TATA MAKERTHON

Object Detection and Tracking

-----------Project By

TF-003179 Aditya Mahajan

TF-003306 Mansi Wakodkar

TF-003312 Aman Agrawal

Abstract:

Finding objects by a computer in an image can be a daunting task but due to neural networks it has become a lot easier as the neural networks can be trained and used for various purposes, similarly for object detection we will be using the neural network to find a particular object from a given image (given by phone) then searching in the same object is present anywhere in the surrounding which is done by a trained model. The images would be processed on CVB (Nvidia Jetson TX1). It would check if the entities are matched, then give the result on the phone if the object is found with the orientation of the object all the database here is managed with Firebase. The images from surrounding would be taken from a camera which would be placed on the gimbal controlled by CCB (Computer vision board). Drone would be given instructions to maneuver and take video and detect objects in it (for prototype we would be using Images). The output data would be uploaded on the firebase. This data will be accessed from the app by pressing the refresh button.

A) Introduction:

The main aim of the project is to find some particular object in real time. This help a lot in real life. We are creating an app which would click a photo, label (Crop) the object in it and sending it to the CVB which would be connected to the CCB which in turn controls the gimbal and the camera, which all will be on the drone to maneuver in the surroundings and check if the object is present return object found or else return error.

B) Hardware and Software specifications:

There would be 2 specifications for hardware that is which we would require for the actual project and the other which we would use for the PoC prototype in round 2 if selected.

FOR PROJECT

1) CVB: Jetson TX1
2) CCB: SToRM 32
3) Camera and Gimbal control algorithm
4) Language to develop the software: Python 3
5) Firmware for CCB: PX4
6) Platform for clicking photo: Android
7) Software for developing the app: Android Studio

FOR PoC (Low End PROTOTYPE)

1) CVB: Raspberry pi 3b
2) Language to develop the software: Python 3
3) Platform for clicking photo: Android

C) Detail Explanation:

(All the reasons behind using the software or methods given below are explained in the next section)

We have created a app which is capable of clicking the images and storing it locally, we are using Firebase to handle our database we have created 2 tabs in our app that is input and output input tab will click and store the image and when pressing the submit button there would be authentication between the database and the app on successful connection image will be stored in the Firebase database which would be present on the cloud. Now we will access that image on the CVB for further processing.

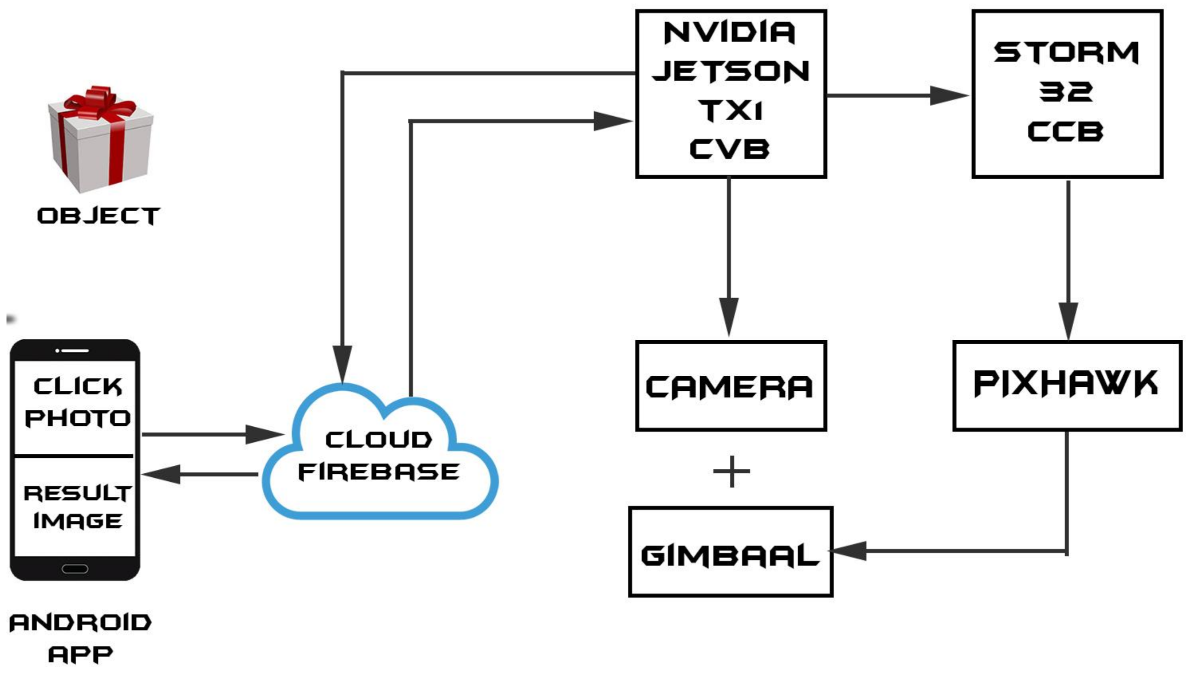
We will be using python3 for all our image processing and object detection, as the file is stored on cloud Database we will access the image from the firebase. For object detection, we would be using darkflow which is actually inter-mediate of tensorflow and yolo.

Then comes the part of taking pictures, we will be connecting CVB to the CCB (StoRM 32) which would be connected to the pixhawk 4 and will control the gimbal according to the PX4 instructions given to it which would be given from the CCB this would be initialized from our main program present in the CVB. Then the drone will move according to the instructions and then take the pictures.

Then our camera on the drone will capture the photos in all the 8 directions namely North, North-East, East, South-East, South, South-West, West, North-West. Then our algorithm would continue, if the object which is present in image given by phone is also present in any of the images then there would we be a message that the object is found, else if object is not found in any of the image then “object not found” message would be given. We are actually using the images (in round 2) as Raspberry pi has less computational power, but in actual project we would try to use the real-time video or record the video and process it on the Jetson TX1 (CVB).

Now as the object is detected, in suppose in one of the images, now we need to find the orientation or degree of the object. We would be doing this by creating the reference, that is each image is taken in the standard 8 directions, North, North-East, East, South-East, South, South-West, West, North-West so the center of each image will point to the direction of the standard directions. Then we will be finding theta(θ) which will be the angle with reference to the center (standard direction) in that image. So, the angle will be plus or minus theta(θ) then according to the number or the standard direction we can easily denote the angle of the object with respect to the north direction.

Now as the “object is found / not found “data (and if found then the image) will be saved or uploaded to the fire base. By the proper authentication, then in our android app from output tab there would be a refresh button which would read the data and access the image which is uploaded from the Computer Vision Board. On refreshing we would get the result.



D) Part wise reasons:

1) Android app and Database used:
2) CVB (Computer Vision Board):
3) Object Detection and image processing:
4) CCB, Gimbal:

1) Android app and Database used:

An Android app which has connection to firebase database.

Design of app:

Basically, App consist of Linear Layout having 3 buttons which are for

a) Clicking image by accessing camera

b) Saving image to gallery

c) Cropping the Image to the area of interest. (In place of labeling)

d) Sending Image to Firebase.

Activity of app:

To make camera app we have used Camera API of android and also, we can edit image (that is crop in our case). We have used all the necessary things for the connection between camera and firebase. Then after successful click of image app will show that image and we will store that image locally. Then comes the major part of sending that image and receiving output

so, for that this app will use firebase.

But Why FIREBASE?

FIREBASE is real-time database. Which allows us to update the data as per database. It has very vast scope and provide us a wide range of utility and application. This is why we are preferring firebase over other databases.

So, on firebase we have connected link to app by implementation of firebase modules and providing link json file. Then after creating FIREBASE and STORAGE REFERENCE variable to access and upload image in it, which can then be accessed from the CVB.

And finally, after processing all the images from CVB we will press refresh and receive output image and co-ordinates and orientation (with respect to the standard directions) of found object if found and show that or else print out object not found.

2) CVB (Computer Vision Board):

For our project, we would be using NVIDIA JETSON TX1.

Jetson TX1 is capable of delivering the performance and power efficiency needed for the latest visual computing applications. It's built around the revolutionary NVIDIA Maxwell architecture with 256 CUDA cores delivering over 1 TeraFLOPs of performance. 64-bit CPU's, 4K video encode and decode capabilities, and a camera interface capable of 1400 MPix/s make this the best system for embedded deep learning, computer vision, graphics, and GPU computing.

This is the main reason why it is used more in deep learning and computer vision projects so by using this we would get Good processing power to detect objects in the real-time video at high quality with more FPS and results will also be better and we can reduce the time of response or so-called delay in video data and the output.

3) Object Detection and image processing:

For object detection, we are using darkflow, is present on the *Github*. Which is actually implementation of YOLO v3, so is used widely for object detection purposes because it has outperformed all the currently existing object detection api's even the R-CNN and its variants giving more FPS and more accurate results and predictions.

For the actual working of the CNN in YOLO you can check it out on the GitHub.

The main thing in this is that Yolo is written in DARKNET (which is an open source neural network library which is in 'C' Language), so darkflow is basically a translation of yolo from darknet to TENSORFLOW.

So, in our code we have used OpenCV for image processing which is the easiest to work with while image processing.

Darkflow: https://github.com/thtrieu/darkflow

Yolo: https://pjreddie.com/darknet/yolov2/

https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection

Also, we would we TRAINING the YOLO model on our custom data set (some clicked images / google images / KAGGLE data-set and labeling the images and then creating tfRecords and training the model) for some of the objects which are very common but they are not included in the original yolo models like trees, buildings, etc so this would increase the prediction accuracy and also will increase the chances of detecting the objects correctly.

So, when we pass an image to the neural network we get name of the object, defining box of the object that is the (x, y) co-ordinates of the top-left and right-bottom corner and the confidence score of the object so we can keep a particular threshold for detecting the object so that it does not recognize the wrong object. Then we would be using those things to match the object from the image from phone to that from the camera.

While finding the angle there is one more twist in it we need the distance of the object with reference to the camera to find the vectors in the 3 dimensions. As we have the x and y pixel positions of the image form passing the image to the yolo convolutional neural network. So, we need the ratio to calculate the actual distance of the object from the camera so we will measure the actual size of the object / dimensions. Then we could compare the actual ratio so as to find the lateral and parallel distance from the camera. In this way, we can find the angle of the object found.

4) CCB, Gimbal:

i)CCB

ii)Gimbal

iii)Pixhawk

iv)MAVLinks

i) CCB: StroM32 gimbal controller

We will be using the StroM32 gimbal controller as a CCB the reasons behind this is that it is cheap and 3-axis brush-less gimbal controller and it is based on a **STM32 32-bit microcontroller**.

Works out of the box;

- Lightweight CNC aluminum alloy frame;
- Direct drive brush-less motors (two pieces of 2206 and one of C2805-145KV);
- 4 pieces of anti-vibration dumping balls;
- 3-Axis control signals (roll, pitch and pan);
- Storm32 brush-less gimbal controller (BGC);
- o323BGCTool GUI for configuration and firmware upgrade.

ii) Gimbals:

Gimbals are used to adjust the position of camera in all direction. Mainly we are using 2-axis brush-less gimbal because its provides the facility of 2-axis camera stabilization and anti-vibration .it also uses the flight controller system of drone hence it can be easily mounted on drone.

Types of gimbals

1) Glidecam iGlide, 2) Glide Gear Action Jib System AXJ-100, 3) VariZoom Stealthy Pro, 4) Feiyu G4, 5) ikan Fly-X3-Plus.

PX4 contains a generic mount/gimbal control driver with different input and output methods. The input defines how you control the gimbal: via MAVLink commands. The output defines how the gimbal is connected: some support MAVLink commands. Any input method can be selected to drive any output.

iii) Pixhawk -series board on px4.

There are Various Pixhawks available in the market.

Pixhawk 4 is an advanced autopilot designed and made in collaboration with Holybro and the PX4 team. It is optimized to run PX4 version 1.7.

we are using pixhawk4 because it is advanced version because it has inbuilt functions and also it is easy to handle.

iv) MAVLINKS

MAVLink is a very lightweight messaging protocol for communicating with drones (and between onboard drone components). MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with re-transmission.