5. Visualization

In this project the workcell and detected objects need to be displayed in a software environment. This is realized by creating a ROS package named “cobot_visualisation” which contains two nodes which are named “distance_node” and “visualisation_node”. The distance_node monitors the distance between objects and the robot and sends a stop and start command to the path planner accordingly. The visualisation_node displays the workcell, detected objects and distance between objects and the robot in RViz. A ROS package called “moveit_visual_tools” is used to make it easier to display objects. This package is only supported in the language C++. Because of this reason the cobot_visualisation package is programmed in C++;

5.1 Node communication

The communication between nodes has been drawn in Figure 11 below. Dashed (- - -) nodes and lines are not part of the cobot visualisation package. The description of the topics can be seen in Table 2 .

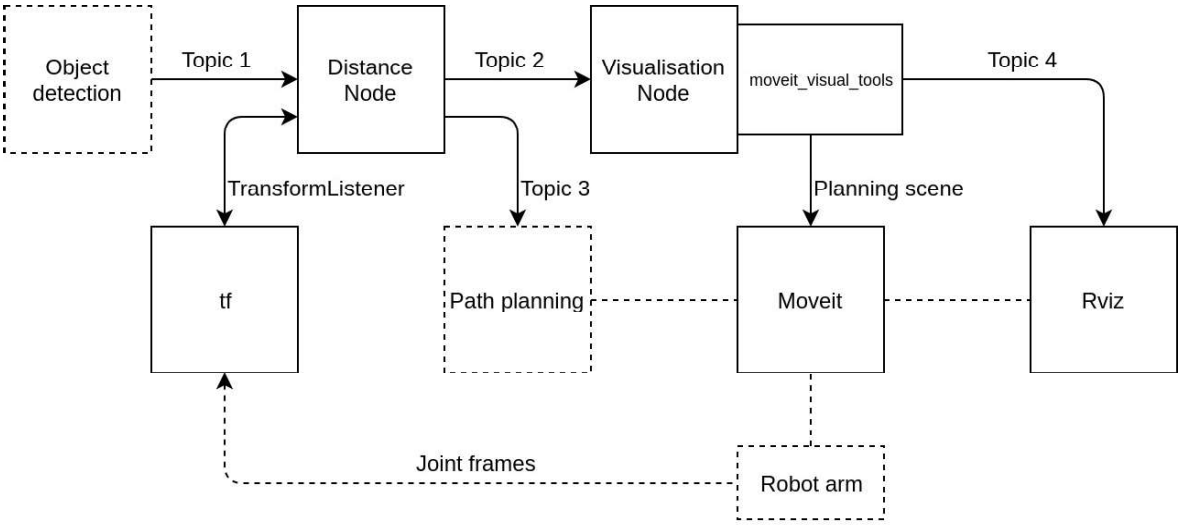

Figure 11- Communication diagram from cobot_visualisation package

Topic	Topic name	Message type
1	/DetectedObjects	visualization_msgs/MarkerArray
2	/cobot_visualisation/visualisation_data	cobot_visualisation/VisualisationData
3	/ur5_control	std_msgs/String
4	/moveit_visual_tools	visualization_msgs/MarkerArray

Table 2 : Topic description

The distance node will subscribe to **Topic 1** where it will receive objects that are detected by the detection system. The distance node will calculate the smallest distance between the objects and the robot arm. The objects and data that need to be visualized are passed to the visualisation node via **Topic 2**. If an object is too close to the robot, a stop signal is sent to the path planning node via **Topic 3**. The visualisation node will add the object to the planning scene of the robot and visualize data.

5.2 Distance node

The distance node calculates the distance between the robot arm and objects received via the topic /DetectedObjects. This topic has the message type: visualization_msgs/MarkerArray. Every object is represented as a Marker which contains the position, scale and color of the object.
Every time when data is received on the topic /DetectedObjects, the distance will be calculated and decided if it's ok for the robot to move or not. This information will immediately be passed through to the visualisation node and the path planner.

Get the robot's location(s)

Three frames of the robot called: tool0, forearm_link and world (see Figure 12) are used to calculate the distance between every object. To get the location of a robot frame in the world frame the ROS package tf is used. This package can do transformations between frames. The distance node does not need to do the complex calculations in this way.

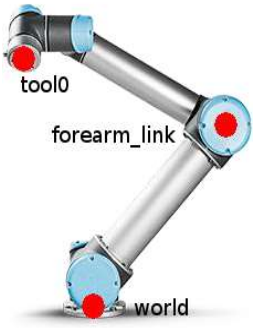

*Figure 12:- Robot frames**Ongeldige***

Move the object's origin towards the robot

The position of an object is defined as the center of the object. The scale defines how far it extends from the center. If the distance is calculated using the center, the resulting distance will be larger than the actual distance between the object and robot because the scale isn't taken into account. Because of this it's necessary to calculate the distance between the body of the object and the robot. This is done by moving the origin of the object as close as possible to the robot without leaving the body of the robot (Figure 13). The scale(size) of the object limits how far the origin can move.

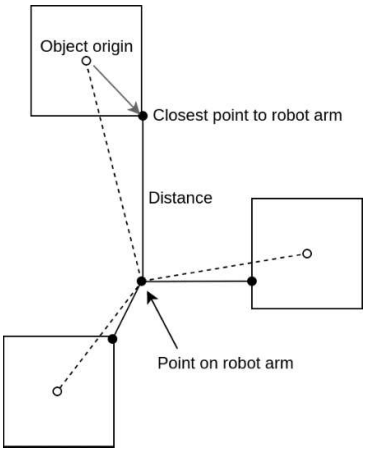

Figure 13 - Moving the object's oriain towards the robot

Distance calculation

A routine is setup to calculate the closest distance between any robot frame and all objects.

- Initialize:
  - Set closest distance to "infinity"
  - Reset closest robot position
  - Reset closest object position
- Do this for every robot frame:
  - Calculate frame position
  - Do this for every object:
    - Move object's origin towards robot frame
    - Calculate distance between object and robot frame
    - Do this if the calculated distance is less than the closest distance:
      - Replace closest distance with the calculated distance
      - Save position of the robot frame
      - Save position of the object

Stopping the robot
The closest distance is used to decide if it's ok for the robot to move or not. The threshold is currently set at 1.26 meters. If the distance is less than the threshold a stop signal will be published in the topic /ur5_control. The path planner subscribes to this topic. If the distance is greater than the threshold for two seconds an empty message will be send in the topic /ur5_control to make the robot move again. The internal status has tree values named: start, stop and cooldown.

Publishing data to be visualized
The Distance node passes all important data to the Visualisation node via the topic: /cobot_visualisation/visualisation_data which handles the custom message type: /cobot_visualisation/VisualisationData. This message contains the following data which can be seen in Table .

Variable name	Message type	Description
objects	visualization_msgs/MarkerArray	Objects received from detection system
closestDistance	float32	Closest distance between object and robot
closestFrame	geometry_msgs/Point	Closest point on the robot
closestObject	geometry_msgs/Point	Closest point on the object
statusMessage	string	Status: start, stop, cooldown
statusDescription	string	Not implemented yet

Table 3 - Data in the message type /cobot_visualisation/VisualisationData

The closestFrame and closestObject points are used by the Visualisation node to draw a line between the closest object and the robot.

5.3 Visualisation node

The visualisation node uses the package moveit_visual_tools which makes it easier to publish (collision) objects to MoveIt. The path planner from MoveIt will use the published collision objects and will try to plan around them. The visualisation node has two main tasks:
- Adding the workcell and objects to the planning scene
- Visualizing the distance between objects and the robot

Receiving data
The visualisation node receives data from the distance node via the topic: /cobot_visualisation/visualisation_data which contains data about the object and the object closest to the robot. This topic uses a custom message type to be able to receive and visualize all information at the same time. Otherwise multiple topics would be needed and updates can be out of sync.
When data is received, all (collision) objects will be removed. The workcell and objects will be re-added afterwards and the planning scene is updated.

Publishing the workcell

To visualize the workcell a STL model file has to be created. A temporary workcell has been designed in SolidWorks (Figure 14) to be able to test the visualization and the path planning of the robot.

In the visualisation node a function has been made to display the workcell. When the node starts, the STL file of the workcell is loaded. When the function is called the desired position and rotation is set. The workcell is added to the planning scene and displayed by RViz afterwards.

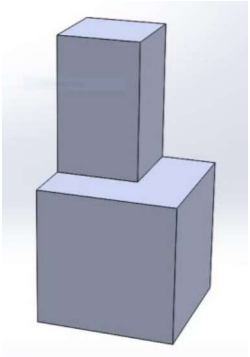

Figure 14 : Workcell in Solidworks

Adding the objects

The received objects have a position, scale(size) and color. moveit_visual_tools doesn't have an easy function to publish a box where x,y and z scale can be applied. The only function available is a cuboid function which need the two opposite corners of the box. The position of these corners can easily be calculated using the position and scale. Here is an example calculating the position of two corners on the x-axis:

corner1.x=position.x-scale.x/2

corner2.x=position.x+scale.x/2

The visualized objects in RViz can be seen in Figure 15.

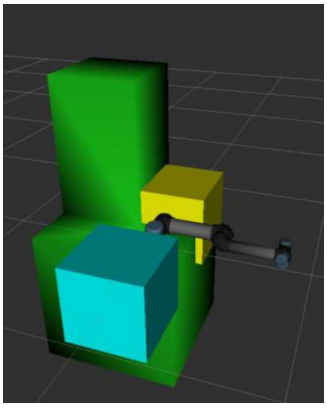

Figure 15 : Workcell and objects in RViz

Visualizing the object distance

The received data is used to visualize the distance between the closest object and the robot. The data contains three variables which are used:

- closestFrame
- closestObject
- closestDistance
- statusMessage

The points closestFrame and closestObject which contain x,y and z coordinates are used to draw a line between the closest object and the robot (Figure 16). This line is constructed out of a cylinder and two spheres which are placed on the ends. The color of the line depends on the value of statusMessage. The color definitions can be seen in 4.

statusMessage	Color
"start"	Green
"cooldown"	Yellow
"stop"	Red
Default	White

Table 4 - statusMessage color definitions

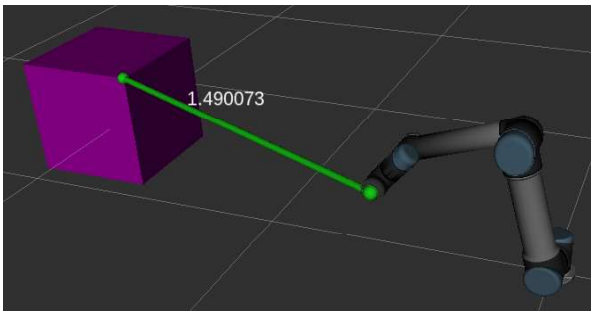

Figure 16 : Distance between an object and the robot