

Unusual Objects on Road

Team (Group 32) name: Teaching AutoPilot to Dodge

Basil Al Zamil, Xilun Guo, and Tanner Fry

CS 461: Senior Capstone Fall 2016

Oregon State University

Abstract



CONTENTS

1	Introduction	3
1.1	Goal	3
1.2	Definitions	3
2	Technology Overview	3
2.1	Algorithms testing	3
2.2	Hardware	4
2.3	Software Overhead	4
3	Method Overview	5
3.1	Theory	5
3.2	Hypothesis	5
3.3	Methodology	5
3.4	Testing	5
3.5	Data Analysis and Results	6
4	Conclusion	6
5	Change Log	6

1 INTRODUCTION

1.1 Goal

The end design goal of this project is to collect and compile a dataset that will test for failure in multiple algorithms. These datasets should target known or expected areas of fault within the algorithms. Those faults being odd lighting and unexpected objects on the road that the algorithms are not familiar with.

1.2 Definitions

Dataset(s) is a grouping of images with the purpose of being ran through an image recognition algorithm for testing.

Tesla is the car manufacturer whose "autopilot" car technology is the reason for our research.

Autopilot, self-driving, and auto-drive are all global terms for a sensor/camera autonomous vehicle that can control itself.

Image recognition refers to software registering objects within a given dataset image based on algorithmic logic.

Algorithm is mathematical and computer science term for a strict process or rule set to achieve a goal, in this case for identifying objects within images.

Semantic labeling is a method of marking the object type for software identification, for this research it is to conform to the standards of the Cityscapes.

Cityscapes is a subset of Daimler Automotive Company that is creating datasets and scripts to process image recognition algorithm outputs.

2 TECHNOLOGY OVERVIEW

2.1 Algorithms testing

There are three major milestones for the algorithm testing process. Environment conditions testing and unusual objects testing. We are planning to keep collecting on road images and video data while working on testing.

By applying database technologies we can collect different weather conditions and categorize them as such. Using machine learning algorithms to deal with the data we can see if the algorithm can provide a safer output than before. Note we are not concerned about real time (0.1 sec) algorithm speed. We will primarily use parameter testing groups with many images in them. For example we have a full of images where it is snowing. Then we keep changing the variables in the test and continue testing.

We can setup a test of moving unexpected objects and test them. Randomly set up various conditions where objects suddenly appear to the front of cars, and then test how quickly the algorithm reacts and stops the vehicle. We put all these conditions onto the columns and all outputs onto the rows in a matrix (more on this later) so that we can combine the information and tell if any algorithms could cause a crash.

Finally, we will set up a global matrix, and divide the columns onto two parts with weather conditions testing and unusual objects testing. In addition, dividing the rows onto three parts as well, for each testing outcome with the algorithms we have. We then put all conditions onto the columns and all output onto the rows in the matrix so that we can cross reference the information to gain insight in how each algorithm performed and whether it is failing with a given condition. From that information we can try to determine which parts of the software algorithm we can try to fix.

2.2 Hardware

The hardware consists of the recording device: An HD dash camera, and storage unit: SD card and an external hard drive.

The video footage that will be used to test the algorithms are primarily taken through the camera. The camera will be mounted on a car's windshield, and powered through a 12 volts outlet. The camera has a 170 degrees lens, which would capture a wider view of the road. The camera's recording dimension is 2306x1296P resolution. The camera can be used in both day and night time, since it supports night video recording.

The SD card is of 128 GB, while can take up to 20 hours of video footage. The external hard drive, which is a 5 TB, will be used to store the footage from the SD card when the SD card is full. The footage has to be stored from the SD card roughly every 10 to 20 hours of video recording. Otherwise, the camera will overwrite the oldest file in the SD card.

The camera is turned on manually before the driver starts to drive the car. The video recording would take place in both the day and the night. However, during the night time, the recording will be limited to the beam of headlights. Hence, video taking during the day is preferred.

Video recording will take place mostly in Oregon, including highways, freeways and country sides. Thus, will give us an opportunity to test different types of objects on the roads.

After recording, we are going to test the image-recognition algorithms on different objects. Since there is so much footage we determined that the best method of analysis is to grab frames from the video feeds at an ten second interval and then down scale the image to something manageable for the hard to handle. For example, regular cars are most likely to be recognized by the algorithms, whereas unusual types of vehicals could fail to be given a label correctly.

For all of our video frame capture we used VLC an open source tool for videos. For the image down scaling we used default OS tools and even some scripting for certain images.

Additional videos will be requested from Oregon's Police Departments and Oregon State Car Club.

2.3 Software Overhead

The available algorithm sets run off mostly python scripting. The code to handle different algorithm labeling and formats is available on GitHub at github.com/mcordts/cityscapesScripts with a decent amount of documentation alongside it. This code is mostly for handling and analysis of other algorithms semantic labeling. However from my research the code is designed to only parse our with the CityScapes dataset by default. A decent amount of overhead work will be required on our part to get the functionality in place that we need. Along with the documentation the team members at Cityscapes are more than happy to take questions, suggestions, or input should we run into any of the above. The scripts themselves are broken down into five main parts. Viewer scripts are designed to view the images and annotations within the datasets. Helper scripts and files are included by other scripts to support functionality. Preparation scripts are used to convert the "ground truth" annotation into a format used by the specific algorithms approach. Evaluation scripts are for validating of image recognition methods. And finally an annotation branch of scripts are used for labeling datasets passed through the code. Thankfully the writers of the scripts also stuck a basic documentation level at the top. Along with this is a list of the core scripts we will need to use per dataset.

After more thorough review of the algorithms available to us we determined that it would be best to try only the best algorithms. The main issue with this is that each algorithm available to us is from a different supplier and thus there is no consistency between setup and environment variables. For example the Dilated Convolutions team requires the

use of Anaconda (a Python compiler) and Caffe (a Python Utility) for using their algorithm. This means we will have to run our testing methodologies across all the different algorithms that we test, so we will take on algorithms as we see fit. This only includes options that there is code available for which can be found at <https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task> in the 'Code' column. The datasets we collect need to be broken down into smaller chunks of frames, our aim is for one frame per ten seconds. That might not sound like much but for an hour of footage that adds up to 360 images to verify. We have also been given permission by our client to acquire products if we deem them necessary, a video editing tool might be a possible purchase. Please note many algorithms already have a pre-learned database for the CityScapes dataset.

3 METHOD OVERVIEW

3.1 Theory

We theorize that most image recognition algorithms for self-driving cars do not handle certain conditions correctly. This is based on a crash of a Tesla car in 2016 where it was determined that the image recognition systems incorrectly recognized the side of a white semi-truck as a cloud. The system then drove into the side of the semi assuming the path was clear, killing the driver in the crash.

3.2 Hypothesis

Since the algorithm of Tesla 2016 car failed to recognize the white semi-truck, we are approaching the research with the hypothesis that the algorithms would fail to recognize novel objects. Novel objects are objects that the algorithms producer did not program or test the algorithms to recognize, or did not test the objects under different weather or light conditions.

3.3 Methodology

To thoroughly test as many cases as possible we are setting up a matrix of testing scenarios and comparing that to the results of other algorithms. Each algorithm will be ran with the same datasets and then the results will be semantically marked up using the python scripts provided by Cityscapes. These markups can then be compared across each other to figure out which ones perform better. However each frame must be checked manually to make sure the object in the frame was registered correctly.

3.4 Testing

White box testing is our main idea for all algorithms testing. We are planning to deeply look at the source code and base on clearly understanding which algorithm is used for which condition before the testing. It is hard for us to have enough knowledge to read the source code in limited time, which is one of the main disadvantage. However, we will keep in touch with someone in the algorithms developing team for further help. On the other hand, we have the advantage of having a white box testing. This is helpful since there are large amount of data that we need to test, and due to the need auto testing.

3.5 Data Analysis and Results

There are two methods of analysis we will plot. The first method is individual testing on objects and lighting per algorithm. These will be analyzed manually by checking frames with a testable conditions to see if the algorithms correctly handle the situation. We then will make note on each testable image of either success or failure for each algorithm we test. Thus creating a matrix of testing parameters for analysis and comparison. From there, we can determine not only which factor failed in a given image, but also determine how each algorithm performed for each test image.

4 CONCLUSION

We have described a direction of our project and some milestones with how we planned to process.

5 CHANGE LOG

- Revised goals to be more focused on algorithm testing quality over number of algorithms tested.
- Minor updating to definitions for accuracy.
- Updated algorithms testing section to exclude camera blind spots, not something in our scope as of now.
- Updated hardware and software information based on products we have used for certain tasks
- Made changes to reflect that we are going to be testing algorithms from different developers and that those algorithms have their own machine learning knowledge base.
- Added change log section

REFERENCES

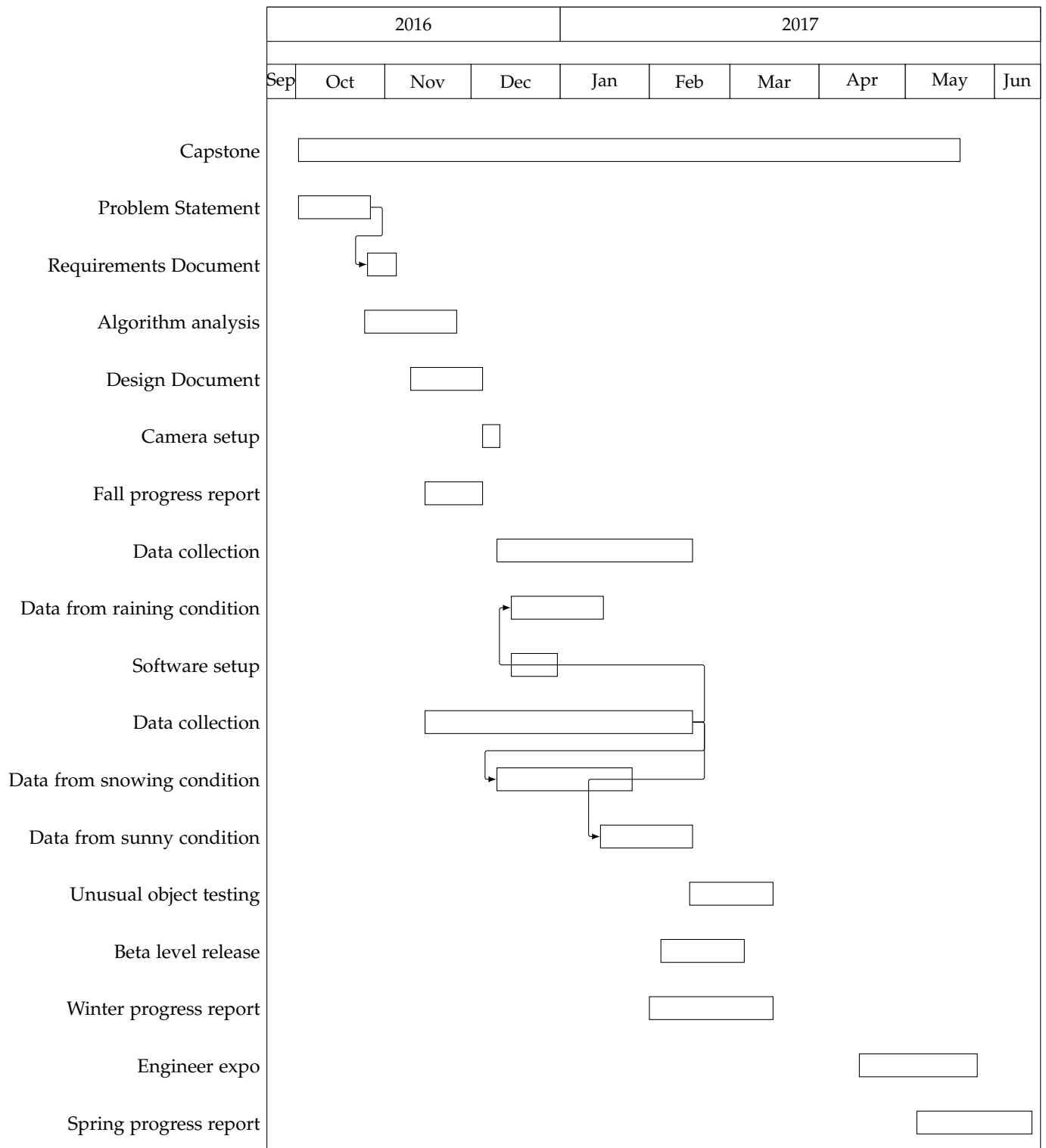


Figure 1: Gantt Chart: Timeline of Project Tasks