EU Horizon 2020 Research & Innovation Program

Advanced Robot Capabilities & Take-Up

Safe Robot Navigation in Dense Crowds

www.crowdbot.eu

Technical Report

D 1.3: Specification of Scenarios Requirements Update

Work Package 1 (WP 1)

Scenarios Co-Design & Evaluation

Task Lead: INRIA France

WP Lead: University College London, UK

Table of Contents

Table of Figures

Executive Summary

Deliverable D1.3 is an updated version of D1.1. To enable the reader to have a complete understanding of the document without need to refer to previous ones, some existing portions have been reported here, even though they did not experiment a major update. To clarify things, for each section, a similar orange text box provides information about the nature of modifications compared to D1.1 (major update or minor update).

Scenarios are descriptors that portray use cases and operational procedures of mobile robots in human crowd environments such as hospitals, shopping malls, train stations and other public or private venues. Nowadays we are witnessing the presence of robots in both public and private places but their efficacy and technological features are rather modest due to their limited mobility and interaction with humans. The main focus of the Crowdbot Project is to demonstrate safe and efficient mobile robot navigation in a dense crowded human environment.

This report is an update from the deliverable *D1.1* entitled "*Specification of scenarios requirement*". Indeed, since the writing of this first report, the technology developed within the project evolved. We list them in the **Section 5** (update from **Section 4 in D1.1**) of this document. In consequence, scenarios have been adjusted in order to fit better the current state of the technologies and the planned implementations during the year. This report details the updated navigation scenarios we plan to test and validate as part of the overall project goal of research, innovation, ethics and feasibility study for technology transfer in mobile robotics.

This report is based on the report D11, we have reported some of its content, updated or completed it according to the progression of the CrowdBot project. The introduction of each section summarizes the changes and addition to D11.

In **Sections 1** and **2** we provide a thorough coverage of our proposed definitions and relationships among the terms *general/social*, *operational* and *test* scenarios. Full description of various navigation test scenarios that the team plans to test and evaluate is provided in **Section 5**.

Closely related to scenarios are test events and system-level requirements to evaluate the outcomes (both success and failure) of such tests. Programmatic aspects of requirements development and test execution are covered in the latter part of **Section 2**. This section also describes the process by which we decided on the next scenarios and how we updated them.

We have prepared a System-Level Test Plan (STP) and its associated requirements list. An STP is prepared a month before the commencement of a test event. Two test events (the 1st in late 2019/early 2020 and the 2nd in early-to-mid 2021) are planned over the 42-month project span. A requirements list pertaining to all proposed Crowdbot navigation scenarios is presented in **Section 6**.

The T&E team is also responsible for evaluating test data and publication of a test report after the conclusion of each test event. The test report is shared with both internal teams in charge of Technology Development (TD), Robot System Integration (SI) and Design & Quality Control (QC) and external stakeholders. Based on test evaluation and recommendation from the T&E team and external stakeholders, the remaining internal teams (TD, SI and QC) devise technology enhancements to improve robotic navigation. Specific details are provided in **Sections 6** and **7**.

For validation and verification of test scenarios and robotic performance outcomes by external stakeholders, the team plans to solicit advice and feedback from technical experts as well as potential user communities via interview sessions and related engagements, coordinated meetings and joint publications. This topic is covered in **Section 7**.

A unique feature of the Crowdbot Project is its utilization and integration of navigational technologies onto three different types of robots. The team has already provided technical descriptions and specifications of system components and onboard sensor suite of our robots in already released reports *D2.1: Sensor Specifications* and *D5.1: System Architecture*. In **Section 3** we provide supplementary material of our robots with relevance to scenarios, requirements and test events: namely, details about each robot's physical frame structure, locomotion profile and human-machine interaction & communication options.

Additionally, the technologies developed during the project that are not robot dependant, and which are evaluated in the described scenarios, are described in the **Section 4**.

Finally, this report (D3.1) is the second in a series of two scenarios and requirements reports we planned to prepare and submit as official deliverables. The previous-on report *D1.1* (First version of Scenarios & Requirements) is available on the website www.crowdbot.eu.

1. Introduction

This section is a minor update compared to *D1.1*. The update concerns the last part of the section that provides new definitions for the categories of scenarios that CrowdBot will explore in the next evaluation round of tests, and provides an overview of the experimentation plan through Table 1.

The title of this report "Specification of Scenarios Requirements" is somewhat long-winded but in essence, this is a *requirements* document. Immediately, this raises two relevant questions:

Q1: What are scenarios?

Q2: Who is preparing scenario requirements and evaluating test outcomes?

The answer to **Q2** will be provided in **Section 2** and further elaborated in **Sections 6** and **7**. First, we address **Q1**.

Since the main focus of the Crowdbot project is safe and intelligent navigation of a robot in a dense human crowd environment, the term *scenario* refers to a space-time event in which a mobile robot moves from an initial location (say, Point A) at stop watch time t_0 (time zero) to its final position (say, Point B) by time t_{fin} (finish time). To factor in the human crowd effect to a scenario, we must specify the physical environment (e.g. building structures, furniture, roads, trees, etc.) to bound the available space for movement and also characterize the behavior and motion profile of humans that the robot is likely to encounter during its traversal from Point A to Point B. A robotic navigation event can be summarized more precisely in terms of an *Operational Scenario*.

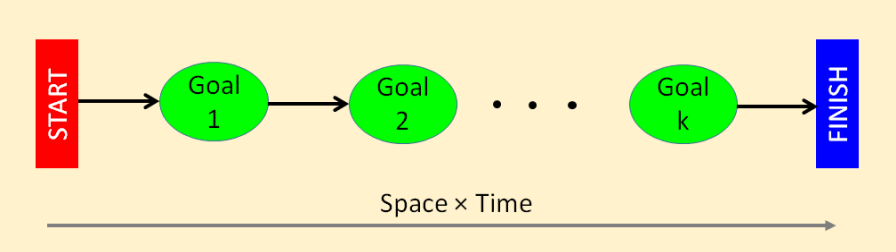

Figure 1. Description of a Navigation Scenario in terms of Space-Time Markers

As shown in Figure 1, the START marker is associated with start position (Point A) at time t_0—denoted as (A, t_0) — and the FINISH marker as (B, t_{fin}). Note that several intermediate steps (k steps in Figure 1) are involved before reaching the finish line. In the robotics community each step is known as a goal and the position of the robot after the completion of a goal is called its goal location. In general, the overall START-to-FINISH operational scenario is broken into smaller goals because in each intermediate step, a unique task-specific operation must be carried out to reach a designated goal location. Mathematically, the goal location for Goal *m* can be denoted as L_m and its corresponding finish time as t_m. Hence, every START-FINISH operational scenario is a succession of goal-oriented, task-specific robotic operations, all executed in the correct order to reach the specified finish point.

START: (A, t_0) -> G_1: (L_1, t_1) -> G_2: (L_2, t_2) -> . . . -> G_m: (L_k, t_k) -> FINISH: (B, t_{fin})

We now provide an example to illustrate and elaborate on a real-world scenario. Figure 2 shows a tele-operated robot by Starship Technologies [1] adapted for delivery of Domino's pizza. An employee loads the pizza into the robot's cargo bay (Figure 2 – left) and is later retrieved by a customer (Figure 2 – right).

Figure 2. Domino's Pizza Delivery Robot by Starship Technologies [1][2][3]

It is important to note that this pizza delivery scenario falls under the category of a real-world *Social or General Scenario*. From an operational perspective, robotic pizza delivery is simply a navigational operation of a service robot carrying a package from Point A to Point B with possibly an upper bound on the total delivery time (to maintain crispiness of a freshly baked pizza). Strictly speaking, an operational scenario is task-specific and goal-oriented whereas a social or general scenario almost always has a story to tell (i.e. use case) in some social context. Other plausible examples of general/social scenarios for robotic navigation are:

- Hospital aid robot assisting patients, possibly guiding from one room/area to another
- Railway station guide providing directions to departure platforms and exits to city streets
- A service courier robot delivering mail or packages to offices and residential homes

Figure 3. Two very different Physical Environments for a Service Robot (Patient waiting area and hallway of a typical hospital floor [4](left). Heavy foot traffic at Paris Gare du Nord train station [5](right).)

It is obvious from above examples that general scenarios are not insightful in revealing underlying operational procedures and robotic technologies needed for a successful outcome. As the name suggests, they are "general" in describing a navigational event. In fact, from the above examples, we can see that the role of a robot guide in a hospital setting or at the train station appears similar even though the former takes place in an indoor cluttered office space (Figure 3, left) while the latter is a crowded, mostly open-space environment with disorderly human traffic patterns (Figure 3, right). Hence from an operational viewpoint, they are very different. On the other hand, the service courier and the pizza delivery robots are almost identical in their operational roles, the physical environment and human behavior profiles they each encounter. Nevertheless, general scenarios are still useful when communicating with the following target audiences:

- *General Public*: When a layperson hears that the Crowdbot team is developing a robot that moves and co-exists among humans, the immediate response is: "What's the purpose?" or "Why is it behaving this way?" When a robot's behavior is portrayed by telling a story in an acceptable social context via a general scenario, members of the general public are more receptive and accepting, and thus are more likely to tolerate or participate in actual tests.
- *Potential Customers*: How robots are used in a socially setting in the near and far future is anybody's guess. Promoting a robot for a specific task is also outside the scope of the Crowdbot project. However, the team cannot interact and explore potential use cases with a potential customer unless we have a plausible story to tell and the general scenario description is a meaningful and effective ice-breaker.
- *Stakeholders*: Stakeholders are individuals, groups and other entities that have a personal, professional or institutional interest in the Crowdbot project. A good example of stakeholders' concern is the safety and ethical implications of robots co-existing in human-only environments. In an indoor office setting, local, state and national level safety laws (c.f. OSHA Laws and Regulations [26]) prohibit the placement of bicycles nor the roaming of animals nor crying babies. Using general scenarios, the Crowdbot team communicates with external stakeholders in resolving potential ethical issues and drafting new standards proposals for safe operation of robots in cluttered and crowded environments.

Further details of general scenarios and their ethical implications are detailed in our companion report, *D6.1: Overview of Risks when Using Robots in Crowds* [6].

Now that we have provided clear examples delineating the differences between an operational from a general/social scenario, we return to the original pizza delivery scenario to elaborate further on the required operational procedures. One possible navigation scenario of the Domino's robot is as follows: A pizza is delivered from a local Domino's bakery store (START marker) to a customer staying at a nearby hotel (FINISH marker). Other intermediate steps are:

1) *Transitional* (stop-go-merge) Operation: The robot starts from the store front of a local Domino's pizza store. It leaves the store premise and merges onto the sidewalk with human pedestrian flow and heads in the direction of the customer (see Figure 4, left).
2) *Flow with Traffic* Operation: The robot traverses across city streets and pedestrian sidewalks. In some cases, it follows the human crowd. In another case, it negotiates with the cross traffic (i.e. vehicles) when it has to cross a street (see Figure 4, middle).
3) *Mapping, Localization & Path Planning* Operation: The robot has knowledge of a path it has to take to arrive at the hotel entrance where the customer is staying. The finish position can be the hotel room of the customer (at a higher floor level) or merely the reception desk (ground level) or the front entrance area of the hotel.
4) *Safety & Robustness* Operation: During the entire journey the robot must be mindful of potential hazards: collision with pedestrians; collision with man-made and natural objects; getting stuck in a pothole or train track (see Figure 4, right); falling down via the bank of a sidewalk; getting lost and unable to arrive at its next goal location, etc.

Figure 4. Task-specific Operations of a Pizza Delivery Service Robot (Robot traversing along with pedestrians on a paved walking path [28](left). Robot attempting to cross along the zebra against vehicular cross traffic [29](middle). Robot maneuvering over city tram rail tracks and vehicular traffic [30](right).)

The relationship between a general scenario and its corresponding task-specific robotic operations are shown in the top half of Figure 5. That is, every general scenario can be described in terms of an equivalent operational scenario which is a succession of task-specific and goal-oriented operations arranged in a specific order:

General Scenario -> Operational Scenario = (List of Operations + Execution Order)

At each intermediate step, a navigation goal can be defined in terms of its task-specific operation. For example, for the Domino's robot, its initial stop-go-merge operation is equivalent to reaching Goal 1 from its START position.

In general, the order of execution (of operations) is important. For example, the operation of a robot exiting via a door (say, from a furnished and cluttered office room) and walking along a corridor requires a different set of technologies from the case where a robot walks along the corridor first and then finds an exit via a door to an office room.

As shown in the bottom half of Figure 5, each task-specific operation is associated with one or more technology profiles. A *Technology Profile* is a unique operational procedure that lays out all the required technological components as well as their integration for a particular robot such that the desired goal is met.

Task-Specific Operation -> Collection of Technology Profiles (robot specific)

In essence, a technology profile is a recipe description for cooking an operational dish. Since the ingredients in the recipe are derived from the robot itself (its sensors, its processors, its software modules, its body frame and other hardware components), a technology profile is relevant and meaningful only when paired with a specific robot and its technical specifications. Therefore, the same operation for robot navigation (say, entering/exiting a door) may require a different technology profile for a humanoid robot from that of a smart wheelchair.

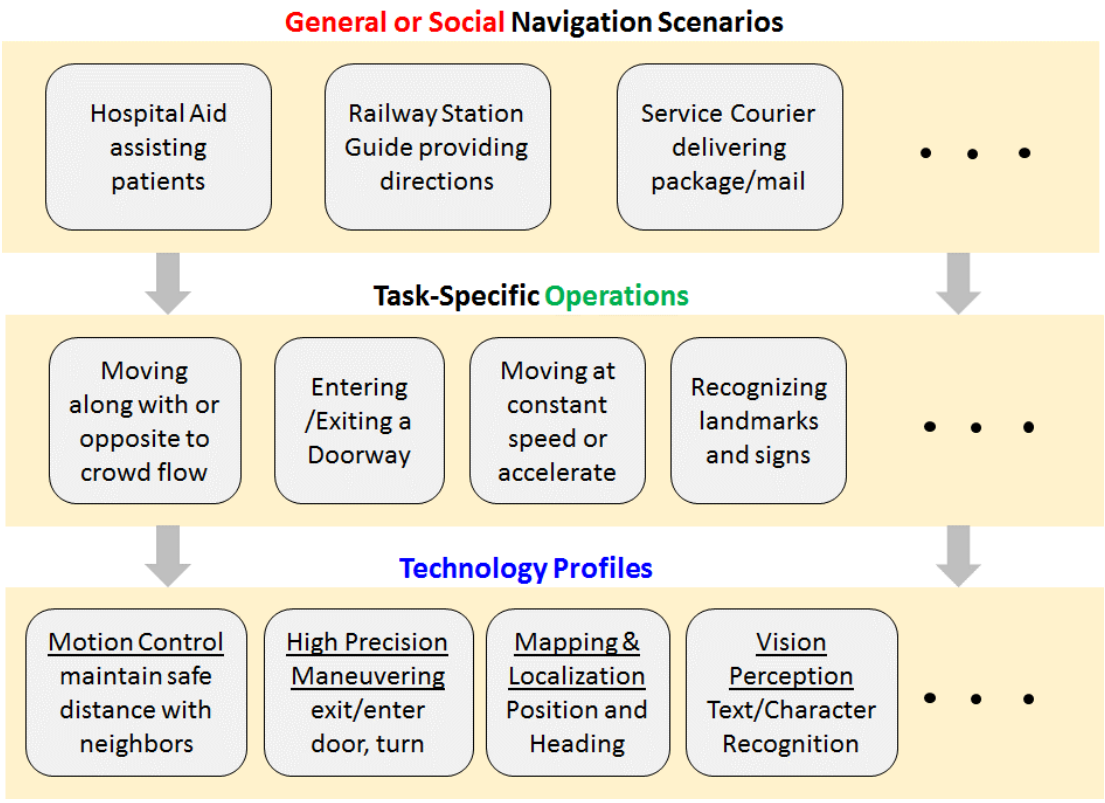

Figure 5. Categorization of a Different Navigation Scenarios

Note that this report is a requirements document and thus we do not stipulate or promote a certain type of sensing or processing technology, or robotic hardware design to fulfill a particular operational objective. We, however, do take into account specific hardware limitations that exist in some types of robotic platforms and are mindful that certain operational tasks cannot be handled by such robots. In this case, this operational task is excluded from its originating operational and general/social scenarios. We again use the pizza delivery robot to give a final example.

In the Domino's pizza delivery scenario, we stated earlier that the finish location can be:

1) The customer's hotel room (at a higher floor level)
2) The reception desk at ground level
3) The hotel entrance area (possibly outside the main entrance door)

It is apparent that the operational complexity for each option is different from the other two. Equivalently, different technology profiles are needed to fulfill the goal in each option. Option 1 is the most difficult since it requires the robot to enter through the main door, operate an elevator to reach the desired floor level and navigate in the corridor and accurately locate the final position ─the customer's hotel room. In contrast, option 3 is the simplest. The customer or a hotel staff must retrieve the pizza from the robot just outside the main entrance. Figure 6 shows three different types of robots (ETH Zurich's ANYmal [7], Savioke room service robot [8] and a TUG robot [10] for luggage delivery), all capable of using an elevator. The ANYmal robot is equipped with a limb to press the elevator's buttons while the other two are embedded with radio devices to wirelessly communicate with the elevator operator to request elevator use privileges.

Figure 6. Three different types of robots with technology profile to use an elevator (ANYmal robot with dexterity and precision to press the elevator button [7](left). Wally room-service Robot using the elevator via its radio communication module [8][9](middle). TUG luggage delivery robot maneuvering its way to the elevator [10][11](right).)

For a robot without either of these technology profiles, this type of operation is excluded from the available list. In **Section 2.3,** we outline the operational scope and technical capabilities of our robots. In short, none of the Crowdbot robots support technology profiles that require high-precision movement of body parts. Interaction with humans via head and joint movement, vision sensors, audio and visual cues and communication devices applies to the humanoid Pepper only.

As the project evolved and progressed, diverse technology profiles have been precisely identified: “Reactive navigation”, “Low-level control”, “Social signaling/Gestures”, “Human Motion Prediction”, “FollowBot”, “Tracking”, “Lidar FLOW”, “Reinforcement Learning”, “High-level Multi modal Behavior Planning”, “Global Planning”, “Physical Interaction”, “Shared Control”. Those technologies profiles developed within the CrowdBot project to provide crowd navigation capability are detailed in the **Section 4.**

Those technology profiles need proper evaluation in an adequate context.

Therefore, scenario categories have emerged: “Static”, “Flow”, “Sparse”, and “Mixed” (Figure 7). This categorisation has been thought to be strong enough to englobe the use cases in which the developed technology profiles are relevant.

The following cross table (Table 1) gives the correspondence between each technology profile, scenario category, and CrowdBot robot on which the implementation is done.

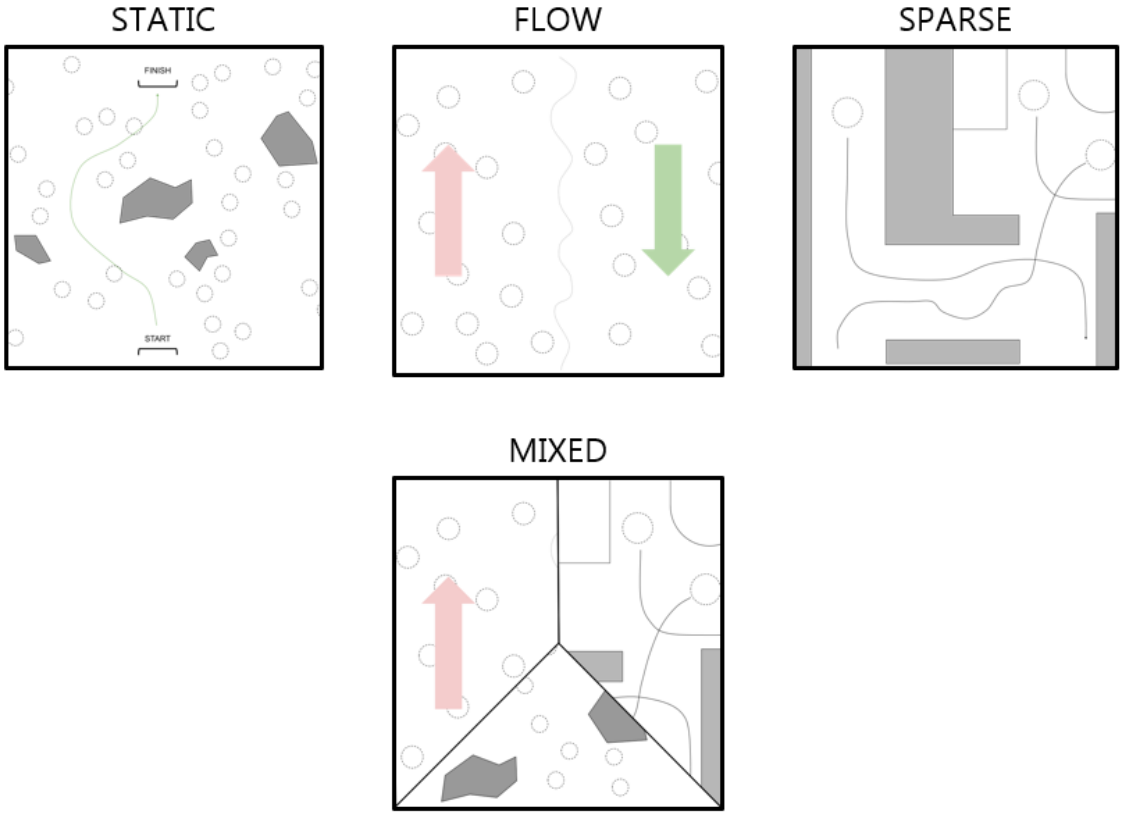

Figure 7. Schemes of scenarios

Table 1. Scenarios associated to technology profile for the purpose of evaluation with specified robots

Technology profile		STATIC	FLOW	SPARSE	MIXED
Reactive	EPFL	PCQ	P	Q	
Low-level	ETH	P		C	
Social Signaling / Gestures	SBRE	P	P		
Prediction	INRIA		WP	WPC	
Followbot	INRIA		P	P	
Tracking	RWTH	PWC	WQ	WCP	
LiDAR FLOW	RWTH		PC		
RL	ETHZ			P	

High-level Multi-mode	ETHZ				P
Global Planning	ETHZ	P		C	
Physical Interaction	SBRE			P (at SBRE)	

P: Pepper / Q: Qolo / W: Wheelchair / C: CuyBot

2. Scenario Requirements Overview

This section is reported from *D1.1*. and introduces some terminology for scenarios requirements descriptions. This has not been changed during the progression of the project.

In the previous section we provided ample detail and relevant examples to clarify the terms "operational" and "general/social" scenarios. Both terms are useful and serve different purposes in conveying the main objectives of the Crowdbot project. In this section we introduce another related term called the "test" scenario which is used to derive *requirements*. We will use mathematical notations in order to provide concise definitions of general, operational and test scenarios.

If we denote the list of all conceivable general/social scenarios as **W**:

$$\mathbf{W} = (W_1, W_2, W_3, \ldots)$$

and the equivalent operational scenarios as **V**:

$$\mathbf{V} = (V_1, V_2, V_3, \ldots)$$

Each operational scenario V can be described in terms of several individual goals bounded by START and FINISH markers. This is denoted as (without indexing)

$$V = [\text{START} \rightarrow G_1 \rightarrow G_2 \rightarrow \ldots G_k \rightarrow \text{FINISH}]$$

START, FINISH and Goals are space-time milestone posts. To transition from one post to its successor, a robot must complete a task-specific operation. To reach Goal *m*, it carries out Operation m (O_m). Of course, O_1 is the first operation initiated from the START marker. Likewise, the $(k+1)^{st}$ operation (O_{k+1}) is executed to reach the FINISH marker:

$$O_1: \text{START} \rightarrow G_1, \qquad O_m: G_{m-1} \rightarrow G\_m, \qquad O_{k+1}: G_k \rightarrow \text{FINISH}$$

An equivalent representation of an operational scenario in terms of task-specific operations is:

$$V = [\, O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow \ldots O_k \rightarrow O_{k+1} \,]$$

A key takeaway from above description is that the sets **V** and **W** for operational and general scenarios are very large, and possibly infinite in size whereas the set of robot operations $\{O_m\}$ is finite. Let's denote this set of operations as **D** (i.e., D_m for "do this operation *m*"):

$$\mathbf{D} = (D_1, D_2, \ldots, D_p)$$

Every operation O_m belongs to the set D of finite entries up to **p**. Furthermore, if we describe all possible technology profiles required by a particular robot to successfully complete a goal as

$$\mathbf{T} = (T_1, T_2, \ldots, T_n)$$

where T_m is a unique technology profile of type *m*, then **T** is also finite.

Referring to the elevator example in **Section 1**, we deduce that floor-to-floor navigation is an operational task (say, D_m) and its associated technology profiles are radio communication (say, T_j) or mechanical button press (say, T_k). Of course, each D_m is usually associated with more than one technology profile. In our example, for a robot to press an elevator button, other technology profiles such as high-precision vision to locate the button and high-precision motor control to move the limb/finger to the right position are required. The implication of these realizations to Crowdbot robots is two-fold:

1) If a certain robot is lacking a specific technology profile, then the operation that requires its execution will not be successful. Therefore, such operation must be excluded when testing and evaluating an operational scenario. Even better is to exclude operational scenarios that involve such an operation due to the fact that operational scenarios are operation-order dependent (see **Section 1** for additional details).
2) Since each operation relies on the execution of one or more technology profiles in unison, test results suffer from compound effect; that is, when there is a test failure, it is difficult to determine the root cause or culprit that is responsible. For this reason, tests should be structured in a

hierarchical fashion where each base test is used to evaluate and validate one or two technology profiles only.

We are now ready to provide a concise definition of *Test Scenario*. As shown in Figure 8, there is an equivalence mapping from a general to an operational scenario. However, an operational scenario test cannot be executed successfully using a particular robot due to its lack of required technology profiles. Furthermore, even if a robot supports such technology profiles, a test should be excluded if the cause of failure cannot be analysed without ambiguity. Therefore, test scenarios are a subset of operational scenarios where we take into account:

- Available technology profiles of the robot under test
- Other test constraints (physical environment, human participants, etc.)
- Clear and concise test objectives that lead to further enhancements to the robot

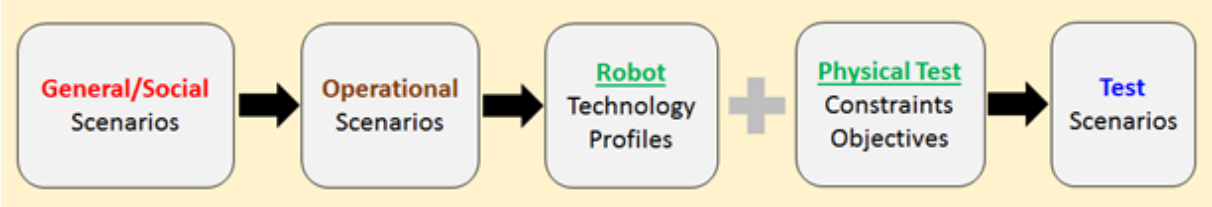

Figure 8. Relation between General and Test Scenarios

From this point forward, we will use the term "scenario" to refer to a test scenario. If there is potential for confusion, we will explicitly state whether a scenario is general, operational or test. **Section 5** covers all aspects of (test) scenarios relevant to the Crowdbot project. In fact, there are only seven main scenarios, denoted as:

$$\mathbf{S} = (S_1, S_2, \ldots, S_7)$$

Each main scenario can have a number of derivative or sub-scenarios. For test and evaluation purposes, there are requirements associated with each scenario. They are all listed and elaborated in **Section 6**. The notational convention is as follows. All requirements associated with Scenario *m* are denoted as $\mathbf{R}_m$ and the complete set of all test requirements is **R**:

$$\mathbf{R}_m = (R\ m.1, R\ m.2, R\ m.3, \ldots), \quad \mathbf{R} = (\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \ldots)$$

Before leaving this section, we have one more important topic to discuss. Thus far we have provided reasoning and justification (mostly due to technology constraints) for conducting well-controlled test scenarios instead of the more realistic operational or general scenarios in real-world environments such as a hospital floor or a railway station platform. There are also obvious logistical, financial and legal reasons that prohibit the team from conducting such real-world tests. Nevertheless, our tests and gained insights are meaningful and of value to stakeholders only if we can infer or deduce from test results their applicability to operational or general scenarios. In the commercial industry, this step is known as *requirement trace*. We noted above that a set of requirements $\mathbf{R}_m$ is associated with each test scenario $\mathbf{S}_m$. In reverse, we can specify an operational or a general scenario in terms of its requirements from a user or customer point-of-view (as opposed to goals and operations for the technology developers). Using mathematical notation,

Requirements for V or W **->** R *i.j*, R *m.n*, R *p.q*, . . . **(Test Scenario Requirements)**

That is, we can trace all requirements from test scenarios that can be used to define an operational or a general scenario. Note, however, some requirements for V or W may not exist in the set **R**. In this case, they are known as *deficiencies* to be met by further technology enhancements by the Crowdbot technology team or carried over to future research teams.

2.1. Requirements Authority

In the beginning of **Section 1** we raise the question **Q2**: *Who is preparing scenario requirements and evaluating test outcomes*? This section provides the answer. Unlike commercial product development contracts, the Crowdbot project is focused on research and innovation. We do not have an external customer that prepares a requirements document as part of contractual obligations; however, we are guided and overseen by our funding authority, the European Commission and are chartered to work closely with external stakeholders. Details of our interaction with stakeholders are discussed further in **Section 7.3**. Here we provide a brief summary.

For the Crowdbot project, the internal Test & Evaluation (T&E) team is in charge of requirements specification (including this report), interaction with relevant stakeholders, preparation of potential use cases, and foremost, evaluation of test data and delivering recommendations for further technology enhancements. For the purpose of this discussion, the Crowdbot team as a whole can be grouped into four sub-teams (see left side of Figure 9):

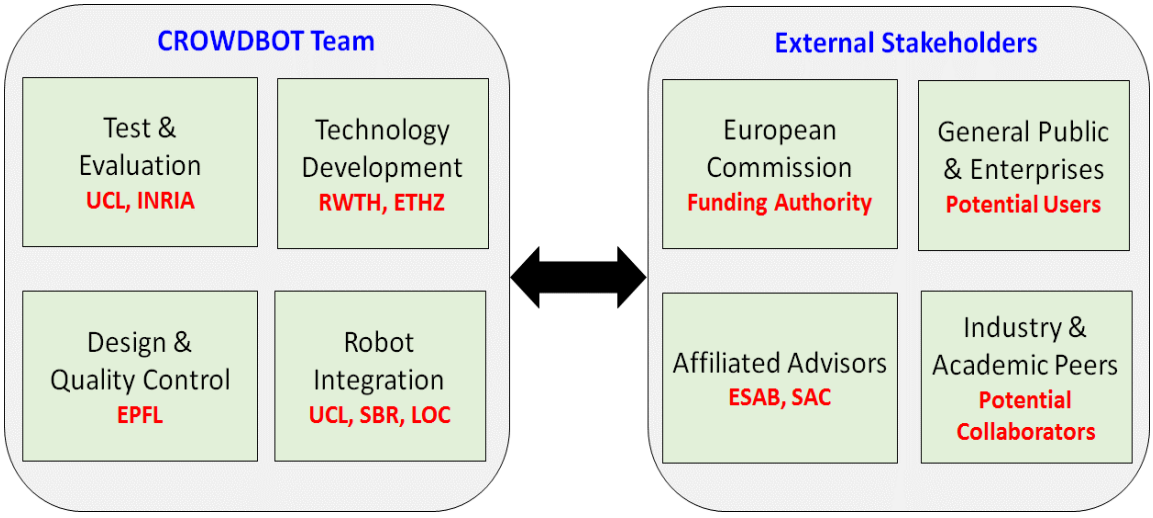

Figure 9. Crowdbot team composition and interaction with external stakeholders (ESAB: Ethics & Safety Advisory Board, SAC: Scientific Advisory Committee)

The lead partners in each sub-team are highlighted in red. The Test & Evaluation team, which is responsible for this report as well as evaluation of test results, is led by UCL and INRIA. Two other partners RWTH and ETHZ are the main developers and enablers of key technologies in sensing, processing and navigation. They lead the Technology Development (TD) team. Integration of such technologies into specific robots are assigned to the owners of such robots: UCL (Smart Wheelchair), SBR (Humanoid Robot) and Locomotec (CuyBot), collectively known as System Integration (SI). The T&E team is concerned with the maturity and adaptability of new robotic technologies developed by RWTH and ETHZ for specific navigation tasks whereas the Design & Quality Control (QC) team led by EPFL is focused on the overall robotic product. Specifically, it is in charge of safety and design enhancements, interaction with stakeholders concerning standards and use cases plus potential ethical issues that may arise when deploying particular types of robots in different social environments. Testing of scenarios (venue planning, logistics and execution) is carried out by the entire team. As can be seen from the sub-team composition in Figure 9, different partners are in charge of each sub-team. This assures integrity, fairness and neutrality in assessing and judging work products of each sub-team by remaining partners. Furthermore, the team also schedules periodic review sessions with EC and solicits feedback from external stakeholders as an additional layer of checks and balances.

2.2. What's New

Mobile robotic research has been ongoing for several decades now. Navigational tests using electric wheelchairs started from early 1990's when proximity sensors (e.g. sonar) and portable computers were within the budget range of academic researchers. Nowadays, with the ubiquity of vision sensor modules, fast microprocessors and cheap memory resources, the focus has shifted from hardware-constrained robot prototype tests to exploitation of advanced artificial intelligence algorithms and real-time fusion and interpretation of multi-sensor data streams. The emphasis now is less on hardware design and more on software technologies. Any suitable robotic platform ─commercial model or an academic prototype─ is adequate as long as it supports a fairly open software architecture and additional hardware can be augmented with little difficulty. In fact, this same approach is taken by the Crowdbot team. Several members of the team have also gained knowledge and experience by participating in prior and ongoing European/EU national research projects such as EUROPA, EUROPA2, SPENCER, CV-SUPER, ADAPT, Mummer, STRANDS, iCub and so on. Nevertheless, we highlight several unique features and goals of Crowdbot that stand out when compared to other mobile robotic projects:

- *Use of Multiple Robots*: Crowdbot will test its navigation system and associated technologies on three different robotic platforms. The emphasis is obvious: the team's goal is to develop robotic software that is portable or adaptable across various kinds of autonomous mobile machines. Integration of software onto an existing robotic hardware is always a challenge. The team will pursue such integration using three very different robotic platforms: a closed, proprietary commercial humanoid (Pepper by SoftBank Robotics), a commercial electric wheelchair (Quickie mid-drive model) with extensive room for modification and a new service robot (CuyBot by Locomotec) in prototype/test phase.
- *Multi-Disciplinary Effort*: Robotics is already a multi-disciplinary field. Here, we are referring to cross-pollination of robotics and human crowd analysis. The study of human crowd behavior and motion profiles is the convergence of several disciplines: psychology, thermal dynamics, traffic engineering, mathematics, and computer science. In the past humans are treated merely as dynamic objects in a robotic test. Likewise, crowd analysis is restricted to human-to-human interaction only. The Crowdbot team has expertise in both robots and crowds; this synergy will be used to tackle an emerging problem in mobile robotics: social navigation of robots among dense human crowds.
- *Technology Transfer*: There are three main emphases here: 1) Academic to Commercial: Since two of the seven partners are non-academic (commercial) members, the team aims to demonstrate methodical transitioning of stand-alone intelligent software (developed by academic partners) into an embedded module of a commercial-grade robotic system. 2) Platform Portability: The same intelligent software as well as sensing and computing hardware will be adapted from one robotic platform to another. In general, this is not a simple task since each robot has its own design constraints (dimension, shape, software architecture and embedded hardware). 3) Current generation to the Next: Two rounds of navigation tests are planned and it is expected that 2nd round tests will undergo further enhancements based on lessons learned from round 1. Since sensor and computing technologies are advancing at a rapid rate, we also anticipate upgrading both sensing and computing technologies in 2nd round tests. The team will then gain insight into technology migration potential using the same robot.

In short, all three goals of Crowdbot aim to maintain its relevance after the completion of the project. They facilitate subsequent follow-on work by Crowdbot researchers and their peers.

2.3. What's Outside of Scope

The following system features are neither available nor activated among Crowdbot robots for navigational operations. Exceptions are noted whenever applicable.

- *Body Part Movement*: All Crowdbot robots move on wheels. Crowdbot sensing and navigation technologies may not apply to bi-pedaled (two-legged) or multi-legged robots with lateral and vertical motion profiles while moving forward. Humanoid robots tend to support pitch, raw and roll

movements of certain body parts. These features are generally not exploited with the exception of head, limb and torso movement of Pepper in social navigation scenarios (see **Section 5.7 of D1.1**).

- *Human-Machine Interaction (HMI)*: Crowdbot robots use Human-Machine Interfaces (HMI) for in-seat (e.g. wheelchair) or remote control for testing, monitoring and troubleshooting by a human operator. However, once a test event starts, HMI for navigation tasks by a human is permitted for the wheelchair only. (See below for the restriction against tele-operation.)
- *Radio Communication*: It is anticipated that two-way wireless communication equipment will be used during various test phases to monitor system parameters as well as activate the kill switch when there is a safety concern. However, Crowdbot robots are designed to be semi- or fully-autonomous, self-sufficient and self-powered. Therefore, they cannot use off-site processors (e.g. cloud computing), nor real-time information exchange (e.g. map updates, navigation and timing data). They are not equipped with electronics for indoor or outdoor positioning or timing services (e.g. Global Positioning System, ultra-wideband localization, cellular networking timing). Here the reason for exclusion of such system features is not due to technology limitation. We do not allow circumvention of a technical problem by utilizing an easier work-around or by substitution of another technology.
- *Other Modes of Communication*: In socially interactive scenarios for Pepper and the wheelchair user, verbal and non-verbal communication with by-standing or moving humans using audio-visual cues (via its speakers and light/video displays) or use of a voice synthesizer and microphone to talk and listen to sound and speech is permitted. Of course, the wheelchair operator is allowed to communicate with other humans.
- *First-Person View & Teleoperation*: Since Crowdbot robots are likely to be equipped with vision cameras and two-way communication devices, they naturally lead to the possibility of First-Person View or tele-operation. This option is not permitted for navigation tasks. The robot must navigate using its own programmed or learned intelligence or be under continuous control of the human operator in the case of the smart wheelchair.

3. Review of Our Robots

This section reports some evolution of the robots we consider in the CrowdBot project. Updates with respect to *D.1.1* concern: the evolution of the Pepper robot used by the partner ETHZ, the description of Qolo robot which is new to the project, the evolution of the wheelchair used at UCL, as well as the CuyBot robot from Locomotec. As a result, most of the section content has been updated.

The main message to be conveyed in this section is that a single set of requirements cannot be prepared for all three Crowdbot robots. The reader is referred to our companion reports *D2.1: Sensor Specifications* [15] and *D5.1: System Architecture* [16] for more in-depth technical description of our robots. Here the focus is to highlight features that are different and unique in each of our robots (from an operational perspective) and how they may result in specification of separate requirements for each robot. Key features relevant to requirements specifications are:

- *Frame & Body Composition*: A robot's physical dimension is critical for obstacle avoidance and safety measures when navigating through clutter and human crowd. The Pepper robot has the smallest footprint (size and weight) of a child while the CuyBot is similar to that of an adult. The wheelchair is the largest and heaviest robot with a rectangular, rigid frame. The composition of body parts limits the type and number of sensing and computing modules that can be attached or augmented to a particular robot. This constraint is applicable to both mechanical and electrical enhancements that may be necessary for successful navigational tasks.
- *Motion Profile*: Some of our robots are holonomic while some are non-holonomic. Even though all robots can move forward and backward, they may require a 180-degree turn to reverse its heading since vision sensors do not have a 360-degree field of view. In such a scenario, a non-holonomic robot requires additional clearance space to complete a turn.
- *Technology Integration*: The issue here is forward/backward compatibility and openness of the onboard digital system to technology enhancements. If the system is proprietary, new sensing and computing technologies cannot be integrated. The backward/forward compatibility issue is related to the integration of Crowdbot hardware and software onto the robot's pre-existing hardware, software and interconnect (data bus) architectures.

3.1. Humanoid Robot (HR)

3.1.1. Pepper Description

The Pepper robot (left, Figure 10) is the model chosen for Crowdbot navigation tests. Other humanoids are also shown in Figure 10 to convey that humanoids come in different shapes, sizes and locomotion profiles. The Romeo (middle, Figure 10) has legs (i.e. bi-pedaled or two legs) whereas the remaining two are wheel-driven. As noted in **Section 2.3**, Crowdbot sensing and navigation technologies are limited to robots on wheels. As shown in Figure 11, a humanoid can be decomposed into several body parts: base, body, limbs and top or head. The PR2 robot (Figure 10, right) is not very human-like but it has limbs (arms and claws) and a top where most of its vision sensing modules reside. As noted already, Crowdbot navigation tasks do not exploit dexterity or limb movements of a humanoid. The body parts most relevant to Crowdbot are base, body and top. The joints/limbs are not exploited (see Figure 11, left). Sensors exist as embedded (already integrated inside the robot), attached (placed or mounted externally to a body part) or augmented (attached or substituted in place of the previous embedded sensor).

When humanoids are compared to service robots (**Section 3.3**), they are both similar in many attributes except that the latter lacks joints/limbs and may or may not have a prominent top/head.

For the sake of technology integration, the Pepper humanoid is best viewed as a commercially available, closed-form marketplace robot that will undergo technology enhancements for navigation in a crowd. Its base (wheels, motors and power electronics circuitry) cannot be modified and is used as-is. The digital electronics boards in the main body and head are also tightly integrated with embedded sensors and thus cannot be replaced or augmented. The availability of computing resources for running additional tasks and interconnection with external devices is highly dependent on a particular model and the age of its circuitry.

On the other hand, a service robot or a smart wheelchair is a prototype that is already modular for augmentation or can be modified from its base design to support crowd navigation.

Figure 10. Different types of Humanoid Robots (The three-wheeled Pepper robot by SoftBank Robotics [13] is used in Crowdbot (left). The bi-pedaled Romeo robot by SoftBank Robotics [13](middle). The eight-wheeled PR2 robot by Willow Garage [14](right).)

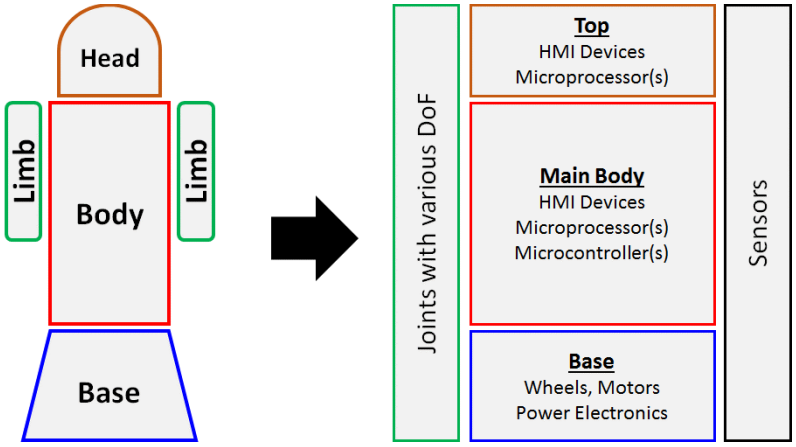

Figure 11. Framework & Composition of a generic Humanoid Robot

3.1.2. Pepper Augmentation

The Crowdbot Pepper robot has undergone two rounds of sensor augmentation as described in D5.2 and D5.3. Here we provide a summary of the augmented sensors, sensor placement and electronics integration. For full details, please refer to the relevant deliverables.

Additional navigation sensors in the form of two 2D planar LiDARs mounted around the base of the robot as well as a ceiling-facing visual-inertial (VI-)sensor [36][37] mounted on the back of the robot head have been integrated into Pepper (see Figure 12). These additional sensors provide robust and complementary mapping and localisation modalities. The LiDARs provide high fidelity information on obstacles in the robot's lateral frame, while the VI-sensor maintains localisation quality when there are severe occlusions from dynamic obstacles around the robot that prevent the LiDARs from observing the static environment.

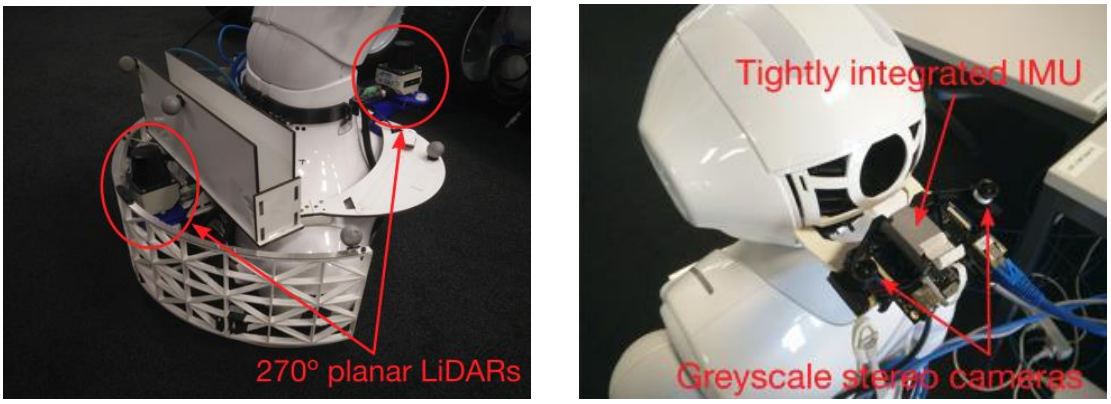

Figure 12. Two planar LiDARs are mounted front and back of Pepper's base. (left) Each SICK TiM571 LiDAR [38] has a 270° field of view, together enabling 360° planar coverage of obstacles around the base. Upward mounted visual inertial sensor enables localisation and mapping using visual features on the ceiling(right).

Pepper also includes an additional forehead-mounted RGBD camera (Intel RealSense D435 [39]) as shown in Figure 13. The custom mount was designed and produced by SBRE. The sensor provides colour as well as depth images at approximately 30fps. This data stream is used in the Crowdbot perception pipeline to perform pedestrian detection and tracking.

Figure 13. Forehead-mounted Intel RealSense D435 (which provides colour and depth images)

The LiDARs and VI-sensor as well as the core Pepper robot computer are connected to an external laptop via an ethernet switch. An external LiPo battery powers the ethernet switch, as well as the LiDARs and VI-sensor with power routed via a custom power board. The Intel Realsense is powered through its direct USB connection to the external laptop. The Pepper robot also contains its own internal power source.

In addition to these electronics augmentations, we have also installed a 3D printed fender around the robot base (also visible in Figure 12 (left)). The purpose of the fender is two-fold: first, since Crowdbot Pepper will be required to navigate in close proximity to humans, the fender will protect human participants in cases of collision by dissipating the collision force across a larger area. Second, the fender is mounted such that contact forces are routed directly to the robot base and do not directly impact the sensors and electronics installed around the robot base.

3.2. Smart Wheelchair (WC)

3.2.1. Wheelchair Description

A smart or an intelligent wheelchair is a commercially available electric wheelchair that has been modified to support semi-autonomy features for navigation. In the past, researchers have custom built electric wheelchairs from the ground up when developing navigation technologies and testing autonomy features. However, we can safely assume that current and future smart wheelchairs will be augmented or modified

versions of commercially available electric wheelchairs. Three different models of an electric wheelchair by Quickie Wheelchairs [12] are shown in Figure 14. Such wheelchairs are also known as *motorized* or *power/powered* since they all run on battery power and electric motors for mobility. The electric wheelchair chosen for Crowdbot is similar to the mid-drive model (left side, Figure 14) where the motorized wheels are placed in the mid-section of the wheelchair. It has casters as front guiding wheels and casters in the rear for anti-tilt. We also show other possible drive mechanisms: rear-wheel and front-wheel drives (middle and right respectively in Figure 14). Note that each electric model has a structure that is slightly different from the other two and this has performance implications when augmenting sensing and computing technologies for intelligent and safe navigation purposes.

Within the Crowdbot project, we have modified the Quickie Salsa M² powered wheelchair (through the addition of various sensors, processors, and software modules) to enable it to perceive the environment around it. The smart wheelchair control system runs across two computers; a gaming laptop used for image processing and more CPU intensive operations and an onboard Odroid X4 which interfaces with the motor controller and low-level Arduino controlled sensors. The RGB-D sensor and the Lidar both connect directly to the gaming laptop, whereas all the other sensors (four ultrasonic clusters, an IMU, joystick emulator, and wheel encoder) are connected to the Odroid, some of which are pre-processed by Arduinos.

We will revisit this topic shortly; we now cover the meaning of the term *autonomy* applied to the electric wheelchair.

Figure 14. Three different models of a power wheelchair from the same manufacturer (Push handles, exposed side panels, caster wheels to the side of footrest (left). Adjustable armrest, folding headrest, folding footrest in front of caster wheels (middle). Front-wheel drive, anti-tilt wheels in front, reduced space under seat (right).)

A smart or intelligent wheelchair differs from a conventional powered wheelchair because it is able to perceive the environment (to some extent) and is equipped with some degree of autonomy. Just like in the automotive industry and as first described by Sheridan and Verplank (1978) [27] several different levels of autonomy are possible. Smart wheelchair autonomy can be broadly categorized into three different types:

1) *Full Autonomy*: In this mode the human (in chair, remote location or aid) has no control over the wheelchair's motion behavior as it traverses from START to FINISH markers. Of course, the initial setup and programming of an operational scenario is done by a human. In practice, most end users prefer to maintain authority and develop skills where possible, so this type of control is not generally desired.

2) *Override Autonomy*: In this mode the human operator relies on the machine to achieve all goals in the navigation profile, but maintains the capability to halt, modify or abandon any ongoing or future operations. This is the kill switch option and the human operator can restart or resume an operation as appropriate. This option is also known as *reactive navigation*.

3) *Assisted Autonomy (Shared Control)*: In this mode both the human and the machine work in tandem by sharing the navigation task. When viewed in terms of space-time markers and goals, goals are achieved by combining the human input with the machine input to realize a final goal location output. If the human stops providing a continuous input, the wheelchair stops moving. In the literature this option is known more widely as *shared-control navigation* and will be the focus of the wheelchair control paradigms used in the Crowdbot project.

Options 2 and 3 are collectively known in the robotics community as semi-autonomous navigation. Option 1 can be viewed as machine-based navigation without a human-triggered kill switch. Note that different levels of autonomy exist because each is more suited to a particular type of wheelchair user. It is important to note that although wheelchairs are used by people with severe mobility impairments, many have compound and complex needs. For example, aside from the motor impairment, some users also have cognitive and/or sensory impairments. This topic is outside of the scope of this report and the reader is referred to these excellent sources [31, 32] for additional information regarding medical aspects of assisted wheelchair use.

From an operational perspective, these three options require different technologies and operational profiles for reaching a goal. Option 1 (full autonomy) is the most complex in design, use of sensors and computing resources since it assumes no human involvement. Option 2, reactive navigation, could be viewed as option 1 with a kill switch. Since the human operator has the final override authority, he/she must have the mental capacity and sufficiently fast reflex to make a quick judgment, but may lack mechanical means or long-term concentration to provide continuous control input to the system. Thus, option 2 is less complex than option 1 since it can rely on the human operator for safety concerns and against navigational hazards. Option 3 (shared control) assumes regular continuous human-machine interaction and the human operator plays a more significant role in the navigation task. Although one might imagine that having a human in the loop would lessen the design complexity and workload for the machine, it is actually still quite complex, especially given that the user input is likely to be fairly noisy (if the user input were perfect, there would be no need for shared control.) A typical example of assistance would be proactive obstacle avoidance e.g. helping to steer around a table. However, in some cases, the user may actually wish to dock to the table. This type of ambiguity makes the system design rather challenging and is only further complicated by the type of dynamic "obstacles" (people) that we will be dealing with in Crowdbot.

Referring to the three models (from the same manufacturer) shown in Figure 14, we observe slight differences in the overall body structure of one model from the other two. This variation is to be expected since a different model is tailored for a certain type of user. This, however, complicates our task of requirements specification. As noted in **Sections 2** and **3**, certain operations cannot be performed by a robot due to limitations in its technology profiles.

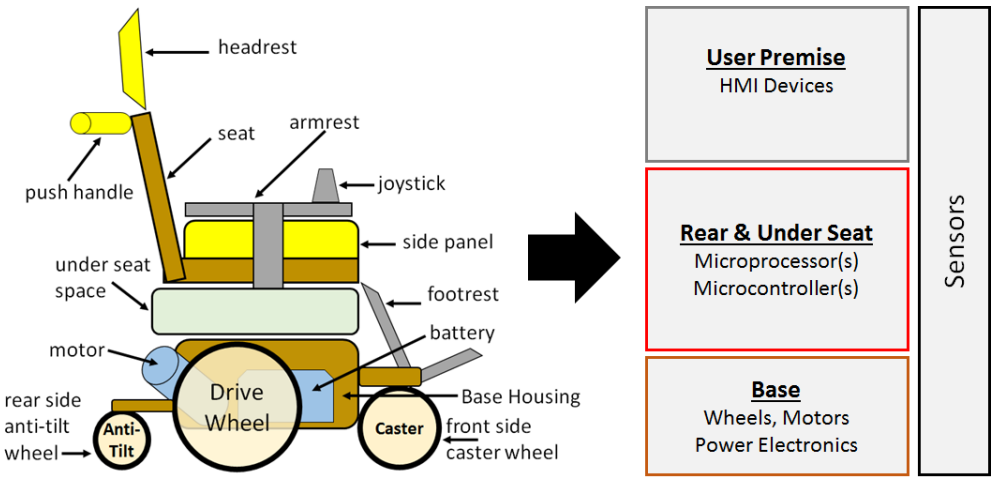

Figure 15. Framework & Composition of a generic Smart Wheelchair (Yellow-coloured items are optional in certain models; Gray-coloured items are non-rigid, i.e. movable)

For the electric wheelchair, we highlight in Figure 15 structural components that are rigid (consistent across all models) and those that may be optional (i.e. non-existent) in certain types. We also show structures that are non-rigid (movable, foldable, extendable, etc.) such that any sensor attached to them may change position relative to the fixed base. Unless noted in our requirements specifications, we assume the smart wheelchair is composed of fixed base structures (excluding yellow-colour parts) as shown in Figure 15. The available space for storage of computer and other electronics is also highly dependent on the model; the mid-drive model has the largest rear space due to its large anti-tilt casters in the rear. The rear space behind the back support can also be used to house augmented computing equipment but this area is generally reserved for storage of handbag or backpack of the operator.

Compared to a humanoid or service robot, the smart wheelchair is the most flexible in terms of hardware attachment and technology integration since none of these features exist in the baseline electric model. However, unlike its counterparts, the wheelchair exists in many different physical models (even from the same manufacturer), each with a unique frame and body structure, such that care must be taken in both design and integration of Crowdbot technologies such that intelligent modules tested in one model could eventually be transferred to or adapted to a different model. In Crowdbot, we will focus on developing and evaluating the proof of concept using the popular Quickie Salsa M^2 mid-wheel drive. However, an optional requirement item is included in the list (see Table 3) that addresses this specific topic: Applicability and adaptability of a technology profile from one model to another.

3.2.2. Wheelchair Augmentation

The Crowdbot Wheelchair robot has undergone several rounds of sensor hardware augmentation as described in D5.3 and **Section 3.2** above. Here, we provide a summary of the augmented sensors, structural modifications and electronics integration.

The main structural modifications are the adjustable sensor frame mount for the RGBD camera, an inverter to power the wheelchair's main processor (a laptop), and an emergency stop button. The Intel RealSense camera (Figure 16) captures images and depth information from the surrounding environment for use in tracking and route planning algorithms. The emergency stop button is a safety measure to disable the wheelchair in an emergency situation.

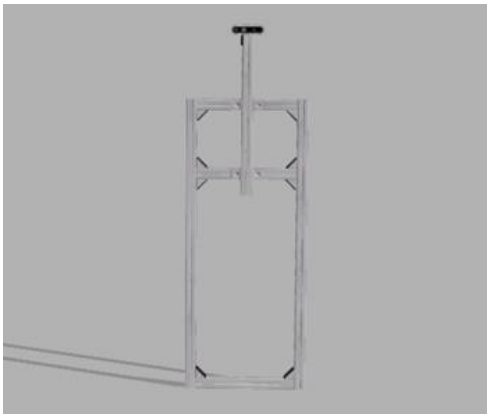

Figure 16. The Intel RealSense camera, mounted on a custom-built frame, which is then attached to the rear of the wheelchair

Several sensors were instrumented onto the wheelchair. An overview of the hardware integration on the wheelchair is shown in Figure 17.

Figure 17. CAD models (left) and smart wheelchair (right) with hardware modifications

The RGB-D sensor and the Lidar both connect directly to the gaming laptop, whereas all the other sensors (four ultrasonic clusters, an IMU, joystick emulator, and wheel encoder) are connected to the Odroid, some of which are pre-processed by Arduinos. The Lidar is physically mounted on the wheelchair footrest. The wheel encoders are mounted on the chair using a 3d printed housing, they are then connected via a pulley to a 3D printed cog attached to the wheel (Figure 18).

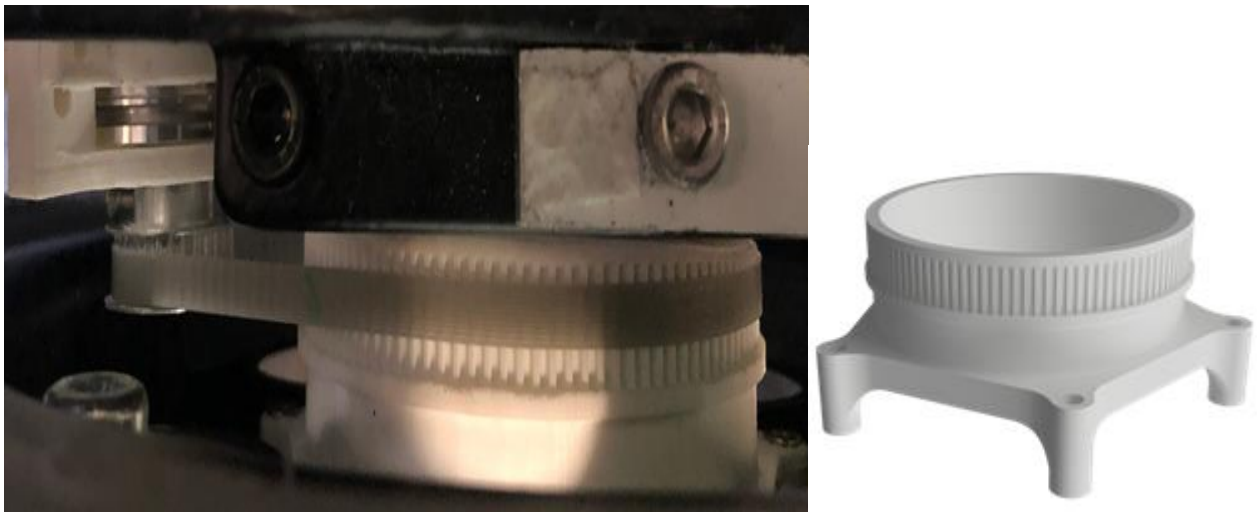

Figure 18. Custom built wheel encoders are installed on the wheelchair

The IMU (SparkFun 9DoF Razor IMU M0) provides angular velocity, linear acceleration and orientation data. A 3D printed casing was designed to help protect the sensor and mount it on the chair. The sensors are arranged in clusters of 3 at each corner of the wheelchair (Figure 19). Arduinos are used to read data from the sensors and publish readings as ROS messages. One Arduino reads the front two clusters and one the rear two.

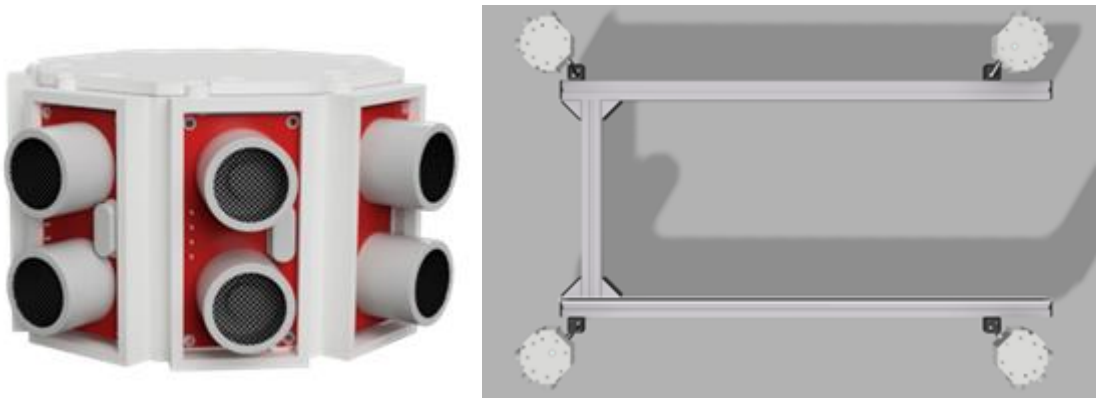

Figure 19. CAD rendering of our ultrasonic sensor cluster (left) and CAD rendering of four sensor clusters, mounted at the four corners of the sensor frame, which in turn is affixed to the wheelchair chassis (right).

The main drive interface of the Salsa M² quickie wheelchair is an R-Net joystick controller. This virtual joystick controller was programmed to function in two distinct modes namely the Crowdbot shared control mode and the manual manufacturers mode. The manual mode comes with four different profiles each with five different velocity acceleration selections. The virtual joystick provides an R-Net interface allowing the wheelchair to be driven via ROS velocity commands. The architecture of this virtual joystick interface is implemented via a Tiva microcontroller which interfaces with the drive control unit built into the wheelchair (via the R-net bus) and connects to the Odroid via USB. Essentially it forms a proprietary interface that is used to read the joystick messages from the R-Net bus and place motor velocity messages back on to the R-Net bus, which in turn are fed to the motor controller unit.

It is important to note that in order to send a velocity command to the motors a continuous input must be provided by the user via the joystick interface. This is indeed the case in the shared control semi-autonomy navigation paradigm.

3.3. CuyBot (CB)

The cuyBot robot developed by Locomotec is an example of a non-humanoid service robot. During the course of the Crowdbot project the robot has undergone multiple changes. The latest version is shown in Figure 20.

Figure 20. CuyBot robot

The goal of cuyBot was to build a compliant robot that can safely execute a range of service and logistics tasks. It features two main elements, an outer hull compliantly mounted via force sensors on the robot base, and an omnidirectional drive mechanism based on the newly developed so-called SmartWheels. SmartWheels are compact drive units consisting of two wheels with independently controlled direct-drive motors, enabling an omnidirectional, quasi-holonomic motion of the robot.

The general structure of the robot is typical for these kinds of devices, as depicted in Figure 22, consisting of base, body and top. In case of cuyBot the base and body are kind of combined with each other, connecting to the wheels and including battery, PC and other electronic equipment. In addition the top in form of the trunk of the robot carries additional sensors and HMI elements.

The robot includes the sensors used in Crowdbot, in particular one (or two) 2D Lidars at the front (and back) and two Realsense D435 RGB-D cameras in the trunk facing forward and backwards. Based on initial tests using Crowdbot software, the processing power of the robot has been increased with an Intel NUC i7 and a Nvidia AGX Xavier. The robot is still in the prototype phase and may be changed further.

Figure 21. Different types of wheeled Service Robots (Sharship's pizza delivery robot (left), Savioke's room-service robot (middle), Aethon's luggage carrier (right))

Other service robots similar to the CuyBot are shown in Figure 21. Unlike a humanoid, they come in many shapes, sizes and weight classes. We can, however, define all wheeled service robots into the physical blocks shown in Figure 22 (left): base, body, top and cargo space. In some designs there is no separate structure for the top module but is blended with the main body. The main electrical and electronic components are grouped into its physical structures (Figure 22, right). Compared to a humanoid, a service robot is likely to be equipped with a cargo space to carry objects; it lacks joints or movement of any body parts. Similar to all other robot types, sensors are placed as needed in all structural parts of a robot.

As evident from the three service models shown in Figure 21, not much can be said about the motion profile of this class since they vary from one model to another. Compared to a humanoid, a service robot to likely to have a sturdier and more stable base construction with sufficient battery power and motor torque to achieve higher acceleration and greater manoeuvrability. Some are targeted for indoor smooth surfaces only but others can navigate through rough terrain.

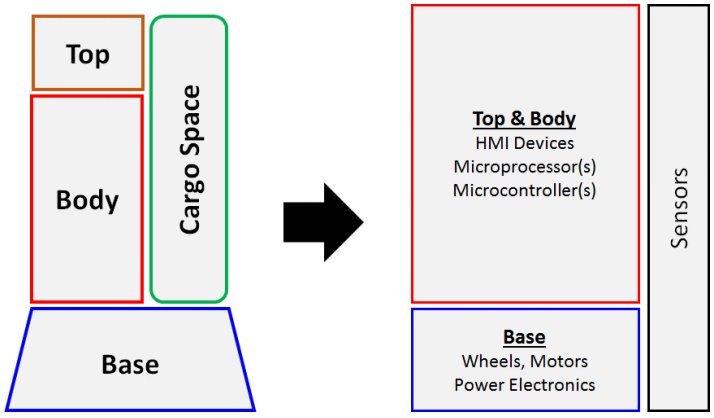

Figure 22. Framework & Composition of a generic Service Robot

3.4. Qolo

The Qolo robot developed by the artificial intelligence lab at the University of Tsukuba [Paez-Granados D. et al 2018, Eguchi Y. et al 2018]. could be considered a device very much like a powered wheelchair, as it accomplishes the same task of transporting a person with lower-body impairment. Nonetheless, Qolo is as well a standing mobility vehicle which falls in the category of person carrier robot defined by ISO:13482-2014. As such, it provides a different perspective for crowd navigation. In contrast to a wheelchair, a user standing in Qolo would be more visible at a distance, even within a crowd. Additionally, motion constraints would play a role similar to that of Pepper where acceleration limits are stricter for ensuring safety of the

user (preventing tip-over). Moreover, the user interface and potentially the shared control strategy are different in type and level of autonomy required.

Some examples of these type of vehicles which are becoming popular worldwide as a last-mile mobility solution, and short-mid distance commute proposals are standing scooters or smart scooters, as show in the Figure 23.

Figure 23. Different person carrier robots (a Segway standing scooter (rigth), and a Toyota e-scooter (left))

The objective of the Qolo robot is to achieve a standing mobility vehicle for users with lower body impairments, such as spinal cord injury (SCI), while allowing the usage of their remaining motion capabilities in the upper body. In contrast to powered wheelchairs or standing wheelchairs, no external power source is required for achieving the sit-to-stand or stand-to-sit transition [Paez-Granados D. et al 2018, Eguchi Y. et al 2018].

The robot Qolo is a slim design for a powered wheelchair with an exoskeleton that attaches to the user's lower-body (shown in Figure 25), and weights 35 Kg in total. The robot is composed of a mobile base with 2 standard wheelchair in-wheel motors, 2 rear caster wheels, and the exoskeleton base in the middle (Figure 24).

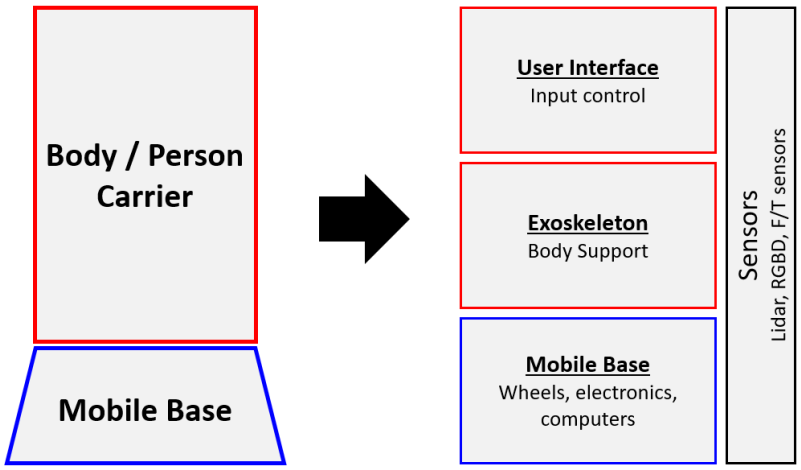

Figure 24. Framework & Composition of a generic Service Robot

For human input to drive the robot, Qolo provides 2 types of interface. First, an embedded hands-free input interface in the torso bar. This interface is based on an array of pressure sensors which allows to control linear and angular speed through upper body motions [Chen Y., et al 2019]. Second, a standard joystick control for wheelchairs is provided in case of need or preference by the user.

Currently, the controller of the robot is executed from an embedded computer UpBoard Squared (intel Celeron 2.4Ghz), integrated within the exoskeleton structure, which handles low-level control and user interface inputs.

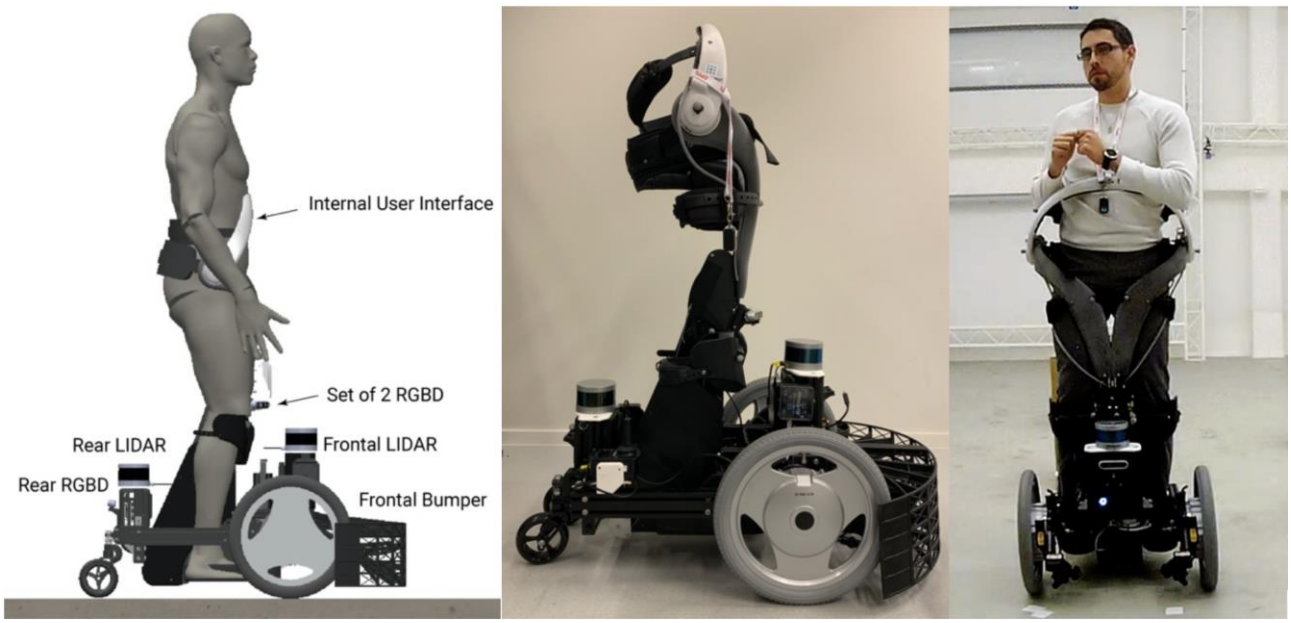

Figure 25. Qolo robot used in CrowdBot project (CAD of the robot with sensors (left), robot profile with full set of sensors installed (middle), user driving the robot (right))

For the Crowdbot project, we have modified the standard version of Qolo by integrating sensing and computing capabilities for onboard autonomous control of the robot. Following the recommendations in D6.2, three Intel RealSense D435, as well as, two 3D Lidar sensors Velodyne VLP16 are installed in the robot. Currently, two cameras are set forward and one heading rear of the robot, although, its location could be re-arranged accordingly to the desired testing scenario.

Moreover, we have included a frontal Force/Torque sensor (Botasys Rokubi 2.1) for contact detection attached to a bumper, which could be used for detecting small contact with feet or other surfaces, and potentially controlling for compliance in such scenarios (Figure 25 center).

For the integration of the lidars we have set a second UpBoard computer which can run independently of the low-level controller for a distributed control architecture. This second computer is embedded in the lidar's base for a compact configuration. Moreover, a Nvidia Jetson AGX is installed in the rear of the robot with its own compact power source, for processing up to 2 RGBD sensing information for people tracking.

We have followed a modular approach for distributing the computing power as proposed for the CrowdBot project robots, as shown in Figure 26. Where all the separate components of the control have been constructed with a ROS communication architecture, so that, it would be compatible with all the partner's developed modules.

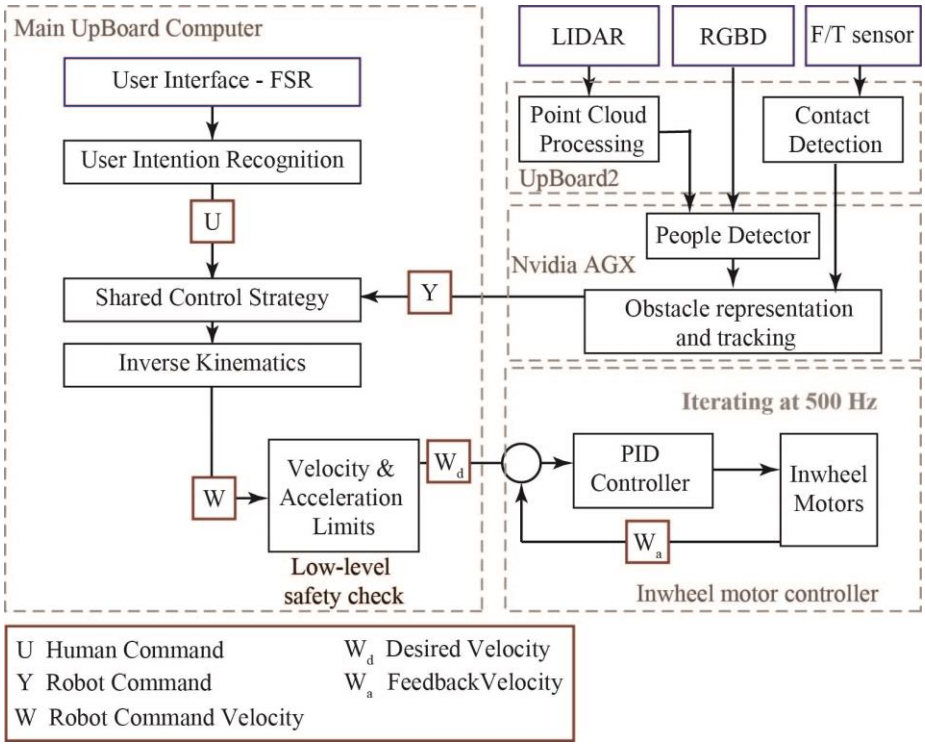

Figure 26. Overall control distribution with sensors, computing units and actuators, for the robot Qolo

3.5. Motion Profiles

All Crowdbot robots run on wheels, therefore, we summarize here the main two general types of motion profile available. Non-holonomic motion (as in **Smart Wheelchair** or in **Qolo**) uses two differential-drive wheels for both rotation and translation of its frame. Therefore, it cannot move sideways (lateral translation without rotation), creating an underactuated motion control with 3 degrees of freedom (DOF) in the plane (x and y translations and yaw rotation) and only 2 actuators. Within non-holonomic constraints there are different configurations, e.g., there are wheelchair models with a drive train that allows wheels to rotate simultaneously in any direction, thus, achieving an omni-directional turn only, but still requires the same maneuvers as a standard wheelchair for a sideway displacement.

Another type of constraint exists for the **CuyBot** robot. The robot can move sideways, in fact it has full mobility in all directions and rotation. But it cannot always change its velocity instantly. For example if the robot was moving forward and comes to a stop, in order to continue sideways, the wheels first have to rotate into the proper direction before the robot can start moving sideways.

In contrast, a full motion control profile is available in a holonomic robot such as **Pepper**, which can control each DOF independently. Both holonomic and non-holonomic motion profiles are illustrated in Figure 27.

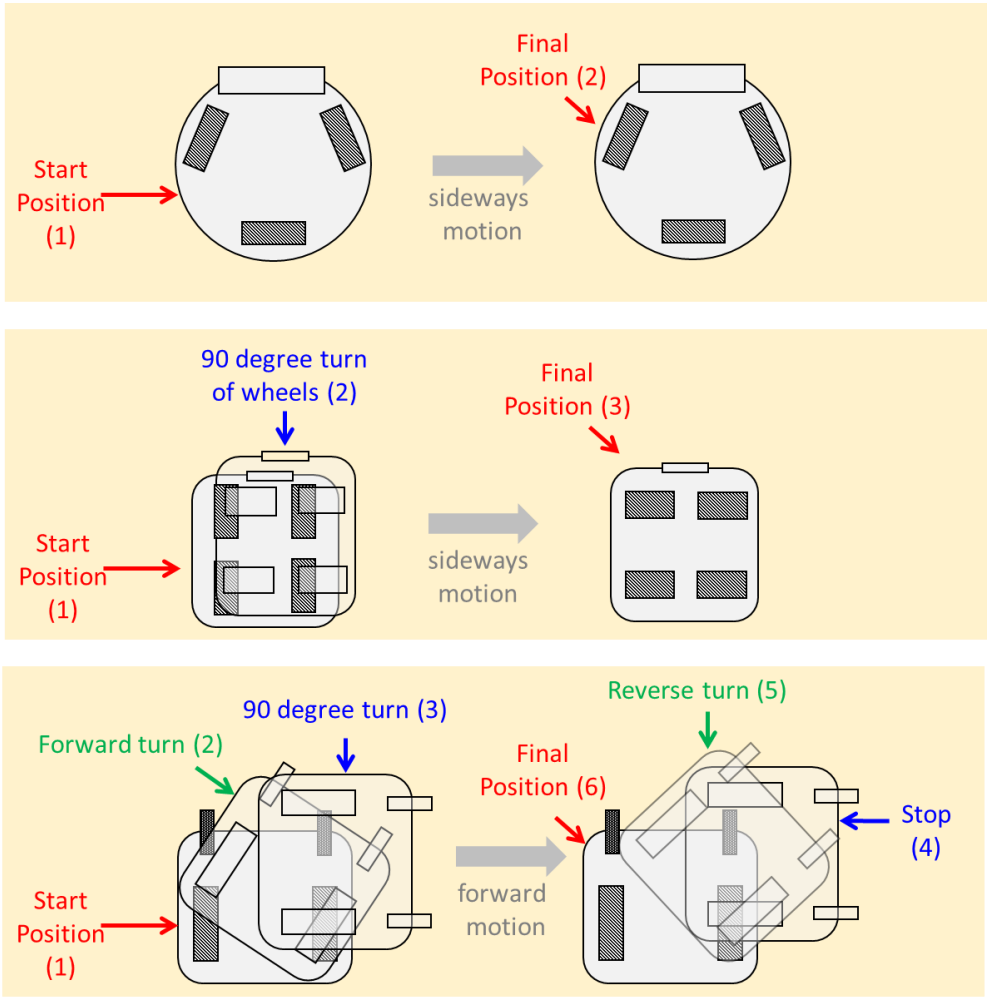

Figure 27. Motion Profiles for sideway movement of a Holonomic (top) vs. Non-Holonomic (bottom) robot. Robots like cuyBot are in between both extremes (middle).

3.6. Human-Machine Interface (HMI) Options

Crowdbot robots are designed to run in either semi- or full-autonomy mode of navigation. Even in full autonomous mode with no human intervention during actual navigation runs, it is expected that human-machine communication is still required for 1) pre-run setup and troubleshooting, 2) remote control and monitoring during run by a human operator, and 3) post-run data dump. Of course, for the smart wheelchair, continuous in-seat HMI device control is assumed in actual navigation runs since its default mode is semi-autonomous, shared control operation.

Commonly used human-robot communication methods are Ethernet, WiFi, Bluetooth or a computer communication bus (e.g. USB, serial port) or a microcontroller bus (e.g. CAN, I2C). As a safety measure, each robot must be equipped with a kill switch that can be activated remotely. This is listed as a requirement item in Table 3. A physical “red button” kill switch on the robot may not suffice since in an emergency it may be difficult to approach and make contact with a machine that is misbehaving. Other options are remote kill via wireless communication or blockage of future motion using a physical obstacle.

For in-seat control of wheelchair maneuvering, many technology options have been developed over the years to accommodate a user/operator with a specific type of disability or impairment. Details of these HMI options are outside the scope of this report. A summary can be found in [24, 25]. In Figure 28 we provide a pictorial summary of such HMI options. In-seat maneuvering involves changes to both direction and speed. If an HMI device is sensitive to speed adjustments, then it is called proportional. Non-proportional HMI switches are limited to simple tasks: stop-and-go, forward-and-backward and so on. In all Crowdbot tests the most common HMI option ─the proportional joystick─ will be used, with the exception of the Qolo robot

which is controlled by a torso HMI developed for hands-free standing mobility [53]. No further HMI device requirements beyond the generic joystick are specified in the requirements list of Table 3.

Figure 28. HMI Options for the Smart Wheelchair Operator

A subject closely related to HMI options is the process flow of human-to-machine communication. A high-level description is portrayed in Figure 29. The left-side block "Human Operator" can be viewed as a remote controller, an in-seat human operator or a computer program that executes a programmed navigation task. Its output is typically (translational) speed (m/s or km/hr) and direction. All objects in the figure (human operator, HMI device, translator and motor control) are integrated hardware-software modules of a robot. The thick black arrows denote communication interfaces from one object to the other. In the case of a physical joystick, the communication interface is the human hand that jolts the stick to a certain direction. In the case of remote or computer control, contact with the joystick is non-physical and its output is simulated. Typical HMI outputs are forward, backward, left, right, neutral (mid-point in a joystick) and a proportionality value that corresponds to translational speed. The translator then converts speed and direction to rotational speeds (rpm's) of all motors with the aid of readings from other sensors for motor/wheel position, rotational speed and inertia measurement units.

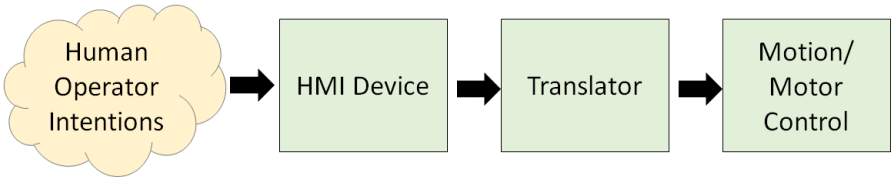

Figure 29. Human-Machine Interface Process Flow

In fully functional commercial machines (marketplace robots and electric wheelchairs), the motor control object is an integral component in the base of a robot with little or no room for design change or modification. In an electric wheelchair the HMI device is a physical component mounted on the body (e.g. the joystick sits on the top of an armrest). If the HMI device is physical, then the translator object already exists as an internal module, developed by the robot/wheelchair manufacturer. In this case the Crowdbot team can maneuver the robot by sending signals that imitate HMI device outputs using the same HMI-translator communication interface. On the other hand, if the robot is not equipped with a physical HMI device, the team must find an option to communicate with the translator object ─the communication protocol

and accepted input data formats of the translator must be known. In terms of requirements, no specific option is specified regarding the HMI process flow. The Technology Development (TD) and System Integration (SI) team are allowed to use any available means for robot maneuvering. The chosen methods are reported as requirements items.

3.7. Structural, Electrical & Electronics Augmentation

Crowdbot robots have been augmented with additional sensors, computer processors and related accessories to meet robot safety, design and navigation goals laid out in this report. Electrical and electronic components and modules have been embedded or attached to the baseline robot platform. Structural modifications such as gluing, fastening and harnessing have been realised. Furthermore, our robots are all mobile and tether less, without any dangling cables or power cords. Thus, electrical DC or AC power is now supplied to augmented electrical and electronic devices via an onboard primary energy source such as a battery or a source such as a DC power supply module. Augmentation descriptions are available in **Sections 3.1, 3.2, 3.3 and 3.4**. For the sake of replication, reproducibility and technology migration purposes, the team have reported on the following modifications to the baseline robot:

- Electrical and electronic device augmentation
- Additional computing resources used
- Structural modifications to accommodate augmentation
- Changes to dimension, weight and locomotion profile
- Removal or disabling of any embedded sensors and electronic devices
- Power supply or energy source modifications to accommodate augmentation

All modifications to the base structure and hardware components are annotated using robot design figures similar to those shown in Figure 30 and Figure 31. For example, when a new sensor is augmented, its placement location as well as additional structural support to mount this sensor is shown on the base structure of Figure 30. Electrical power and data bus connections for this new sensor are annotated using a functional block drawing similar to the format shown in Figure 31.

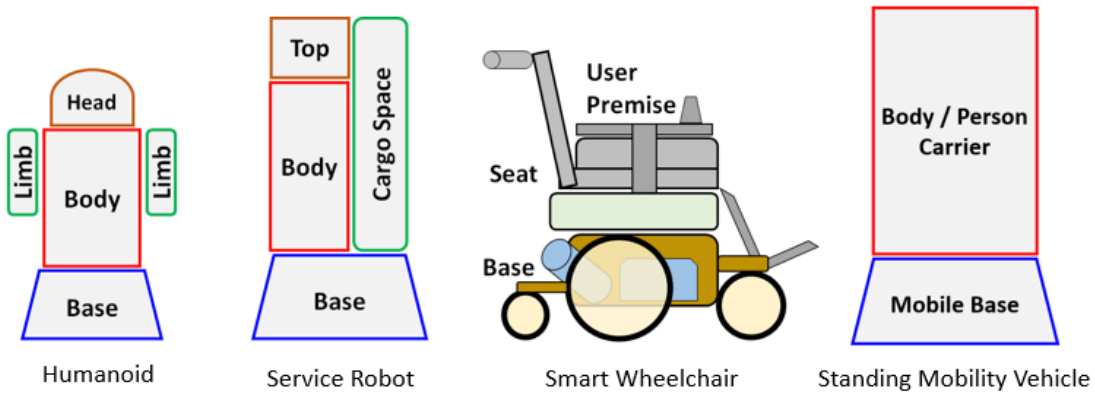

Figure 30. High-Level Structural Description of Crowdbot Robots

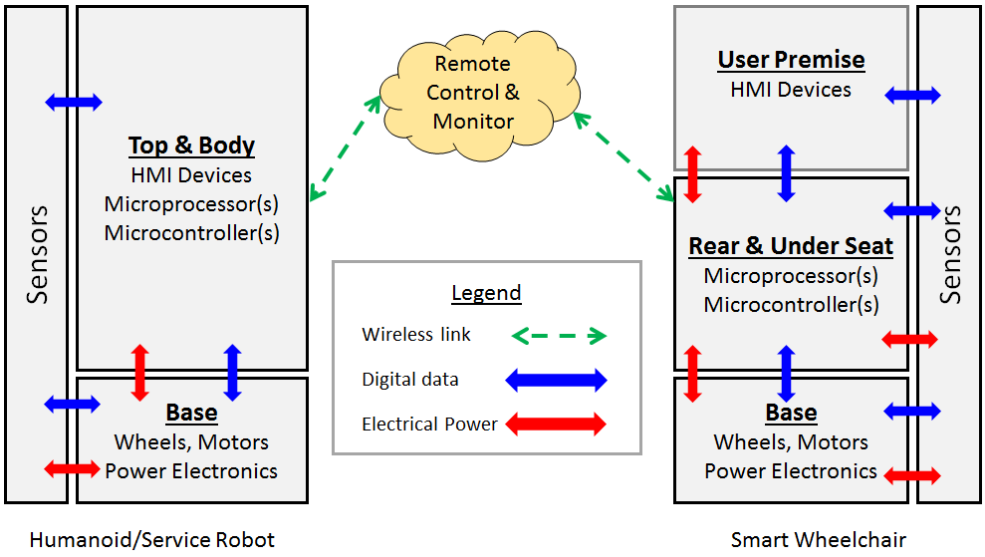

Figure 31. Hardware Composition and Interconnects of a Robot System

4. Review of our technologies

This section is new compared to *D1.1*. It has been introduced to list all the technological components that need to be evaluated in the upcoming test round. The content may have been taken from other deliverables of the CrowdBot project, in the idea to avoid the reader to go through external references.

4.1. Reactive navigation

The main objective of the reactive control module is to provide a layer for local navigation assistance and immediate responsiveness to unmodeled or unpredicted object occurrences or motions.

The Modulated Dynamical System (MDS), as reported in period 1 (see also [57]) represents obstacles analytically as star-shaped level sets of a distance function and reduces the robot to a point moving in a cartesian space. It guarantees to lead the robot to the goal by exploiting the assumptions that the robot is holonomic and circle-shaped and that the objects are star-shaped. In case of a crowded environment (density of 1ppsm or higher) and for a wheel-based platform with non-circular shape, they demand a rather conservative simplifications of the robot shape and kinematics. Furthermore, obtaining an analytic representation of the objects around the robot introduces an additional pre-processing step.

Therefore, as a supplementary approach, the Method termed Redirecting Driver Support (RDS) has been developed, which replicates the behaviour of the MDS locally. While it sacrifices the guarantee to reach the global goal (by itself, i.e. without an additional path planner), it allows to represent more accurately the local object and robot shapes' details and non-holonomic constraints of robot kinematics.

The RDS approach has been designed to be functional when using unprocessed measurements of objects and to be executed with a high control frequency. It locally deflects the robot velocity to slide along or around objects rather than colliding with them, and it can therefore complement a high-level motion planner or assist a human driver or operator. Preliminary experimental results and practical demonstrations have been presented in the deliverable D1.4. The following sections describe the technical details and theoretical formulation.

4.1.1. Method MDS (Modulated Dynamical System)

4.1.1.1. Objects Representation

The method represents each object by a corresponding distance function $\Gamma(x)$ which evaluates to one when the robot is in a position x which implies contact with the object and which is greater than one for positions which imply a distance greater than zero from the object. The normal direction pointing away from the object's surface can be retrieved from the distance function's gradient as $[n_1, n_2] = \frac{\partial \Gamma}{\partial x} / |\frac{\partial \Gamma}{\partial x}|$, which allows to construct the tangential unit vector as $[e_1, e_2] = [-n_2, n_1]$. Additionally, the method defines a reference point x_{ref} in each object's interior and a corresponding unit vector r pointing from the reference point to the robot, i.e. $r = (x - x_{ref})/|x - x_{ref}|$. This reference point must be chosen such that the object is star-shaped with respect to it, i.e. every ray starting from the reference point must cross the object's boundary once but not more.

4.1.1.2. Dynamical System Modulation

For a given goal location x_{goal}, the method constructs a nominal velocity field $f(x)$ which guides the robot from every possible position in space straight to the goal according to $f(x) = -k(x - x_{goal})$. Multiplication of this nominal velocity with the modulation matrix M gives the actual velocity command for the robot to avoid collisions with the objects in its way. The modulation matrix M has the form $M := EDE^{-1}$, with the diagonal matrix D and with the matrix E having the vectors r and e as the first and second column, respectively. The entries on the diagonal of D have therefore the effect to rescale the nominal velocity's components in the direction towards the reference point and the tangential direction. Thus, the method lets the first entry on the diagonal decay to zero as Γ approaches one and the second entry increase by the respective amount, which absorbs the nominal velocity's normal component (as the reference point is within

the object and the object is assumed to be star-shaped) and amplifies the tangential component. Thereby, it guarantees both collision avoidance and reaching the goal.

4.1.2. Method RDS (Redirecting Driver Support)

4.1.2.1. Shapes Representation

The method approximates the robot's shape as the union of circles, and it also represents the surrounding objects' shapes in this fashion. Let C^R denote the set of circles to represent the robot's shape. Write $C^R = \{c_1^R, \dots, c_N^R\}$, where each c_i^R is a circle whose radius is denoted as r_i^R and whose center's global position with respect to a fixed reference in the Euclidean plane is denoted as p_i^R.

Similarly, let C^O denote the set of circles to represent all the surrounding objects' aggregate shape, i.e. $C^O = \{c_1^O, \dots, c_M^O\}$,where a given circle c_j^O represents a part of the surrounding objects and is characterized similarly by its radius r_j^O and its position p_j^O.

4.1.2.2. Vehicle Kinematics

Assuming that the robot is a vehicle which has two actuated wheels that do not slip and that rotate around the same axle and which has additional caster wheels to keep it stable, the following kinematic relations hold. Define the robot's local frame as the Cartesian coordinate frame whose x-axis coincides with the main wheels' axle and whose y-coordinate points forward and intercepts the axle at its midpoint. With respect to this frame, let x^p, y^pdenote the position coordinates of a point P on the robot, and let v_x^P, v_y^P denote P's velocity vector's coordinates.

Further, consider the axle midpoint's velocity, whose x-component must vanish to not slip. Denoting its y-component by v and the robot's angular velocity by ω, it holds for any point P on the robot that$v_x^P = -y^P\,\omega$ and $v_y^P = v + x^P\,\omega$. If the point P does not lie on the line through the wheel axle, this mapping is invertible and the inverse is given by $v = v_x^P x^P/y^P + v_y^P$ and $\omega = -v_x^P/y^P$. Choose two different points P and P' and chain the aforementioned mappings to obtain a direct mapping from the velocity of P to the velocity of P', namely $v_x^{P'} = v_x^P y^{P'}/y^P$ and $v_y^{P'} = v_x^P(x^P - x^{P'})/y^P + v_y^P$. This relation maps the velocity of any point P which does not lie on the x-axis to any other point's velocity.

4.1.2.3. Velocity Constraints

Pair each circle c_i^R from C^R with each circle c_j^O from C^O and create a constraint for the velocity of the center of c_i^R to limit the velocity's component in the direction of c_j^O. Let$v_x^{R,i}, v_y^{R,i}$ denote the components of the velocity $\mathrm{d}p_i^R/\mathrm{d}t$ when expressed in the robot's local coordinate frame (i.e. the expression in the local coordinate frame of the rate of change vector of the center point's global position). Further, let $n_x^{i,j}, n_y^{i,j}$ denote the local coordinates of the normal vector $(p_i^R - p_j^O)/|p_i^R - p_j^O|$. Then, the constraint for the pair (c_i^R, c_j^O) reads

$$n_x^{i,j} v_x^{R,i} + n_y^{i,j} v_y^{R,i} \le (D/T)\sqrt{(|p_i^R - p_j^O| - r_i^R - r_j^O)/D - \delta} := b_{i,j}$$

where the positive time and distance constants T and D define together the maximum deceleration to impose on any robot circle c_i^R to be D/T^2 and $\delta > 0$ defines the minimum separation (normalized by D) between the robot and object circles up to which the robot can approach the objects.

Using the kinematic relations introduced in the previous sections, one can transform all the constraints for the velocity of different circle centers into equivalent constraints for the velocity of a single reference point P. Expressing the velocity in the above inequality by the velocity of the reference point P, one obtains

$$n_x^{i,j} v_x^P y^{R,i}/y^P + n_y^{i,j} v_x^P (x^P - x^{R,i})/y^P + n_y^{i,j} v_y^P \le b_{i,j}$$

Where $x^{R,i}$, $y^{R,i}$denote the local coordinates of the vector which connects the local origin (axle midpoint) and the center of c_i^R.

4.1.2.4. Constrained Optimization

The method computes the velocity command tuple (v^*, ω^*) which the robot will execute by minimizing its deviation from a given nominal command$(\underline{v}, \underline{\omega})$ coming from the driver or any higher-level module. It achieves collision avoidance by imposing the above constraints on this optimization. The metric during the optimization is the Euclidean norm of the difference between the nominal velocity and the chosen velocity of the reference point P. For readability, P's velocity components are simply denoted by v_x, v_y (without the superscript) in this section.

It is therefore the easiest to solve this minimization problem directly in the space of the reference point velocity, and to transform the nominal command into a corresponding nominal reference velocity $\underline{v_x}(\underline{v}, \underline{\omega})$, $\underline{v_y}(\underline{v}, \underline{\omega})$ via the aforementioned kinematic relations. The problem is formally written as

$$(v_x{}^*, v_y{}^*) = arg\ min_{v_x, v_y}(v_x - \underline{v_x})^2 + (v_y - \underline{v_y})^2$$

$$\text{s.t.} \quad n_x^{i,j} v_x^P y^{R,i}/y^P + n_y^{i,j} v_x^P (x^P - x^{R,i})/y^P + n_y^{i,j} v_y^P \leq b_{i,j},$$

$$\forall \quad i \in \{1, \ldots, N\}, \forall\, j \in \{1, \ldots, M\}.$$

From the solution, the actual velocity command for the robot is then computed as $v^*(v_x{}^*, v_y{}^*)$, $\omega^*(v_x{}^*, v_y{}^*)$ according to the kinematic relations.

4.1.2.5. Implementation

To use object information directly from sensor data without introducing any latency or potential for missing out object parts during further processing, the current implementation creates a circle and corresponding constraints for each measured point provided by a planar laser scanner.

In our case, the shape of the robot, Qolo, is approximated by two circles to take into account the elongation from the back to the front.

The implementation uses a geometric algorithm to solve the quadratic program above, which is also being used in the ORCA method [54]. It slightly adapts an algorithm which is known as incremental linear programming [55], [56]. Besides being efficient on small-dimensional problems like this one (which is in 2D), its advantages include that it converges in a finite number of iterations, which is desirable to ensure real-time execution, and that it is simple and rather easy to implement.

4.2. Low-level control

Planning algorithms described in the previous section are dependent on the process of detecting and classifying objects and agents in the environment. This process, however, presents several challenges, mainly: i) it can induce additional latency ii) it presents additional failure modes, such as false/true positive/negatives, which then propagate through the subsequent planning steps, leading to unpredictable or undesired behaviours. This fact can serve to motivate the development of end-to-end planning techniques: motion planning methods in which the planner takes sensor readings as inputs directly, and outputs velocity commands. In order to research the viability and test the limits of such methods in the case of CROWDBOT, a low-latency planner was designed to function on the Pepper platform modified by ETHZ.

The low latency planner described in this section is similar to the Dynamic Window Approach, and Velocity Obstacle, in that it creates an estimate of reachable velocities at the current time, and searches in the sampled space of those velocities for those which maximize an objective function $J(v)$. The space of reachable velocities is defined as the intersection of two sets, the set of velocities which satisfy the platform velocity limits, V_s, and the set of velocities reachable from the velocity at current time according to the platform acceleration constraints, V_a,

$$V_s = \{v \in R^2, \|v\| < v_{max}\},$$

$$V_a = \left\{v, v_t \in R^2, \|v - v_t\| \frac{1}{dt} < a_{max}\right\}.$$

The objective function J at current state S is defined as,

$$J(v,S), \qquad s.t.\ S = \{x_{goal}^{t}, O^{t}\},$$

with x_{goal}^{t} as the spatial coordinates of the goal in the robot frame at the current time, and O^{t} as the set of obstacle coordinates $\{x_{obstacle_1}^{t}, x_{obstacle_2}^{t}, \dots\}$. Furthermore, $J(v,S)$ is,

$$\begin{aligned}
&undefined \quad if \quad d_{goal}^{t+1} > D \ and \ p_g < 0,\\
p_g \quad &if \quad \left(d_{goal}^{t+1} \le D \ or \ p_g \ge 0\right) and \ d_{closest}^{t+1} \ge R \ and \ d_{closest}^{t} > R,\\
0 \quad &if \quad \left(d_{goal}^{t+1} \le D \ or \ p_g \ge 0\right) and \ d_{closest}^{t+1} < R \ and \ d_{closest}^{t} > R,\\
p_o + 0.1p_g \quad &if \quad \left(d_{goal}^{t+1} \le D \ or \ p_g \ge 0\right) and \ p_o > 0 \ and \ d_{closest}^{t} \le R,\\
0 \quad &if \quad \left(d_{goal}^{t+1} \le D \ or \ p_g \ge 0\right) and \ p_o \le 0 \ and \ d_{closest}^{t} \le R,
\end{aligned}$$

where,

$$x^{t+1} = v\,dt,$$

$$p_g = \|x_{goal}^{t}\| = \|x^{t+1} - x_{goal}\|,$$

can be understood as the progress towards the goal in meters.

$$d_{goal}^{t+1} = \|x^{t+1} - x_{goal}\|,$$

$$d_{closest}^{t} = \left(\|x_{obstacle_i}^{t}\|\right)\ for\ x_{obstacle_i}^{t} \in O^{t},$$

$$d_{closest}^{t+1} = \left(\|x^{t+1} - x_{obstacle_i}^{t+1}\|\right)\ for\ x_{obstacle_i}^{t+1} \in \hat{O}^{t+1},$$

$$\hat{O}^{t+1} = Predictor(O^{t}),$$

and

$$p_o = d_{closest}^{t+1} - d_{closest}^{t},$$

can be understood as the progress away from the closest obstacle in meters. R is a constant representing the comfort radius of the robot. D is a constant for the maximum distance the robot is willing to be away from its goal.

If no prediction of the future position of the obstacles $\hat{O}^{t+1}$ is available, one can make the assumption that obstacles can be considered as static over the planning timescale and replace $\hat{O}^{t+1}$ with $\hat{O}^{t}$ in the calculation of $d_{closest}^{t+1}$.

In words, the objective described above can be outlined as follows:

(1) If the robot is too far away from the goal and does not make progress towards it, J is undefined.

Otherwise:

If the robot is free from obstacles:

(2) If the resulting position is also free, J equals progress towards goal in meters.

(3) If the resulting position is not free, J equals 0.

If the robot is not free from obstacles:

(4) If the resulting position yields progress away from obstacles, J equals progress away from obstacles plus a small contribution of progress towards goal.

(5) If the resulting position yields no progress away from obstacles, J equals 0.

In the case of Pepper, the highest field-of-view and precision sensor for obtaining an estimate on the position of obstacles is LIDAR. For this reason, the planning algorithm is run with each new LIDAR sensor reading. The LIDAR scan is filtered to remove noise, and the set of (x, y) coordinates of LIDAR points is assigned directly to the set of obstacle coordinates O^{t}. x_{goal} is obtained from Pepper's state estimation.

The execution of the algorithm is described in Algorithm 3.

Algorithm 3: Local, low-latency planner		
1	At given time t, obstacle coordinates O^t, goal coordinates x_{goal}^t	
2		$S = \{x_{goal}^t, O^t\}$
3		$J_{best} = 0$
4		$v_{best} = (0,0)$
5	ForEach $v \in V_a \cap V_s$	
6		$J^{t+1} = J(v, S)$
7	If $J^{t+1} > J_{best}$	
8		$J_{best} = J^{t+1}$
9		$v_{best} = v$
10	End If	
11	End ForEach	

4.3. Social Signaling / Gestures

In recent years, robotic platforms have been deployed in public environments making evident the need for human-aware navigation capabilities. Tackling this need, numerous researchers have made efforts in order to include social conventions such as proxemics to create models for robot navigation. Nevertheless, few efforts have been made concerning the problem of labeling humans as an interactive agent when blocking the robot motion trajectory. Current state of the art navigation planners will either propose an alternative path or freeze the motion until the path is free. We propose a strategy to fulfill this gap in the social navigation capabilities of robots in Human-Robot Interaction.

In the scope of the social navigation work package, Softbank Robotics Europe realized a preliminary study concerning social clues of Pepper while navigating and guiding a participant. Our results showed that people perceived the robot more as an assistant and less as a machine when it exhibited social behaviours. Also, we asked them to select the specific behaviours they preferred during the guiding task with the robot. These were: speech; coloured lights; maintaining an appropriate distance from obstacles, humans or objects; approaching obstacles at an appropriate velocity, humans or objects, and narrow places; gesture; head and body orientation; or write down any social cues not included in the previous ones. One participant did not answer the question, the remaining participants unanimously preferred a robot that is able to talk.

The problem of freezing when the robot cannot continue its navigation due to the presence of people is more accentuated in a semi-crowded environment, for example in a cocktail party. Then, our approach is to consider the people as interactive agents who can react to social cues of the robot to clear its path.

There are some situations where a physical touch by the part of the robot could be useful, for example when the robot is in a noisy environment and the person does not listen to the petition of the robot to clean the path. In such situations, the robot should be able to detect the human position and if possible, the positions of the arms and hands of the person. Such information is of vital importance for achieving physical contact while preserving the factors of safety, comfort, naturalness and sociability. We develop a system to detect human arms and hands to perform a robot arm movement to touch the hand or the arm of a person in front of him, which is useful for asking for permission to pass in a semi crowded environment, and in this way clear its path and continue with its navigation.

4.4. Human Motion Prediction

In order to predict the state of the crowd surrounding the robot, we use a neural network prediction system that is able to learn multi-modal trajectory distributions. The system is designed to take the set of N observed trajectories of detected people, and to return K different sets of N plausible predicted trajectories.

The system uses Long Short-Term Memory (LSTM) cells to encode each of the observed trajectories separately. LSTMs are a class of recurrent neural networks that are designed to memorize long-term information in time-series data. Hence, having N agents around the robot, we will get N encoded vectors of the observed trajectories.

In parallel we also compute 3 social (interaction) features between every pair of agents, say p_i and p_j:

1) Euclidean distance between p_i and p_j
2) The bearing angle of p_j from the point of view of p_i
3) Distance to closest approach or DCA (assuming constant-velocity motion for both agents)

In the prediction system there is another layer called Attention Pooling. The role of this block is to take the social features, and decide which agent will apply stronger impact on the future motion of any other agent. In other words, the interaction between agent i and j is assigned with a weight $0 \leq w_{ij} \leq 1$ where $\sum_j \quad w_{ij} = 1$. We sum up the encoded trajectories of all other agents than p_i encoded and the output will contain social features of p_i

Finally the social features and encoded trajectory of p_i will be concatenated and fed to a decoder layer to generate the predictions. The decoder is also an LSTM network that generates the prediction step by step.

The system learns the prediction from a dataset of trajectories recorded in the same environment. Having this dataset, we divide the trajectories into observation/prediction segments, and we use a Generative Adverserial Network (GAN) architecture to train the model. In such a system the training takes place between a 'Generator' and a sample 'Discriminator'.

The samples generated by the generator and also the ones in the dataset are given to the discriminator. The task is to learn to discriminate a real sample from a fake sample (one coming from the generator).

On the other hand, the task of the generator is to 'fool' the discriminator by making its output more and more similar to the real samples. Then the generator ideally learns to generate samples which are not distinguishable for the discriminator anymore.

The source of the diversity in a GAN is a random variable (z) that is given to the generator. By changing this variable various prediction can be achieved. But in practice there is a phenomenon which is called ‘mode collapsing’. It happens, when the generator produces just a subset of real samples in the dataset, and misses some 'modes' of the data.

To deal with this problem, we use a technique called info-GAN that tries to increase the mutual information between the input random variable and the output trajectories. Using this technique another random variable, which is called latent code (c) is given to the generator. Then the discriminator is trained to guess the variable associated with each generated trajectory.

Then, by changing this code we would achieve more various trajectories from the generator and it resolves the mode collapsing effect.

The block diagram of the prediction network is depicted in the Figure 32

To make this system able to predict trajectories in a new environment, we are modifying the system to receive the semantic information in the map around the agents. This way, the predictions will be associated with local structures in the environment, and the prediction model can be applied in new environments. In parallel we are developing the Followbot framework, to deal with the prediction from a robot perspective.

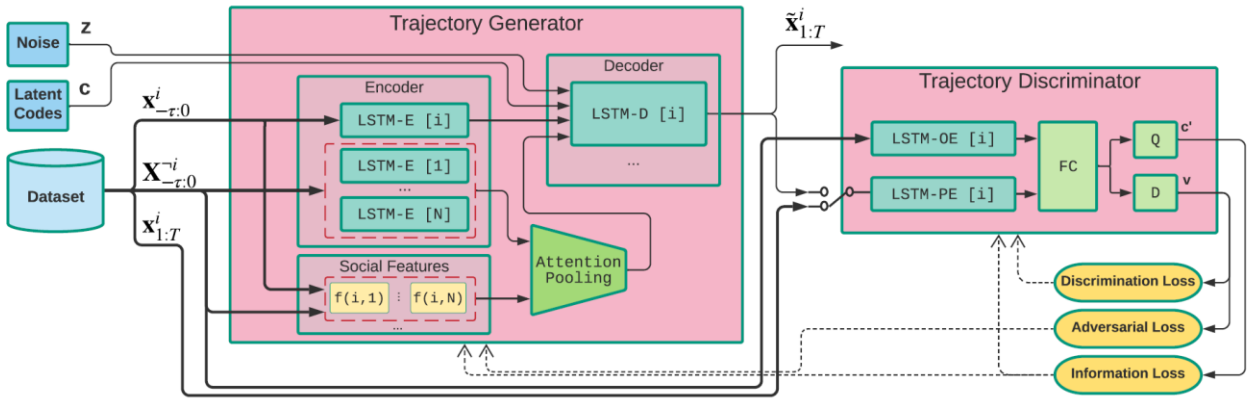

Figure 32. Block diagram of the prediction system

4.5. FollowBot

The motivation of the "FollowBot" module is to design a framework for the task of a robot following someone. Such feature is useful in a 1D flow situation where the robot will have to limit the perturbation of the crowd flow by going with the crowd. To test the limits of such coupling, the robot will have to follow the leader as close as possible.

The main idea of the Followbot scenario is to put the robot in a scenario to follow a leader person as close as possible in a crowded environment.

The framework is designed for development and evaluation of the crowd prediction system, from a robot perspective, using range data coupled with a navigation technique.

The framework is shown in Figure 33.

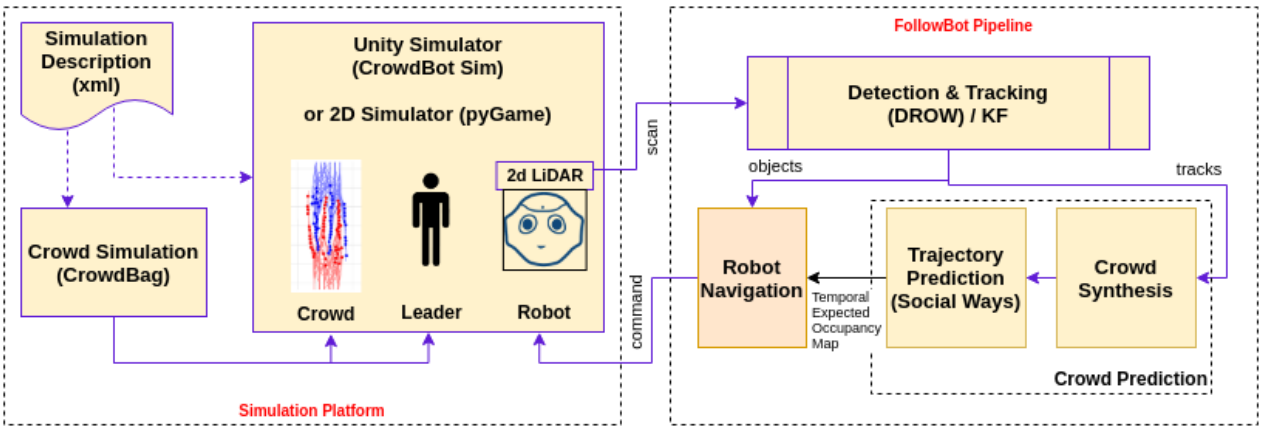

Figure 33. Followbot framework

In this diagram, a simulation platform is shown on the left, and the process pipeline is on the right side. The simulator is developed by Unity (a graphical game engine which contains a robot, the leader person, and the crowd. And the components communicate through ROS.

The motion of the crowd is controlled by an external module, called CrowdBag. This module, provides a ROS service to simulate a crowd, using a specified crowd simulation algorithm. The supported algorithms include:

- Helbing/Boids model (social forces)
- Power-law model (collision forces)

- RVO2 model (Reciprocal velocity obstacles)

Also the configuration of the simulation environment is stored in an XML file and is read by both the main simulator and the crowd simulator.

The Followbot pipeline is composed of 3 main components:

1) **Detection and Tracking**: the range data, produced by simulated LiDAR, is given to a detection and tracking module. It will detect and track pedestrians.
2) **Crowd Prediction**: crowd prediction is composed of two sub-modules:
    a. **Crowd synthesis**: since the camera of the robot can not see some of the objects in the blind spots (because of occlusion). Hence, we add some synthetic pedestrians to see the effect of them on the detected people. This way we may achieve prediction samples that are not generated in the normal way, i.e. by only considering the detected pedestrians. This idea is still in the stage of proof of concept.
    b. **Trajectory prediction**: this module is the same described in previous section.
3) **Robot navigation**: We use an RVO robot navigation and the leader location as the goal point.

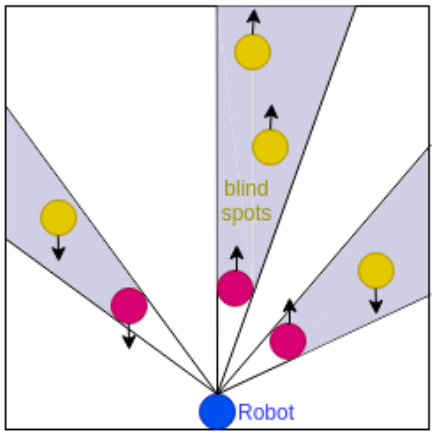

Figure 34. Crowd in blind spots (shown by yellow)

Crowd Synthesis module tries to synthesize some unseen pedestrians in random locations, in order to improve the prediction samples of the leader person and the other detected pedestrians (Figure 34).

4.6. Reinforcement Learning

In Crowdbot, we aim to achieve smooth and socially compliant navigation among pedestrians. That is, robot navigation that considers the potential movements and interactions of all participants in the crowd. Several end-to-end deep reinforcement learning approaches to robot navigation have been proposed in recent literature. These methods aim at finding motion control policies for ground robots that can implicitly account for pedestrian-robot interactions that arise from the robot's executed motion as well as potential downstream interactions between pedestrians in the observed crowd. Examples include socially aware collision avoidance with deep reinforcement learning (SA-CADRL) [40], its extension to the asynchronous advantage actor-critic framework (GA3C-CADRL) [41], as well as CrowdNav [42], which uses a self-attention mechanism for identifying pertinent agents in the crowd. Each of these existing methods use some combination of the robot state, goal location, a (partial/local) map of the static environment, and some representation of the surrounding pedestrians as input to the robot control policy (see Figure 35 for an example from our prior work [43]).

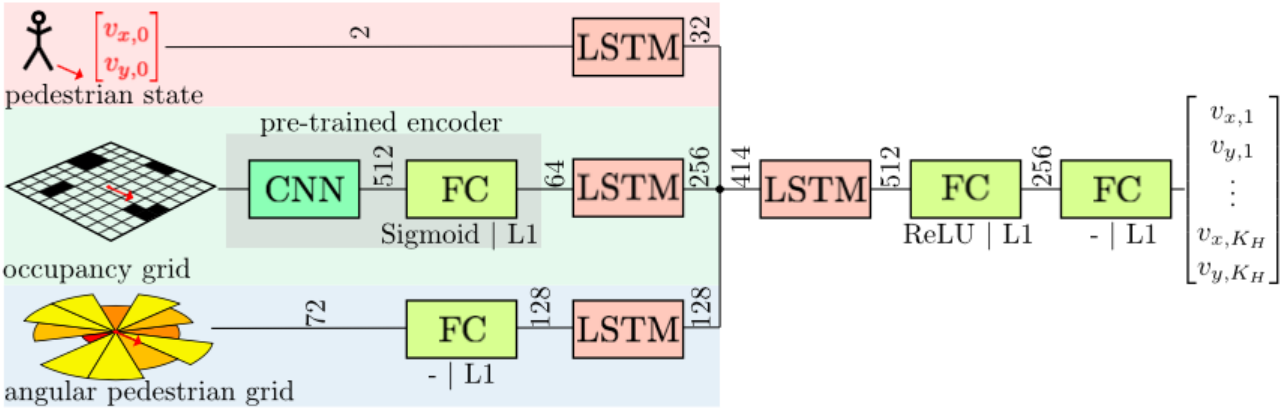

Figure 35. Example pedestrian prediction neural network architecture from [43]

In Crowdbot, pedestrian tracking and prediction are handled by separate modules whose outputs could directly provide the necessary inputs to a reinforcement learning framework tasked to derive a control policy for a navigating robot. This would reduce the computational burden on the RL framework since the learned policy would not need to derive a complete implicit model of the crowd interaction, which is already handled by the external prediction module. Nevertheless, the RL policy has the representational capacity to capture any residual errors that are not explicitly modeled by the inputs. By focusing on modeling the residuals, we can potentially use a lighter policy architecture and accordingly reduce the training effort.

Indeed, the particular challenge in learning an interaction-aware and socially compliant motion policy is that training examples which adequately reflect true pedestrian responses to robot navigation are difficult to collect. Existing simulators use simplified interaction models, such as the social forces model [44], to control synthetic crowds. Moreover, robots still represent novel entities in human environments and so humans often react differently to the presence of moving robots than they do when moving among other humans. One option for collecting high fidelity pedestrian-robot interaction data that we will explore in Crowdbot is to use virtual reality. This will allow us to place real humans and robots in the same virtual environment as well as include simulated pedestrians to evaluate different crowd scenarios. At the same time, we are able to guarantee the safety of the human participants since the robot and participants can be moving in different physical spaces and thus will not collide.

4.7. Tracking

Our perception pipeline is designed following the well-known tracking-by-detection paradigm. Under this paradigm, objects are detected for each frame independently, and a tracking algorithm is used to associate detections that belong to the same object instance over multiple frames. We also adapt a modularized design approach. For each perception sensor, a detection module is instantiated that operates independently of the other sensors. The detections produced by the different modules are then transformed into a common 3D coordinate frame; detections from sensors with overlapping fields-of-view are fused; and the resulting fused 3D observations are fed to the tracking algorithm. The final output is a set of 3D object trajectories on the ground plane in either world coordinates (if robot odometry is provided) or in local 3D coordinates relative to the robot (if no odometry is available). The overall architecture is shown in Figure 36.

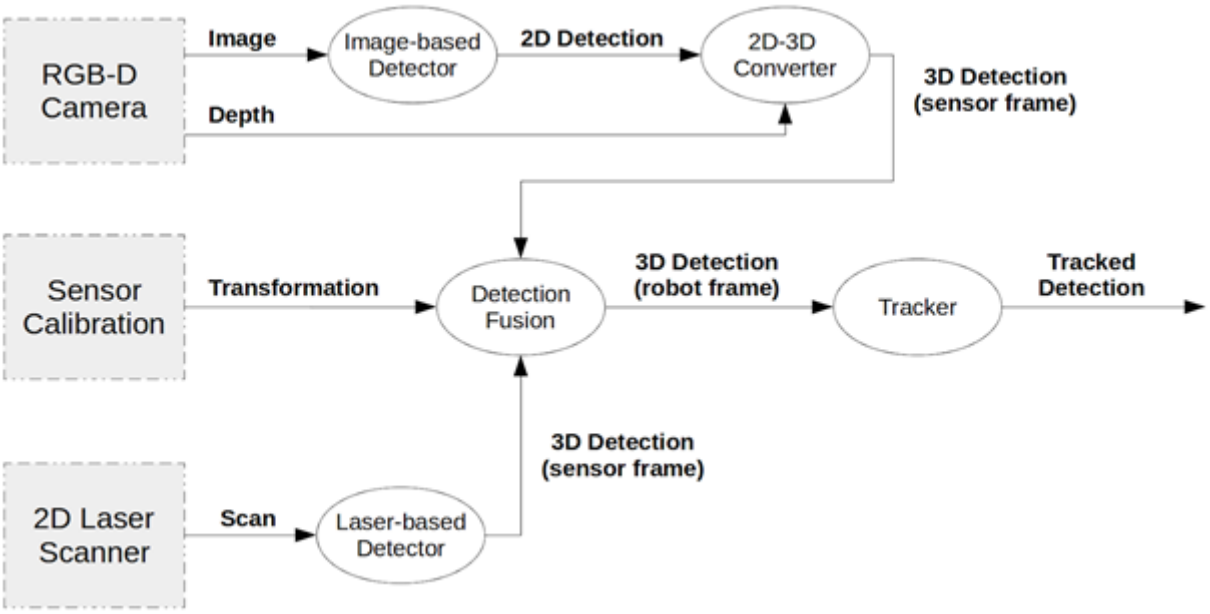

Figure 36. Architecture of our perception pipeline

We use a modularized design approach that enables a flexible combination of different numbers and arrangements of sensors. Dedicated detection modules detect pedestrians from each sensor independently. These detections are then transformed into a common coordinate frame; detections from sensors with overlapping viewing fields are fused; and the resulting measurements are passed to the 3D tracking algorithm.

Our modularized design approach allows for easy adaptation to changes of sensor configuration. The only requirement for adapting to a new set of sensors or a different arrangement of the existing ones is an extrinsic calibration of each sensor relative to the robot coordinate system (as reflected in the ROS coordinate tree). Our current implementation provides dedicated person detection modules for RGB/RGB-D cameras and for 2D LiDAR sensors, which covers the two most important sensor types used in CROWDBOT. As a result, our tracking pipeline can be deployed for all four CROWDBOT robots (ETH's Augmented Pepper, UCL's Wheelchair, Locomotec's CUYBOT, EPFL's QOLO), even though they use different numbers of sensors in different configurations. For further details, please refer to D2.2.

4.8. LiDAR FLOW

When the crowd density becomes very large, conventional tracking-by-detection systems based on RGB data might fail due to heavy occlusion. In such cases it would be helpful to fall back to low level vision techniques such as optical flow, scene flow, or LiDAR flow. Optical flow provides motion information at the pixel level for a small neighborhood within an image. This information can be useful to make motion predictions in cases where the tracker fails due to occlusions, for example when people close to the camera obstruct the field of view.

For this purpose, RWTH is exploring approaches to modify the tracking pipeline by incorporating optical and/or LiDAR flow. This is currently work under development and will be part of a later deliverable D2.3.

4.9. High-level Multi-modal Behaviour Planning

Notably, when humans navigate through crowded and dynamic environments, they show signs of joint planning to move efficiently and interact physically with each other in complex ways to reach their goals. Humans are also able to plan according to global flows and select paths that avoid harder-to-navigate areas. On the other hand, state-of-the-art robot navigation still lacks the ability to move naturally in crowded environments. Existing methods employ monolithic motion planners, which focus solely on moving the robot into free space. In dense and dynamic environments such as human crowds, free space is often fleeting,

leading to what's known as the "freezing robot problem". These methods are also severely limited in that they do not consider other behaviours, such as gesturing or speaking, which can facilitate the robot-crowd interaction. Thus, we aim to develop various behavior modalities that a robot can apply in crowd navigation scenarios. By investigating the efficacy of individual and combinations of modalities, we aim to formulate high level robot planning methods that can reason over the state of the crowd and execute the sequence of behaviours that gets the robot to its goal safely and in the shortest time.

The set of behaviours that we will investigate include variants on the movement of the robot base (speed, heading), gestures with the robot's upper limbs and head (pointing, indicating direction of motion), as well as speech and sounds. Robot behaviours are parameterized according to low level control commands that can be sent directly to the robot for execution, for example, forward velocity of the base, angular velocity of the joints, etc. A basic set of behaviours is outlined in Table 2; this set was defined from observed human behaviours for navigating through crowds.

Table 2. Description of robot behaviors

Behavior name	Description
Intend	The robot base follows the planned path to goal with a strict tolerance on the allowed path deviation. The robot will travel with a forward velocity inversely proportional to the density of obstacles around it to facilitate smooth acceleration. The maximum commanded velocity in this mode $v_{move,max}$ is equal to the top speed of the robot base. The robot will stop if no movement command will allow it to make forward progress along the path without (i) coming within a specified radius r move of an obstacle and/or (ii) violating the path deviation constraint.
Crawl	Similar to the "Intend" behavior except that $v_{crawl,max} << v_{move,max}$ and $r_{crawl} << r_{move}$. The resulting behaviour allows the robot to slowly move closer towards nearby obstacles.
Say	The robot executes a verbal utterance such as "hello" or "excuse me", which is commonly used to get a person's attention. This can be accompanied by a corresponding arm motion.
Nudge	From a neutral pose, the robot raises and points its arm in the direction of intended motion before returning to a neural pose.

We aim to iterate on the individual behaviour definitions (parameters) through in situ evaluation of the robot navigation performance. Our ultimate goal is to design a control policy that can adjust the optimal behaviour parameters online given the response of the crowd to the robot actions, rather than using hand-tuned parameters.

Our planning framework is based on Monte Carlo Tree Search, which expands and searches through possible outcomes of executing behaviours according to the sensed robot state and the state transition probabilities. Given sensor information specifying the relative location, orientation and movement of pedestrians, the planner outputs a sequence of behaviours to execute. The planner must also generate new plans in cases where the observed state changes or when commanded behaviours fail, e.g. the "Intend" behaviour terminated prematurely due to newly observed obstacles, or pedestrians do not make room for the robot to pass after a "Say" action. The main performance metrics for the planner are related to the computational and memory complexity of the solver. In addition, we are also interested in the navigational performance of the robot, i.e. time to reach goal, travel distance, deviation from shortest path, etc. Initial testing in a simulation

environment has shown that the planner is capable of discovering behaviour sequences that allow the robot to reach to goal in scenarios where tradition motion planners fail and that real time replanning is possible even in reasonably complex environments. Figure 37, Figure 38 and Figure 39 show an example of a simulated crowd navigation scenario. The high-level behaviour planner takes as input the perceived crowdedness of the environment and searches for the sequence of behaviours that result in the lowest path cost for reaching the goal.

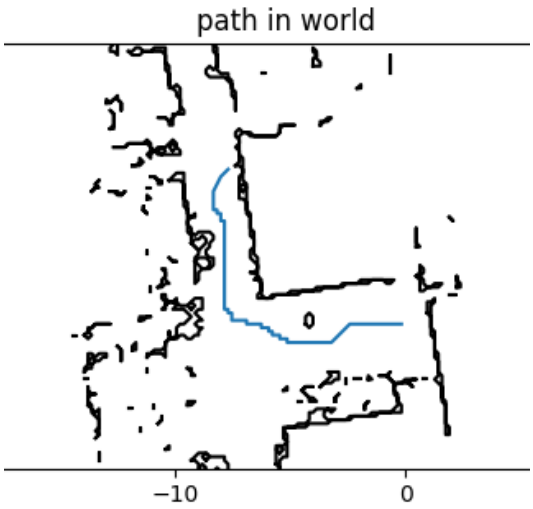

Figure 37. Planned path of the robot base through the static environment

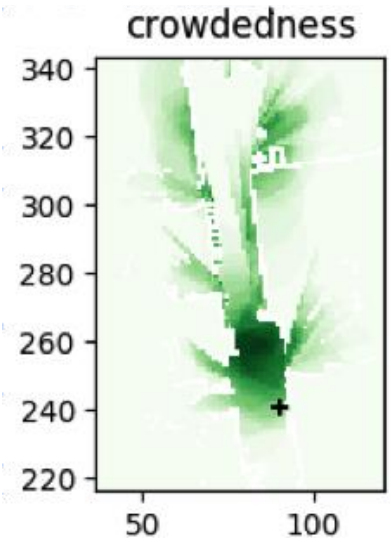

Figure 38. Perceived crowdedness from the robot's sensors (the darker the more crowded)

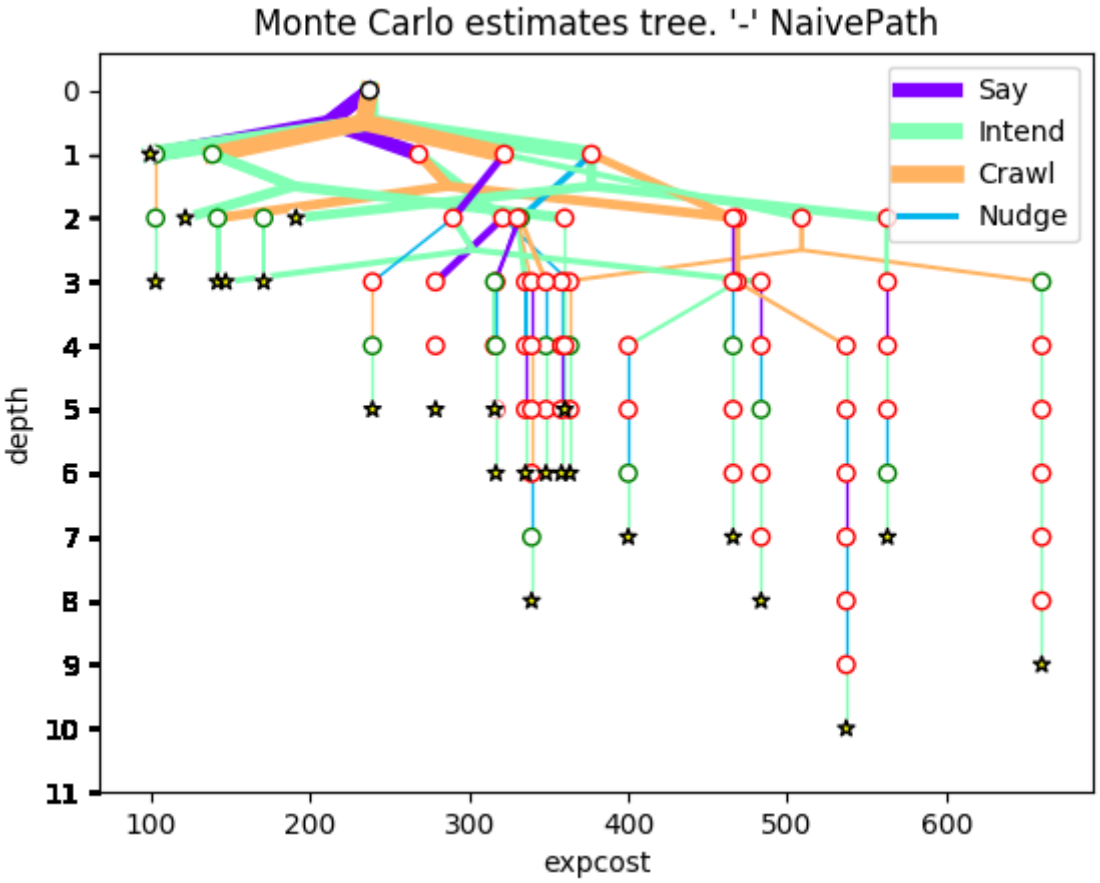

Figure 39. Search tree produced by the high-level behaviour planner

4.10. Global Planning

A versatile and proven approach for global planning in static environments is to use the Dijkstra algorithm to calculate the minimum distance-from-goal value at every point in the map. A common way of adapting the Dijkstra algorithm to data represented on 2D grids is to treat each grid cell as a graph node, with edges connecting cells to various nearby cells. Though cells can be arbitrarily connected in the graph, a useful and common method is to use a regular pattern of connectivity, for example one where each cell is connected to its four direct neighbours. This will be referred to as ‘4-connectivity’ for the purpose of this discussion. Inclusion of more nearby cells to the neighbours set can be envisioned, resulting in 4, 8, 16, 32-connectivity and so on. Figure 40 is an example graph representation of a grid cell’s neighbours in 4, 8, 16 and 32 connectivity. Note that the set of neighbours and edges in 8-connectivity is a superset of the set of neighbours and edges in 4-connectivity, the same is true for 16- vs. 8-connectivity, and so on.

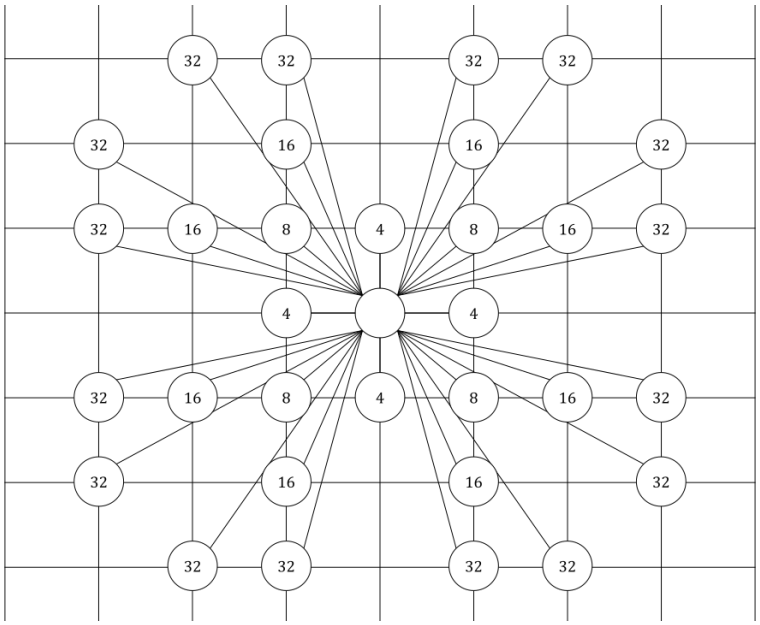

Figure 40. Example graph representation

Higher connectivities present attractive properties, as they increasingly approximate a 2-dimensional Euclidean distance transform which is blocked by obstacles. Such a field can be intuitively understood as representing a sampling of the minimum distance an agent would have to travel on a continuous 2D space if they could move in any direction. However, higher connectivities also introduce complexities, such as the possibly undesired possibility of 'jumping' over obstacles. This issue can be mitigated by performing local pruning of high-connectivity neighbours based on whether an obstacle lies between them. However, this would increase the computational complexity of the search. In addition, 16-and-higher-connectivities present the possibility of introducing local minima to the resulting field, since the propagation of minimum cost in space during the Dijkstra algorithm is no longer wavefront-like but can 'jump' over cells.

In order to obtain a global path from such a distance field, a simple algorithm can be used. At each node, the algorithm checks the distance field value for each neighbour and picks the neighbour with the smallest value. This guarantees that the path is the shortest possible path to the goal.

In some cases, several neighbours present the same distance value. This means that both paths are viable shortest paths. Expanding these choices for every alternative can yield a large number of equivalent shortest paths to goal. If only a single shortest path is desired, a proposed method is to stochastically pick which of the equivalent possibilities should be conserved. This method is used to obtain all path examples presented in this section, for ease of visualization.

Algorithm 2: Naïve algorithm for shortest path

1 Initialise $n = n(x_{robot})$

2 $$Path = [x(n)]$$

3 While true:

4 $$n_{best} = arg\ arg\ \left(D(n_i)\right)\ ,\ \ n_i \in Neighbours(n, connectivity)$$

5 If $D(n_{best}) \geq D(n)$

6 Break // reached local minimum

7 End If

8	$Path = [Path, n_{best}]$
9	$n = n_{best}$

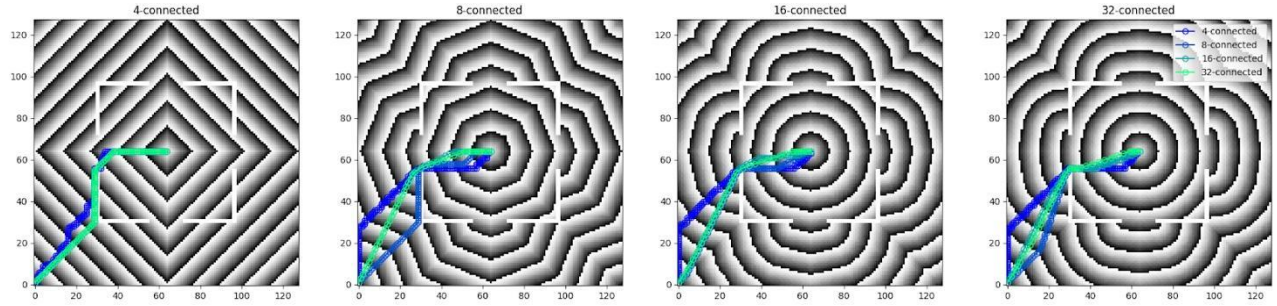

Figure 41. Comparison of contours for distance-to-goal fields

Figure 41 present a comparison of contours for distance-to-goal fields calculated with the Dijkstra algorithm, using 4, 8, 16, and 32-connectedness on a 2D example map. The agent is located at (0,0) and the goal is located at (64,64). A set of four angled obstacles constrain the path to exit the center of the map from one of four "doorways".

In practice we observe that higher connectivity causes the algorithm to yield paths which are found qualitatively to be smoother and more intuitive than lower-connectivity solutions.

Another result of testing the Dijkstra algorithm for global planning is that the shortest paths it yields pass close to static obstacles. This is to be expected as the algorithm presented so far does not consider distance to obstacles at all. One possible solution is to pass the path to a local planner, which then adds obstacle avoidance objectives, such as an elastic-band planner.

On the other hand, it is possible to include this objective in the distance field directly, by adding distance penalties in areas close to obstacles. This presents the advantage of incurring the additional computational cost only once instead of many times as in the case of a local planner, so long as the map remains static. A proposed implementation of this objective is to first calculate the Euclidean Signed Distance Field of every point to its closest obstacle in the map (an efficient algorithm for this is presented in [46]. Then, for every edge in the Dijsktra graph, add a penalty based on the ESDF, for example:

$$edge\_cost_{new}(n_i, n_j) = edge\_cost(n_i, n_j) + \frac{1}{ESDF\left(\frac{x(n_i) + x(n_j)}{2}\right)},$$

where n_i and n_j are the two nodes connected by this edge.

The concept of adding extra edge costs in the Dijkstra algorithm to serve as area penalties could in theory be extended to other navigation objectives, such as dynamic obstacle avoidance. However, due to computational cost, it is useful to use this method for static elements of the scene only and add local planning components for dynamic elements. This makes it possible to compute the weighted distance field only once, assuming the map is known (in the case of SLAM, at a low-frequency update rate). Thus, we avoid recomputing the fields every time a dynamic agent's position changes.

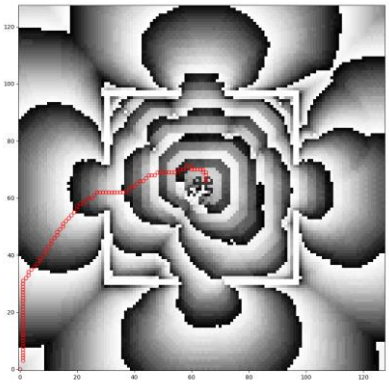

Figure 42. Contours of a 32-connected Dijkstra field, which includes area penalties both close to obstacles and to dynamic agents

One concept which was not explored is the possibility of using Dijkstra distance fields for waypoints instead of a single (x,y) goal objective. Using parameters, this field could encode some objective function representing progress towards a goal while passing through waypoints (Figure 42).

4.11. Physical Interaction

We consider two kinds of physical interaction, the first one is generated by the robot intention to clear its path if it is in a deadlock situation, which was described in the section of Social signaling/Gestures. The second type of physical interaction is generated by other agents that can be an issue for the security of the robot and the people around it. While at the base the Pepper robot has bumpers to detect collisions, in its body it only has sensors at the back of its hands and at the top of its head. In order to detect collisions in other parts of Pepper, we developed an anomaly detection module working on the position of the joints, based on a deep learning approach using a long-short term memory, trained on a dataset of 1 million samples of joint positions of Pepper performing different movements. This is done because the commanded joints position and the sensor positions can differ due to the limitations of the hardware. Then our model would predict when an anomaly (due to an external force) happens on each of the joints.

4.12. Shared Control

The control strategy for a semi-autonomous wheelchair can mainly be divided into two categories. Hierarchical framework where the user was in charge of high-level tasks such as choosing a desired maneuver whilst the wheelchair took control of the low-level tasks [45]. Although this control strategy could be effective for some scenarios, it may not be suitable for navigation in crowds due to the highly dynamic and uncertain environment.

Different from this hierarchical control paradigm, another kind of shared control is achieved by continuous blending input from both user and the motion planner [46-47]. The wheelchair will not move unless input commands from both parties are received. This characteristic allows the process to be more controllable and allows more collaboration between the user and the wheelchair.

One of the widely used blending strategies is called linear blending (LB). Intuitively, the final blended trajectory is a weighted sum of the user's intended trajectory and the trajectory computed by the motion planner.

$$u(t)^{LB} = k_h u(t)^h + k_R u(t)^R$$

$$k_h + k_R = 1$$

Where k_h and k_r are the weights assigned to the human and the motion planner relatively.

Linear blending (LB) has been extensively explored by researchers [47]. In most cases k_h and k_r are dynamic values and are defined based on certain criteria. As a result, the main focus for this approach lies in the weight negotiation between the user and the planner in order to optimize certain objectives. However, linear

blending has some defects. Firstly, the simple blending of the two trajectories does not guarantee a collision free resultant trajectory [48]. Secondly, due to the uncertain nature of the user's intention, in the case where there are several similar k_h and k_r, the failure of the linear blending leveraging this information may result in a final trajectory which does not best describe the user's intention. To address these issues, we adopted a new blending strategy called probabilistic blending strategy which was first proposed by Trautman (2015) and was further adapted by Ezeh et al. (2017) for practical use [46].

Probabilistic shared control (PSC) models the human robot interaction more accurately by taking interaction uncertainty into account. In addition, this approach guarantees safety mathematically [48] and was experimentally demonstrated to be safer than linear blending [46].

In general, PSC is implemented in four steps.

Based on the assumption that the motion planner's trajectory follows a probability distribution, each candidate velocities is associated with a probability.

$$u_{t+1}^{R_i} \sim p(u_{t+1}^{R_i}|z_{1:t}), i = 1{:}N$$

Where N is the total number of candidate trajectories generated by the motion planner and z is the sensor measurement.

Based on the assumption that the user's trajectory follows a probability distribution and it depends solely on the current input, each user's intended velocities are associated with a probability.

$$u_{t+1}^{h} \sim p(u_{t+1}^{h}|z_{1:t}) = p(u_{t+1}^{h}|z_t\ ) = \delta(p(u_{t+1}^{h} - z_t\ )$$

Where u_{t+1}^{h} is the estimated user intended trajectory, which is represented as $u_{t+1}^{h} = [v_{t+1}, w_{t+1}]'$. By assuming the user's intended goal trajectory only based on its current input which is sampled at 10Hz, the problem is simplified and the trajectory is assumed to be memoryless.

Shared control is implemented by taking a joint probability of the user's intended trajectory and the one generated by the motion planner.

The interaction among user, motion planner and the obstacles are modelled as:

$$p(u_{t+1}^{h}, u_{t+1}^{R}, c) = \varphi(u_{t+1}^{h}, u_{t+1}^{R})p(u_{t+1}^{h}|z_{1:t})p(u_{t+1}^{R}, u_{t+1}^{E}|z_{1:t})$$

Where u_{t+1}^{E} is the obstacle' trajectory. It should be noted that this term is ignored in [46] as only static obstacles were considered. However, in the Crowdbot project, it is necessary to include u_{t+1}^{E} as moving obstacles such as pedestrians are also involved. In order to study the interaction between pedestrians and the wheelchair, an experiment has been conducted in PAMELA (the Pedestrian Accessibility Movement Environment Laboratory) in August 2019. The details of the experiment have been described in D1.4.

In the equation above, the sensor measurement vector $\underline{z}_{1:t} = [z_{1:t}^{h}, z_{1:t}^{R}]'$, and the "agreeability function" $\varphi(u_{t+1}^{h}, u_{t+1}^{R})$ are modelled as:

$$\varphi(u_{t+1}^{h}, u_{t+1}^{R}) = exp\left(-\frac{1}{2\gamma}(\hat{u}_{1+t}^{h} - \hat{u}_{1+t}^{R})(\hat{u}_{1+t}^{h} - \hat{u}_{1+t}^{R})'\right)$$

Where $\hat{u}_{1+t}^{h}$ and $\hat{u}_{1+t}^{R}$ are normalized trajectories. The parameter γ models how strongly $\hat{u}_{1+t}^{h}$ and $\hat{u}_{1+t}^{R}$ are correlated.

The final control law at each time step is produced by the path planner's velocity that maximise the joint distribution.

$$u_{t+1}^{PSC} = u_{t+1}^{R*}$$

$$u_{t+1}^{PSC} = argmax_{u_{t+1}^{R}} p(u_{t+1}^{h}, u_{t+1}^{R}|\underline{z}_{1:t})$$

Past experiment results indicated that although PSC and LB have similar performance in scenarios with static obstacles, PSC was considered safer than LB in terms of number of collisions. As for user acceptance, LB was slightly preferred over PSC in the second experiment.

In general, PSC provides a new blending strategy which not only guarantees safety in the shared control, but also models the interaction between the user and the motion planner more accurately by considering the

uncertainty. In Crowdbot, we are working on combining PSC with reactive navigation to avoid moving obstacles. In the future, we intend to improve the user experience by adding intuitive feedback to the system.

5. Technology specific scenarios

This section is new compared to *D1.1*. It replaces the **Section 4 of D1.1**, "Test Scenarios". This new section describes in the details the scenarios introduced in **Section 1**.

This part details the scenarios used as references in CrowdBot, for testing, evaluation, and benchmarking the technologies defined in **Section 4**. The **Section 4 of D1.1** defines the crowd density and personal space, as it is mandatory to define scenarios with various types of crowd.

In this update, we specify new scenarios which are Technology specific. Indeed, in *D1.1*, General scenarios are defined in order to test a complex system, implementing various technologies, which the cluster constitutes the final robot. Such scenarios are relevant for testing the final product features: navigation in the crowd, maneuvers, safety, robustness and social aspects. Those scenarios can be read in the *D1.1*.

Technology specific scenarios, detailed here, put the focus on the technologies defined in **Section 4**, which are the primary elements of what constitute a robot. The goal of such scenarios is to have the focus of the evaluation on each technology, giving more granularity to the overall evaluation.

The described scenarios constitute our reference, and thus should serve the purpose of evaluating the different technologies described in **Section 4**.

The table specified in the introduction of this report specifies the link between those technologies and categories of scenarios.

This part classifies the scenarios into the different scenario categories.

5.1. Scenario Category: Static

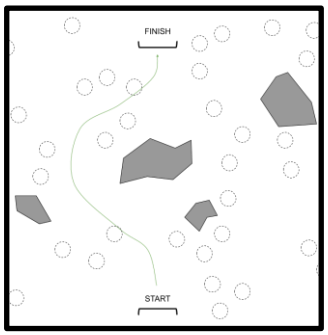

Figure 43. STATIC

The category *static* (Figure 43) regroups the scenarios which are relevant to evaluate most of the technology profiles at the simpler level possible within CrowdBot. Indeed, in this context, the crowd is considered as static or quasi-static, which means the dynamic component of the crowd is neglected. Each agent of the crowd is considered to have no goal whatsoever, and will then stay mostly in space. As an example, such a crowd can be found in a train station when people are waiting for their trains in front of the screens (Figure 44). Also, in social events such as conferences or parties where a lot of people are gathered in the same place around a buffet (Figure 45).

Figure 44. People waiting for their train in the train station

Figure 45. People eating at a buffet

Therefore, depending on the context, the crowd is either considered as a set of obstacles (similar to fixed pillars) which is like a maze in which the robot has to evolve, or like humans with which the robot can interact with. Also, a less dynamic crowd means less uncertainties in the sensing.

Such a scenario allows the tests and evaluation of the following technologies described in the **Section 4**:

- The reactive navigation (EPFL - **Section 4.1**) with Qolo Pepper, and Cuybot
- The low-level planner (ETH - **Section 4.2**) with Pepper
- The social signaling / gesture (SBRE - **Section 4.3**) with Pepper
- The tracking (RWTH - **Section 4.7**) with Pepper, the smart wheelchair and Cuybot
- The global planning (ETH - **Section 4.10**)

This category of scenarios will have different levels of complexity. Such scenarios consider a medium (between 0.5 p/m2 and 1 p/m2) to high density (more than 1 p/m2), in order to benchmark the limits of the technology developed in terms of crowd density.

Also, the presence of obstacles is considered, but we consider small obstacles that can be easily avoidable with short term navigation. For instance, pillars, or short walls, can be easily avoided by the robot. Maze or complex corridor layout is not part of this category.

5.2. Scenario Category: Flow

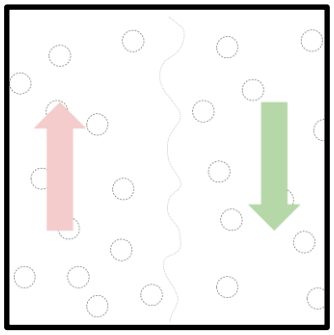

Figure 46. FLOW

The *flow* category (Figure 46) is similar to the static category in the sense that the environment remains simple, with no or few obstacles, and large space or simple layout of corridors.

However, the major difference between the static and the flow category is that the crowd is now dynamic. Indeed, in this category of scenarios the crowd is moving, in particular, each member of the crowd is moving in a particular direction. We consider that there are no more than 2 possible directions of flow, and that an agent of the crowd moving in one direction will never change the direction of its goal.

Such scenarios are pretty common in our everyday life, for instance, in busy walking streets (Figure 47).

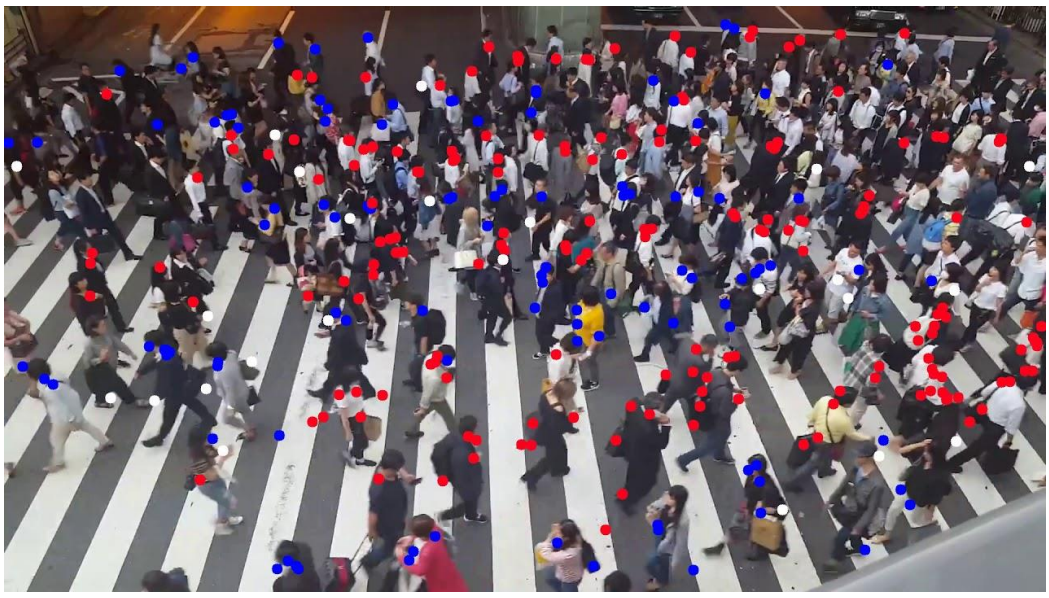

Figure 47. 2D flow in a busy street, blue and red labels oppose each other

Also, another example is a train station (Figure 48), when people are coming out of the train, they are all heading toward the exit, and while they move at various speeds, each agent in the crowd can over-take people when necessary.

Figure 48. 2D flow in a train station

Such scenarios are necessary to test further the navigation stack of CrowdBot, i.e. when various planning approaches are appropriate and when different modes of motion and interaction are needed. For instance, in such context, we already know that the global planner will fail as it becomes too complex for the robot and its sensors to observe enough features of the static environment without uncertainties, that are used for the localisation of the robot and the mapping of the environment. Also, in such a situation, we consider that even if the robot is able to localize itself, the crowd is at a density level that the robot is constantly moving according to its avoidance algorithm and reactive navigation which should take priority over the global planning.

Also, some technologies are only useful in the context of a dynamic crowd. For instance, the crowd prediction and the LiDAR FLOW module are only truly tested if the crowd is moving. Also, the Followbot technology is only useful in the context of a moving crowd.

Overall, with such scenario category, we intend to test, evaluate and benchmark the following technologies:

- The reactive navigation (EPFL - **Section 4.1**) with Pepper
- The social signaling / gesture (SBRE - **Section 4.3**) with Pepper
- The crowd prediction (INRIA - **Section 4.4**) with the smart wheelchair and Pepper
- The Followbot module (INRIA - **Section 4.5**) with Pepper
- The tracking (RWTH - **Section 4.7**) with the smart wheelchair and Cuybot
- The LiDAR FLOW module (RWTH - **Section 4.8**)

Similarly to the static category, this category has several parameters that define different levels of difficulty. First, the density of the crowd, from medium (between 0.5 p/m2 and 1 p/m2) to high density (more than 1 p/m2), complicates the task of navigating the robot. Also, the number of flows, 1D or 2D, and their angle of incidence, from 0 to 180, has an effect on the maneuvers the robot has to perform to move in a safe way. Finally, the disparity of the behaviors in the crowd, in particular, the difference in the preferred velocity between each agent, influences the number of overtakes in the crowd, thus leading to a higher entropy, which challenges the reactive navigation.

5.3. Scenario Category: Sparse

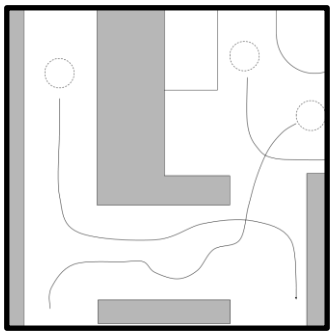

Figure 49. SPARSE

The category *Sparse* (Figure 49) is regrouping the scenarios in which the crowd is scattered in a complex environment. In such scenarios, the density of the crowd is low, and the goal of each person in the crowd is undefined. Such scenario is very common in real life. For instance, the main square of a city center on a normal weekday will not be very crowded, and the people in the crowd are just going through the place.

Figure 50. A public place with a few people

Another example will be an open space (Figure 51) in which people are mostly static at a desk, but might move at some point.

Figure 51. An open space

Such a situation is less likely to have occlusions of people by other people than the flow and static categories. As a consequence, it is less challenging for the crowd tracker and crowd prediction.

However, as the layout of the environment increases in complexity, the ability of the robot to navigate is challenged. This is the classical problem of navigating a mobile robot that has to be evaluated for the CrowdBot robot.

Note that the presence of people in the environment will occlude some of the features used by the robot to localize itself. Therefore, the localisation is challenged by the crowd density and activity.

In such a category of scenarios, we intend to test, evaluate and benchmark the following technologies:

- The reactive navigation (EPFL - **Section 4.1**) with Qolo
- The low-level planner (ETH - **Section 4.2**) with Cuybot
- The crowd prediction (INRIA - **Section 4.4**) with the smart wheelchair, Pepper, and Cuybot
- The Followbot module (INRIA - **Section 4.5**) with Pepper
- The tracking (RWTH - **Section 4.7**) with the smart wheelchair, Cuybot, and Pepper
- The reinforcement learning (for neural-net training) (ETHZ - **Section 4.6**) with Pepper
- Global planning (ETH - **Section 4.10**) With Cuybot
- Physical Interaction (SBRE - **Section 4.11**) with Pepper

Once again, the sparse category covers different scenarios which are more or less challenging for the different technologies. In particular, the crowd density, from low (0+ p/m2 to 0.5 p/m2) to medium (0.5 p/m2 to 1 p/m2), and crowd activity, from static to various levels of velocities, will influence the scenario complexity.

Also, the layout of the environment, from a wide public place (Figure 50) to a maze with narrow corridors, will help us to identify the limits of our technologies.

5.4. Scenario Category: Mixed

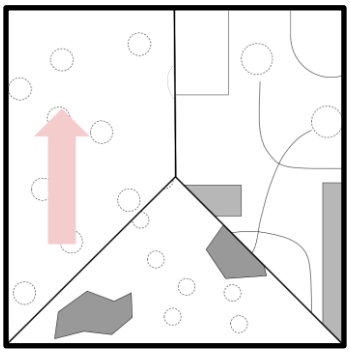

Figure 52. MIXED

While we believe that the categories *static, flow, and sparse* already cover a lot of the required scenarios, a more realistic situation is actually a mix of each of these (Figure 52).

The category *mixed* covers very specific category scenario in which the crowd and the environment change in time and space. Such a situation is very common in a train station (Figure 53): near the train platform, people will wait for their trains, and when it finally arrives, they will rush toward their cabin, while other people will exit the train and walk in the opposite direction. Also, as we walk away from the train platform, we reach a part of the train station which has a more complex layout, with corridors, and the crowd is sparse, walking around between the shops.

Figure 53. A train station with mixed scenarios

The most important feature of such scenarios is the notion of change in the situation. This will allow us to evaluate and benchmark one specific technology: The High-level Multi-modal Behaviour Planning with the robot Pepper.

This category challenges the ability of our robots to handle the changes in situation.

6. Specification of Requirements

Compared to *D1.1*, the updates of this section are mainly about the performance metrics that we introduce so as to perform evaluations (Section 6.1). We updated the requirement table as well so as to focus more directly on scenarios of major interest for the project.

This section serves the main theme of this document: specification of requirements for all test scenarios that are planned for the Crowdbot project. Note that the test scenarios (**Section 5 in D1.1**) and requirements list (Table 3) have been updated by the consortium as a result of our initial tests and stakeholder engagement activities. In particular, we have re-focussed our mandatory requirements to better fit with the identified scenarios that deal with the primary contributions of Crowdbot (i.e. navigation in crowds) and made some of our earlier identified requirements optional, where they are more suited to general mobile robotics sensing and planning (e.g. the ability to deal with varying terrain etc.) An additional key contribution to this document is the consortium's agreement about a set of common metrics (**Section 6.1**) that will be used, as appropriate, in the evaluations. Not all metrics will be applicable to all robots and scenarios, but where they are used, it will allow us to better compare between robots, scenarios and techniques.

As noted in **Section 5 in D1.1**, there are seven main test scenarios among which some have additional sub-scenarios for further testing. Additionally, there are 4 categories for technology specific testing (cf. **Section 5**). For ease, we use the same terminology as in *D1.1*.: each requirement is notated using the convention R x.y for easy referencing (x is the main scenario ID). For example, R 4.2 is the 2nd requirement item from the 4th main scenario. Additional details associated with each requirement item are:

1) *Robot*: Humanoid (H), Wheelchair (W), Service (S) or All (A)
2) *Metric*: see **Section 6.1** for complete details
3) *Venue*: Physical (P), Simulation (S) or both (B)
4) *Type*: Mandatory (M) or Optional (O)

Some other constraints for testing and requirements fulfilment:

1) Since the Crowdbot team has access to three different types of robots (say, A, B and C), each requirement item must state which platform(s) it applies to. If the main scenario is applicable to robot type A only, then its derived requirements (sub-scenarios and requirement items) are not applicable to another platforms B or C.
2) The physical test venues have so far been indoor, such as a partner's laboratory facility, a gymnasium, sporting facility or some other large room or warehouse. Office hallways and corridors can also be used to test 1D and 2D flow scenarios. Outdoor testing is optional and limited to the wheelchair, where we have already collected data from the UCL Bloomsbury Campus (c.f. D1.4) and anticipate further evaluations on the Queen Elizabeth Olympic Park.
3) Physical tests are carried out under the leadership of Work Package 1 and 5 (WP1 & 5). Test cases are strictly limited to availability of human and robot resources, access to a physical venue and regulatory clearance from local and national authorities. In contrast, computer-based simulation tests have no such restrictions except for the team's ability to complete and run high-fidelity simulation models and tools. Simulation based tests are conducted by the leadership of Work Package 4 (WP4) and make use of Virtual Reality and Real Lab environments.
4) Since test cases may be conducted at different partner sites, the same test setup (physical space, size and type of humans and object clutter) at one test site may differ from another, resulting in different test outcomes. This issue in test data discrepancy, if it exists, will be investigated, recorded and clarified in the Test Report (TR).
5) If a requirement is labelled as "mandatory", then the test scenario associated with this requirement must be completed, corresponding test results must be published and shared with stakeholders. In case a mandatory requirement is not fulfilled, the Test Report (TR) must provide an explanation.
6) Optional requirements are listed as "to do" items if there is additional time for completion or the scenario is an advanced extension of a mandatory requirement. If completed, optional

requirements and their test outcomes are included in the test report. If not, they are simply omitted from the Test Report (TR).

An example of a requirement item to be populated in Table 3:

Item No.	Description	Robot	Metric	Venue	Type
R 0.1	Robot must move on wheels.	A (All)	B	Both (B)	Mandatory (M)

6.1. Performance Metrics

Each requirement item is labelled exclusively as one of the following three metric types:

1) Success Threshold (Quantitative or Type A): Whether a requirement from a test scenario is met or not is assessed by comparing a measured quantity to a success threshold. For example, the execution time to reach from Point A to Point B is measured in seconds. If a robot achieves the task under a pre-defined total time elapse value, then this requirement is met. In most cases, due to the lack of prior experimental data, the requirement item may not explicitly state the threshold value but it is left to the Technology and Test & Evaluation teams to propose an acceptable threshold before actual tests commence.
2) Success Threshold (Qualitative of Type B): In certain test scenarios, it may not be possible to measure a robot's performance quantitatively or with a meaningful metric. For example, in the same example as above, in moving from Point A to Point B, a robot may traverse via many different paths with no clear winning path. However, if a robot maneuvers in a motion profile deemed to be unsafe or unpredictable, then this requirement is unmet. Typical metric values are Yes/No, Pass/Fail or a numeric scale (1 to 10).
3) Discovery (Measured Data & Assessment or Type C): Since Crowdbot is exploring new robotic features and test scenarios that have never been documented nor replicated, in certain cases, we do not have a baseline to gauge the success level of a test outcome. For such test scenarios, we simply report our findings (test set up, collected data, analysis and assessment) and they serve as discoveries/outcomes that meet this requirement. We propose to analyze according to the evaluation metrics.

6.1.1. Evaluation Metrics

The team of CrowdBot has agreed on 6 categories of evaluation metrics in order to analyze the path in different ways. Indeed, a particular path might be very efficient but have a substantial effect on the crowd.

The 6 categories are *Path efficiency, Effect on crowd, Pedestrian-robot similarities, Crowd-robot interaction, Collision, and Shared control quality, and are described below*

6.1.1.1. Path efficiency

The path efficiency regroups the metrics which evaluate the robot efficiency to reach the goal when it is alone in its environment, compared to its efficiency when it is doing the same task surrounded by a crowd. Such metrics are usable in controlled environments (simulation or experiments with informed crowd).

- Relative time to goal

The relative time to goal compares the time taken by the robot to reach its goal when it is alone to the time it takes when it is in a crowd. The metrics is given below:

$$T_{rtg} = \frac{T_{robot\ alone}}{T_{crowded}}$$

- Relative path length

The relative path length compares the length of the path taken by robot to reach its goal when a crowd is present or not. Mathematically it is given by the following formula:

$$L_{rp} = \frac{L_{robot\ alone}}{L_{crowded}}$$

- Relative jerk

The relative jerk evaluates the smoothness of the path taken by the robot to reach its goal when it is alone compared to when it is in a crowd. It is given below:

$$J = \int_0^{T_{robot}} (\dot{a} + \ddot{\theta})$$

$$J_{rp} = \frac{J_{robot\ alone}}{J_{crowded}}$$

6.1.1.2. Effect on crowd

This category regroups the metrics evaluating the effects the robot has on the crowd.

- Effect on time

The effect on time is measured by comparing the average time taken by the agents of the crowd to reach their goal when there are no robots and the average time taken by the crowd when the robot is present. It is given by the following formula

$$E_t = \frac{T_{crowd\ alone}}{T_{crowd\ +\ robot}}$$

- Effect on velocity

The effect on velocity compares the average velocity of the crowd when there is a robot on not. The formula is given below:

$$E_v = \frac{V_{crowd\ alone}}{V_{crowd\ +\ robot}}$$

- Neighbor's time to goal

The neighbor's time to goal compares the average time taken by the neighbors of the robot to the one taken by the crowd. The mathematical formula is given below:

$$N_{ttg} = \frac{T_{robot\ neighbors}}{T_{crowd}}$$

- Neighbor's velocity

The neighbor's velocity compares the average velocity of the neighbors of the robot to the one of the crowds, on average. The mathematical formula is given below:

$$N_v = \frac{V_{robot\ neighbors}}{V_{crowd}}$$

6.1.1.3. Pedestrian-robot similarities

This section of metrics compares directly the robot path to the one of the agents, showing how similar a robot path and an agent path are. We use statistics on the crowd as a reference for the agent. A value of "1" for means a high similarity between the robot path and the average agent of the crowd.

- Time to goal similarity

The time to goal similarity compares the time taken by the robot to reach its goal to the time taken by an agent of the crowd (average time of the crowd). The metrics is given below:

$$\Delta_{ttg} = \frac{T_{robot}}{T_{agent}}$$

- Path length similarity

The path length similarity compares the length of the robot path to the length of the average agent path. The formula for this metric is given below:

$$\Delta_{L} = \frac{L_{robot}}{L_{agent}}$$

- Heading similarity

The heading similarity shows similarities between the variation of rotation between the robot and the average agent of the crowd. The formula for this metric is given below:

$$\Delta_{\theta} = \frac{1}{T_{robot}} \sum_{0}^{T_{robot}} \left(\frac{\dot{\theta}_{robot}}{\dot{\theta}_{agent}}\right)$$

- Velocity similarity

The velocity similarity compares the robot velocity to the one of the average agent. The mathematical formula is given below:

$$\Delta_{v} = \frac{1}{T_{robot}} \sum_{0}^{T_{robot}} \left(\frac{V_{robot}}{V_{agent}}\right)$$

In the same idea we compare the velocity of the robot to the average velocity of its neighbors with the following formula:

$$\Delta_{v|N} = \frac{1}{T_{robot}} \sum_{0}^{T_{robot}} \left(\frac{V_{robot}}{V_{neighbors}}\right)$$

- Overtaking count

This metric counts the number of overtakes the robot performs, to be compared with the average number of overtakes of an agent of the crowd.

A high score means the robot is more aggressive than the rest of the crowd, while a low score means the robot is blocking the crowd.

6.1.1.4. Crowd-robot interaction

- Proximity

We define proximity as the shortest distance to agents during the whole trajectory of the robot. The formula is given below:

$$P = \sum_{0}^{T_{robot}} d_{min}^{-1}$$

This metric is robot dependant and crowd dependant, but can be used to compare different robots in similar situations.

- Normalized proximity

This metric is defined as the normalized distance between the robot to the barycenter of its neighbors.

This metric is density independent, thus relevant in scenarios with different crowd density.

$$\hat{P} = \frac{std(d1, \ldots, dn)}{avg(d1, \ldots, dn)}$$

Where “std” is the standard deviation and “avg” is average for distances between the robot and its neighbors.

6.1.1.5. Collision

This category of metrics is regrouping anything collision related.

- Risk

We first define the notion of distance to closest approach (dca), which is, at a given time, the minimal distance in meters between an agent of the crowd and the robot if both keep their current velocity in the future. If this distance is 0, then they will eventually collide if they do not try any avoidance maneuver. The time of closest approach (ttca) is the corresponding time in seconds, at which the collision happens.

The risk is defined as the sum of the minimal ttca if dca is 0 at each time step.

$$R = \sum_0^{T_{robot}} (ttca_{min}\ if\ dca = 0, 0\ otherwise)$$

- Expected Severity

We define the severity in a similar way as the risk, but we take the relative velocity into account:

$$S_R = \sum_0^{T_{robot}} (v_{relative}^2 \times ttca_{min}\ /if\ dca = 0, 0\ otherwise)$$

- Number of collisions

This metric is just the count of collisions, noted C, that occurred in the scenario. Note that a collision is not necessarily dangerous, as a touch is already a collision.

- Collision severity

We define collision severity as the sum of the momentum of each collision that actually occurred during the scenario. It is defined as below:

$$S_C = \sum_0^{T_{robot}} (m_{robot} + m_{agent})/2 \times (v_{robot} - v_{agent})^2\ \ ||when\ colliding\ (ttca = 0)$$

6.1.1.6. Shared control metrics

- Agreement

We defined agreement in terms of the deviation of the direction of user's commands from the direction of the final shared control's velocity. Mathematically, agreement is given below:

Define direction $$\theta(u) = tan^{-1}(\frac{v}{w})$$

$$a_i = 1 - \frac{|\theta(z_h^i) \ominus \theta(u_{SC}^i)|}{\pi}$$

$$agreement = \frac{\sum_{i=0}^{N} a_i \cdot \Delta t_i}{\sum_{i=0}^{N} \Delta t_i}$$

where v and w are the translational and rotational velocities $u \sim [v\ w\ ]$, a_i is the normalised agreement at time step, u_{SC}^i is the final output of the

shared control (PSC or linear blending). N is the number of samples available in which data from both the measured input of the user, z_h^i coincide in time with u_{SC}^i . Δt_i is the duration of user's input command z_h^i.

For the same wheelchair user, we expect the agreement to be high when there are no or only a few obstacles in proximity. On the contrary, when there are obstacles in proximity, agreement would be lower as the wheelchair is providing more assistance. Indeed, this metric would be more informative if analyzed with other metrics such as user input entropy and objective metrics.

- User input entropy

Entropy determines how much information or disorder is in the user's command. The more erratic and jerkier the user's input, the more entropy the user's input exhibits. Thus, entropy can indicate the degree of difficulty in a similar manner to jerk. Entropy is estimated from the angular component, $tan^{-1}(\frac{v(z_h)}{w(z_h)})$ of the user's input [49].

- Fluency of Commands

we observe the fluency of commands of the user (temporal continuity) given that the reactive navigation is intervening with its desired motion, as a second reference measurement of the disagreement with the decisions of the assistive navigation.

$$F \quad = \frac{1}{N} \sum_{t=t_0}^{t_N} 1 - \ |u_h^t - u_h^{t-1}|$$

where u_h^t represents the user given command at time t, and N the number of samples taken in the interval $t_0 \ \ to \ t_N$.

- Objective metrics

In a controlled physical experiment, self-assessed metrics of performance can be used to evaluate shared control from the perspective of the participant [50]. The perceived workload is captured by the NASA-TLX. The idea here is that better performing shared control will reduce the perceived workload in comparison to other shared control. The USE questionnaire captures how well a user is satisfied with the shared control, and it is important because we want users to be content using our shared control.

6.1.2. Requirement table

All requirement items are listed in Table 3, which we have updated since *D1.1*, based on test results and recommendations from external stakeholders. In particular some of the mandatory requirements have been made optional to better fit with the identified scenarios and focus on the primary contributions of Crowdbot (i.e. navigation in crowds), rather than general mobile robotics sensing and planning (e.g. the ability to deal with varying terrain etc.).

Table 3. Test Scenario Requirements List

Item No.	Description	Robot	Metric	Venue	Type
R 0.1	Robot must move on wheels.	A	B	P	M
R 0.2	Robot navigation test is conducted with indoor ambient (natural or artificial electric) lighting.	A	B	P	M
R 0.3	Wheelchair navigation test is conducted with outdoor ambient natural lighting.	W	B	P	O
R 0.4	Wheelchair navigation test is conducted on rough terrain.	W	B	P	O
R 0.5	Applicability and adaptability of a technology profile used for navigation to another robot in the same class (humanoid, service or wheelchair) must be presented.	A	A	P	O

R 0.6	Human-Machine Interface device and communication method used for the robot must be specified.	A	A	P	M
R 0.7	Two-way wireless communication method, if used, must be specified.	A	A	P	M
R 0.8	Robot must be equipped with a kill switch that deactivates and stops its forward/backward/sideway motion.	A	B	P	M
R 0.9	Structural augmentation of robot must be specified.	A	A	P	M
R 0.10	Electrical/electronic augmentation of robot must be specified.	A	A	P	M
R 0.11	Additional computing resources used must be specified.	A	A	P	M
R 0.11	Changes to dimensions, weight and locomotion of robot, if any, must be specified.	A	A	P	M
R 0.12	Removal or disabling of embedded sensors, processors and electronic devices or modules must be specified.	A	A	P	M
R 0.13	Power supply and energy source modifications to accommodate system augmentation must be specified.	A	A	P	M
R 1.1	Robot follows humans in 1D flow in a straight-line path; all agents have the same speed; no over-taking	A	B	B	M
R 1.2	Robot follows humans in 1D flow in a straight-line path; all agents have the same speed but a different value from a prior test; no over-taking	A	B	B	O
R 1.3	Robot follows humans in 1D flow in a straight-line path; Some human agents have different but constant speeds; over-taking may occur.	A	B	B	M
R 1.4	Robot follows humans in 1D flow in a straight-line path; Human agents increase acceleration and robot must adapt its motion profile to continue following; no over-taking	A	B	B	M
R 1.5	Robot follows humans in 1D flow in a straight-line path; Human agents decrease acceleration and robot must adapt its motion profile to continue following; no over-taking	A	B	B	M

R 1.6	Robot follows humans in 1D flow in a non-straight curve line path; all agents have the same speed; no over-taking	A	B	B	O
R 1.7	Robot follows humans in 1D flow in a straight-line path. A human agent that the robot is following over-takes another human. The robot then follows another human.	A	B	B	M
R 1.8	Robot follows humans in 1D flow in a straight-line path. A human agent behind the robot over-takes the robot.	A	B	B	M
R 1.9	Robot follows humans in 1D flow in a straight-line path. The robot over-takes the human that is in front of the robot.	S, W	B	B	M
R 1.10	Robot follows humans in 1D flow in a straight-line path. Flow congestion is induced by adding human agents to flow.	A	B	B	M
R 1.11	Robot follows humans in 1D flow in a straight-line path. Flow congestion is induced by narrowing the passage.	A	B	B	M
R 1.12	Robot follows humans in 1D flow in a straight-line path. Flow congestion is induced by having all agents exit a door at the end of passage.	A	B	B	M
R 1.13	Robot follows humans in 1D flow in a straight-line path; obstacles (physical objects) are added along the passage. All agents have the same speed; no over-taking	A	B	B	M
R 1.14	Robot follows humans in 1D flow in a non-straight curve line path; obstacles (physical objects) are added along the passage. All agents have the same speed; no over-taking	A	B	B	O
R 2.1	Robot follows humans in 1D flow in a straight-line path; another 1D flow in the opposite direction also exists; all agents have the same speed; no over-taking; opposite flow only on one side of the robot.	A	B	B	M
R 2.2	Robot follows humans in 1D flow in a straight-line path; other 1D flows in the opposite direction also exists; all	A	B	B	M

	agents have the same speed; no over-taking; opposite flows on both sides of the robot.				
R 2.3	Robot follows humans in 1D flow in a straight-line path; obstacles are added along the passage; other1D flows in the opposite direction also exists; all agents have the same speed; no over-taking; opposite flows on both sides of the robot.	A	B	B	M
R 2.4	Robot follows humans in 1D flow in a straight-line path; another 1D flow in the opposite direction also exists; over-taking in the opposite flow is induced; opposite flow only on one side of the robot.	A	B	B	M
R 2.5	Robot follows humans in 1D flow in a straight-line path; another 1D flow in the opposite direction also exists; over-taking of the robot by a human in the same flow is induced; opposite flow only on one side of the robot.	A	B	B	M
R 3.1	Robot marches forward in a 1D x 1D, 90-degree cross flow. All human agents also march forward using social norm.	A	B	B	O
R 3.2	Robot marches forward in a 1D x 1D, 90-degree cross flow. Some human agents march forward using social norm and some make left or right hand turns in front of robot.	A	B	B	O
R 3.3	Robot marches forward in a 1D x 1D, 90-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is partially occluded.	A	B	B	O
R 3.4	Robot marches forward in a 1D x 1D, 90-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is fully occluded.	A	B	B	O
R 3.5	Robot marches forward in a 2D x 2D, 90-degree cross flow. All human agents also march forward using social norm.	A	B	B	O
R 3.6	Robot marches forward in a 2D x 2D, 90-degree cross flow. Some human agents march forward using social norm and some make left or right hand turns in front of robot.	A	B	B	O

R 3.7	Robot marches forward in a 2D x 2D, 90-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is partially occluded.	A	B	B	O
R 3.8	Robot marches forward in a 2D x 2D, 90-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is fully occluded.	A	B	B	O
R 3.9	Robot marches forward in Shibuya 2D x 2D 90-degree cross flow. All human agents also march forward without social norm.	A	B	B	O
R 3.10	Robot marches forward in Shibuya 2D x 2D, 90-degree cross flow. All human agents also march forward without social norm. Cross flow human density is high such that the forward path is partially occluded.	A	B	B	O
R 3.11	Robot marches forward in Shibuya 2D x 2D, 90-degree cross flow. All human agents also march forward without social norm. Cross flow human density is high such that the forward path is fully occluded.	A	B	B	O
R 3.12	Robot marches forward in Shibuya 2D x 2D 60-degree cross flow. All human agents also march forward using social norm.	A	B	B	O
R 3.13	Robot marches forward in Shibuya 2D x 2D, 60-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is partially occluded.	A	B	B	O
R 3.14	Robot marches forward in Shibuya 2D x 2D, 60-degree cross flow. All human agents also march forward using social norm. Cross flow human density is high such that the forward path is fully occluded.	A	B	B	O
R 3.15	Robot marches forward in Shibuya 2D x 2D 60-degree cross flow. All human agents also march forward without social norm.	A	B	B	O
R 3.16	Robot marches forward in Shibuya 2D x 2D, 60-degree cross flow. All human agents also march forward without social norm. Cross flow human density is high such that the forward path is partially occluded.	A	B	B	O
R 3.17	Robot marches forward in Shibuya 2D x 2D, 60-degree cross flow. All human agents also march forward	A	B	B	O

	without social norm. Cross flow human density is high such that the forward path is fully occluded.				
R 4.1	Robot make a left or right hand turn at the junction of a 1D x 1D 90-degree cross flow. All human agents also march forward using social norm.	A	B	B	M
R 4.2	Robot make a left or right hand turn at the junction of a 1D x 2D 90-degree cross flow. Cross flow traffic is 2D. All human agents also march forward using social norm.	A	B	B	M
R 4.3	Robot make a left or right hand turn at the junction of a 2D x 2D 90-degree cross flow. All human agents also march forward using social norm.	A	B	B	O
R 4.4	Robot follows humans along a curved corridor in 1D flow. All agents march at the same speed.	A	B	B	M
R 4.5	Robot first follows and then leaves 1D flow at a specified angle and marches forward.	A	B	B	M
R 4.6	Robot first follows and then leaves 2D flow at a specified angle and marches forward. Robot must cross over incoming flow from opposite direction.	A	B	B	O
R 4.7	Robot joins 1D flow from initial rest position.	A	B	B	M
R 4.8	Robot leaves 1D flow from initial following motion and then stops at any location outside of flow area.	A	B	B	M
R 4.9	Robot leaves 1D flow from initial following motion and then stops at predefined location outside of flow area.	A	B	B	O
R 5.1	Robot squeezes between two boundaries without colliding either of them.	A	A	P	M
R 5.2	Robot makes tight 90 degree left or right turn without colliding boundaries.	A	A	P	M
R 5.3	Robots makes 180-degree U turn without colliding boundaries.	A	A	P	O
R 5.4	Robots makes 180-degree U turn in a corridor without colliding boundaries and using forward motion only.	A	A	P	O
R 5.5	Robot maneuvers over or around a protruding obstacle on the ground plane.	A	A	P	O

R 5.6	Robot maneuvers over or around a small indented obstacle on a ground plane.	A	A	P	O
	Robot maneuvers around a large indented obstacle (a ditch) on a ground plane.	A	A	P	O
R 5.7	Robot maneuvers around a thin pole obstacle on the ground plane.	A	A	P	O
R 5.8	Robot maneuvers around a thick pole obstacle on the ground plane.	A	A	P	O
R 5.9	Robot maneuvers around a glass object on the ground plane or on the side as a wall or partition.	A	A	P	O
R 5.10	Robot identifies and stops at a frontal cliff edge.	A	B	P	O
R 5.11	Robot identifies and avoids a cliff edge on the side of path.	A	B	P	O
R 5.12	Robot moves along an inclined (uphill) surface.	A	A	P	O
R 5.13	Robot moves along an inclined (uphill) surface and stops at pre-defined localized position.	A	A	P	O
R 5.14	Robot moves along an inclined (downhill) surface.	A	A	P	O
R 5.15	Robot moves along an inclined (downhill) surface and stops at pre-defined localized position.	A	A	P	O
R 5.16	Robot moves along side-inclined surface while maintaining forward march.	A	A	P	O
R 5.17	Robot moves along side-inclined surface and stops at pre-defined localized position.	A	A	P	O
R 6.1	Robot can detect one-time contact with an external object.	A	B	P	M
R 6.2	Robot can detect non-contact with an external object after engaging in one-time or continuous contact.	A	B	P	O
R 6.3	Robot can detect continuous contact with an external object while at rest.	A	B	P	O
R 6.4	Robot can detect continuous contact with an external object while moving.	A	B	P	O
R 6.5	Robot can measure force impact during contact with an external object.	A	A	P	O

R 6.6	Robot can measure pressure exertion after contact with an external object.	A	A	P	O
R 6.7	Robot can measure change in pressure exertion after maneuvering to and from an external object.	A	A	P	O
R 6.8	Robot can detect multiple contacts originating from different directions.	A	B	P	O
R 6.9	Robot can simultaneously detect multiple contacts originating from different directions.	A	B	P	O
R 6.10	Robot can detect and differentiate multiple contacts originating from different directions.	A	B	P	O
R 6.11	Robot can simultaneously detect and differentiate multiple contacts originating from different directions.	A	B	P	O
R 6.12	Any navigation malfunction during test must be reported.	A	A	P	M
R 6.13	A red team is formed to deliberately trigger a malfunction.	A	A	B	O
R 6.14	Robot's margins before falling are measured by injecting force from the side.	A	A	P	O
R 6.15	Robot's margins before falling are measured by injecting force from the front.	A	A	P	O
R 6.16	Robot's margins before falling are measured by injecting force from the back.	A	A	P	O
R 6.17	By tilting the robot from the side (if possible), its force exertion of the external object is measured.	A	A	P	O
R 6.18	By tilting the robot from the front (if possible), its force exertion of the external object is measured.	A	A	P	O
R 6.19	By tilting the robot from the back (if possible), its force exertion of the external object is measured.	A	A	P	O
R 6.20	Robot navigation is assessed after disabling a sensor module via electrical, software kills or physical blind fold.	A	A	P	M
R 6.21	Robot navigation is assessed after disabling a processor module via electrical or software kill.	A	A	P	O
R 6.22	Robot's navigation accuracy is tested after long time-elapsed operation by repeating a squeeze test (through two boundaries) with collision.	A	A	P	O

R 6.23	Robot's navigation accuracy is tested after long time-elapsed operation by repeating a multi-waypoint localization and navigation test.	A	A	P	O
R 6.24	Robot's navigation accuracy is tested after long time-elapsed operation by repeating a relevant test procedure.	A	A	P	O
R 6.25	Robot is deactivated using onboard kill switch while moving.	A	B	P	M
R 6.26	Robot is deactivated using remote kill option if exists.	A	B	P	M
R 6.27	Robot is deactivated using at least two kill switch options.	A	B	P	M
R 6.28	Any abnormal robot behavior such as stall or freeze is reported.	A	A	P	M
R 7.1	Robot passes by stationary and 1D flow moving human and uses audio-visual cues and verbal announcements as needed.	H	B	P	M
R 7.2	Robot passes by stationary and 1D flow moving human and uses audio-visual cues and verbal announcements as needed. If passage is blocked, robot asks for clearance via verbal communication until passage is cleared.	H	B	P	M
R 7.3	Robot passes by stationary and 1D flow moving human and uses audio-visual cues and verbal announcements as needed. If passage is blocked, robot uses its gaze in the direction of humans that are blocking and asks for clearance via verbal communication and also until passage is cleared.	H	B	P	M
R 7.4	Robot passes by stationary and 2D flow moving human and uses audio-visual cues and verbal announcements as needed.	H	B	P	O
R 7.5	Robot passes by stationary and 2D flow moving human and uses audio-visual cues and verbal announcements as needed. If passage is blocked, robot asks for clearance via verbal communication until passage is cleared.	H	B	P	O
R 7.6	Robot passes by stationary and 2D flow moving human and uses audio-visual cues and verbal announcements	H	B	P	O

	as needed. If passage is blocked, robot uses its gaze in the direction of humans that are blocking and asks for clearance via verbal communication and also until passage is cleared.				

Those requirements scenarios reference the General scenarios detailed in **Section 4 of D1.1**. Required Technology specific scenarios are not detailed here as the technology is subject to evolution during the project. It is therefore more relevant to specify such requirements in the test plan document prepared during the first step of validation of each technology, one month before the actual validation.

6.2. Validation & Verification of Tests

The following order of execution are planned for each round of tests:

1) System-Level Test Plan (STP): A month prior to test events, the T&E team in collaboration with other internal teams (Technology Development, System Integration, Design and Quality Control) prepares a test plan document. It provides a list of test cases planned during one- to two-month long test event. Test cases are labeled as:
Example: T1.1: 1D Flow test with all agents moving at same constant speed
T1.2: 1D Flow with agents moving at different constant speeds and so on...
Note that specific details of each test case (time length of test, number of repetitions, number of human agents, speed value, etc.) may not be stated in STP since it is a system-level document. Actual values of test parameters used in each test case is recorded by the test execution team. A copy of STP is distributed (prior to the test event) to stakeholders to solicit their feedback and recommendations.
2) *STR Mapping*: Each test case is a representative of a test scenario detailed in **Section 4**. There may be not a one-to-one equivalence between a test scenario and a test case. A test case may be a subset of a test scenario but it may contain features and constraints not stated or considered in a test scenario. The purpose of STR mapping is to clarify the relationship between a test case **T** x.y, its best-matching test scenario S_m and its associated requirement items $R_{i.j}$, R_{m},n, …. The STR map is prepared by the T&E team during or after the test event since it must witness the test cases themselves to understand what are being tested and what are omitted. The STR map and its related information are documented in the main body of the Test Report (TR).
3) *Test Phase & Data Collection*: The test event duration is from one to three months. See Figure 54 for timelines. First- and second-round tests are planned for M22─25 and M39─42, respectively. Test cases can be spread across these months based on availability of the test team, a specific robot platform, human participants, test facility and other logistics and physical resources. Test cases can also be spread across different partner locations. STP will provide high-level information of each test case (date, location, test team, etc.) but test details are gathered and documented in the Test Report.
4) *Test Evaluation & Reporting*: Test data evaluation and preparation commences from the end of the test event and lasts exactly one month. A total of two test reports─TR1 and TR2─will be completed by end of project months M25 and M42, respectively. TRs are also released and distributed as official deliverables D1.4 and D1.5, respectively.

7. Scenario Specification Updates

This section is reported from *D1.1*. with minor updates.

This report D1.3 is the second of two requirements documents for the Crowdbot project. Its predecessor *D1.1: Specification of Scenarios Requirements* was published at the beginning of the project in October 2018 before the completion of 1st round physical tests and its accompanying evaluation report (see Figure 54 for details of the timelines). This assures that this deliverable considers lessons learned from the 1st round tests as well as feedback from stakeholders.

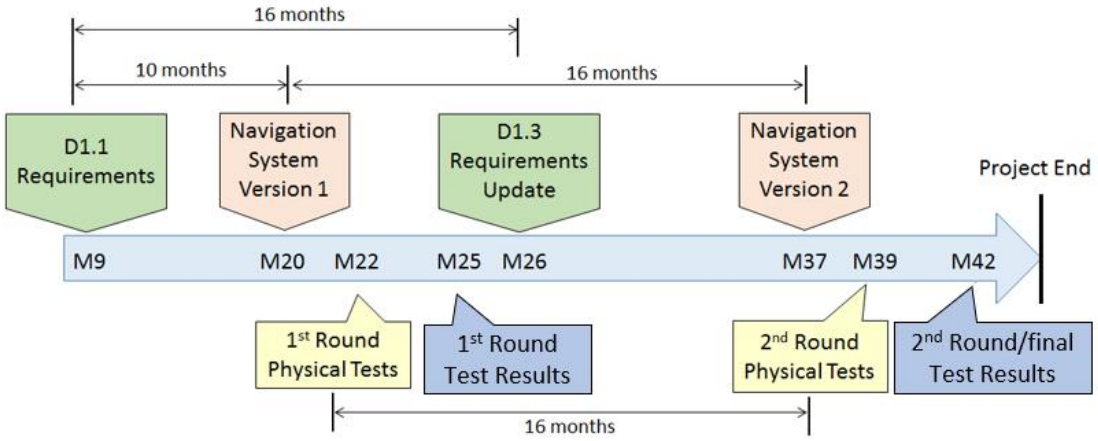

Figure 54. Initial timelines & dependencies among requirements

From Figure 54, we see that there are two rounds of physical tests with a time gap of roughly sixteen months in between for technology enhancements. Likewise, the time gap between completion dates for 1st and 2nd navigation systems is also sixteen months. The UCL Bloomsbury data collection and the associated quality check of the data were completed by 4th December 2019.

7.1. System-Level Test Plan (STP)

Two rounds of physical tests were planned on months 22 and 39 (October 2019 and March 2021) over the 42-month project time span. However, no specific dates were set in the original proposal for the delivery of a test plan. Furthermore, no action item was assigned to a Crowdbot team member as a Work Package task to prepare a test plan. This ambiguity was resolved in deliverable D1.1: *Specification of Scenarios Requirements.* The Test & Evaluation (T&E) team has taken the lead in the preparation of test plans for both 1_{st} and 2_{nd} round tests. It works closely with the remaining internal teams (Technology Development, System Integration and Design & Quality Control) teams to script specific tests to be carried out in each round.

The 1st round of physical tests took place at several partner sites (UCL, ETHZ, LOC, SBR) between July 2019 and January 2020. The results of these tests are reported in D1.4.

7.2. Simulation Tests

Another topic not covered in the proposal was the role of computer-simulation based navigation tests in fulfilling scenario requirements. The Work Package 4 (WP4) team is in charge of crowd simulation but this effort may not extend to robotic navigation system simulations. Two deliverables were anticipated for the crowd simulator: month 20 (August 2019) for the intermediate version and month 36 (December 2020) for the final version. No specific dates are assigned for the delivery of a robotic navigation simulator.

The proposed solution by the T&E team was as follows: The developer of the crowd simulator (INRIA) works closely with other Crowdbot teams (Technology Development, Design & Quality Control and Robot

Integration) to jointly develop a robotic navigation system simulator. Since the simulator itself is not undergo the process of independent validation and verification, it cannot be used to fulfill requirement items marked as "Quantitative Success Threshold." Its contribution is limited to the fulfillment of qualitative success and discovery requirement items only. More importantly, the simulator can be used as an aid in debugging software modules and robot integration efforts before the commencement of physical tests. Further details on the System Level Test Plan (STP) can be found in **Section 6.2.**

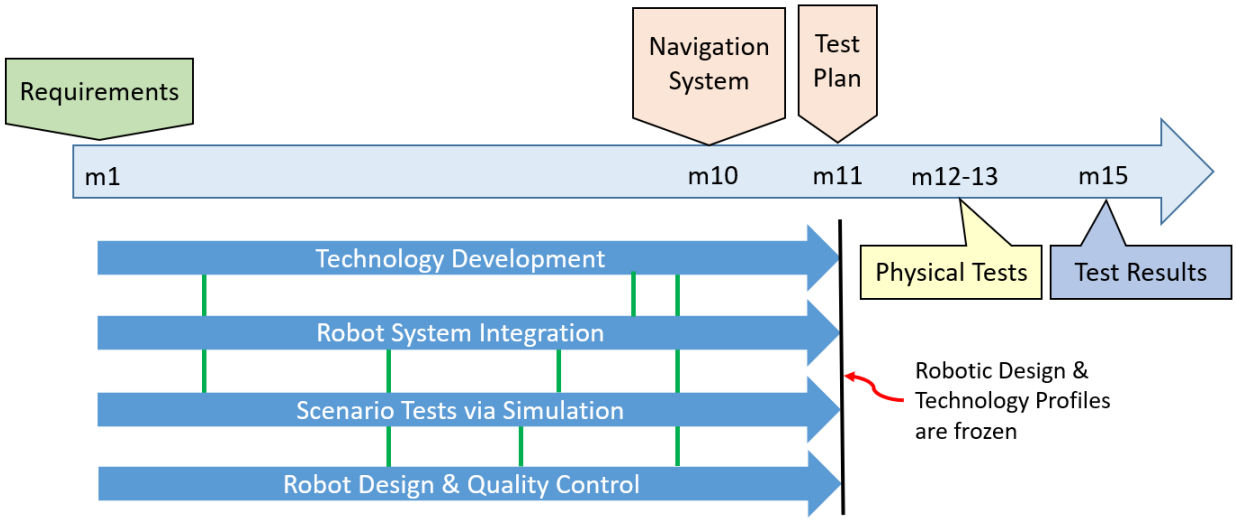

Figure 55. Fifteen-month Timeline & Activities for each round of Physical Tests

A summary of timelines and Work Package activities for each round of tests is provided in Figure 55. During the 15-month time span from the publication of the requirements document to the commencement of physical tests, parallel efforts are underway among Crowdbot teams and at certain project months, integration takes place. This is illustrated using vertical green lines in Figure 55 signifying team-to-team collaboration. Approximately two months before the commencement of tests, the Technology Development team releases its Navigation System (software code). The Test & Evaluation team then prepares a test plan to outline specific test scenarios and relevant requirement items. The specific robot design and navigation technology profiles are frozen before the preparation of the test plan.

At this stage, we have tested our collision avoidance algorithm with probabilistic shared control in the simulator, with the user input coming from an actual joystick. The first-round evaluation result has been reported in D1.4. Future tests will be carried out for proposed scenarios in the simulator first, before moving to the real-world evaluations.

7.3. Stakeholder Engagements (SHENG)

As detailed in Figure 55, there are several types of external stakeholders that the Crowdbot team will continually engage with and seek feedback regarding project progress and milestone achievements. In this section we limit our scope to the following stakeholders most pertinent to scenario development and test evaluation: general public, governmental and enterprise customers. These stakeholders are collectively known as the user community since they all play critical roles realizing in feasibility and implementation of Crowdbot technologies for real-world practical use. An outline of how we use stakeholders' feedback in improving test scenarios and technology enhancements are illustrated in Figure 56. Specific details of the process and procedures used for stakeholder engagement is provided in the following sections.

Our informal exchanges with stakeholder and our observations of the robots' navigation performance and interactions with the crowd (in particular, difficulties in indicating intent to move) have influenced the new test scenarios (described in **Section 5**).

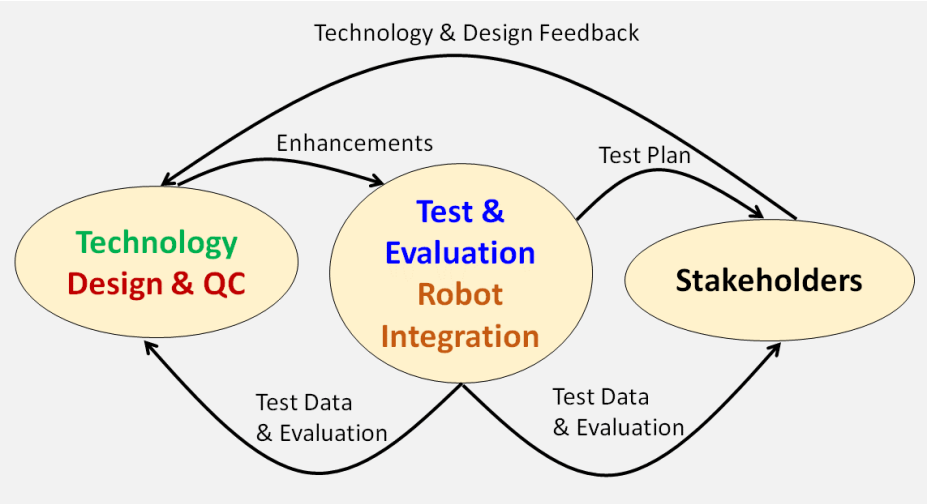

Figure 56. Feedback Cycles for Updating Test Scenarios

Involving stakeholders in CROWDBOT is crucial for defining and refining scenario requirements. The CROWDBOT project investigates safe robot navigation in dense crowds. The aim of the stakeholder engagement is to understand the type of interactions between robots (smart wheelchairs or humanoids) and crowds of people in order to define concrete scenarios in which to evaluate and test technologies developed within the project scope. UCL has submitted an ethics approval proposal for the CROWDBOT stakeholder engagement studies, which has been approved with ethics reference number UCLIC/1617/024/Staff Holloway/Herrera.

7.3.1. Recruiting Participants

In terms of the three categories of stakeholders, we further identify sub-groups for the general public to be ‘wheelchair users’, ’medical practitioners’, ’carers’, ‘building/property managers’, ‘Robot/wheelchair manufacturers’ and ‘other types of general public'. As for governmental stakeholder, we will recruit policymakers in the area of innovation and assistive technology.

The recruitment process involves sending out invitations for participation in either a questionnaire, structured interview or focus group. The invitations can be written or verbal and a consent form must be signed by each participant prior to participating in any study. The recruitment channel includes participants pool, social media, posters, charities, robotic and assistive technology events.

The first-round stakeholder engagement was conducted in March 2019, where we recruited 22 members of the general public globally for online questionnaires and 23 participants for semi-structured interviews.

7.3.2. Procedure

This study follows two main procedures, namely the structured interview and focus group study. A structured interview occurs in a public place and a participant is asked a fixed set of questions with his response’s video/audio recorded. Each participant is asked exactly the same set of questions for consistency. A structured interview is expected to last, at most, 60 minutes. The interview recordings will be outsourced to an external company for transcription. In the interest of safety and transparency, the researcher will never be alone in a room with a participant. All structured interviews will be conducted in the presence of other people. A focus group involves a group of six to eight participants discussing wheelchair scenarios. The discussion is moderated by the researcher who will also be taking notes during this process. A focus group is expected to last, at most, 90 minutes. Reasonable travel costs will be reimbursed for both the structured interview and focus group studies.

After the first round of engagement, we think it would be more efficient and cost-effective to include a questionnaire in our study, especially for getting responses from the general public. The questionnaire consists of closed questions with options and open questions where the participants can leave their comments. During the first round, we made an online questionnaire, and received responses from 22

participants worldwide. As this was done at the early stage of this project, only images of robot prototypes were shown to the participants, which may be different from our latest robot prototypes. As a result, the questions will be revised and videos with (partially) functioning robots will be included for the second-round engagement.

In addition, we are planning to test our wheelchair at UCL's Here East facility in London, with both controlled scenarios and natural crowds. Stakeholders including the general public, building manager and robot manufacturers will be invited to take part in the experiments which will be followed by a semi-structured interview.

7.3.3. Ethics/Privacy

Every structured interview and focus group will be video/audio recorded, however the participant can choose whether or not he wishes to be video recorded. All audio recordings will be anonymized in transcription and video recordings will not be shared or viewed by anyone outside the ethics approval, unless prior consent is given by the participant. Where consent is given, the video will be used for presentations at conferences or teaching material and will be anonymized. Apart from contact information (name, email address, telephone number, address) and recordings, no other personal information will be collected. Furthermore, contact information of participants will remain in the country where it was collected and will be deleted upon finalising this study.

The anonymized data collected will only be disclosed to the principal investigator, project partners, and researchers involved in this project. The results will be published in journals or conference proceedings.

In summary, stakeholder engagements are planned twice: before 1st and 2nd round tests. Feedback from each engagement will be used to tailor and enhance test cases in each round of test event.

7.4. Technology Enhancements

From Figure 55 one can infer that technology enhancements are part of ongoing R&D activities. New features are added after successful collaboration among different sub-teams. Milestones serve as time markers for major achievements in the project but fine-grained enhancements are targeted at monthly or bi-monthly intervals. These are not official deliverables but internal reports/documents and software builds. A major technology upgrade is expected after the release of first-round test reports and the publication of scenario updates D1.3. This includes the use of new sensing and computing devices, development of new test scenarios and modification of navigation algorithms and software code. Enhancements continue up until the commencement of 2nd round tests in month 39. By then the entire team shifts focus to test planning, preparation and execution. All technology enhancements are reported in the first and final test reports.

Figure 56 summarizes all entities involved in each round of tests. The Crowdbot team consists of four sub-teams: Technology Development (TD), Robot Design & Quality Control (QC), Test & Evaluation (T&E) and Robot System Integration (SI). The external entities are various stakeholders (see **Section 2.1** and Figure 9 for additional details). The two combined teams (TD and QC) develop technologies applicable to each robot platform. Each version of design and software build is submitted to the SI team where each robot undergoes technology enhancements. Based on current state of the resulting Crowdbot robot features, the T&E team prepares a test plan and lays out relevant test scenarios and applicable requirement items. Beyond internal circulation the test plan is shared with certain stakeholders as a courtesy. After tests are completed, T&E evaluates test data and prepares a report which is shared with all other Crowdbot teams and certain stakeholders. Feedback from stakeholders is incorporated into further technology enhancements. The above process occurs in two cycles, once for each physical test.

References

The references 1 to 35 are reported from *D1.1*. The references 36 to 57 are new compared to *D1.1*

[1] Starship Technologies, https://www.starship.xyz/
[2] Domino's Pizza delivery robot, https://qsrmedia.co.uk/technology/news/dominos-and-starship-launch-eu-pilot-program-autonomous-robots
[3] https://www.seattletimes.com/business/technology/delivery-robots-are-showing-up-on-city-sidewalks/
[4] Emergency waiting area in hospital, http://brazosportregional.org/healthcare_services/emergency_services.aspx
[5] Paris Gare du Nord train station: https://parisbytrain.com/wp-content/uploads/2013/02/gare_du_nord_platforms_eurostar_thalys_optimized.jpg
[6] Crowdbot Project D6.1: Overview of Risks when using Robots in Crowds, available in Dec. 2018.
[7] ANYmal Robot by ETH Zurich, http://www.rsl.ethz.ch/robots-media/anymal.html
[8] Savioke Robot, http://www.savioke.com/
[9] Savioke Relay robot for hotel room service, https://wtvox.com/robotics/relay-robot/
[10] TUG Robot, https://aethon.com/
[11] TUG Robot for luggage delivery, https://ktla.com/2018/02/19/luggage-robots-sheraton-hotel-san-gabriel/
[12] Quickie Electric Wheelchair, https://www.quickie-wheelchairs.com/
[13] SoftBank Robotics, https://www.softbankrobotics.com/emea/en/robots/pepper
[14] Willow Garage PR2 Robot, http://www.willowgarage.com/pages/pr2/overview
[15] Crowdbot Project D2.1: Sensor Specifications.
[16] Crowdbot Project D5.1: System Integration.
[17] Cross flows at Shibuya traffic junction: https://www.blogto.com/city/2007/10/could_we_walk_across_bay_and_bloor_diagonally/
[18] Tracking of large dense crowds: http://crcv.ucf.edu/projects/crowd/
[19] Presidential Inauguration Crowd, https://www.pbs.org/ image taken from live video coverage.
[20] Helbing, Dirk, Anders Johansson, and Habib Zein Al-Abideen. "Dynamics of crowd disasters: An empirical study." Physical review E 75.4 (2007): 046109.
[21] Plaue, Matthias, et al. "Trajectory extraction and density analysis of intersecting pedestrian flows from video recordings." Photogrammetric image analysis. Springer, Berlin, Heidelberg, 2011. 285-296.
[22] Liddle, Jack, Armin Seyfried, and Bernhard Steffen. "Analysis of bottleneck motion using Voronoi diagrams." Pedestrian and evacuation dynamics. Springer, Boston, MA, 2011. 833-836.
[23] Safe Robot Navigation in Dense Crowds, CROWDBOT Project, EU Horizon 2020 proposal to European Commission, H2020 ICT-25 2017 RIA.
[24] A. L. Barriuso et al, "Agent-based Intelligent Interface for Wheelchair Movement Control", Sensors, May 2018, vol. 18, 1511.
[25] Sunrise Medical Power Wheelchairs and Accessories; http://www.sunrisemedical.com/
[26] United States Department of Labor, Occupational Safety & Health Administration, https://www.osha.gov/law-regs.html
[27] Sheridan, T. B., & Verplank, W. L. (1978). Human and computer control of undersea teleoperators. Massachusetts Inst. of Tech., Cambridge Man-Machine Systems Lab.
[28] Groceries delivery via robots, https://bgr.com/2017/01/31/robot-delivery-pizza-starship-technologies/
[29] MediaMarkt package delivery robot, http://www.mediamarktsaturn.com/en/press/press-releases/pilot-project-launched-media-markt-test-delivery-customer-delivery-robot
[30] Domino's Pizza Robot crossing tram track in the Netherlands, https://www.nu.nl/119076/video/dominos-test-pizzabezorging-met-robot-in-amsterdam.html
[31] World Health Organization, Fact Sheet on Wheelchairs; Geneva, Switzerland, 2010.
[32] United Nations. Standard Rules on the Equalization of Opportunities for Persons with Disabilities; San Francisco, CA, USA, 1994.
[33] National Transportation Safety Board, https://www.ntsb.gov/_layouts/ntsb.recsearch/RecTabs.aspx
[34] Federal Aviation Authority, https://www.faa.gov/regulations_policies/
[35] Cuybot robot, Locomotec; http://www.locomotec.com/en/
[36] Janosch Nikolic, et al. "ROS Wiki: visensor". http://wiki.ros.org/visensor. Accessed: 20-09-2018.
[37] 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.
[38] 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
[39] Intel Corporation. "Depth Camera D435 - Intel RealSense Depth and Tracking Cameras". https://www.intelrealsense.com/depth-camera-d435/ Accessed 06-02-2020

[40] Y. F. Chen, M. Everett, M. Liu, and J. P. How, “Socially Aware Motion Planning with Deep Reinforcement Learning,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sept. 2017, pp. 1343-1350.
[41] M. Everett, Y. F. Chen, and J. P. How, “Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2018, pp. 3052–3059.
[42] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning,” in 2019 International Conference on Robotics and Automation (ICRA), May 2019, pp. 6015–6022.
[43] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, “A Data-driven Model for Interaction-Aware Pedestrian Motion Prediction in Object Cluttered Environments,” in 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018, pp. 5921–5928.
[44] D. Helbing, and P. Molnar, "Social force model for pedestrian dynamics," Physical Review E, 52(5), May 1995, pp. 4282-4286.
[45] M. R. M. Tomari, Y. Kobayashi, and Y. Kuno, “Development of Smart Wheelchair System for a User with Severe Motor Impairment,” Procedia Engineering, vol. 41, pp. 538–546, 2012.
[46] C. Ezeh, P. Trautman, L. Devigne, V. Bureau, M. Babel, and T. Carlson, “Probabilistic vs linear blending approaches to shared control for wheelchair driving,” in 2017 International Conference on Rehabilitation Robotics (ICORR), London, 2017, pp. 835–840.
[47] Qinan Li, Weidong Chen, and Jingchuan Wang, “Dynamic shared control for human-wheelchair cooperation,” in 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 4278–4283.
[48] P. Trautman, “Assistive Planning in Complex, Dynamic Environments: A Probabilistic Approach,” in 2015 IEEE International Conference on Systems, Man, and Cybernetics, Kowloon Tong, Hong Kong, 2015, pp. 3072–3078.
[49] M. A. Goodrich, E. R. Boer, J. W. Crandall, R. W. Ricks, and M. L. Quigley. Behavioral entropy in human-robot interaction. Technical report, BRIGHAM YOUNG UNIV PROVO UT, 2004.
[50] T. Carlson, Y. Demiris. Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload. IEEE Transactions on Systems, Man, and Cybernetics, PartB (Cybernetics), 42(3):876–888, June 2012. ISSN 1083-4419. doi: 10.1109/TSMCB.2011.2181833.
[51] Paez-Granados, D. F., Kadone, H., & Suzuki, K. (2018). Unpowered Lower-Body Exoskeleton with Torso Lifting Mechanism for Supporting Sit-to-Stand Transitions. In IEEE International Conference on Intelligent Robots and Systems (pp. 2755–2761). Madrid: IEEE Xplorer.
[52] Eguchi, Y., Kadone, H., & Suzuki, K. (2018). Standing Mobility Device with Passive Lower Limb Exoskeleton for Upright Locomotion. IEEE/ASME Transactions on Mechatronics, 23(4), 1608–1618.
[53] Chen, Y., Paez-Granados, D., & Suzuki, K. (2019). Torso Control System with A Sensory Safety Bar for a Standing Mobility Device. In MEXT (Ed.), International Symposium on Micro-Nano Mechatronics and Human Science (MHS-2019) (pp. 2–5). Nagoya, Japan: IEEE.
[54] Van Den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In Robotics research (pp. 3-19). Springer, Berlin, Heidelberg.
[55] Seidel, R. (1991). Small-dimensional linear programming and convex hulls made easy. Discrete & Computational Geometry, 6(3), 423-434.
[56] Van Kreveld, M., Schwarzkopf, O., de Berg, M., & Overmars, M. (2000). Computational geometry algorithms and applications. Springer.
[57] Huber, L., Billard, A., & Slotine, J. J. (2019). Avoidance of convex and concave obstacles with convergence ensured through contraction. IEEE Robotics and Automation Letters, 4(2), 1462-1469.