Autonomous Vehicle Processor

Design Document

Dec1710 - Team VOLDEMORT
Rockwell Collins - Josh Bertram
Advisors - Philip Jones, Joseph Zambreno
Team Leader - Alex Orman
Communications Leader - Sean Jellison
Webmaster - Chris Kelley
Key Concept Holders - Lixing Lin, Evan Lambert
Scribe - Lucas Ince
dec1710@iastate.edu
http://dec1710.sd.ece.iastate.edu/

Revised: 04/20/2017, Ver. 3.0

Contents

1 Introduction

1.1 Project statement

The purpose of this project is to utilize an onboard system in realtime to provide information to flight systems regarding location of objects in a drone camera's view. To accomplish this we use a pre-trained neural network to identify complicated objects and then a series of other, simpler methods to extract additional information such as distance away from the drone.

1.2 Purpose

This software will utilize a system's GPU to be able to more accurately and efficiently apply deep learning concepts to embedded real time system processing. It will be able to detect and analyze objects and provide relevant information about it.

The use of a neural network operation with real time data on onboard hardware has a wide range of applications beyond this project. While many military applications come to mind, there are plenty of civil uses as well to better the world. A good example of this would something like Amazon's drone delivery service. If they could utilize this system, it could save time and money for the company by avoiding objects and ensure more happy customers by putting packages in more convenient and secure locations.

1.3 Goals

When finished, we want our system to be able to be installed onto any autonomous system and be able to provide accurate and useful information about the surroundings. This is intended for use on remote controlled drones, but this could also be applied to autonomous vehicles to detect objects near the vehicle or surveillance systems to detect intruders. The ability to detect a generic object has a multitude of uses already, but our system will be able to be trained for specific purposes. Additionally, the ability to easily the network to detect novel objects.

Lastly, here is a list of tentative features we hope to achieve in order of under taking:

- Simulation data collection
- Distance and relative object orientation calculation
- Geographical position detection based on landmarks
- Object trajectory predictions
- Multi Camera and alternative camera support
- Wingmate and other aerial object tracking

2 Deliverables

We aim to provide software that can be installed onto an embedded board for use in autonomous aircraft. The software will be able to detect landing strips, runways, known airports, and be able to analyze additional flying objects. It will also be able to track objects and point out landmarks or significant structures.

Specifically, these items will be released to the client upon completion:

1. Source code for application developed, unless otherwise specified we(the students) may share and reuse any and all source code.
2. A PDF guide explaining installation and usage of source code aforementioned
3. A powerpoint explaining the project
4. Any videos generated during this project regarding the project.

3 Design

Conventional Computer vision: We researched standard computer vision techniques, and determined that standard methods were slow and unadaptable. We were looking for a system that could be updated in the future to determine multiple different object types. We also wished to limit the amount of computation necessary.

Neural Networks: We decided to use a neural network to solve the problem because of its lightweight approach to solving complex tasks. We are using a convolutional neural network to limit the level of computation needed for the image processing. We are avoiding recurrent networks to simplify the program also. .

3.1 System specifications

Initially, use NVIDIA tutorials to implement some type of visual object recognition or similar task and use a mockup environment in the lab. (Toy airplanes on sticks, etc.) As confidence grows, Rockwell Collins can potentially provide some video footage from one of our drone flights this spring or summer. The specific objective of the visual system can be negotiated; examples might be runway detection, detection of another nearby aircraft, etc.

We will not receive any outside data besides what is collected by the camera. We will need to generate any required information through image processing.

Any goals not stated above are choices the design team have chosen to implement and are assumed about the project, such as image processing speed and quality:

- Board: Jetson TK1 board mounted
- Camera: oCam-1MGN-U Global Shutter Greyscale
- Machine Learning Framework: TensorFlow
- See functional and nonfunctional requirements

3.1.1 Non-functional

- Embedded system must fit inside the drone
- Must be able to process an image at least once per second from onboard camera

- Ideally, target goal is 30-60 FPS

3.1.2 Functional

- Runable on the embedded system.
- Can detect basic objects.
- Must be able to take in images from a camera
- Be able to locate key features in an image
- Be able to identify known objects or features from 400ft - 600ft

3.1.3 Standards

As of yet we have not encountered a need to have a standard to follow or set up. As a general rule we will follow established good programing practices for C, C++, and Python. These are the main languages used for the training and of our neural network , and the post processing.

3.2 PROPOSED DESIGN/METHOD

As a team, we have decided to implement a machine learning/deep learning approach to this problem. We plan to utilize a convolutional neural network to identify objects such as runways, airports, and other planes/drones. A camera will provide input for the network. Pre-processing is likely only to be resizing the image to fit the network. The network would use that input to determine what type of object is in the frame. The network output would be sent for post-processing which could be used for detecting the orientation of an object, the distance of an object, and detecting other friendly drones.

3.3 DESIGN ANALYSIS

The camera we are currently planning on using is a Greyscale oCam-1MGN-U with Global Shutter. We may end up using our original choice of a Color image See3Cam if the greyscale images are not sufficient, but the oCam is a much cheaper alternative.

oCam-1MGN-U Technical Details

- Sensor : OnSemi MT9M031 CMOS image sensor

- Lens : Standard M12 Lens with focal length of 3.6mm

- Field Of View(FOV) : 65 degree at full resolution of 1280x960 (images of less resolutions are copped from the full resolution image to have less FOV accordingly.)

- Image sensor size : 1/3inch

- Pixel size : 1.4 μm x 1.4 μm

- Shutter : Electric Global Shutter

- Interface : USB 3.0 Super-Speed

- Supported OS : Linux, Plug-and-Play by UVC(USB Video Class) Protocol

- Power : USB BUS Power

- Rating : 5V/180mA

- Operation Temperature : 0°C ~ + 70°C

- Camera Control : Brightness, Exposure

- Frame Rate : 45fps@1280x960, 60fps@1280x720, 80fps@640x480, 160fps@320x240

- Weight : 33.5gram approx. (Including protective case)

- PCB size : 39x39mm

- Case size : 42x42x17mm

Alternative See3Cam_10CUG_CHL

- Sensor : 1.3 MP AR0134 ⅓” format RAW image

- Pixel size : Pixel size:3.75μm x 3.75μm

- Shutter : Global Shutter

- Interface : USB 3.0 device with USB 3.0 Micro-B connector

- Supported OS : Windows 10, Windows 8, Windows 7 & Linux

- Power : USB BUS Power

- Operating Voltage : 5V ± 250mV

-Power Requirements : 500mA for USB 2.0, 900mA for USB 3.0

- Operation Temperature : -40°C ~ + 85°C

- Focus : Fixed Focus

- Frame Rate : USB 3.0 [VGA(binned) 45fps, HD(720p) 60fps, Full Resolution 45fps], USB 2.0 [VGA(binned) 30fps, HD(720p) 12fps, Full Resolution 9fps]

- Weight -

  - 23 Grams (without lens and without enclosure)
  - 100 Grams (with lens and without enclosure)
  - 72 Grams (without lens and with enclosure)
  - 149 Grams (with lens and with enclosure)

- Size : 40x44mm (lxb)

We have a small collection of data at the moment, roughly 1000 images, to train our network with. However we would like to have many more images to train and test with. To meet this goal we are both working independently and collaboratively with our client to find a way to generate such a large volume of correctly labeled data.

Some ways to generate the images:

- Use a simulator and take screen-shots of relevant objects
- Find online images and databases
- Take pictures manually or from a drone
- Use video streams with the relevant objects

4 Testing/Development

4.1 Interface specifications

Our interface for testing will be data output to a console/terminal showing a list of the objects detected, a percentage of confidence for each object, and other our intended data from each image such as the camera's distance to the object.

If we feel it necessary, we may make a simple gui to better display the information we want and possibly be able to change parameters during runtime, but this is not a priority.

4.2 Hardware/software

Tenserflow: A machine learning framework that helps us quickly and accurately generate neural networks, and then goes a step further by allowing us to connect our network to different data sources via Python.

Jetson TK1: The Jetson board is a GPU accelerated computer on a chip module. It houses a full micro PC with hefty computing power in a compact and robust package.

oCam-1MGN-U: Greyscale Global Shutter Mono Camera with up to 1280x960 Resolution

Alternative:

See3CAM_10CUG_M : 1.3 MP Global Shutter Monochrome Camera with 720p HD and support for trigger modes.

4.3 Process

As of now we have only been using theory and a test Tenserflow neural network on a Python script. All tests have been manual tests feeding in images to the network.   We have set up scripts to automatically gather test photos from the simulator and have found datasets online.

5 Results

5.1 Preliminary testing

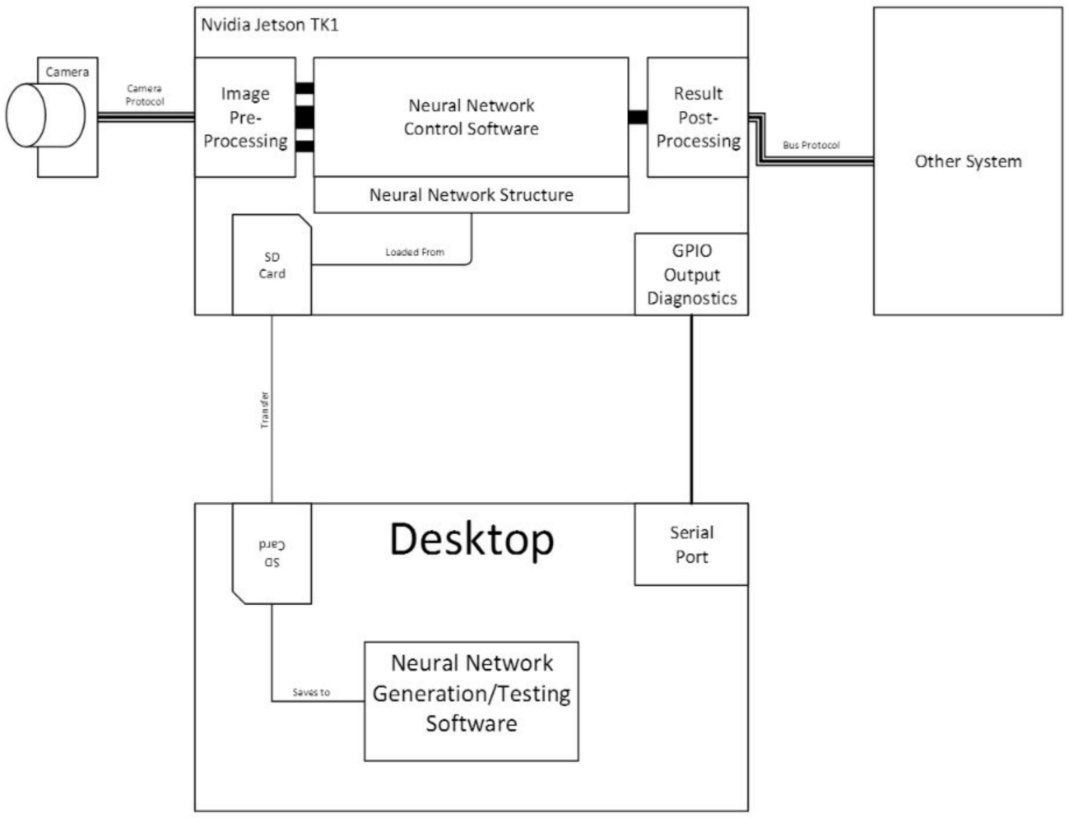
At the very beginning of the project we downloaded and ran simple neural net projects to do character recognition and later simple image classification. We found that the demos work exceptionally well at being able to recognize novel data, or images. As expected, the object character recognition program(OCR) worked well as expected but a more exciting result was a simple image classification test we performed. We scraped roughly 200 images from google, half of roses and the other half of jetliners. We then found a novel image of a rose, a jetliner, a number of pictures of ducks.

When testing a number of novel images against the network, after we had trained it, we found that the network was almost always right. This lead to great confidence that we were moving in the right direction.

6 Conclusions

By project's close, we shall have a system utilizing an onboard system in realtime to provide information to flight systems regarding location of objects in a drone camera's view. We have some ambitious goals for this project, but we are dedicated and eager. We will have it detecting airports, providing useful information about objects it sees, and be able to be trained to detect new objects.

7 References



8 Appendices