

MID REVIEW REPORT

MID REVIEW REPORT INDIVIDUAL PROGRESS

ABSTRACT

This document will provide insights and current progress of the Fetch Robot Path Following.

Esteban Andrade Zambrano

Sensors and Control for Mechatronic Systems

1. Introduction

This document will provide some insights and individual contributions to the Sensors and Control Project. The agreed project was Fetch robot following the path. The main Purpose of the project is to develop a control algorithm to control the fetch robot to follow a path of a guider which is in front of the robot.

The Fetch Robot will follow a guider in a certain environment and within a specific distance.

1.2 Scope

The Constrain Project is based on developing an algorithm to control the Fetch robot based on the detection of QR codes from the guider. The control and navigation will be done in a Gazebo environment, where a custom world was developed for the task. The proposed guider will be a Tuttle-Bot. The guider custom Tuttle-Bot will have an attached link that will have a QR texture added at the back of the robot. The guider will be controlled with a play station controller.

The fetch robot will analyse the QR code in order to detect it. Once it is detected it will use the RGB and Depth Images to track the position and orientation of the guider. Based on this images it will track it within a certain angle and distance threshold. Furthermore, the fetch robot will have a laser call back where it will detect obstacles around it. This call-back will be used to detect walls and possible collision with objects around the environment. This call-back will be embedded on the navigation stack and it will improve the overall safety of the environment.

2. Individual Contributions.

For this project the task were divided between the 2 group members. At the moment the only 2 missing components from the project is to complete the navigation stack and modify the QR code node to finalise the detection.

In the below subsections will be individual contributions to the project.

2.1 Repository and Package Setup.

The repository will be hosted in GITHUB with the following URL .

- <https://github.com/esteban-andrade/FetchRobotPathFollow>

As it can be seen, from the Repository I set up 3 Branches that will be used for development of the code. The proposed branches will be master, Esteban and Ajal. The main purpose of dividing it into branches is to split tasks and manage the development of different components of the code.

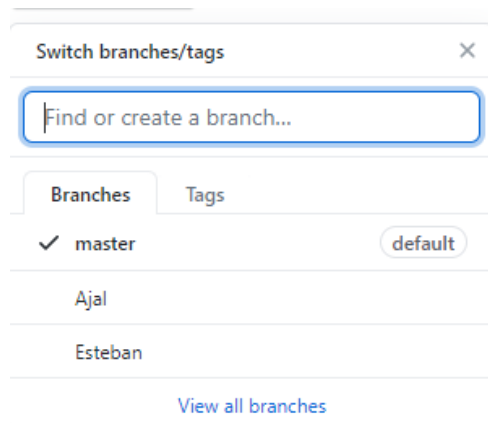
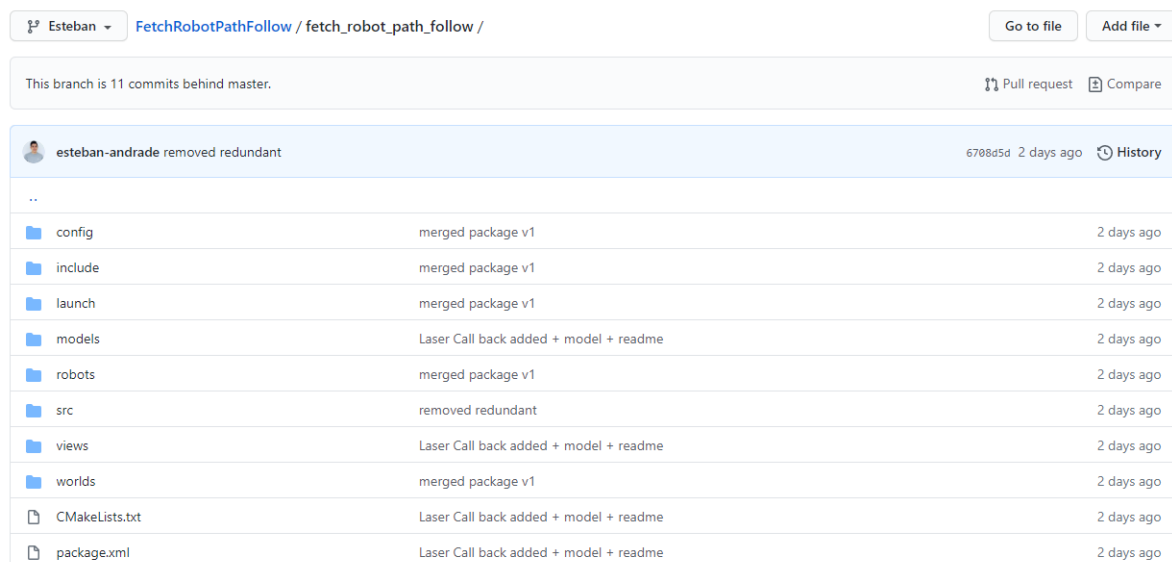


Figure 1: GitHub Branches layout

My personal Branch will have the latest pushed updates of work status of the project.

Furthermore, the initial contribution was to create the initial node and package that will be used for development. The package will be `fetch_robot_path_follow`. It will have the corresponding dependencies that will built on top of the `fetch_robot` and `tuttlebot` packages.

I was in charge of adding the custom RVIZ view, launch files for the world and robot models. Furthermore, it contains all the corresponding models that will be used in gazebo. The information on how to add this is included on the `README.md`.



The screenshot shows the GitHub repository layout for `FetchRobotPathFollow / fetch_robot_path_follow /`. The interface includes a header with the repository name and a status bar indicating the branch is 11 commits behind master. A table lists the repository's structure and recent changes.

File/Folder	Commit Message	Commit Hash	Time
..	..	6708d5d	2 days ago
config	merged package v1		2 days ago
include	merged package v1		2 days ago
launch	merged package v1		2 days ago
models	Laser Call back added + model + readme		2 days ago
robots	merged package v1		2 days ago
src	removed redundant		2 days ago
views	Laser Call back added + model + readme		2 days ago
worlds	merged package v1		2 days ago
CMakeLists.txt	Laser Call back added + model + readme		2 days ago
package.xml	Laser Call back added + model + readme		2 days ago

Figure 2: GitHub Repository layout

2.2 Custom Gazebo World

The proposed custom world will be develop and constructed using gazebo. The custom world will have a custom maze added to it. As well as objects and obstacles with in the same environment.

Furthermore, external and more realistic objects were added. Object such as persons, houses and others will give extra reality to the world. All of the added objects will have the corresponding physics components that will enhance the realism of the world itself.

The Custom objects that were added to the world are included in the `models` folder. The corresponding instruction on how to add these to gazebo were included in the repository `readme` file.

The launch file was customise and will allow connection with gazebo and the create world. All of these mapping in embedded within the package. On the other side the custom world was save with a `.SDF` file format that will allow easy integration with gazebo.

The images below illustrate the world and the locations of all the components within.



Figure 3: Gazebo World layout



Figure 4: Gazebo world realistic components

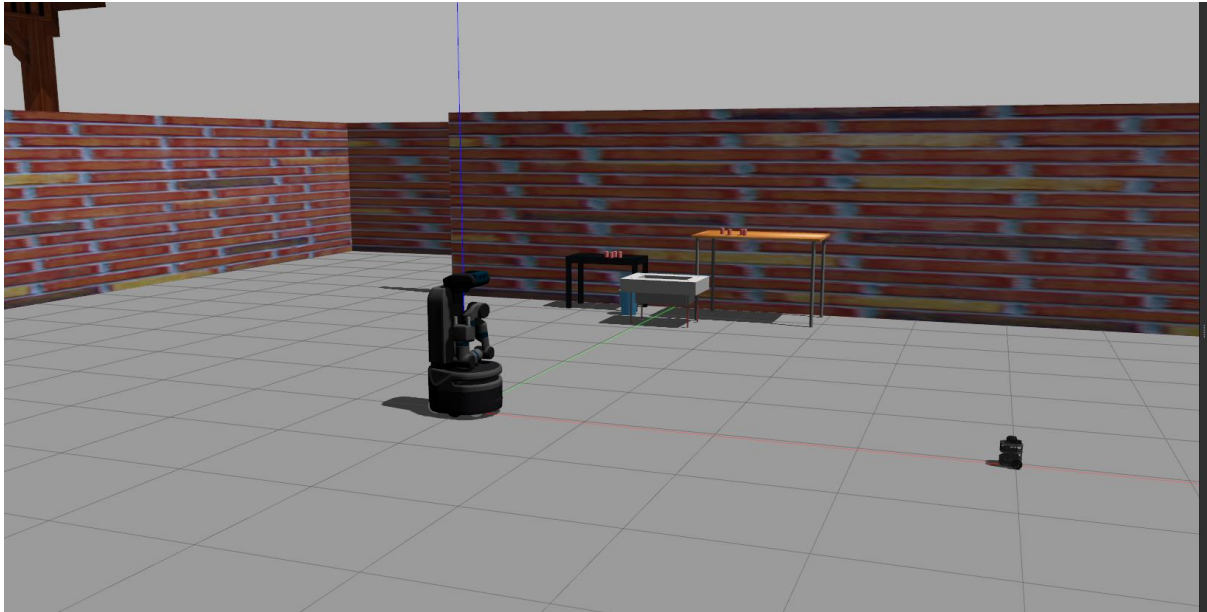


Figure 5: Gazebo world Added robots

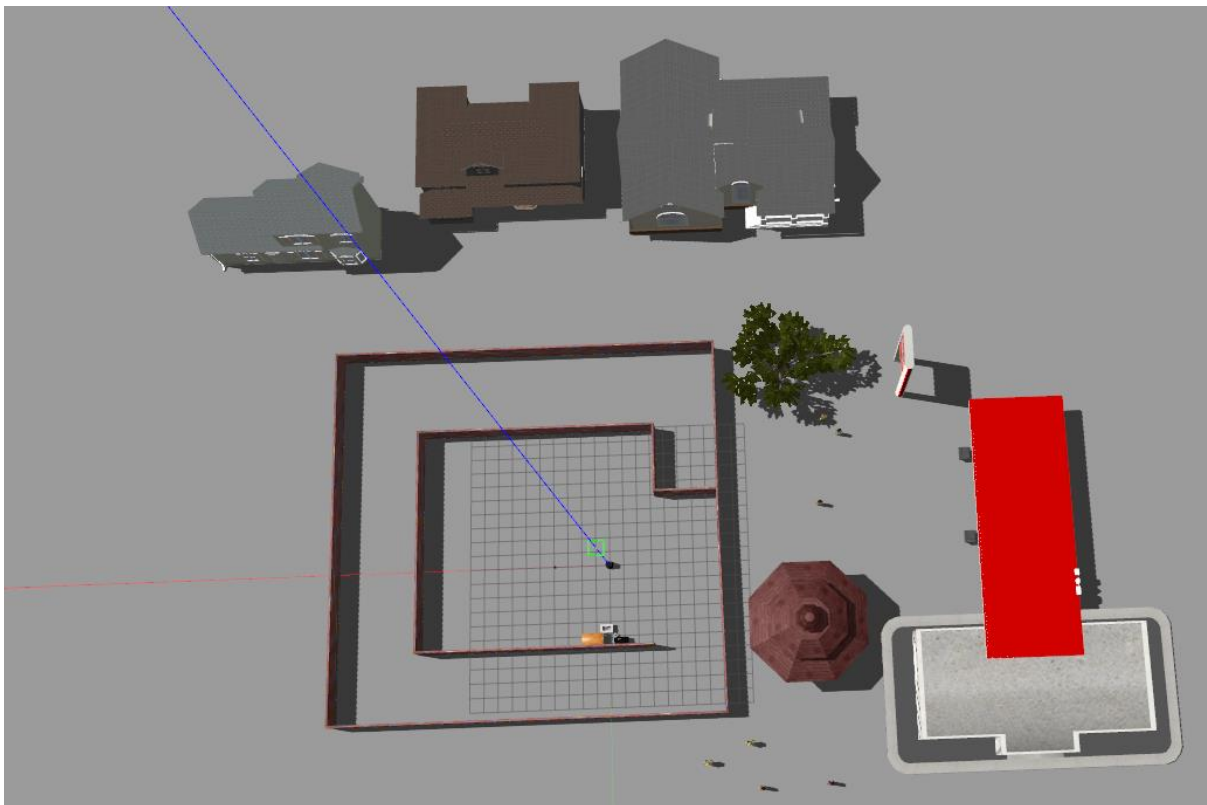


Figure 6: GitHub Repository layout

2.3 Laser Call-back

The laser call back is controlled using the built in laser from the Base of the fetch Robot. The main class was develop with two principal components. These will include a double that will return the reading of the laser scan and a Boolean that will return if an object is detected.

For this application the field of view of the laser was constrained as well as the assigned limit and the radius of the fetch robot base.

The Boolean will return either true or false based on the return minimum laser reading within that constrained field of view. If the laser reading is within the condition of the laser limit and fetch radius, it will return true, else it will return false.

This laser call-back will be embedded into the navigation stack. It will be used to avoid possible collisions with the environment or objects that maybe be in the path of the fetch robot.

Below you can see an attached picture of the fetch Robot detecting a wall.

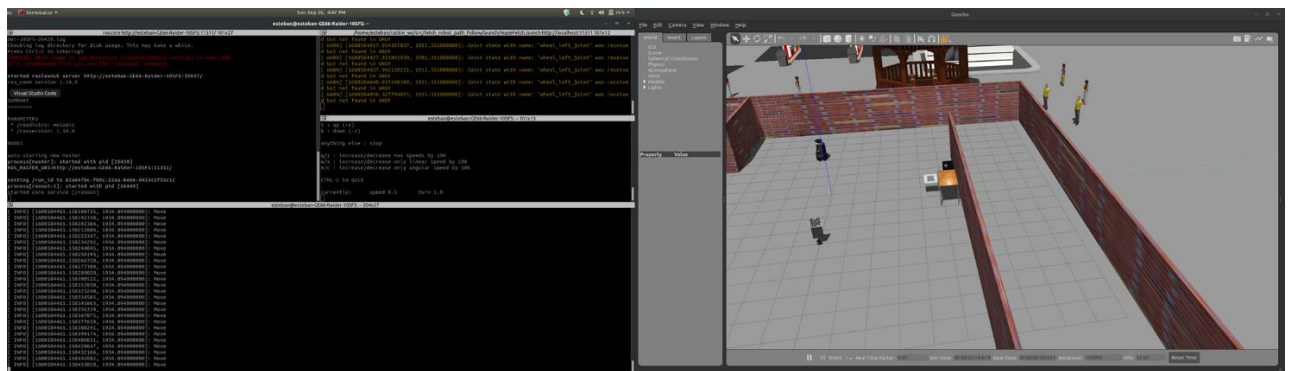


Figure 7: Laser Call back sending info valid movement

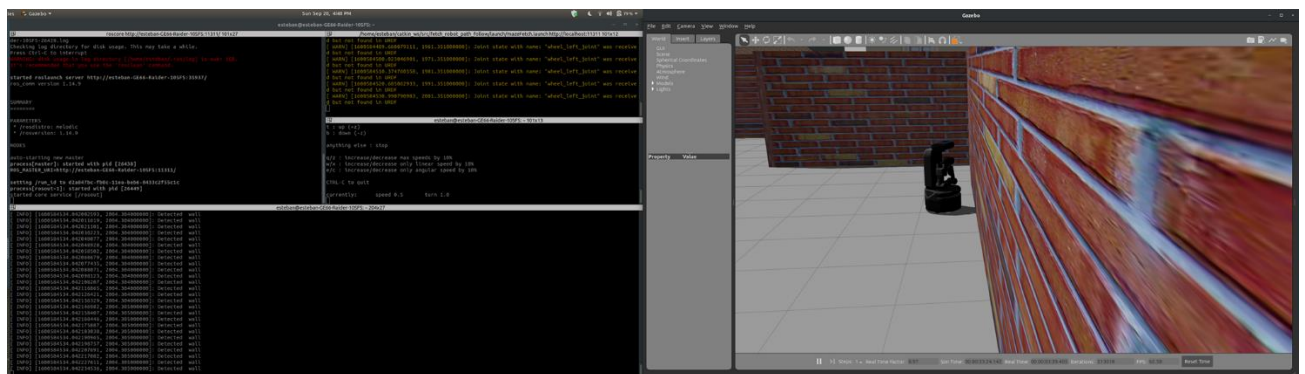


Figure 8: laser Call back notify that wall is detected

Furthermore, the image below illustrates a develop custom RVIZ view. This view will allow easy visualisation of the environment and what the fetch Robot perceives.

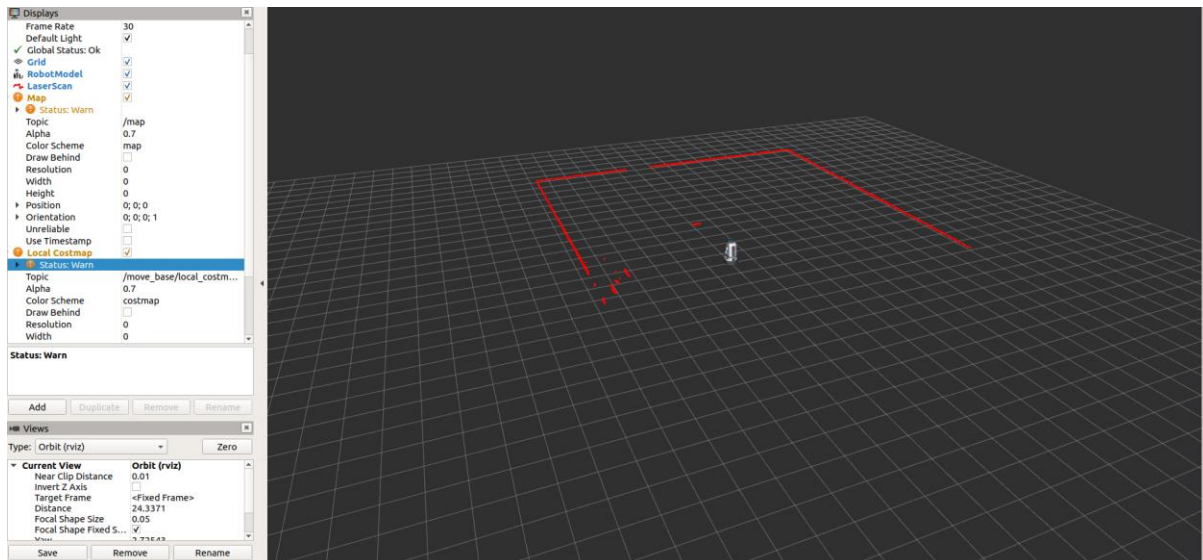


Figure 9: Custom RVIZ view

3. Next Steps

The following next steps for the tasks will be to continue developing the QR code tracking in order to be able to detect the pose of the target. Once the pose of the QR code is detected in the map, it will be possible to track the relative distance and orientation between the guider and the fetch robot. This distance and orientation will be used on the navigation stack. The pose and identification will be analysed using the RGB and Depth images of the camera. These 2 data inputs will be fused for the pose tracking and detection.

Once the pose of the QR code of the guider is tracked successfully, we will aim to control the robot with `/cmd_vel`. Based on the retrieved pose, the linear and angular velocity of the fetch robot will be adjusted accordingly based on the distance and orientation of the retrieved guider QR code.

We will maintain a safety distance between the fetch and the guider in order to maximise efficiency and security. Furthermore, the laser callback will be implemented in these in order to avoid possible environment obstacles. The proposed algorithm to be used will be point and shoot where the fetch robot will orientate to the guider and then maximise the linear speed in order to go straight. Furthermore, another alternative to be used will be pure pursuit. This algorithm will allow to adjust the gain value based on the pose of the target QR code. Therefore, the values of the controller of the algorithm will adjust itself as the data is retrieved and the fetch moves to the target.

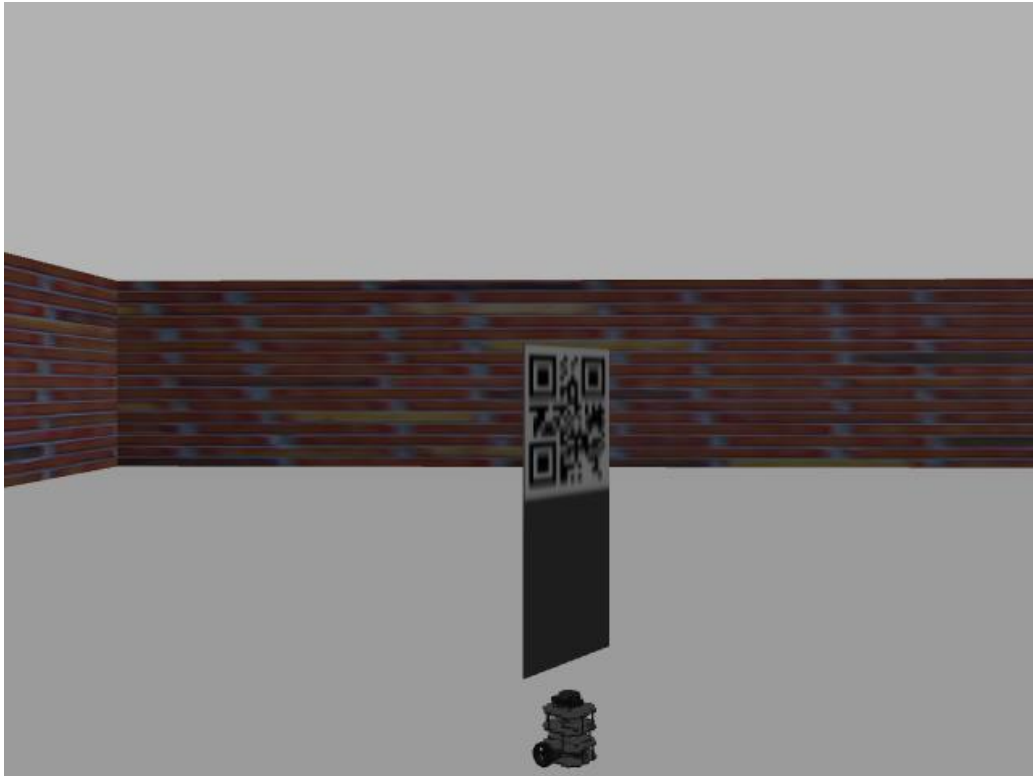


Figure 10: RGB images for tracker

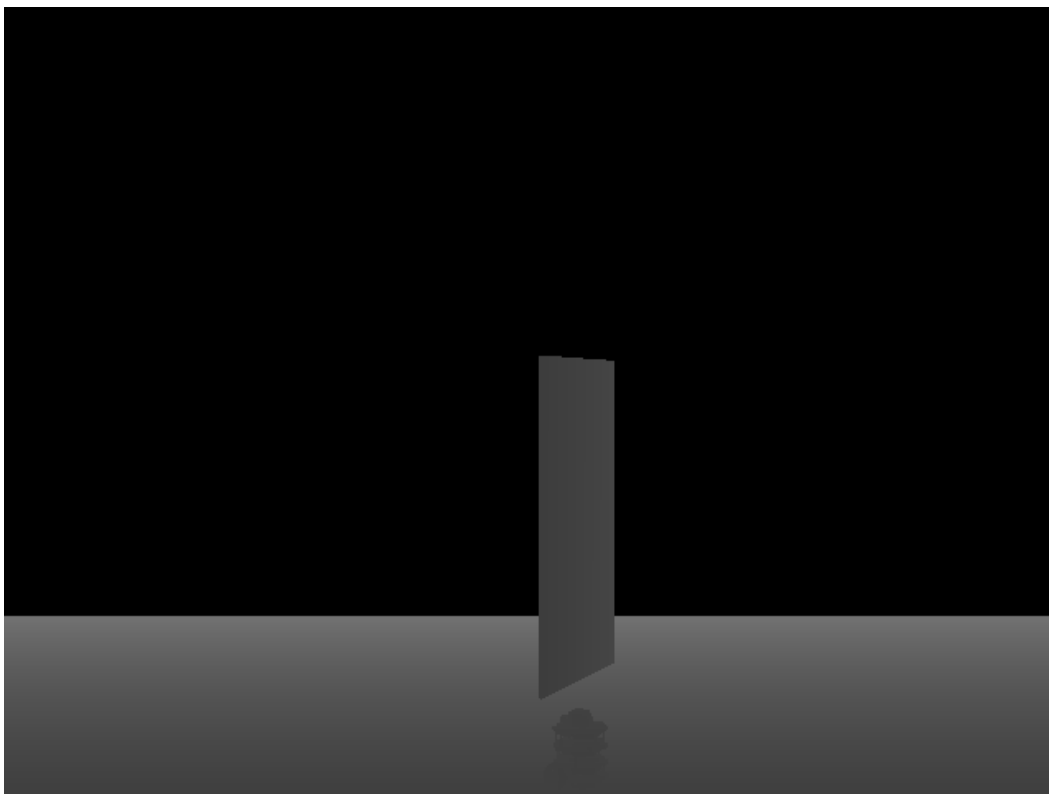


Figure 11: Depth Images for data fusion.