BOT-DRIVES-TO-WAYPOINT PROJECT

Objective: Have the car drive through the gate, autonomously.
Duration: 1.5 weeks. (depends on how fast you do the preliminary ROS tutorials)

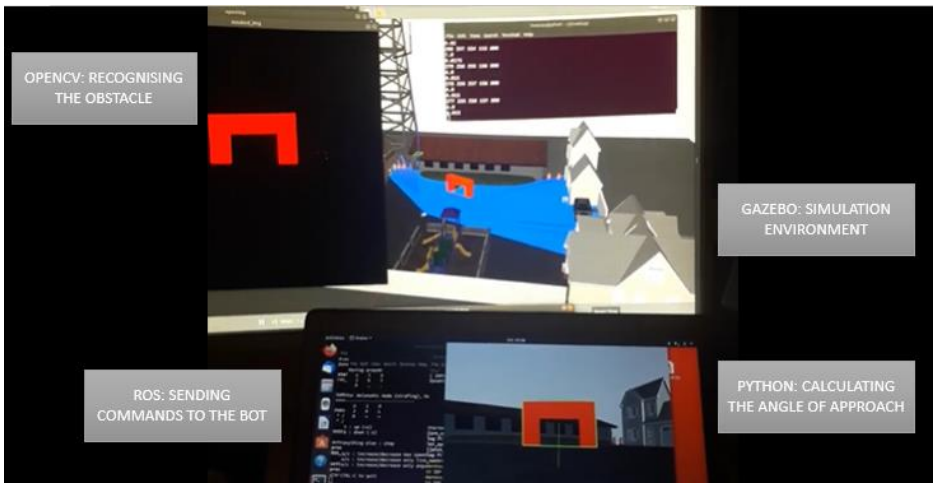
This project uses a wheeled bot in Gazebo and programs it to detect and approach a gate. This project, for me, was a primer in the use of a simulator, **Gazebo** with a programmable middleware, **ROS** in a field that interests me, **waypoint following for robotics**. This project is therefore clearly linked to autonomous drones, as they also can navigate by following waypoints. However, the specific usecase here is about:

- **coding bot behaviours within the Gazebo simulator based on what the bot observes**,
- while also creating a modular environment intuitive enough for other functionality to be added later.

Various functionalities are designed and we will be able to explore them in greater depth in a series of tutorials. This page, instead, looks at what was achieved in the project. See these pages on the site for introductory information.

- *Robot behaviour planning*
- *Basic computer vision for robots*
- *Linux and ROS background*
- *Creating applications with ROS*

The final demo is accessible at https://www.youtube.com/watch?v=rdJYIxeUt-o . It incorporates the technologies shown below.



Note: this document was last updated in October 2020. Please signal any issues at thomas.carstens@edu.devinci.fr .

1. Why ROS?

Usefulness of:

- A networked environment
- A Middleware


1. Niveau Débutant

1. Installation et Configuration de Votre Environment ROS
   Ce tutoriel vous guide à travers l'installation de ROS et la configuration de votre environnement ROS pour votre ordinateur.
2. Navigation dans le Système de fichier ROS
   Ce tutoriel introduit les concepts du système de fichiers ROS, et couvre l'usage des commandes de roscd, rosls,et rospack
3. Créer un package ROS
   Ce tutoriel présente l'utilisation de roscreate-pkg ou catkin pour créer un nouveau package et rospack pour lister les dépendances des packages.
4. Construire un package ROS
   Ce tutoriel détaille la création d'un package ROS à l'aide d'un ensemble d'outils ROS.
5. Comprendre les 'nodes' ROS
   Ce tutoriel introduit les concepts de ROS graph et l'utilisation des outils en ligne de commande roscore, rosnode, et rosrun.
6. Comprendre les Topics ROS
   Ce tutoriel introduit les concepts de Topics sous ROS ainsi que l'utilisation des outils en ligne de commande rostopic et rqt_plot.
7. No Title
   No Description

Project

Objective: Have the car drive through the gate, autonomously.
Duration: 1 week. (depends on how fast you do the ROS tutorials)

---

1. Driving a car in Gazebo

Objective: Have the car drive through the gate, autonomously.
Duration: 1 week. (depends on how fast you do the ROS tutorials)

Steps

Step 1: set up ROS, then Gazebo simulator.
Step 2: set up the environment (the gate)
Step 3: Setting up the car
Step 4: Setting up a ROS script to control the car's wheels.
Step 5: Setting up a camera on the car (and rqtplot)
Step 6: Connecting OpenCV node to recognise obstacles.
Step 7: Making the car move to the gate.

ROS tutorials

Il est toujours bon de réviser ses fondamentaux et je conseille de faire les tutos Débutant, Intermédiaire et On Your Custom Robot pour découvrir toutes les nouvelles fonctionnalités: http://wiki.ros.org/fr/ROS/Tutorials

***Si vous êtes nouveau sur Linux**: Vous trouverez peut-être utile de commencer par faire un rapide tutoriel sur les outils communs en ligne de commande pour Linux. Un bon est ici.*

Pour toute fonctionalité à implémenter sur Gazebo, voir http://gazebosim.org/tutorials

---

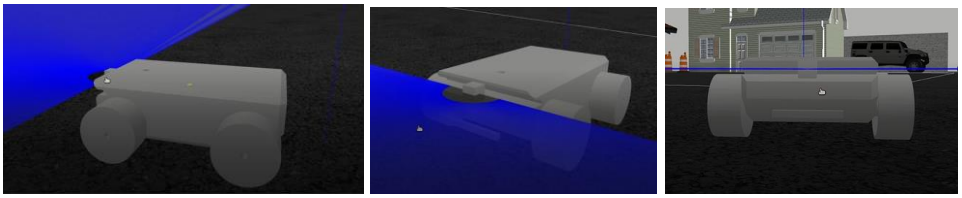
DESIGNING A WHEELED BOT

First we test the URDF in an empty simulator.

The Unified Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot. I followed a tutorial provided by the ROS documentation to create the URDF format, and another from the Gazebo documentation be able to convert it to SDF (XML file format that is readable by Gazebo).

Read the theory to better understand what URDFs consist of. But don't do it yourself 😊 because there are too many mistakes in the textbook!

- **Modelling the robot base:**
- **PG106 OF ROS TEXTBOOK**



As you can see, the front two blocks house the lidar sensor (from which emanates the blue rays) as well as the RGB camera (elongated block).

Note: do not spend too much time on this stage. If needed, ask for help.

GETTING STARTED

I used the ROS manual for this, which you can find in our ROS Resources Page.

You can easily get up and going with the finished version of the URDF accessible on the ROS textbook's github. (contact a friend to learn to use github)

- Git clone https://github.com/PacktPublishing/ROS-Robotics-Projects-SecondEdition
- For your info: Urdf and other file formats found here: **https://github.com/PacktPublishing/ROSRobotics-Projects-SecondEdition/chapter_3_ws/src/ robot_description/**

This workspace (chapter_3_ws) needs to be compiled as a ROS package. Follow instructions from the ROS Textbook:

- **Testing the robot base**
- **PG100 (stop at "Modeling the robot arm")**

SETTING UP POSTOFFICE WORLD

A template world can be loaded from the gazebo libraries.

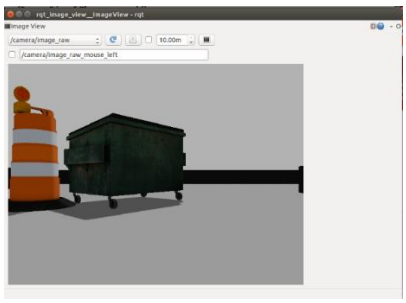
Follow instructions from the ROS Textbook:

- **Setting up the environment in Gazebo**
- **PG168 (stop at "Making our robot base intelligent")**
- **Be careful to spawn the robot base and not mobile_manipulator as they do. You can compile chapter_5_ws as they do, or keep the previous workspace.**

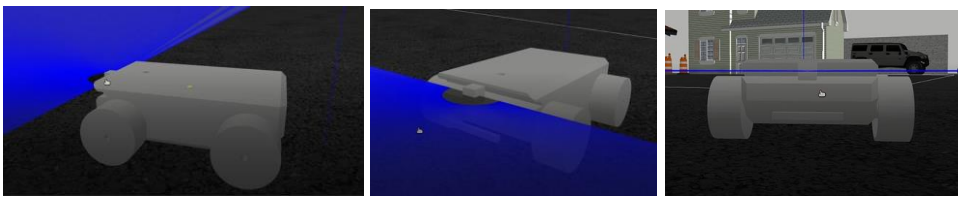
ROS Textbook - https://drive.google.com/file/d/147l-EPXTe4QW_H9m5FHB2yH3JsMd5oMQ/view?usp=sharing (request access first)

ADDING A CUSTOM CAMERA TO THE BOT

Gazebo allows the integration of cameras in order to visualise the simulation environment.



I followed a tutorial provided by Gazebo documentation in order to integrate a custom camera (top). Do similarly to integrate a custom lidar (bottom).

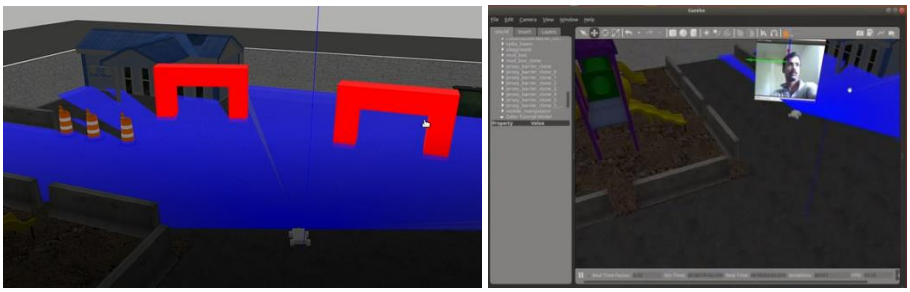


MAKING A RED GATE

In order to make the waypoint itself, it was important to examine how textures and colours are created in the Gazebo world, which I discovered in this tutorial. Once again: don't do it yourself 😊 because there are too many mistakes in the tutorial!

For now: ask txa for his model which you will be able to modify!!

Interesting notes: Adding colour textures, but also custom images, onto an object in Gazebo, actually requires the addition of a separate package for texturing. Secondly, it seemed simplest for me to fetch the model from any simulator instance by storing it in the full object in the root /.gazebo/models folder.



DETECTING OBSTACLES

For detecting obstacles, I used some basic OpenCV.

Watch out: As of May 2020, there was no support for OpenCV6 in the ROS libraries specific to Python3, therefore a ROS node using OpenCV6 was put together using various resources.

For now: ask txa for his package which you will be able to modify!!

- RVIZ OUTPUT
- OPENCV SCRIPT ANNOTATED

READ THIS FOR FUTURE USE: USING ROS WITH A MACHINE LEARNING ALGORITHM

A little video shows the car approaching the face. (**at 1:05)** The rest of the video shows the car setup.

https://www.youtube.com/watch?v=tNqYDqC6wo4&t=65s&ab_channel=ThomasCarstens

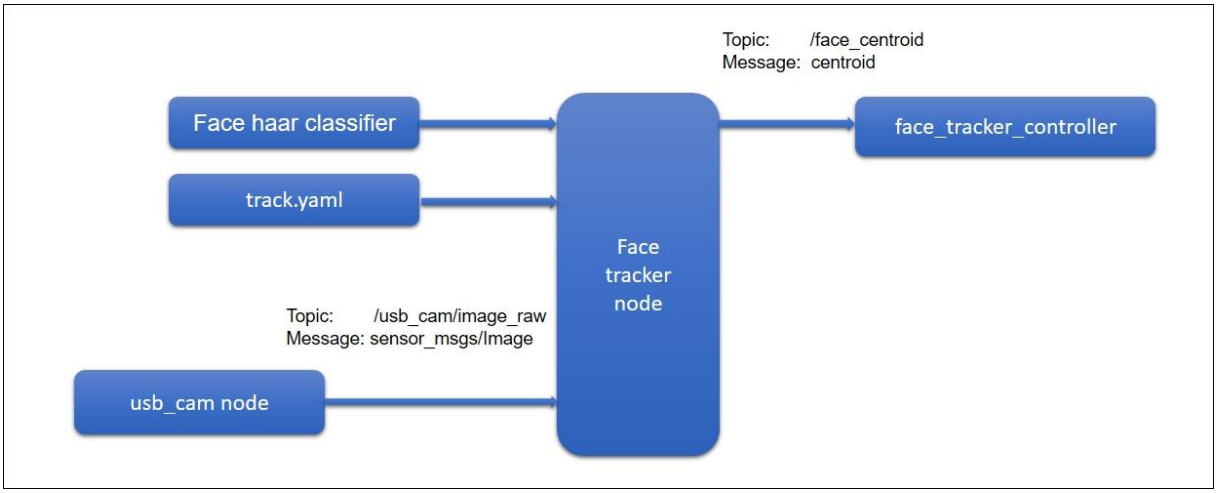
This was done with a **haar cascade.** It is an algorithm that learns to distinguish overarching pixel patterns in an image. Quick overview: https://www.youtube.com/watch?v=jG3bu0tjFbk

For building a haar cascade for other inputs (hand gestures, etc), refer to this series of videos.

- **CUSTOM**
- **Making your own Haar Cascade Intro - OpenCV with Python for Image and Video Analysis 17 (See numbers 16 and 17**
- https://www.youtube.com/watch?v=jG3bu0tjFbk

USING HAAR CASCADES INSIDE ROS

The ROS package for Haar cascades have a particular architecture which would be useful to learn about, on pg397 of the ROS manual.



I replicated a Haar cascade from this project.

STAGE 2: SETTING UP ROS

You now should have:

- control over the robot's motion
- The ability to detect the gate via its colour and dimensions

This part is about interlinking the input to the output as simply as you can.

WHEELED BOT NAVIGATION FROM IMAGE TO WHEELS

Different behaviours had to be coded according to what was detected in the images.

- MY ANNOTATED CODE
- THE RESULT

FINETUNING THE ROBOT MOTION

The robot motion required different speeds for the different parts of the trajectory. Only with specific tuning could I achieve a bot driving toward the gate.

- MY ANNOTATED CODE
- WHAT A CHANGE LOOKS LIKE

Note:

WRAPPING IT UP IN AN ACTION SERVER

The state machine could be attached to an action server.

- THE ANNOTATED CODE
- HOW THIS ENABLES CODE EXECUTION

INTEGRATING A LIDAR INTO STATE MACHINE

Using this ROS stack, we can now add more modules into our code.

- PLACES WHERE THE LIDAR IS INTEGRATED
- FURTHER INFO ON LIDARS