Algorithm Overhead and Setup - Tech Review

Team (Group 32) name: Teaching AutoPilot to Dodge

Tanner Fry

CS 461: Senior Capstone Fall 2016

Oregon State University

Abstract

The aim of my tech review research is to figure out the best way to implement and begin the testing process for the algorithm. Compared to my teammates I will be doing more of the technical overhead of the project, setting up the algorithm and environment necessary to test the datasets we put together. This document is put into three main sections; environment, algorithms, and functionality. Environment focuses on integrated development environment options, operating system limitations, and any outside libraries needed. Algorithms will focus mostly on the options at hand for testing the image recognition code, there are multiple options available to us. Finally on what we expect the code to do and handle be default, and what functionality we need out of it to complete our tasks.

CONTENTS

1 Software Environment

1.1 Code

The Cityscapes algorithm analysis sets run off mostly python scripting. The code is available on GitHub at github.com/mcordts/cityscapesScripts with a decent amount of documentation alongside it. This code is mostly for handling and setting up the image recognition code to run. However from my research the code is designed to only run with the Cityscapes datasets by default. So 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 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.

1.2 Evaluation

Once we test our datasets the Cityscapes team has a handy way to turn in the dataset that fails to that others can use it to test with. The Cityscape team is interested in points of failure in both other people's approach and how to properly analyze those methods. The results need to be formated in the label image names as labelIDs as per standard of the Cityscapes team. The images produced from our testing should have a pixel value class ID defined in labels.py.

1.3 Development Environment

At the highest level the system should work with any machine or system that can run python scripting. That being said there are a lot of options with a lot of positives and negatives. After much deliberation we discuss with Dr. Li and decided to run the algorithms on an OSU based server called Steed. The Steed server is GPU accelerated which will make processing images much faster. It is globally accessible to all of us, so we can work on things concurrently. The environment is Linux based, however we can create our own virtual environments to work in when running the algorithms, thereby eliminating the problem of having to manage extra environments or operating systems.

Researching for the best Integrated Development Environment (IDE) turned up to be easy. Some of the main options were PyDev http://www.pydev.org/ a simplified IDE for Python development, it plugs into Eclipse and is relatively simple to run. Another high profile option is PyCharm https://www.jetbrains.com/pycharm/ which is an expansion of Jet Brains line of IDE options. I believe the later is our best option as it comes with the complete ”package” of utility we need. It has a Python assistant, cross-technology development, and built-in development tools. Not to mention it is similar to the IntelliJ framework my team used in Software Development I. That being said, once we figured out how to run things on the Steed server the need for any sort of GUI environment went away.

2 Algorithms

2.1 Options

There are a multitude of options for image recognition algorithms to test linked on the Cityscapes web page. We have been instructed to test only the best algorithms so we plan to try four of the best algorithms first. Quality is more

important than quantity when we are working with algorithm testing.

All that being said, information on this section is limited as of now because of research. The Cityscapes linked image recognition algorithms are for research only and can only be accessed by approved sources. I have made an account and submitted an ”application” for access to the algorithms through the Cityscape Website, however they must approve our team before we are allowed to download and use the algorithm. In this case approval to access the algorithms is the critical path. This document will be updated when access is granted. Access has been granted and we have found the best two algorithms to test first.

2.2 Methods of Image Recognition

It all depends on the algorithm we run, our first was the Berkeley FCN model which uses a form of machine learning and pixel image breakdowns to analyze images. This method has different option of how many splits to do, but the general theory is to break down the image into 8/16/32 levels of images and combined the analysis feedback from all of those images. Refer to these slides for more detail https://docs.google.com/presentation/d/10XodYojlW-1iurpUsMoAZknQMS36p7lVIfFZ-Z7V_aY/edit. Another method uses different scale qualities of images and does the same analysis on each. From there it then combines the data labeling it got into a single image and outputs that as the result.

3 FUNCTIONALITY

3.1 Dataset Testing

What are the needs of the setup we are planning to put together for this project to meet all requirements? First we must be able to test the algorithms with our own datasets. This requires that we have a way to convert any video feeds or formats into proper form for the algorithms to handle. Thankfully a lot of this is handed to us via the python scripts already created by the Cityscape team. But we also need to get tangible results back out to find places of fault. This is again done for us thanks to the scripting already in place, we just needs to accept it in Cityscapes format and labeling style. Not a hard problem to solve if we chose to migrate away from that, we just need to write our own python scripts for labeling. That being said we see no immediate issues with the given format, and will stay with it unless a better method is found. The dataset feeds themselves must be broken down to a pixel level for object analysis to take place, typically this is done in a gray scale. Along with file conversions we also will needs to handle color manipulation if the algorithm requires it.

3.2 Error Detection

Next we need to be able to detect faults in the image recognition software. Assuming the above labeling process is correctly done unknown objects will either call into the wrong type, or filter into an unknown category which will help us tremendously with testing. With simple scripting we should be able to pull out faults with ease, and in cases where the errors are not caught in the labeling process we will have to do eye level inspection. Our datasets where we want to target something specific should be rather short in interval, the Cityscape team processes and posts 5000 images per week. So we have a reasonable understanding of what we can do with out resources based off that.

3.3 Extras

The topics here are things it would be good to have at our disposal during this projects life span. First thing is a way to organize datasets that need to be analyzed and those that have been. And a way to track and label them accordingly. If we end up processing a lot of datasets keeping track of them manually would be unnecessary. It would be smarter to label them on creation and then pass them through an automated process that does the work for us. This is something to be calculated when the time comes, creating the script to do the process automatically might be more time consuming than doing it by hand. Video or image converting would be nice to have. Thankfully there are many open source tools available to convert video file types, and OSX will convert file types when the .extension is changed between formats. For example changing between a .flv and a .mp4 requires just a change to the those extensions.

4 Change Log

- Major changes to
- 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

[1] 20002016 JetBrains s.r.o. All rights reserved. Developed with drive and IntelliJ IDEA *Jet Brains at PyCharm.com*

[2] Brainwy Software Ltda, 2014-2016 *Brainwy at Pydev.org*

[3] https://github.com/mcordts/cityscapesScripts. *Part of the Cityscapes team* Marius Cordts, Mohamed Omran.

[4] https://www.cityscapes-dataset.com/ *Cityscapes subset of Daimler* 2016 Cityscapes Dataset by Marius Cordts

[5] http://pedrokroger.net/choosing-best-python-ide/ *Choosing the Best Python IDE* Dr. Pedro Kroger