Deliverable 4.1 – The TIREBOT co-worker

Index

I. Abstract

This document provides a description of the integration of the high-level capabilities in the TIREBOT (a TIRE workshop roBOTic assistant) platform. In particular, the description of how the high-level capabilities developed during Task 3 have been integrated with the robot built during Task 2 is presented. The reader is invited to watch the video, attached to this document, which shows the final full working version of TIREBOT.

II. Introduction

This document presents a description of how the activities of Task 3 have been developed and how they have been integrated with the basic capabilities developed during Task 2 (see Deliverable 2 – Mechatronics of TIREBOT). The integration was done during the activities of the subtask 4.1, whose output is a complete and fully working version of TIREBOT.

The integration required a strong cooperation between UNIMORE and CORGHI in order to exploit synergistically the experience on wheel processing and the knowledge about robot control and interaction. The high-level software architecture implementing the robot behaviour and satisfying the requirements and the specifications defined in Deliverable 1 has been implemented.

Activities of Task 3 (Navigation and Cooperation) aim at implementing TIREBOT's capabilities as the safe navigation of the robot, the safe cooperation with human co-workers, the teleoperation of the robot through the haptic device.

This document is organized as follows: Sec. III reports a description of the high-level capabilities implemented on TIREBOT. Section 0 describes how the integration task, which led to the final full working TIREBOT prototype, happened while Sec. V reports the description of the integration experiment. Finally, in Sec. VI, conclusion are drawn.

III. High-level capabilities

In this section, a full description of TIREBOT's high-level capabilities is proposed. Unlike Task 2 basic capabilities, which aim at implementing low-level abilities for the robot, the implementation of the high-level capabilities aim at providing TIREBOT with the intelligence needed to cooperate with the human co-worker and to accomplish its tasks.

Here is a list of the high-level capabilities implemented on TIREBOT:

- **Navigation**:
  - Safe Operator Following: by using the laser scanner and the RGB-D camera, the robot is capable of recognizing the user and following her/him, while keeping a safe distance.
  - Planner with Obstacle Avoidance: the robot, thanks to a path-planning algorithm, can compute the trajectory that can lead it from a starting point to a goal, by also avoiding obstacles thanks to its laser scanner and by building a geometric map (Figure 1) representing obstacles and free space

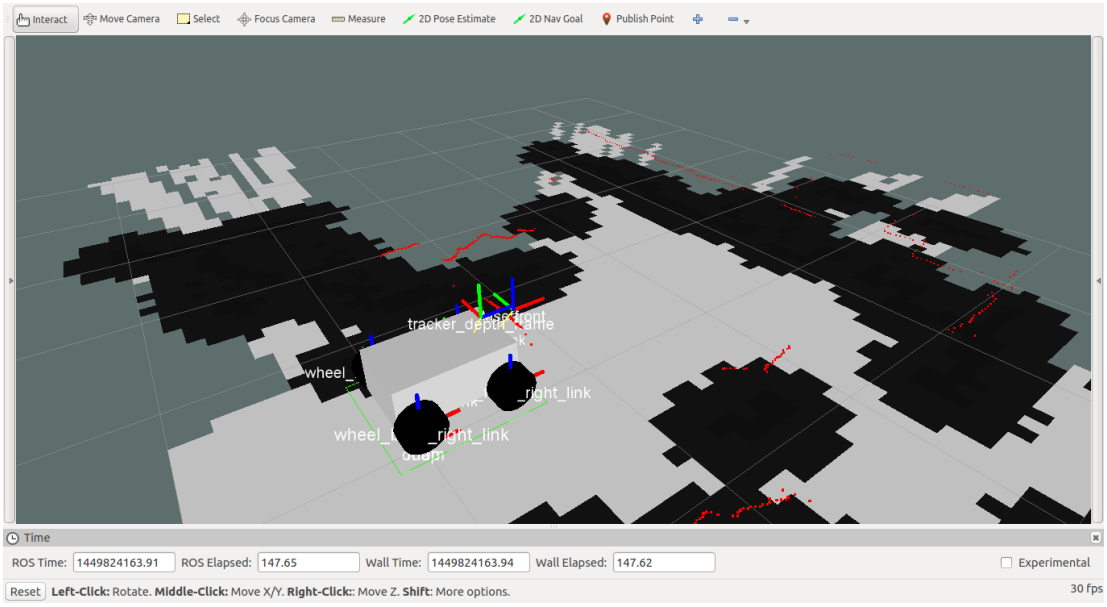

Figure 1: the geometric map. Black areas represent obstacles, grey areas represent free space and red dots represent the laser measurements. Dark grey areas represent the unexplored space.

  - Visual Servoing: when the robot is in proximity of a goal, if it detects a special visual marker, a visual servoing algorithm computes the relative pose of the robot with respect the target and takes the robot in a desired configuration with respect to the marker.
- **Cooperation**:
  - Safe Working Condition Recognition: by using the camera and the laser scanner, the robot can build a local map of the surrounding environment. If there are no obstacle closer than a safety threshold, the robot recognizes the area as a safe working place.
  - Hierarchical Safe Strategy: the robot plans its actions according to a two-layer architecture, where each motion must be evaluated in order to satisfy safety conditions of the environment surrounding the robot.
- **Teleoperation**:
  - Tele/Autonomous Arbitration: When the user interacts with the haptics interface, an arbiter switches the behaviour of the robot from autonomous navigation to teleoperation.

- Teleoperation: a user can manoeuvre the robot through a bilateral teleoperation architecture. Furthermore, the user moves TIREBOT by means of an haptic device and s/he can feel a force feedback proportional to the distance between the robot and the detected obstacles.

In order to better understand how navigation and cooperation work, in the following we propose a description of how the Finite State Machine (FSM) of TIREBOT is implemented and how the user can interact with the robot. Furthermore, the document also reports a full description of the implementation of the safe cooperation and teleoperation.

TIREBOT's Finite State Machine

The behaviour of TIREBOT, while it is operating in autonomous mode, changes according the state the robot is in. The robot can change its state by receiving gesture commands by the user or when certain events happen (i.e.: the user moves the haptics device). The FSM expects TIREBOT working in the following modalities:

- STAND STILL: the robot stands still on its position and rotates while framing the "master-user".
- FIXED GOAL: the robot goes to a fixed goal in the map.
- WANDER: the robot slowly rotates in place while searching for a "master-user".
- MOVING GOAL: the robot follows the "master-user".
- HOMING: the robot returns to its home position.
- WAIT: the robot waits for the wheel to be carried on its forks.
- STOP: the robot stands still on its position waiting for a command from a user.

Figure 2 represents the Finite State Machine for the TIREBOT's Gesture Interaction.

TIREBOT's gesture commands

While in autonomous mode, TIREBOT can change its behaviour by receiving gesture commands from a user. According to the arm pose of the user and the state the robot is in it will change its state.

LEFT/RIGHT	**DOWN**	**UP**	**SIDE**	**FORWARD**
DOWN	UNDEF	FOLLOW ME	GO TO GOAL "A"	STOP
UP	GRAB THE WHEEL	I AM THE MASTER	ACKNOWLEDGE	UNDEF
SIDE	GO TO GOAL "B"	STAND STILL	STOP FOLLOWING ME	UNDEF
FORWARD	RELEASE THE WHEEL	UNDEF	UNDEF	GO HOME

Table 1: TIREBOT's gesture commands

Table 1 summarizes the commands TIREBOT can receive by gestures. Here is a short description of the meaning of each command:

- UNDEF: TIREBOT does not recognize this command and will remain in the same state.
- FOLLOW ME: TIREBOT starts following the user while keeping a safe distance from him.
- GO TO GOAL "A": TIREBOT moves towards the goal "A" a-priori defined by the user (i.e.: near the tire changer). This command needs to be acknowledged before being executed.
- GO TO GOAL "B": TIREBOT moves towards the goal "B" a-priori defined by the user (i.e.: near the wheel balancer). This command needs to be acknowledged before being executed.
- STOP: the robot stops every movement until a user is identified (if the robot has lost its master-user) and orders it to leave this state.
- GRAB THE WHEEL: the robot rotates on itself of a 180° angle and will wait 15 seconds for the user to load the wheel on the forks. Then the robot lifts the lower fork and, if a wheel is detected by the

load cells, the robot actuates the upper fork in order to grab safely the wheel. This command can be executed by the robot only if the robot does not already carry a wheel, otherwise an error message will be printed on the control station of the robot and the robot will not recognize the command. The robot terminates the execution of this command by rotating on itself of a 180° angle.

- RELEASE THE WHEEL: the robot rotates on itself of a 180° angle and will wait 5 seconds for the user to get ready to unload the wheel. Then the robot will release the wheel. This command can be executed by the robot IF AND ONLY IF the robot carries a wheel. The robot terminates the execution of this command by rotating on itself of a 180° angle.
- I AM THE MASTER: this command is needed to be recognized by the robot as the master-user. The robot can recognize this command only if it has not already a master.
- STAND STILL: with this command, the robot enters in the standard state the robot gets in after having identified the master-user and after having terminated the execution of an action. In this state, the robot rotates on itself while trying to frame the master-user with the camera, waiting for further commands. This command is also used to leave the STOP state.
- STOP FOLLOWING ME: this command is only used while the robot is in the “MOVING_GOAL” state. Through this command, the user can make the robot stop following him.
- GO HOME: TIREBOT returns to its home position (i.e.: near its recharging station). This command needs an acknowledge before being executed.
- ACKNOWLEDGE: once the user has sent a command like “GO TO GOAL” or “GO HOME”, the robot needs an acknowledge to start executing it.

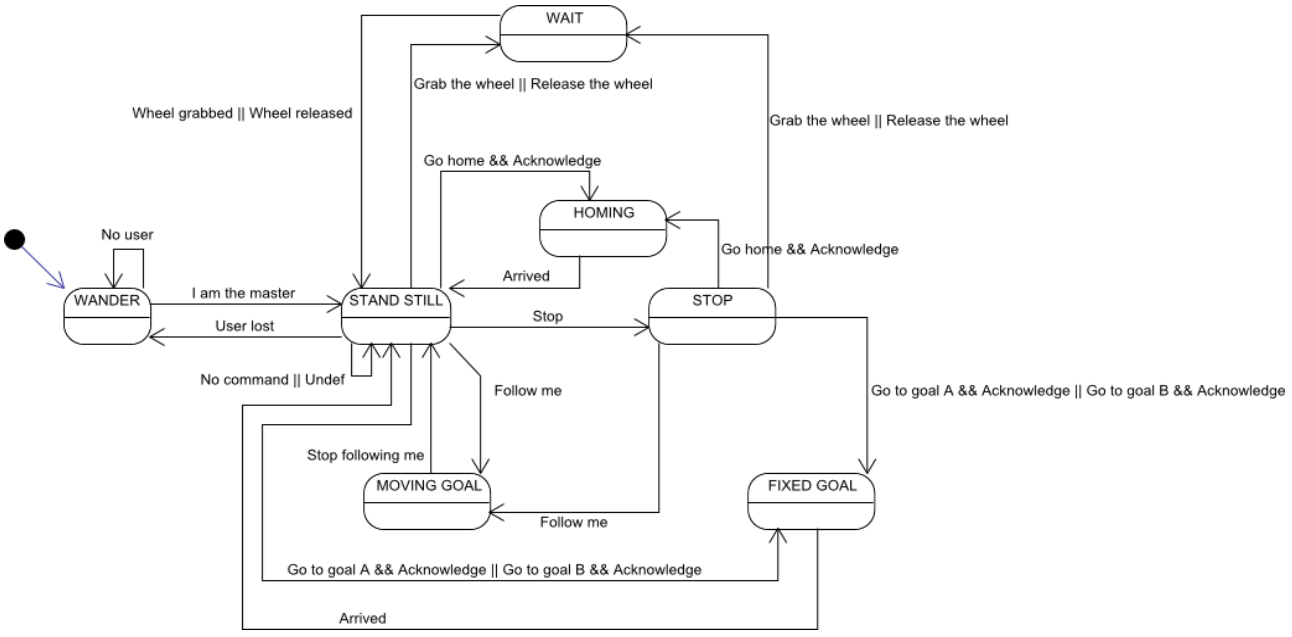

Figure 2: Gesture recognition FSM

NAVIGATION

Free navigation modality is implemented through the "move_base" ROS package[1].

The obstacle avoidance capability is granted by the frontal laser scanner, which can detect obstacles that are mapped on an occupancy grid. The obtained map is then used by the ROS node "move_base"; this node is responsible for the autonomous navigation of the robot. By using the "Dynamic Window Approach" (DWA) provided by the "dwa_planner" plugin, the node computes the optimal path (if it exists) from the actual pose of the robot to the given goal. The DWA also avoids static obstacles through a "global planner", that plans the overall path from the actual pose of the robot to the goal, and moving obstacles through a "local planner" that moves the robot from the planned path in order to avoid sudden obstacles (i.e. a people who approaches the robot while it is moving).

Navigation has a different implementation according to the kind of goal the robot receives from the user. If the user orders the robot to follow him the robot will try to keep its camera framed on the user. The skeleton tracker, that recognizes the user, will send the position of the user's torso to the navigation node. The robot will then compute a point at a certain distance from the user's torso as goal. The robot will keep on tracking the user until he orders it to stop.

When the user orders the robot to go to a fixed goal, the robot starts to search for a visual marker which are positioned in the nearing of the goal. Once the camera has framed the marker, the robot will compute a position according the code of the marker and will go in that position, while constantly correcting its pose in order to be as more precise as possible (visual servoing).

SAFE COOPERATION MODE

The safe cooperation is a particular working modality of the robot; during the free navigation, TIREBOT can move freely from a position to another by avoiding obstacles according to their position. During cooperation with a human the robot should work with a tire workshop operator that, in some phases of his job, must work very close to the robot, for example when the operator, once the tire is removed from the car, has to load the tire on TIREBOT. In this case, the robot should assist the operator without hindering him, moving away if the user has to work in positions occupied by the robot and actuating the wheel grabber without harming the human co-worker.

Standard distance based collision avoidance techniques will always let the robot to move apart when the operator gets closer making therefore cooperation impossible. In order to implement a safe and cooperative behaviour a novel collision avoidance strategy, based on the Danger Field algorithm that depends both on the distance and on the velocity of the detected obstacles is implemented. In this way, if the operator is very close but it moves slowly, TIREBOT will detect cooperation and it will not move. On the other hand, if the distance is small but the operator moves too fast, TIREBOT detects something wrong (e.g. the operator can be escaping from some dangerous situations) and it will move apart to avoid collision with the human.

[1] http://wiki.ros.org/move_base

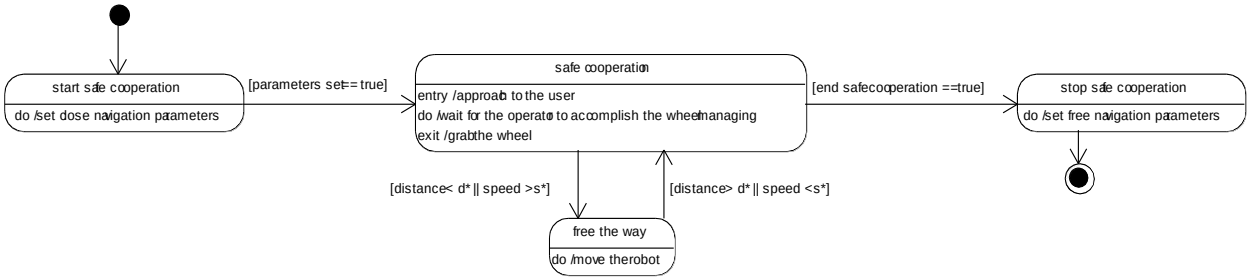

Figure 3: Safe cooperation during wheel grabbing

The “safe cooperation” is a super-state that represents the behaviour of the robot when it works close to a human operator. We can distinguish this state according to the working phase the wheel is in: for example, Figure 3 is the FSM representing the behaviour of TIREBOT when the operator has to load the wheel on it. When TIREBOT has approached the area where the operator is, it changes the parameters for the navigation. These parameters are the safety distance the robot stays away from obstacles and how the position and the speed of obstacles can influence the behaviour of the obstacle avoidance algorithm of the robot. In this phase, the robot also decreases its the speed, in order for the robot to be more accurate in its movements. Once these parameters are set, the safe cooperation starts: the robot approaches to the user that is dismounting the wheel from the car and waits until the operator is ready to load the wheel on the robot. If the operator gets too close to the robot or if a moving obstacle, which can be the operator itself, approaches the robot too fast, the robot moves away. Once the robot has grabbed the wheel, TIREBOT is ready to leave the “safe cooperation” state, it changes back the navigation parameters and enters into the “free navigation” mode.

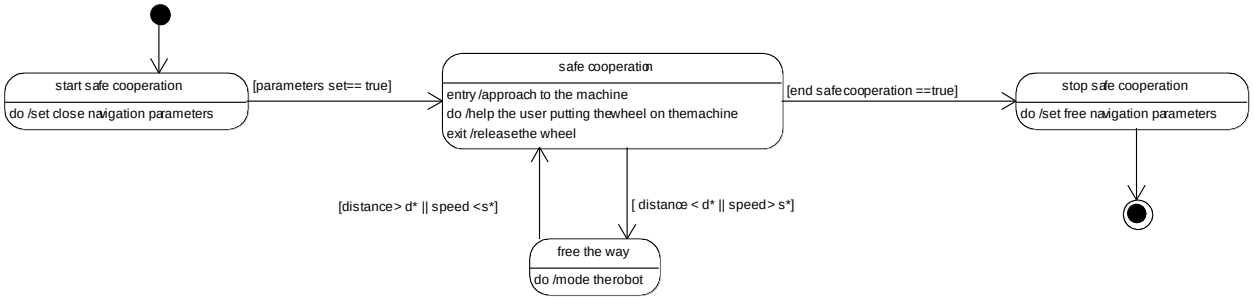

Figure 4: safe cooperation during wheel releasing

Similarly, Figure 4 represents another kind of “safe cooperation”. In this case, the release of the wheel is analysed. Once the robot, which is carrying a wheel, gets in the area near the machine the tire has to be put on, TIREBOT changes the navigation parameters in the same way explained earlier in this section. The robot, then, approaches the operator that unloads the tire from TIREBOT. As in the previous case, the “safe cooperation” expects the robot to move away if an obstacle gets too near or approaches too fast to the robot. Once the tire has been unloaded from the robot, TIREBOT changes back the navigation parameters and leaves the “safe cooperation” state.

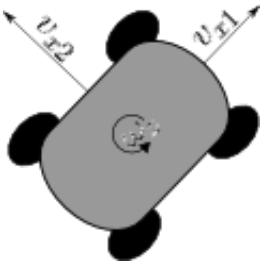

Figure 5: The robot's frames

Let's consider a four wheels omnidirectional robot moving in a workshop-like environment (see Figure 5). The robot is capable of three possible inputs: shifting forward v_{x1}, shifting laterally v_{x2}. The overall movement of the robot is a combination of these three inputs. The state of the robot represents its position and orientation $x = [x_1, x_2, \vartheta]^T$. The robot can switch its navigation modality between a "Free Navigation" mode and a "Safe Cooperation" navigation mode. The "Free Navigation" mode consists in the robot moving through a classical "Artificial Potential Fields" approach. The robot has to travel among known locations and this is done by using a visual SLAM strategy.

Because of the omnidirectional base TIREBOT is mounted on, we model the robot as a kinematic point moving on the plane. The real size of the robot is considered when selecting the safety distances. Let $x \in \mathbb{R}^2$ be the Cartesian position of the robot. We consider the following model for TIREBOT:

$$\dot{x} = u$$

Let $x_o \in \mathbb{R}^2$ be a point representing the position of an obstacle. The speed of the robot and of the obstacle measured in an inertial frame, are respectively $\dot{x}$ and $\dot{x_o}$. The robot is endowed with an on-board sensor (e.g. a laser scanner) capable of measuring the position of the obstacles, and, by derivation over time, their speed. Formally, the robot can measure $(x - x_o)$ and $(\dot{x} - \dot{x_o})$. The control input is given by:

$$u = -\nabla U_{TOT}$$

Where U_{TOT} is a virtual potential field which is defined with:

$$U_{TOT} = U_{ATT} + \eta(t)U_{REP} + [1 - \eta(t)]U_{DF}$$

The term $\eta(t) \in \{0,1\}$ is a discriminant which, if set to 1, nullifies the effect of the "Safe Cooperation" term $[1 - \eta(t)]U_{DF}$, while, on the other hand, if set to 0, it deletes the effect of the "Free Navigation" term $\eta(t)U_{REP}$. The switch between the two working modalities happen as:

$$\eta(t) = \begin{cases} 1 \; if \; \|x - x_H\| \geq \delta \\ 0 \; otherwise \end{cases}$$

Where $\delta$is the distance threshold which delimits the boundary between the "Safe Cooperation" and "Free Navigation".

The other components of U_{TOT} are given by:

- $U_{ATT} = k_{ATT}\|x - x_G\|$: this is the attractive virtual potential field and $k_{ATT} \in \mathbb{R}^+$ is the attractive constant;
- $U_{REP} = \frac{1}{2}k_{REP}\left(\frac{1}{\|x - x_o\|} - \frac{1}{Q^*}\right)^2$ is the classical repulsive virtual potential field. k_{REP} is the repulsive constant and Q^* is the distance beyond which the gradient of the potential field is zero;
- $U_{DF} = U_{SDF} + U_{KDF}$: the Danger Field. It is composed by two components: U_{SDF} is the Static Danger Field produced by static obstacles, while U_{KDF} is the Kinetostatic Danger Field, which is generated by moving obstacles:

$$U_{SDF} = \frac{k_{SDF}}{\|x - x_o\|}$$

$$U_{KDF} = \frac{k_{KDF}\|\dot{x} - \dot{x_O}\|(1 + cos\varphi)}{\|x - x_o\|^2}$$

In these two equations k_{SDF}, k_{KDF} are positive real parameters which can be set arbitrarily to tune the effect of the Danger Field. The term $cos\varphi$ contains information about the direction the danger source is moving along and it is computed as:

$$cos\varphi = \frac{(x - x_{o,i})(\dot{x} - \dot{x}_{o,i})}{\|x - x_{o,i}\| \, \|\dot{x} - \dot{x}_{o,i}\|}$$

The equation for U_{SDF} is very similar to the one of the classical repulsive artificial field, but in this context is preferably to consider two different contributions to the overall potential field. U_{REP} is exploited for implementing the obstacle avoidance during "Free Navigation", where the speed of the robot is high and, consequently, the distance to be kept from the obstacles is high. During the "Safe Cooperation" modality, the robot has to work close to the operator and to other objects (e.g. the car) and, therefore, the safety distance has to be kept small. Thus, the presence of U_{SDF} and of U_{REP} in the definition of U_{TOT} is due to the necessity for changing the safety distance while changing operational modality.

The term U_{KDF} generates a potential field proportional to the relative speed of the robot and of the obstacle. The term $(1 + \cos\varphi)$ has a crucial importance: it is used for determining the direction the robot should move in to avoid the incoming moving obstacle. Furthermore, as stated in Sec. III-B, this term works also as a discriminant for activating or not the danger field. For example, if the relative speed of the robot and of the obstacle indicates that the obstacle is moving away or its trajectory is not intersecting the robot's one, then the danger field is not activated. On the other hand, if the trajectory of the obstacle is leading it to collide the robot, then the direction of the generated command leads the robot on a safer trajectory.

Teleoperation

Consider a four wheels omnidirectional robot, capable of three possible inputs: shifting forward v_x, shifting laterally v_y and rotating ω_z. The total movement of the robot is a combination of these three inputs. Consider now the haptic device Geomagic Touch the teleoperation; the device has six degrees of freedom but only three of them are actuated and can return a force feedback. The device has also a switch inside the inkwell that holds the pen, used to sense if a user is holding the pen or if the device is at rest. The pen has two buttons: if the user does not press any button he will be able to move the robot along the x and y axis. If the user presses the first button (B1) he will rotate the robot along its z axis. If the user presses the second button (B2), then the wheel grabber will be actuated.

The robot is teleoperated in rate mode, which means that the position registered on the haptic device is transformed into a speed which is then commanded to the robot, according to the following equation, where k_v is a proportional constant:

$$v_x = k_v \cdot p_x \Leftrightarrow (B_1 = 0)\&(B_2 = 0)$$
$$v_y = k_v \cdot p_y \Leftrightarrow (B_1 = 0)\&(B_2 = 0)$$
$$\omega_x = k_v \cdot p_y \Leftrightarrow (B_1 = 1)\&(B_2 = 0)$$

Here, p_x and p_y are the distances of the pen from a zero position that is in front of the device. The user takes control of the robot's movements when the pen of the haptic device leaves its inkwell and the teleoperation commands has greater priority with respect the commands sent from the other ROS nodes (i.e. the autonomous navigation). For safety reason, the wheel grabber can be actuated only if the robot is not moving. If the user presses B2 and lifts upward the pen, then the robot will try to grab the wheel. Otherwise, if the user presses B2 and lowers the pen, then the robot will release the wheel (if it is carrying one). The user can check what is framed by the camera mounted on the robot thanks the screen of the control station. Furthermore, the graphical user's interface is provided with a START and a STOP buttons that can be used in case of emergency. The STOP button prevent any movement of the robot. Once the user has stopped the robot, is possible to resume robot's operation by pressing the START button.

The force feedback is computed thanks to the readings of the SICK S300 frontal laser scanner. This sensor returns the position of the detected objects in a vector of N points. Each detected point is represented in polar coordinates(d_i, α_i). If the distance of a point is lower than a predefined range Q^*, then the point generates a virtual repulsive field which contributes to generate the overall force which is communicated to the userthrough the haptic device. Generally, the *i*-th point generates the potential field:

$$U_{rep} = \begin{cases} \frac{1}{2} k_r \left(\frac{1}{d_i} - \frac{1}{Q^*} \right)^2 \; if \; d_i \leq Q^* \\ 0 \; if \; d_i > Q^* \end{cases}$$

In the previous equation k_r is a proportional constant which is used to calibrate the intensity of the repulsive field. The overall potential field is given by the sum of the contribution of each detected point. The force the user will sense at the haptic device can be easily obtained by computing the gradient of the repulsive field $F = -\nabla U_{rep}$.

High level capabilities tests

These high-level capabilities were first tested on a Pioneer P3DX robot equipped with a sensor equipment similar to the one of TIREBOT. The choice of testing Task 3 capabilities on a different robot than TIREBOT, is due to the fact that Task 2, which expect the physical realization of the robot, and Task 3 activities were developed in parallel: this means that the development of high-level capabilities could not have place on the TIREBOT prototype, that was under construction.

Experiments can be seen on the attached video (ECHORD++_exp2.mp4).

The video shows first the robot following the user (which sends commands to the robot through gestures) and navigating autonomously in the environment while also avoiding obstacles. Then the second part of the video shows safe cooperation: if the user moves slowly, he can move close to the robot. On the other hand, if the user gets too close to the robot or moves too fast, the robot detects the fast approaching object as a danger and moves away. The last part of the video shows the teleoperation of the robot through the haptics interface.

IV. Integration

The choice of using ROS for developing the TIREBOT application facilitated a lot the integration of all the pieces of developed software. In fact, the ROS architecture makes easy the reuse of code and the communication between processes (which in ROS are called “nodes”). Any node can publish an arbitrary number of messages, called “topics”, of different types diversified by the topic name, and, similarly subscribe to topics produced by other nodes. In order for a node to subscribe a topic, this one must be of the same type and have the same name. Thus, it is very simple to substitute nodes and the debug of the software is easy.

Here’s a brief list of the main nodes running on TIREBOT:

- neo_relay_board: this node represents the drivers for the Neobotix MPO-500 mobile robot.
- sick_s300_laser_scanner: this node represents the driver for the SICK S300 laser scanner.
- map_server: the map server that handles the map explored by the robot.
- move_base: this node handles the autonomous navigation of the robot. It subscribes to information regarding the map published by the map server and to laser readings. It generates a path the robot should follow and generates the speed commands that the robot has to actuate to follow the desired trajectory.
- video_view: this node, which is running on the control station, shows the video captured by the camera the robot is endued with.
- skeleton_tracker: this node is the driver for the ASUS Xtion Pro Live and publishes information about the skeleton of the framed users. In particular, this node can recognize the limbs of a human user and publishes their positions as frames.
- marker_identifier: this node recognizes the visual markers placed in important places in order to help the robot in its navigation. Once a marker is framed by the camera and recognized, this node sends a new goal to be tracked to move_base.
- tim_drivers: this node implements the drivers for the localization laser scanner SICK TIM310.
- localizer: this node implements a localization algorithm that computes the position of the robot according to laser scanner’s measurements of reflective markers.
- kalman_filter: this node merges odometric information given by the robot with the position computed by the localizer node. This was done in order to smooth the position information given by the localizer and to make possible navigation if reflective markers readings are temporary not available.
- geomagic_driver: this node implements the drivers for the haptics device. It provides data on the position of the haptics device’s end effector, its buttons, the speed of its joints and the their angular position. It is also responsible of the actuation of the haptics device’s motors that implement the force feedback.
- haptic_teleoperation: this node translates the haptics device commands into speed commands. Furthermore, if the user presses the haptics device’ pen buttons, the node sends commands to actuate the wheel gripper.
- force_feedback: this node, by subscribing to laser measurements, generates the force feedback commands.
- gripper: this node implements the drivers of the wheel gripper of TIREBOT.
- danger_field: this node implements the Danger Field algorithm that makes possible the human/robot safe cooperation.
- arbiter: this important node is responsible for the arbitration on which node is actually sending speed commands to the robot. A brief description of how this node works is reported in the following of this section.

- sound_play: this node makes the robot capable of speaking. This was done in order to give a feedback of the action the robot is doing to the user.
- <<generic_tf_responsible_nodes>>: these nodes are responsible of publishing data on how the frames of the robot are placed with respect a fixed frame represented by the map.

The nodes network (commonly called Node Graph) is shown in Figure 6.

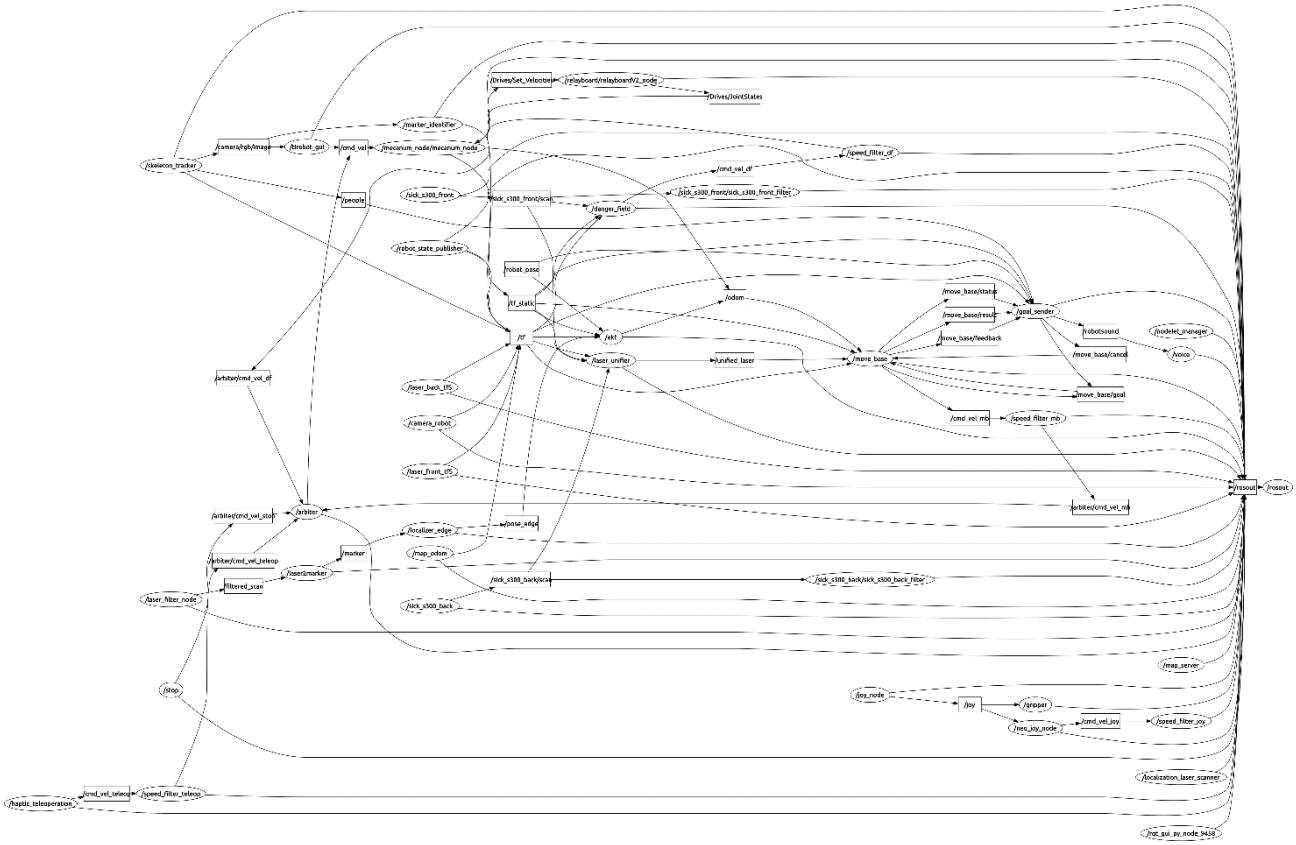

Figure 6: The ROS's node graph

Integration was done while paying great attention to safety. A ROS arbiter node checks the identity of the speed publisher node. Every command has a priority according to the identity of the publisher. For example, the speed published by the teleoperation node has a much higher priority than the speed published by the autonomous navigation. This happens because the teleoperation is actuated directly by the user that must intervene in case of danger.

This arbiter node filters also speed commands sent by the ROS nodes for navigation, teleoperation, etc. The arbiter decides which speed command will be forwarded to the robot according to priority: for example, the Danger Field node, that gets activated by the distance of the robot from the goal, has higher priority with respect to the autonomous navigation node; consider, for instance, the robot following a person. If the user suddenly stops and draws back fast, the robot must stop and draw back too in order to free the way to the user.

Of course, the highest priority is given to the teleoperation node: if the user sees a dangerous situation for people or for the robot, s/he must intervene through teleoperation to prevent damages. The arbiter will then ensure that the speed command generated through teleoperation has higher priority than any other speed command received, and then will move the robot according to the user's commands.

The grabbing of the wheel procedure and the actuation of the forks have a dedicated arbiter that must acknowledge the command before allowing the movement. For example, if a node asks the "wheel grabber" node to grab a wheel, it first lifts the lower forks of few centimetres, letting the load cells to "sense" the presence of the wheel. The upper fork will be actuated only if the wheel is actually loaded on the lower forks. Furthermore, the wheel gripper can be moved only if the robot is stopped.

V. Experiments

The integration experiment took place in a workshop-like environment, in order to make as similar to a real application as possible. In particular, the experiment took place in the CORGHI's Car tyres mechanic School (see Figure 7).

Figure 7: TIREBOT during the Integration Experiment

A video showing all the capabilities of the robot is attached to this document. Table 2 reports a list of all the tasks is proposed with a brief description of the capabilities and where, in the video, each capability is shown.

		Activity	Description	Position
Task 2 – TIREBOT Design and Basic Capabilities	**T2.1 - Construction**	A2.1.1 - Lifting platform and installation	Mechanical design and realization of the lifting device.	The robot can be seen in every part of the video. In particular, the lifting platform in action from 1:09 to 1:44.
		A2.1.2 - Camera installation and calibration	Installation and calibration of a camera on TIREBOT.	This capability can be seen in every part of the video
		A2.1.3 - Control unit installation	Installation of a computer for controlling the robot.	The control unit can be seen in every part of the video
	T2.2 - Interaction	A2.2.1 – Visual recognition	Develop an algorithm for recognizing wheels, tire changer, wheels balancer and operators.	From 0:19 to 1:47
		A2.2.2 – Geometric map	Build a geometric map by using the output of the laser scanner.	From 0:48 to 1:08
		A2.2.3 – User interface	Design a simple user interface that allows to user to specify simple commands. The user can interact with the robot with gestures.	From 0:19 to 1:44
	T2.3 - Control station	A2.3.1 – Setup the control station	Design a control station equipped with a haptic device for implementing teleoperation.	From 0:06 to 0:18
		A2.3.2 – Visual stream from TIREBOT	Make the robot capable of streaming	From 0:06 to 0:18 and from 0:48 to 1:08
		A2.3.3 – User interface	Design a user interface on the control station.	From 0:48 to 0:18
Task 3 – Navigation and Cooperation	**T3.1 – Navigation**	A3.1.1 – Safe operator following	Use laser scanner and camera for recognizing the operator and for following him while keeping a safe distance.	From 0:19 to 0:47
		A3.1.2 – Planner with obstacle avoidance	Build a planner and a trajectory tracking control for taking the robot from its current position to a desired position. Use laser scanner for detecting obstacle and avoid them while keeping a safety distance.	From 0:19 to 1:08
		A3.1.3 – Visual servoing	Design a visual servoing algorithm for precision positioning of the robot according markers in the environment.	From 0:19 to 0:47 and from 0:48 to 1:08
	T3.2 – Cooperation	A3.2.1 – Safe working condition recognition	By using laser scanner, build a local map of the environment surrounding the robot.	From 1:45 to 2:34
		A3.2.2 – Hierarchical safe strategy	Build a two control architecture	From 1:45 to 2:34
	T3.3 - Teleoperation	A3.3.1 – Tele/autonomous arbitration	Develop an arbitration for enabling teleoperation and disabling the autonomous control and vice versa. When the user moves the phantom, the autonomous behaviour is disabled. When the phantom is set into its rest position, the teleoperation is disabled and the autonomous behaviour is switched on.	From 0:06 to 0:18
		A3.3.2 - Teleoperation	Implement a bilateral teleoperation architecture. Implementation was done in "*rate mode*" and force feedback is associated to the distance between the robot and the surrounding obstacles.	From 0:06 to 0:18

Table 2: Task and capabilities list

The integration experiment video will be also uploaded on YouTube. The video starts with the experiment on teleoperation (see Sec. III). Then the video shows the robot recognizing the user, who lifts his arms above his head in order to be recognized, and receiving the order to follow him. The video then shows the user's interface (on the left) and the geometric map building (on the right). Furthermore, the video shows also the path (the red line) built by the planner and how the map evolves, while the robot explores its surrounding.

In the next scene the robot receives from the user the command to grab a wheel. The robot rotates of 180° and waits for the operator to put the wheel on its forks. Then the robot moves the lower forks, first, to lift the wheel and then grabs it firmly with the upper fork.

The last part of the video shows the safe human/robot cooperation. If the user approaches slowly the robot, it stands still in its position. On the other hand, when the user approaches fast the robot or moves too much near to it, the robot senses the operator ad a danger and moves away.

VI. Conclusions

This document reported the integration work of Task 2 and Task 3 activities. In particular, this document described Task 3 activities, which first were tested on a mobile robot different then TIREBOT, and how they were integrated with those developed during Task 2 (see Deliverable 2 – Mechatronics of TIREBOT). A video that shows these high-level capabilities is also provided with this document.

This document also reported a description of how the user can interact with the robot: through gestures or by teleoperation. A subsection of the manuscript is dedicated to the description of how safe human/robot cooperation is achieved.

The implementation of safety procedures is described, with a particular attention to the description of the arbitration of which running node is actually commanding the robot.

The integration work was then tested experimentally that shows the robot working in different modalities in a workshop-like environment while assisting a tire repairer worker.