Programming assignment for HRI - People Tracking & Following.

In this programming exercise you will use ROS-modules for people tracking to implement a person following module, that should make one TurtleBot (or a simulation of it) keep a certain pose relative to one person (or a simulated target, a movable “blob” is fine).

For this exercise, it is recommended to work in pairs (or small groups of three), both due to the complexity of the task but also in order to have one/two operator(s) who can observe and identify issues with your solution and one user, i.e., the person being tracked and followed. Also, you should make sure to have a rather clutter free environment and not too many people around the robot, as you will make use of general off-the-shelf components with probably limited capabilities and robustness. Pair / group up so that there is competence in the group for handling the TurtleBot or Gazebo / other simulation tools and ROS in general.

Background:

Many applications of mobile robots require that persons in the vicinity of the robot are tracked, to simply know where they are for person suitable avoidance or to actually identify user(s) as interaction partners. Such an interaction can also include some form of coordinated movement (the robot guiding a person, the robot following the person, or even a side-by-side movement), which can be seen as a basic capability of a mobile robot platform.
A basic “robot follows person at predefined distance” behaviour is the one of these different movement related interactions that requires least in terms of planning or prediction capabilities, but it still offers some interesting aspects to discover, as more sophisticated following behaviours can emerge from a simple one—and become quite complex in terms of other capabilities like planning being needed as well.
Thus, it is assumed here, that implementing a basic following behaviour will give you some understanding for the potential complexity of a rather simple HRI-related task.

Task:

- Make the robot (Turtle) follow one person (try to keep track of this particular person) at some distance (i.e., implement a following behaviour in a ROS-node).

- Close in until roughly 50cm distance, when the person stops.

- Keep adjusting the robot’s heading to always “face” the person, but do not move forward if the person only makes small movements.

- Start moving again if the person starts moving away and reaches a distance of more than one meter from the robot.

- In case the person is lost from the field of view, the robot should slow down and notify about this through the ROS-package <sound-play>. Resume tracking and following, when possible. Indicate this through a respective verbal comment. The robot should only indicate having lost / found the user once per such a situation.

You will probably encounter problems, e.g., with the very simple dialogue that would not allow for explanations or corrections, with situations in which several people are around the robot and confuse the tracking module, or with door passages (specifically if you have the robot adjust velocity as directly related to the angle and distance towards the user). This can be circumvented for a basic solution by making sure the robot has passed the door (or other narrow passage) before turning and moving perpendicular to the direction of the door passage.

Bonus:

If you feel you have the time and interest, you can add mechanisms

- to resolve problematic situations (user lost from view, etc) in a smarter way,
- to resolve situations in which the robot gets confused by several people being around and has to distinguish a user from bystanders,
- to handle doorway passages or obstacles in a smarter way than by having the user move in a respective way (requires some navigation package / obstacle avoidance in combination with the following behaviour).

Tips and assumptions:

- You can do the following by continuously specifying and updating a virtual goal point at a suitable distance from the person (if you have a respective navigation tool running), or by directly setting the velocity depending on the distance to the person in the respective direction.
- For obtaining a person's position you should use the ROS-package to be found at https://github.com/wg-perception/people.git (documented at http://wiki.ros.org/leg_detector). In case you get multiple "people", it is allowed to engineer the environment (use a hallway, build "walls" around the area your robot moves in and avoid crowds) or use a simple heuristic (e.g., the "person" closest and most clearly "in front" is the "user") to determine which person to follow.
- In case of resource constraints (robots), you can use a simulation, e.g., in Gazebo, where you should create a person simulation (check for opportunities to use results from other, specifically the visualisation task(s)) that you can control. Here, it is allowed to assume "perfect" tracking data, i.e., the simulated person (target) is allowed to produce <people> messages (perfect position data wrapped into position relative robot) instead of using a simulated laser range finder and the person tracking packages. You can even build upon the idea of the original turtle simulation (ROS tutorial) for a start, e.g., by making one turtle follow another which you control (and tweak / wrap its data to send out <people> messages instead of "classic" position data).
- If you use a simulation, make still sure that the "person" (target) is controlled by a real human, who reacts spontaneously to any verbal / textual output of your following module.

Evaluation:

The task is considered solved when you can make the robot / simulated robot follow one (simulated) person — not (controlled by anyone) from your working group — inside a room (changing directions), through at least one (simulated) doorway (it is allowed to use the basic user movement strategy based coping method not to get stuck in the door passage), and along a "corridor" (along a more or less straight trajectory covering at least 10m of distance). At least one situation of losing the user out of sight should be handled as specified, but you may assume the user as being cooperative (not trying to trick the system).