Task 0 – Problem Statement

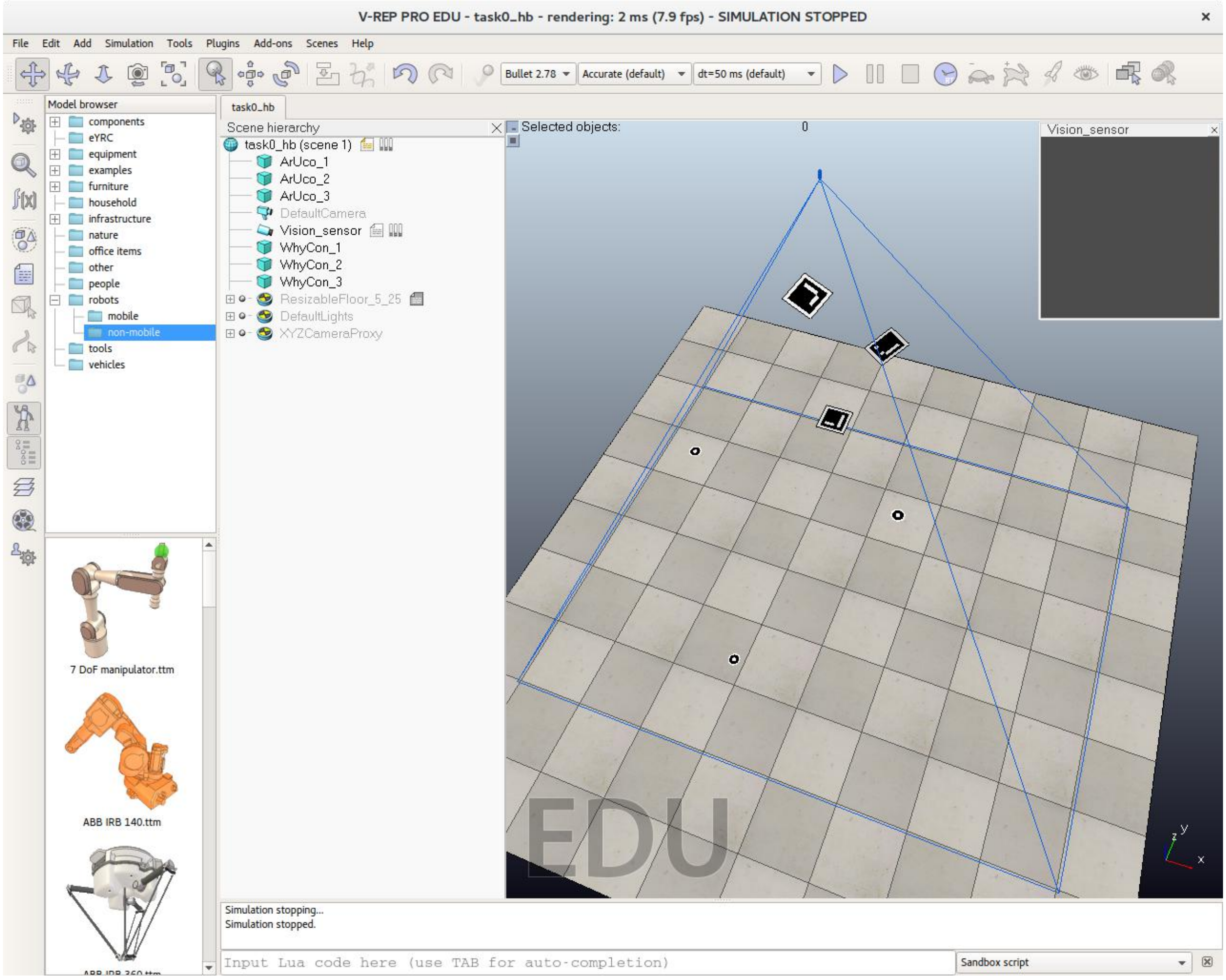
Detecting WhyCon and ArUco markers

From the tutorials, you have learned

- Basics of V-REP : scene, child-script, objects
- Basics of ROS : creating workspace, creating package, launch file, writing a subscriber
- WhyCon and ArUco marker : topics which the package subscribes and publishes

Scene Description:

Load the given scene *task0_hb.ttt* in V-REP simulator. The scene looks as shown in Figure 1:



V-REP PRO EDU - task0_hb - rendering: 2 ms (7.9 fps) - SIMULATION STOPPED

Bullet 2.78 | Accurate (default) | dt=50 ms (default)

Simulation stopping...
Simulation stopped.

Input Lua code here (use TAB for auto-completion) | Sandbox script

Figure 1 : task0_hb.ttt

Following are the various objects in the scene:

ArUco : There are three ArUco markers with names ArUco_1, ArUco_2 and ArUco_3.
WhyCon : There are three WhyCon marker, namely WhyCon_1, WhyCon_2 and WhyCon_3.
Vision_sensor : This gives an image within the blue region with a resolution 640 x 480. The image of the vision sensor is visible in the floating view window named Vision_sensor present in the side of the scene on running the simulation.

Problem Statement:

Fetch WhyCon and ArUco marker coordinates of the static markers placed in the V-REP scene by subscribing to the markers' corresponding topics and print the coordinates on the terminal.

Procedure:

1. Change the directory to your catkin workspace and clone the github repository in "src" folder of workspace by typing the following command:
   ```
   >> git clone https://github.com/fayyazpocker/hungry_bird.git
   ```
2. Take a look at the "*marker_detect.launch*" present inside the launch folder.
3. In it you will find that the changes to the *.launch* files mentioned in *Introduction_to_ArUco_markers.pdf* and *Introduction_to_WhyCon_markers.pdf* are already made.
4. The lines of remap are commented, please uncomment and complete them.
5. Load ***task0_hb.ttt*** in V-REP simulator after launching `roscore` in a terminal.
6. Run the simulator and type `rostopic list` to find the topics being published by the vision sensor.
7. Use `rostopic type /topic_name` to see what is the type of topic being published by vision sensor.
8. The following topics gets published on running the simulation:
   - **a) /visionSensor/image_rect**
     - i. This message is of type *sensor_msgs/Image*
     - ii. This corresponds to the image published by the vision sensor in the vrep scene
     - iii. Run the following command on a terminal to see the image published by the vision sensor :
       ```
       >> rosrun image_view image_view image:=/visionSensor/image_rect
       ```
   - **b) /visionSensor/camera_info**
     - i. This message is of type *sensor_msgs/CameraInfo*
     - ii. This corresponds to the camera information being published by the vision sensor present in the V-REP scene.
9. Remap the topics subscribed by nodes: *aruco_marker_publisher* and *whycon* to the topics being published by vision sensor in the *marker_detect.launch.*
10. Run the simulator and then launch the *marker_detect.launch* file. This launch file must ideally run required nodes in whycon and aruco_ros package to detect markers in the image published by vision sensor in the V-REP and should output two images of detected ArUco markers and WhyCon markers as shown in Figure 2.

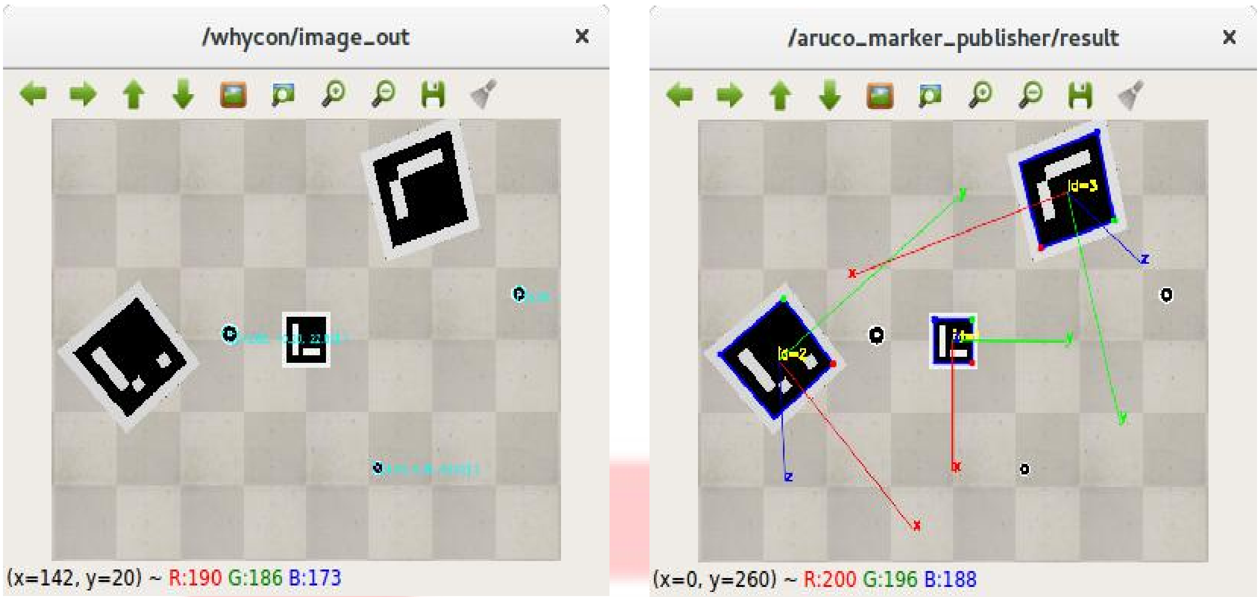
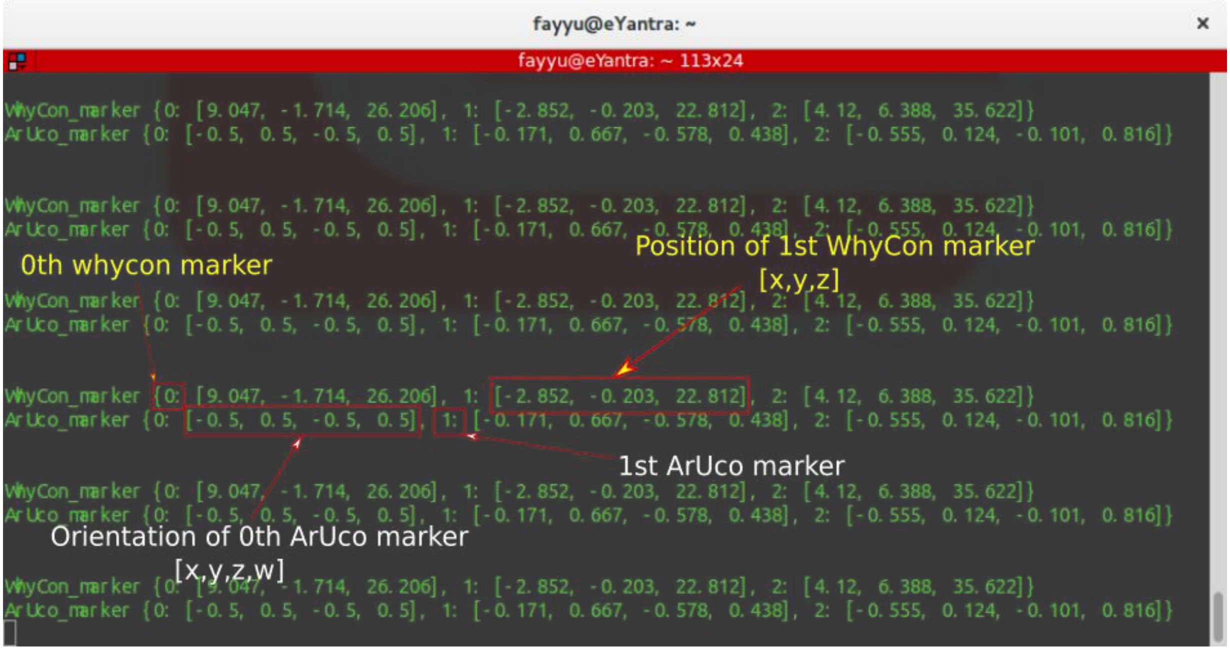

/whycon/image_out
(x=142, y=20) ~ R:190 G:186 B:173
/aruco_marker_publisher/result
id=3
id=2
(x=0, y=260) ~ R:200 G:196 B:188

Figure 2 : WhyCon and ArUco output image

11. Open the python script named “***get_marker_data.py***” in the “scripts” folder present in the folder, “hungry_bird”. This rosnode must subscribe to the ***/whycon/poses*** and ***/aruco_marker_publisher/markers*** and output the **position(x,y,z)** of WhyCon markers and the **orientation(x,y,z,w)** of ArUco markers on the terminal in the same format as shown in Figure 3. The Figure indicates that you should create two separate dictionaries for WhyCon and ArUco markers; in the which key field is the marker id and values field corresponds to a table containing position(x,y,z) in case of WhyCon and orientation(x,y,z,w) in case of ArUco marker.

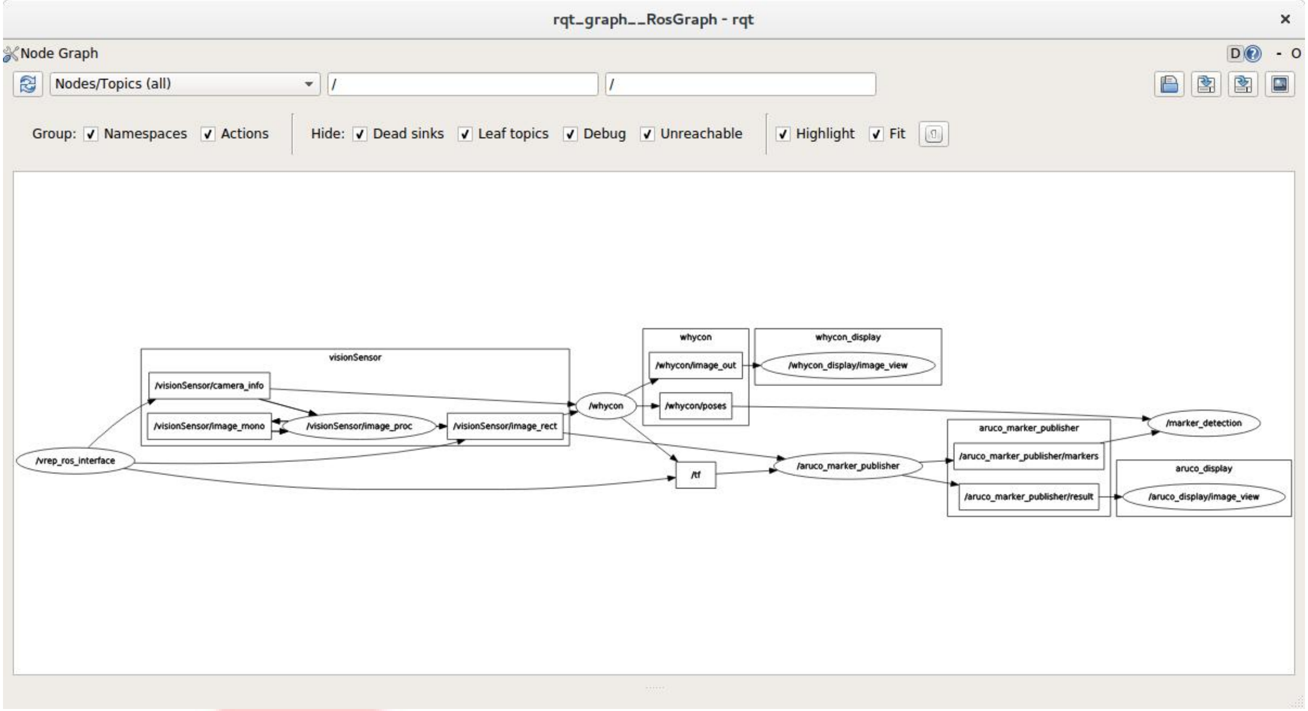


```
WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}


WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}


WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}


WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}


WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}


WhyCon_marker {0: [9.047, -1.714, 26.206], 1: [-2.852, -0.203, 22.812], 2: [4.12, 6.388, 35.622]}
ArUco_marker {0: [-0.5, 0.5, -0.5, 0.5], 1: [-0.171, 0.667, -0.578, 0.438], 2: [-0.555, 0.124, -0.101, 0.816]}
```

Figure 3 : Terminal output image

Points to remember:

- The following simulation settings are default and **should not be changed**.
  - Dynamics engine : Bullet 2.78
  - Dynamics settings : Accurate (default)
  - Simulation time step : dt = 50 ms (default)
- You are not supposed to change any of the templates.
- Make sure you remap the topics properly to detect markers.
- Run "*rosmsg show topic_type*" to see what is the message type of the corresponding topic.
- Run "*rqt_graph*" in a seperate terminal to check your topic and node relations. It should look something like in Figure 4.



Nodes/Topics (all) / /

Group: ☑ Namespaces ☑ Actions Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug ☑ Unreachable ☑ Highlight ☑ Fit

Figure 4: The rqt_graph

Submission Instructions:

Follow the instructions below to submit your Task.

1. Bag File:

a) Launch your *marker_detect.launch* file after running simulation in V-REP:

```
>> roslaunch hungry_bird marker_detect.launch
```

b) Run your python script, *get_marker_data.py* by running the following command in another terminal:

```
>> rosrun hungry_bird get_marker_data.py
```

c) Run the rosbag command to record the topics for a specific duration. The following command records the topics */whycon/poses* and */aruco_marker_publisher/markers* for 5 seconds and saves it with a name *markers.bag* in the directory from where you executed the command:

```
>> rosbag record -O markers.bag --duration=5 /whycon/poses /aruco_marker_publisher/markers
```

d) Compress the *markers.bag*

```
>> rosbag compress -j markers.bag
```

e) This will compress the *markers.bag* and now the size will be below 5mb approximately. Look the size of the bag file in its properties to distinguish between the original and the compressed bag file.

f) Rename the **compressed bag file** as **<team_id>.bag**. For eg: if your Team ID is HB#105, rename it as 105.bag

2. Image File:

a) Take a **screenshot** of your PC screen showing the output of your Python script and the detected markers (**Figure 3 and Figure 2 in a single image**)

b) Rename the image file as **<team_id>.png.** For eg: if your Team ID is HB#105, then rename it as 105.png

3. Code:

a) Rename your python script as **<team_id>.py**.

Compress these three files into a .zip file before uploading. Name the .zip file as your <team_id>.zip. For eg: if your Team ID is HB#105, then rename it as 105.zip

NOTE: You must upload all of the following: (i) bag file, (ii) screenshot of your PC screen containing both the terminal printing the marker and the images of detected WhyCon and ArUco markers (iii) your code in order to be evaluated. Please place all these files inside a .zip file before uploading. Name the .zip file as your <team_id>.zip. Please follow the naming convention strictly as specified in each step.

Instructions for uploading the folder will be provided on the portal.