresponded to an upscaled depth map of a QQVGA (Quarter of QVGA) depth map, produced from the sensor in order to minimise data transfers. The data gathered without this compression appears to be of a better quality and should be more useful for processing algorithms. In addition, for data collection and benchmarking purposes, recording capacities (here streaming and data synchronization) of all data streams is very important. Experiments along these lines are summarized in subsections 5.1 and 5.2. However, the synchronization between the RGB and Depth streams may remain an issue for joint processing of the two streams.

Depth sensors make direct measurements of the environment's 3D geometry. This information is strongly complementary to the standard vision modality and has been exploited more and more in recent years to address a wide variety of computer vision problems, such as path/motion planning, robot navigation or obstacle avoidance. This has also been facilitated through the availability of cheaper and acceptable quality sensors, such as the Microsoft Kinect (1 and 2), Asus Xtion, Primesense or Intel RealSense. Moreover, when based on active technologies like structure light or time-of-flight, these sensors are fairly robust when providing the object's texture and scene illumination.

Depth usage in WP2 and WP3. Not only the depth sensor embedded in Pepper's head is therefore of high value to address the tasks defined in these two work packages, but also the comparison in terms of quality and rate with respect to more recent sensors such as the Intel RealSense. The main objectives are the following:

- Mapping.
- Navigation and path planning.
- Reactive obstacle avoidance.
- Pedestrian detection and tracking.

In all them, depth information can help to accurately estimate the position of elements in the 3D space so that the robot is aware of its position within environment and the near-structure of its surroundings.

4.1 Depth Sensing Quality

The depth sensor on Pepper, according to the specifications, is an embedded Asus Xtion camera. In the process of employing this data for developing WP2 and WP3 tasks, the data was found to be of significantly lower quality than an off-the-shelf Asus Xtion and similar consumer depth sensors. In this section, we describe the steps that were taken to validate these issues and to find possible solutions thanks to the previous analysis done in MuMMER project. This will set up the baseline for improvements proposed in section 5.2.

4.2 Depth Sensor Improvements

The quality of a depth sensor, intended to be used jointly with an RGB sensor, is evaluated according to following criteria: the depth noise (along the camera optical axis), the depth

tangential and radial distortion in the image, the calibration with respect to the RGB sensor as well as the resolution of the depth data, linked to the density of the associated point cloud.

RGB-D calibration. In Figure 5 we show representative results of the initial data quality. Figures 5b indicates the effect of missing appropriate calibration parameters. We used both visualisation tools to discard software usage related errors, and the results were consistent. In order to calibrate the sensors, the process consisted on a standard stereo camera calibration procedure using a checkerboard pattern, while the IR projector in Pepper is covered to avoid capturing the scene with the structured illumination pattern. Therefore, it is a calibration between the IR camera and the color camera. We obtained noticeable improvements in terms of the color and depth alignment (through intrinsics and extrinsics parameters) but the calibration parameters have no influence on the depth quality itself.

Lens distortion. The Mummer project has shown that several versions of Pepper provided planar objects (like walls) were highly curved when 3D rendered using a live stream from Pepper's depth sensor. This is due to the distortion applied to the sensor by the lens. Removing the lens might be an option to avoid such issue while a more suitable material is found in order to maintain Pepper's look for the end customer. A clear improvement was observed for the 3D data in general as planar objects were indeed planar in their 3D rendered version.

(a)

(b)

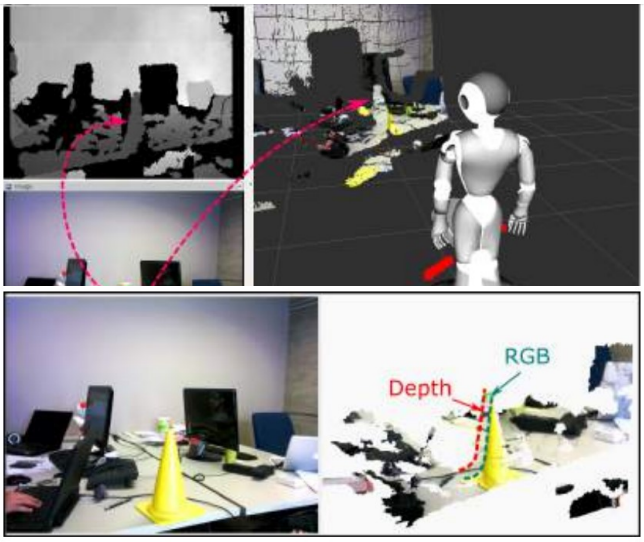


Figure 5: [7] Qualitative analysis of the RGB-D camera provided by the Pepper robot. Figure (a) shows the colored point-cloud rendered by ROS, note how the cone does not get the yellow texture mapped into it when 3D rendered. Figure (b) shows a textured 3D mesh rendered from the RGB and Depth data using Pepper's camera calibration parameters and the ROS rgbd module for visualisation. Note the misalignment between shape (depth) and texture (RGB) due to inaccurate calibration parameters.

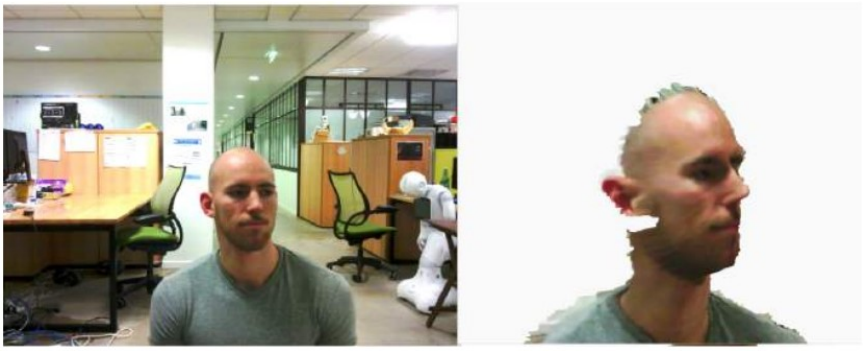
Depth registration and resolution. The registration process consists on projecting the depth map into the RGB camera such that there is a one to one pixel alignment. However, this does not affect the depth noise along the camera optical axis, thus it does not improve the data quality. During Mummer project, it was concluded that the provided QVGA data provided by Pepper actually correspond to an upscaled depth map of the QQVGA depth map. The reason is that this minimises

the data transfers. As the qualitative results shown in Figure 7, acquiring the QVGA and rendering its data clearly suggests a significant improvement on the depth sensing quality.

(a) (b)

(c) (d)

Figure 6: [7] Assessment of the qualitative results obtained scanning the scene (a), before (b) and after (c) the RGB-D calibration in comparison with the results acquired with an Intel RealSense (c), which are significantly better in the matching.



(a) (b)

Figure 7: The image scene (a) and the result using Pepper onboard sensor with depth-rgb calibration, lens distortion removal, depth image registration and QVGA depth resolution.

4.3 Streaming Capacities and Synchronization Accuracy

Recording is a basic and essential part of perception research. Only with enough high quality recorded data, we are able to train models and precisely evaluate them. In preparation for future data collection, we conducted several tests to understand the capabilities and limits of data streaming and recording of Pepper. More specifically, we evaluated the following capabilities:

- Audio-visual synchronization,
- Data streaming rate under different conditions,

- Synchronization between RGB and depth cameras.

Audio-visual synchronization. We tested the audio-visual synchronization by recording a 5-minute video of a person speaking to the robot, and the synchronization is evaluated by subjectively watching the lip movements and listening to the audio recording simultaneously. As the result, we found that there is no noticeable delay between audio and video data.

Data streaming rate. We tested the data streaming rate for recording via naoqi driver (ROS interface of NAOqi) under different conditions. The conditions include different connection settings: via a Gigabit Ethernet cable, 100Mb/s Ethernet cable or via WiFi (802.11n standard); and different robot states: with autonomous life on or off. The following data are streamed simultaneously:

- Front RGB camera images and camera info.
- Depth camera images and camera info.
- 4-channel audio from the microphone array.
- Robot joint states and ROS frame transforms (topic '/tf').
- Sonar, laser and infrared data from robot base.

The streaming rates for videos are shown as in Table 1. Note that in the current NAOqi system, the video data is not compressed before streaming.

Connection	Gigabit Ethernet		100Mb/s Ethernet		WiFi	
Autonomous Life	Off	On	Off	On	Off	On
QVGA RGB + QVGA DEPTH	30	18	18	12	4	1
VGA RGB + QVGA DEPTH	30	15	8	8	1	<1

Table 1: Video streaming rate without compression. The numbers in the table indicate frames per second (fps). Note that as mentioned in Section 4.1.2, the QVGA depth images are actually upsampled from QQVGA depth images captured by Pepper.

The audio streaming rate is fixed under any condition at 4-channel 48000Hz without data loss. And the other data are streamed without interfering the video and audio streaming. In addition to streaming with naoqi driver, we also tried other methods. One of which is running a GStreamer video server on the robot and the server streams RGB video data compressed by JPEG. This approach significantly improves the streaming rate. The result is shown in Table 2.

Data streamed	Ethernet	WiFi
QVGA RGB Compressed	30	24
VGA RGB Compressed	30	15

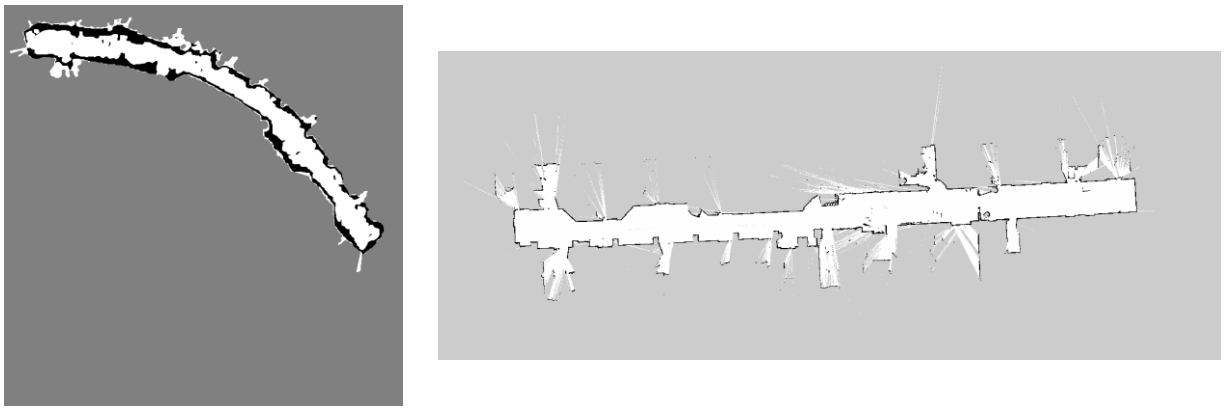
Table 2: Video streaming rate with compression. The numbers in the table indicate the frames per second (fps).

RGB-D synchronization issue. We measured the delay between the arrival time of RGB and depth images, and found that it varies from 10 to 20 milliseconds. While this might be sufficiently good

for some late RGB and depth fusion modules (eg extracting depth measures from localized body parts, to estimate people 3D pose), it will be insufficiency for combined depth and RGB processing, especially gaze tracking which requires very fine synchronization between RGB-D (since in 10 or 20 milliseconds, people will move slightly, and an offset of 1 or 2 pixels in eye localization means 5 to 10 degree error).

4.4 FoV in Scene Mapping and Obstacle Sensing

The results shown in figure 8 provide an insight of the mapping quality of an scenario using the current sensor setup. There two issues to be identified: the thickness of the borders, and the curvature of the final scenario that does not exist in reality.



(a) (b)

Figure 8: Mapping of the same corridor view using different laser technology. (a) shows the result of the Pepper 6 laser sensors and (b) the result using the high resolution laser of another platform.

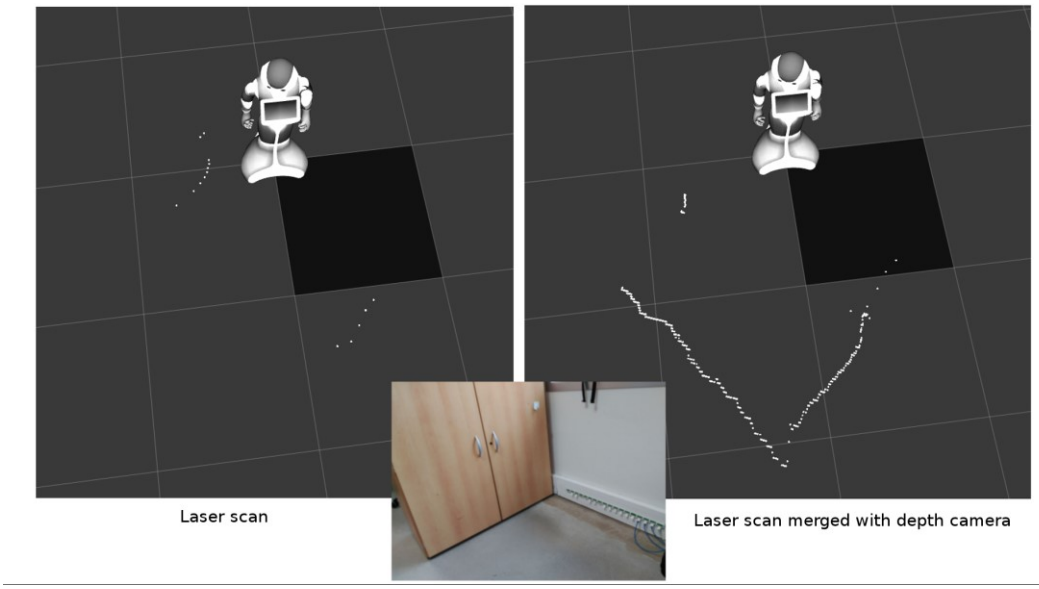
The distortion observed is likely caused by the sparsity of the 3 laser sensors that only provide 15 points per scanner. This, together with its relatively low detection range, can lead to some errors in localization and does not manage to fully compensate for the odometry drift. And thus, the mapping results in a curved area.

Given the fact that obstacles are represented in black, (a) fails at merging consistently when those are seen more than once due to the limited laser resolution. However, (b) merges nicely, reducing at the same time uncertainty areas and providing a results that matches more precisely the reality.

Sensor quality improvement can not only provide better results in terms of mapping performance but also in navigation, so that localization within the given map is achieved more accurately and obstacles are detected at larger ranges. All that together makes a navigation more consistent in terms of trajectories, reducing unnecessary movements and increasing awareness of the surroundings. And improvement of these two issues is described in the following section.

4.5 Improvement of Scene Mapping

The methodology to increase sensor performance has been to merge the depth camera located in the head of the robot platform with the base sensors. The strategy has been to set the head tf frame static so that sense in the same direction the laser do.



(a) (b) (c)

Figure 9: The basic laser scan setup is shown in (a) that maps the scenario shown in (b). After the depth camera and laser scan merging strategy developed, the results are shown in (c).

The conversion from RGBD sensor data to laser scan is not straightforward although some packages from ROS ecosystem already have developed such strategy for other purposes. Once both laser readings are acquired, the following merging strategy described in [2] has been proposed:

1. The simulated laser scan is extracted from a horizontal slice of the depth image acquired from the depth sensor located in the head of Pepper.
2. The 3 laser scans from the base are acquired.
3. A merged scan is created by generating a new scan area that encompasses both the base and depth scans.
4. Scans are overlaid over the new area, with the closest point being selected in areas of overlap.
5. In order to account for the sparse resolution of the laser scans, any new reading that fall between the points of the base scan are then considered invalid readings.

Results. Based on the implementation of the previously described approach the outcome acquired can be observed in figure 9. It is important to note that:

- Only the depth data was used as input for the simulated laser scan (without merging the laser scans).
- The map was generated using a Pepper without lenses on the eyes.
- The mapped room has a length of 23 m and a width of 6 m on its widest part, and a width of 3.8 m on the narrowest part (total surface of 117.04 m^2).
- The room was not entirely mapped.

In comparison with the results acquired using the conventional strategy of mapping, where Pepper maps the surface using the laser scan sensors located in its base, this approach provides the following benefits (see figure 10):

- Decreasement of the thickness of the edges.
- Increasement of the frontal field of view.
- Decreasement of the mapping time (to be tested).
- Mitigation of the error induced by odometry resulting on an inexistent curvature (to be tested).

(a)

(b)

Figure 10: Area mapping result based on the described simulated laser scan (a) and the result of a soft ad-hoc post-processing of the previous (b).

In addition, It is important to notice that in order to enhance the predictability of the motion as much as possible during the robot displacement, a custom motion controller has been deployed in the platform. Such resource correctly accounts for the holonomic nature of the base of the robot when turning. The equations used to build this controller are described in [2] and in [3]:

$$Wfl = theta + V \cdot \cos(150 - \omega)$$

$$Wfr = theta + V \cdot \cos(30 - \omega)$$

$$Wb = theta + V \cdot \cos(270 - \omega)$$

where Wfl, Wfr and Wb represent the front-left, front-right and back wheels respectively (see Figure 11). ω describes orientation of the drive vector of the robot and V, the magnitude of the drive vector. Finally, theta represents the angular rotational value added to the wheels.

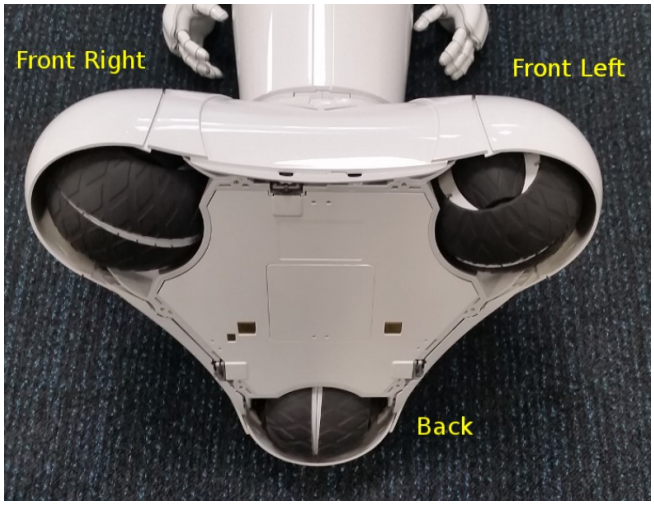


Figure 11: Inner view of Pepper's holonomic base with three omnidirectional wheels. Source: [2]

5. Sensor Augmentation

Justifying the addition of sensors into the Pepper robot has been an important focus of this delivery in order to define the hardware baseline for the project. due to the difficulties in terms of space and trying to maintain the structure of the robot (we will be working on this direction still). In this section, we continue proving the need to add additional sensors to the platform due to the following limitations raised by the experiments conducted in a controlled environment:

- Limitation of the field of view (FoV) for the RGBD camera located in the head due to:
    - Location of the sensor in a constant moving robot frame (head). Head position can't remain static at all times modifying the FoV often.
    - Possibility to include a greater FoV to improve crowd perception and human recognition.
- Base laser sensors have a reduced resolution that prevents performing better at critical project tasks such as:
    - Obstacle detection several meters in advance in order to reduce the displacement.
    - Area mapping for robot localization.
- Limited FoV of the back area due to the absence of any kind of sensor in order to monitor crowd and human beings.

Identifying these areas of improvement and address them properly is key.

5.1 CROWDBOT Custom Sensor Suite

Currently we have mounted a visual inertial sensor (VI-Sensor) for high accuracy odometry and localization, as well as two 2D SICK TiM571 LiDARs at the base of the robot to provide high

resolution range data for mapping and obstacle detection. In the future, we are also planning to mount an RGBD sensor such as the Intel RealSense D435 on Pepper's tablet to achieve the pedestrian detection and tracking requirements of the CROWDBOT project.

5.1.1 VI-Sensor

The VI-Sensor integrates stereo-vision cameras and an IMU (inertial measurement unit) to precisely track ego-motion and provide accurate localization solutions in an indoor environment such as an office space. The VI-Sensor shown in Figure 12 is mounted on the back of Pepper's head with cameras pointed upwards to view the ceiling. Due to the severity of lateral occlusion from human crowds, this may be the most robust way to sense the environment and localize a robot.

The fusion of visual and inertial cues has become common practice for robotic navigation due to the complementary nature of the two sensing modalities. Visual-Inertial (VI) sensors designed specifically for this purpose mount the cameras and IMU on the same board. This alleviates calibration and synchronization challenges to enable tight integration of visual measurements with readings from the IMU. Additional technical details of the VI-Sensor and its driver are available at [9-11].

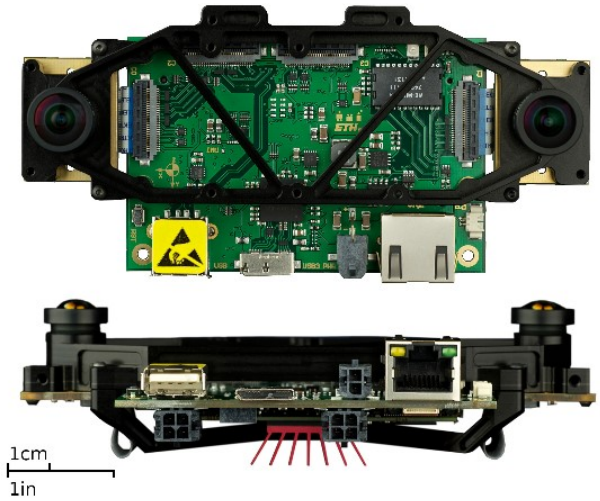


Figure 12: VI-Sensor [10]

5.1.2 SICK TiM571-2050101 LiDAR

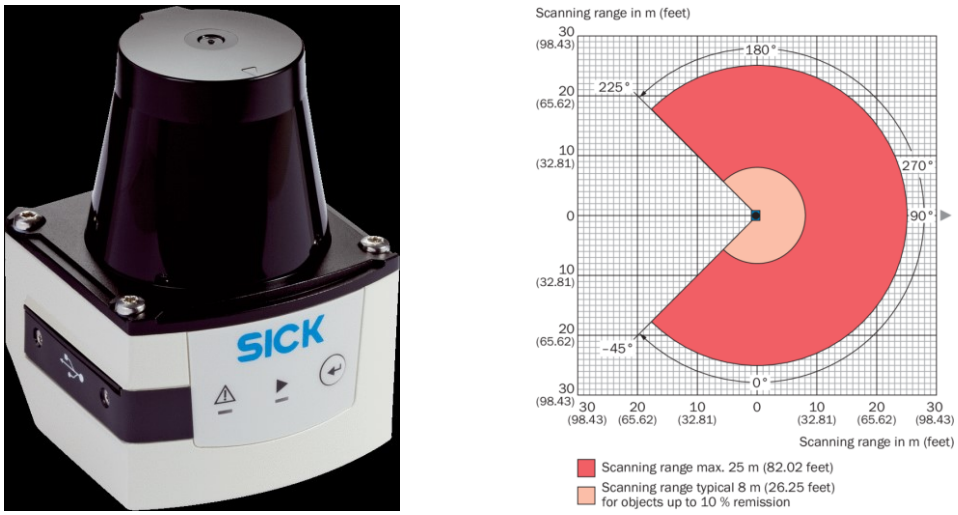
We have mounted two 2D SICK LiDARs near Pepper's base to provide high resolution range data. The sensor and its footprint specifications are shown in Figure 13a. In addition, the sensor has a scanning frequency of 15Hz, an angular resolution of 0.33° and a response time of 67ms.

5.2 Sensor Mounts

The sensor-augmented Pepper robot is shown in Figure 14. The suite includes:

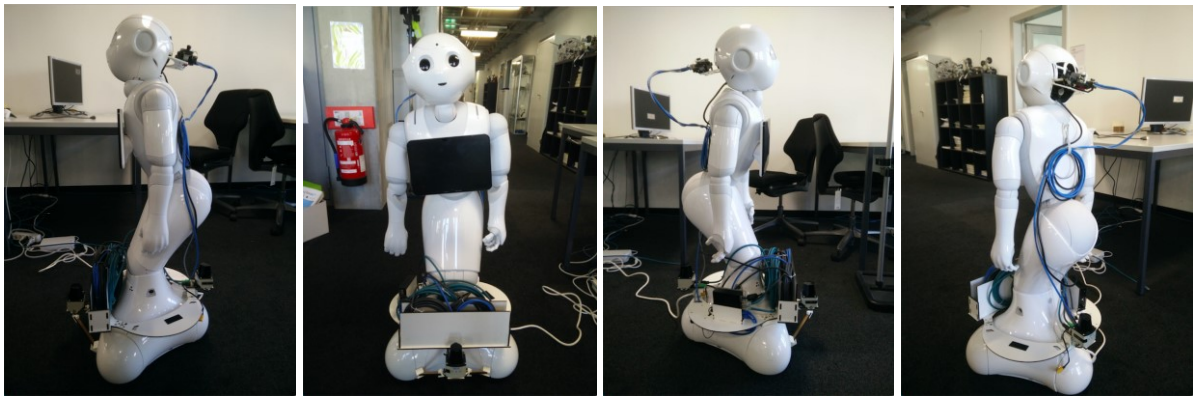
- One 3D printed VI-Sensor mount attached to the back of Pepper's head
- One laser-cut base ring consisting primarily of three pieces
- One laptop tray at the front of the base ring
- One ethernet switch holder on the right side of the base ring
- Two laser-cut LiDAR mounts attached to the front and back of the base ring

- Three 3D printed connectors attached to Pepper's base that form level mounting points for the base ring



(a) SICK TiM571 (b) sensor specifications [12]

Figure 13: 2D LiDARs from SICK [12]

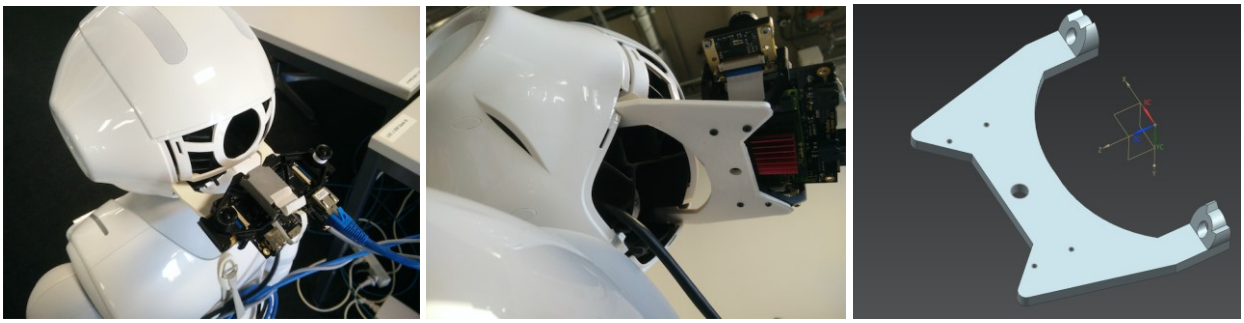


(a) Left side view (b) From view (c) Right side view (d) back view

Figure 14: Augmented Pepper

5.2.1 VI-Sensor head mount

The CAD model of the VI-Sensor mount is shown in Figure 15(c). It is attached to the back of Pepper's head using the available mounting points. Care was taken to avoid occluding the touch sensors on the top of Pepper's head as well as any of the sensors on Pepper's face (see Figures 15(a) and 15(b)). The ethernet and power cables (from the VI-Sensor as well as Pepper) are routed down to the base where the main laptop will be stored. Sufficient cable length is provided to allow the full range of head motion.

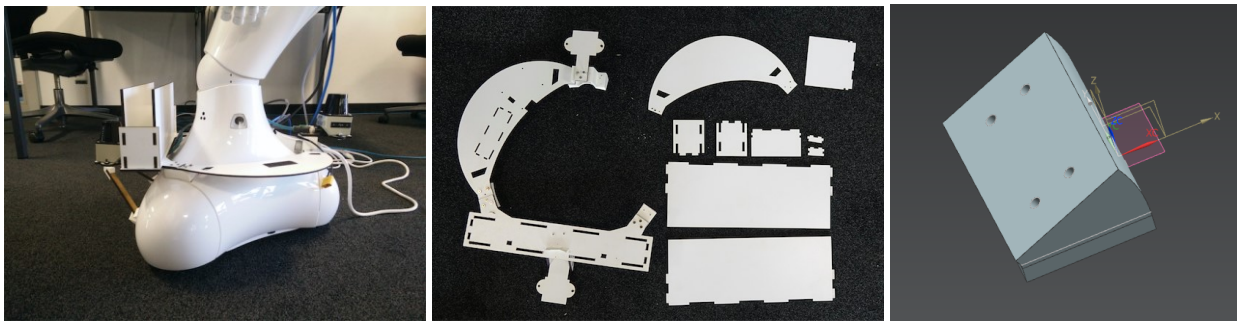


(a) Top view (b) Bottom view (c) Cad model

Figure 15: The VI-Sensor mounted on the back of Pepper's head using the existing mount points

5.2.2 Base ring and mounts

The complete base ring mount as well as its separate components are shown in Figure 16. The ring is constructed from a number of laser-cut pieces (currently 3mm and 5mm white MDF, however our future iterations will use 5mm transparent acrylic, which is stronger) and provides direct attachment points for a LiPo battery and a custom power board (on the underside of the ring) to run the additional sensors. Furthermore, the ring provides space to house a laptop, ethernet switch as well as the two SICK TiM571 LiDARs. More details on each of these components are provided in the following subsections.



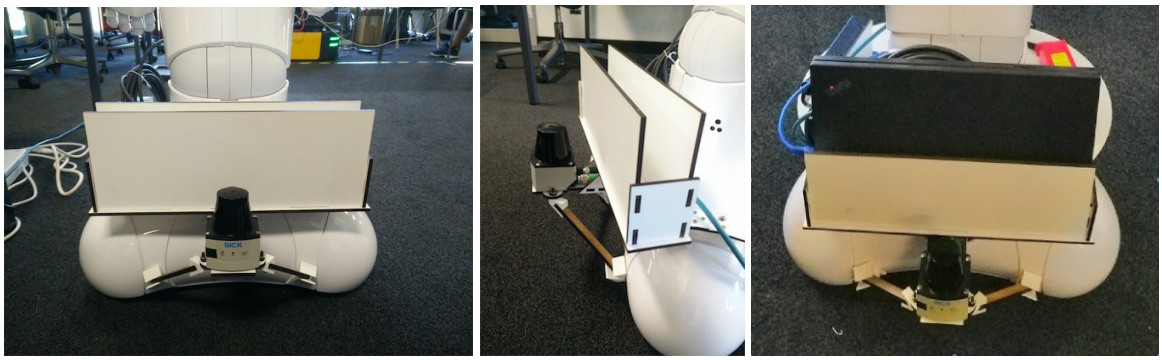
(a) Left side view (b) Laser cut components (c) Cad model of connectors

Figure 16: Ring mount and component pieces

The ring itself is attached to Pepper's base via three 3D printed connectors that conform to the curvature of Pepper's base and provide a horizontal mounting surface. Two of the connectors are of the form shown in Figure 16(c) while the third is combined with the rear LiDAR mount as shown in Figure 16(a). The connectors are attached to Pepper's base using removable doubled-sided foam tape and the ring is screwed into the connectors. The mounting points were chosen such that they provided the greatest stability to the ring without interfering with Pepper's bumpers or ventilation system. The ring can be deconstructed and flat-packed for transportation and remounting on a different Pepper robot. When fully constructed, the base ring and mount occlude two out of three of the original LiDARs and both sonar sensors in Pepper's base.

Laptop tray

The laptop is housed at the front of Pepper's base out of reach of either Pepper's hands or legs. The width of the laptop tray does not exceed the width of the robot base, as can be seen in Figure 17 however it extends marginally in front of Pepper to accommodate the curvature of the base.



(a) Front view (b) Left view (c) With laptop

Figure 17: The laptop tray and forward LiDAR at the front of the ring mount

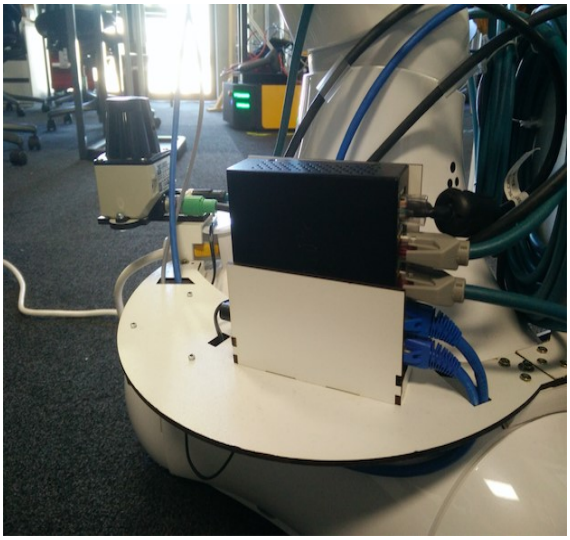


Figure 18: Ethernet switch holder on the right side of the ring mount

Ethernet switch holder

The ethernet switch is housed on the right side of Pepper's ring mount and is also out of reach of any of Pepper's limbs. Several cutouts are provided in the ring to route the ethernet and power cables without exceeding minimum bend radii restrictions on the cables (see Figure 18).

LiDAR mounts

Two LiDAR mounts are attached to the base ring, one in front of the laptop tray as shown in Figure 17, and the other at the back above the charging point as in Figures 18 and 19(a). The two LiDARs together provide full 360° coverage in the lateral plane.

The forward LiDAR extends beyond the footprint of the robot but remains within the circular radius of the base. Two trusses provide additional support and stability with mount points shown in

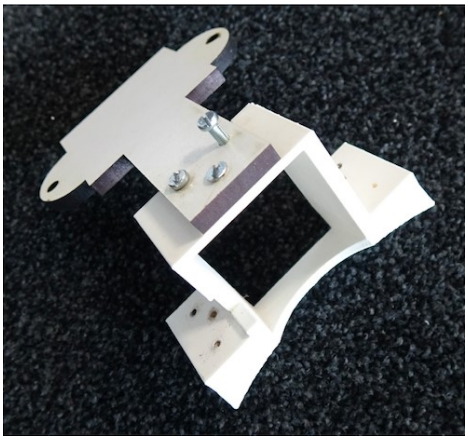
Figure 17 Similar to the ring connectors, the truss connectors are attached to Pepper's base at locations that do not interfere with the bumpers or the ventilation system.

The rear LiDAR sits directly above the rear point of Pepper's base and is elevated slightly above the height of the front LiDAR to accommodate Pepper's charging point without needing to unmount the LiDAR. The offset between the two LiDARs can be seen in Figure 14 and also corresponds to the manufacturer's recommendation for mounting multiple LiDARs in the plane to avoid cross talk in the lasers.

The connector shown in Figure 19(b) is the third mount point for the base ring. As stated earlier, it is 3D printed and attached to Pepper's base using removable doubled-sided foam tape. The LiDAR support platform is laser-cut from 5mm MDF and is screwed into the connector.

Furthermore, unlike the forward LiDAR, the relatively short cantilever arm of the rear LiDAR mount does not require any additional structural supports.

(a) Rear LiDAR mount and charging point access



(b) Rear LiDAR mount components

Figure 19: Rear LiDAR mount assembly

6. Computational strategy

The computational requirements of CROWDBOT are driven by the sensor processing for environmental mapping, pedestrian detection, tracking and crowd prediction needed to conduct safe robot navigation through a human crowd. The bulk of this processing will go towards executing computer vision algorithms based on pre-trained deep neural networks for tracking individual pedestrians as well as the general flow of the crowd around the robot. Current commercially available GPUs will allow for efficient online querying of the visual detection algorithms. Two potential GPU laptops are listed in Table 3.

The mapping and localization algorithms can be handled solely by the CPU and therefore do not require any GPU resources. As shown in Figure 1, the CROWDBOT trajectory planning algorithms will need to interface with both the output of the pedestrian tracking modules as well as the mapping and localization modules to compute socially compliant trajectories. Our current aim is to perform all CROWDBOT-related computation on a single laptop mounted onboard the robot to remove any reliance on external infrastructure for the successful deployment of a CROWDBOT in general open world scenarios. On Pepper, the laptop will be placed in the dedicated tray described previously in Section 6.2.2, with data connections routed via an ethernet switch (see Section 6.2.2).

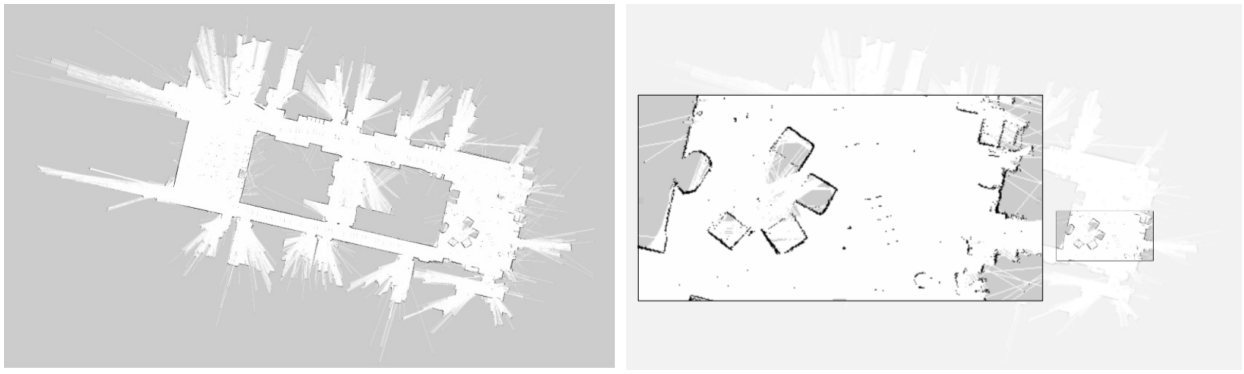
Name	S ROG Zephyrus	Alienware 15 R3
CPU	Intel Core i7 (7th Gen) 7700HQ / 2.8 GHz	Intel Core i7 (7th Gen) 7700HQ / 2.8 GHz
GPU	NVIDIA GeForce GTX 1080	NVIDIA GeForce GTX 1060
Battery	68Wh, 4 cell, Li-ion	68Wh, 4 cell, Li-ion
Power	230 Watt, 19.5 V, 11.8 A	240 Watt, 19.5 V, 12.3 A
Weight	4.85 lbs	7.69 lbs

Table 3: Sources [13] and [14]

7. Experimental Scenario

Preliminary results has been obtained deploying the system described in this document in a controlled environment. The results displayed in figure 20 show the good level of detail acquired during the mapping task by both LiDARs. In addition, the VI-Sensor results are shown in figure 21, where a precise localization using the ceiling during the navigation task is achieved.

Finally, a combined map of the previous sensor inputs is shown in figure 22.



(a) 2D map from LiDAR scans (b) Close up of furniture shows good map coherence

Figure 20: Results from 2D LiDAR mapping

(a) Features found in VI-Sensor (b) Maplab feature map and sensor trajectory

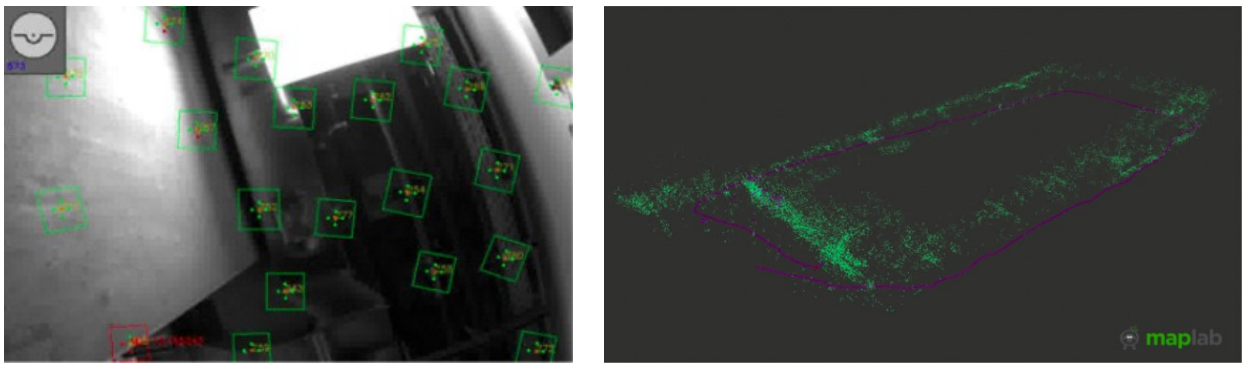


Figure 21: Results from VI-Sensor mapping and localization

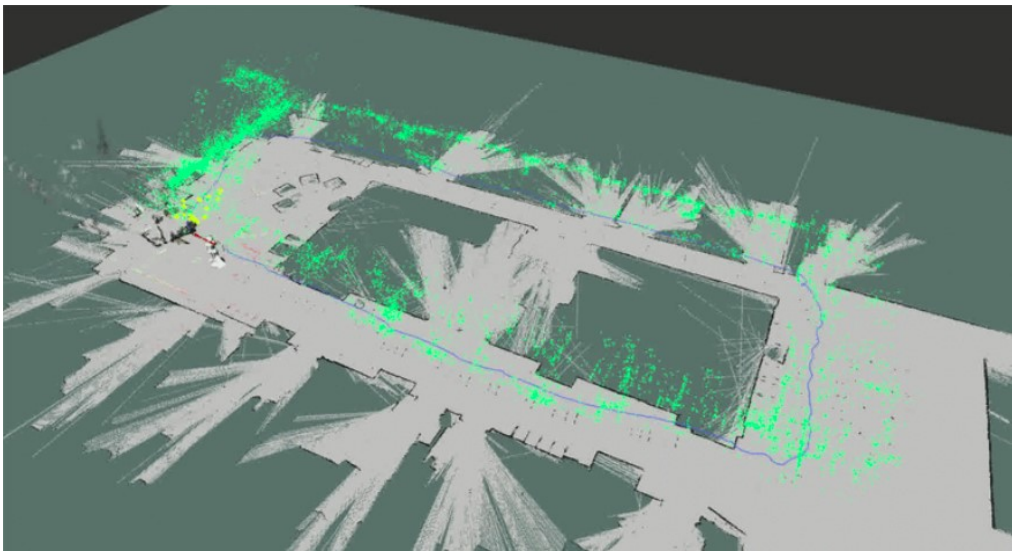


Figure 22: Combined VI-Sensor and 2D LiDAR mapping and localization. Yellow points show matched features found using ROVIOLI. The RGB and depth data are also projected in the scene.

References

[1] Gavin Suddrey and (2018). Enabling a Pepper Robot to provide Automated and Interactive Tours. *CoRR, abs/1804.03288*.

[2] Suddrey, G., Jacobson, A. and Ward, B., 2018. Enabling a Pepper Robot to provide Automated and Interactive Tours of a Robotics Laboratory. *arXiv preprint arXiv:1804.03288*.

[3] A. F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, and N. Pereira, "Three omni-directional wheels control on a mobile robot," 2004.

[4] http://www.ros.org/about-ros/ Accessed: Sep 4th, 2018

[5] http://wiki.ros.org/ROS/Introduction Accessed: Sep 4th, 2018

[6] Foster, Mary Ellen, et al. "The MuMMER project: Engaging human-robot interaction in real-world public spaces." *International Conference on Social Robotics*. Springer, Cham, 2016.

[7] http://mummer-project.eu/media/media_517323_en.pdf

[8] Pandey, Amit, and Rodolphe Gelin. "A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind." *IEEE Robotics & Automation Magazine* 99 (2018).

[9] ethz-asl. "VI-Sensor Factsheet". https://github.com/ethz-asl/libvisensor/blob/master/VISensorFactsheetweb.pdf. Accessed: 20-09-2018.

[10] Janosch Nikolic, et al. "ROS Wiki: visensor". http://wiki.ros.org/visensor. Accessed: 20-09-2018.

[11] Janosch Nikolic, et al. "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM". *IEEE International Conference on Robotics and Automation*, pages 431–437, IEEE, 2014.

[12] SICK AG. "2D LiDAR sensors TiM5xx". https://www.sick.com/us/en/detection-and-ranging-solutions/2d-lidar-sensors/tim5xx/tim571-2050101/p/p412444. Accessed: 20-09-2018

[13] CNET. "Asus ROG Zephyrus Specs". https://www.cnet.com/products/asus-rog-zephyrus/specs/. Accessed: 25-09-2018

[14] CNET. "Alienware 15 R3 Specs". https://www.cnet.com/products/alienware-15-r3-15-6-core-i7-7700hq-16-gb-ram-1-tb-hdd-english/specs/. Accessed: 25-09-2018