OBJECT DETECTION AND TRACKING IN AUTONOMOUS DRONES

1. Abstract

In this study, the methods of deep learning based detection and recognition of threats, evaluated in terms of military and defence industry, using Raspberry Pi platform by unmanned aerial vehicles (UAV) are presented. In the proposed approach, firstly, the training for machine learning on the objects is carried out using convolutional neural networks, which is one of the deep learning algorithms. By choosing the Faster-RCNN and SSD MobileNet V2 architectures of the deep learning method, it is aimed to compare the achievements of the accuracy at the end of the training. In order to be used in the training and testing stages of the recommended methods, data sets containing images selected from different weather, land conditions and different time periods of the day are determined. The model for the detection and recognition of the threatening elements is trained, using 3948 images. Then, the trained model was transferred to the Raspberry Pi 4 Model B electronic board. The method of detecting and recognizing the objects is tested with military operation images and records taken by the UAVs via Raspberry Pi Camera V2 module. While an accuracy rate of %91 has been achieved in the Faster-RCNN architecture in object detection and recognition, this rate has been observed as %88 in the SSD MobileNet V2 architecture. These results are used later to develop applications for drones. The object identification program is developed in Python using the TensorFlow library. The authors have succeeded in implementing and testing this object identification module using the artificial neural network model SSDMobileNet V2 on the Raspberry Pi 3B+. The results in this paper demonstrate the potential of this module for further development in the future. Based on the simulation and real-world results, the authors showed that a good outcome is achievable with limited resources for the AI module. Along with a high precision object detection feature, this module can also estimate the distance and velocity of the "human" object with good accuracy. Besides, the paper also proposes several solutions to increase the performance and most importantly, the real-time feature of the developed module.

The training of artificial neural networks means finding the appropriate set of weights and deviation coefficients for each neuron in the network so that the final output is as closed as possible to the input references (i.e. expectation of the trainer), hence minimizing the objective function (cost function). To train neural networks we use back-propagation techniques in conjunction with optimization algorithms such as Gradient Descent [6]. To solve object detection and identification problems in computer vision, we use a convolutional neural network. The convolutional neural network has a different architecture than a normal neural network. The latter transforms input through a series of hidden layers. Each layer is a set of neurons and fully connected with the neurons on the previous layers. The final layer will represent the prediction of the network. Meanwhile, the convolution neural network (CNN) has neurons that are not fully connected to each neuron of the next layer, but only to a small region (some neurons). This section of the CNN is called the feature extractor, its function is to detect and extract different features in the image.

2. Introduction

$$y_k = \varphi(s_k) = \varphi(\sum_{j=0}^{m} w_{kj} x_j + b_j)$$

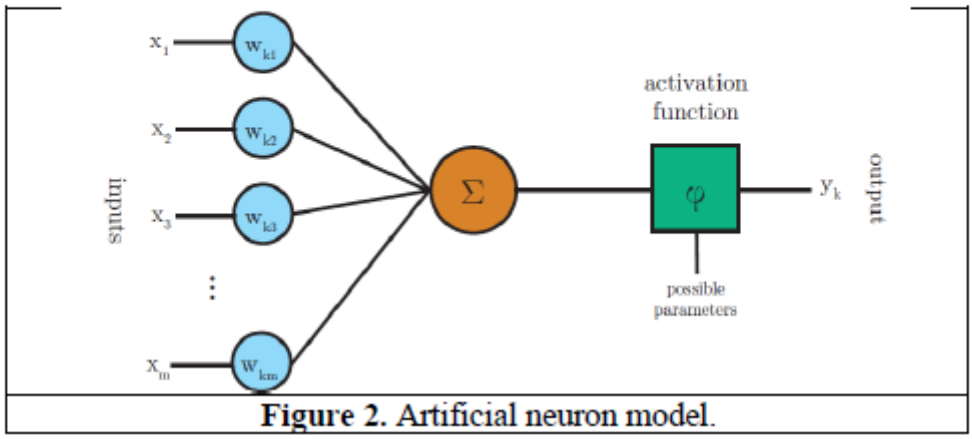


Figure 2. Artificial neuron model.

The training of artificial neural networks means finding the appropriate set of weights and deviation coefficients for each neuron in the network so that the final output is as closed as possible to the input references (i.e. expectation of the trainer), hence minimizing the objective function (cost function). To train neural networks we use back-propagation techniques in conjunction with optimization algorithms such as Gradient Descent [6].

To solve object detection and identification problems in computer vision, we use a convolutional neural network. The convolutional neural network has a different architecture than a normal neural network. The latter transforms input through a series of hidden layers. Each layer is a set of neurons and fully connected with the neurons on the previous layers. The final layer will represent the prediction of the network. Meanwhile, the convolution neural network (CNN) has neurons that are not fully connected to each neuron of the next layer, but only to a small region (some neurons). This section of the CNN is called the feature extractor, its function is to detect and extract different features in the image. Finally, a fully-connected flatten output layer will show the prediction result [7].

From the fundamental convolutional neural network AlexNet in 2012, there have been different convolutional neural network architectures developed so far (for example SSD and YOLO as shown in Fig. 3) Each network architecture has its own advantages in processing speed or resource consumption or accuracy (see Fig. 4).

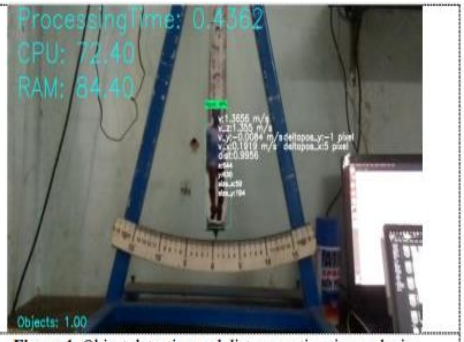


Figure 1. Object detection and distance estimation, velocity estimation results.

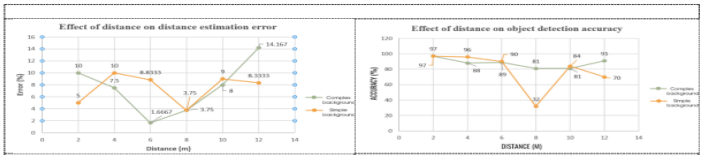


Figure 12. Effect of distance on distance estimation error

Figure 13. Effect of distance on object detection accuracy

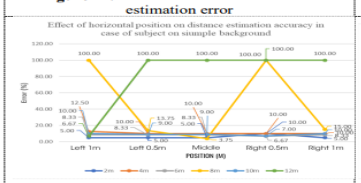


Figure 14. Effect of horizontal position on distance estimation accuracy in case of subject on simple background

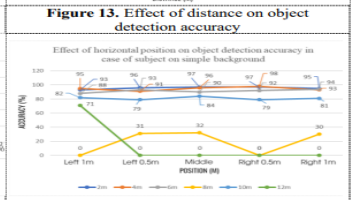


Figure 15. Effect of horizontal position on object detection accuracy in case of subject on simple background

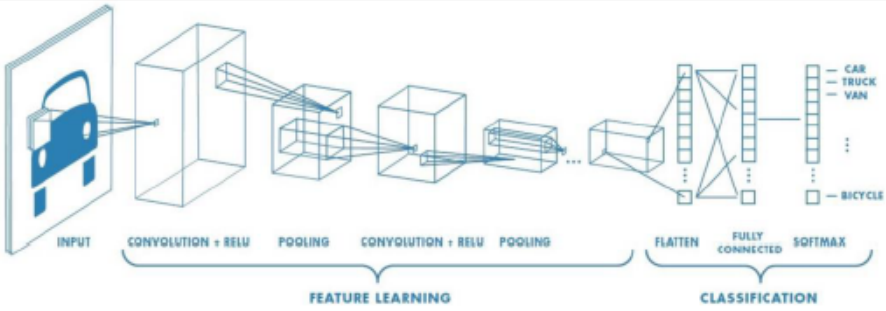


Figure 4. CNN structure for Object detection [2] .

2.2. Distance and velocity measurement

The distance estimation from the camera to the subject in the image is based on the image's formation taken through thin lenses (Fig. 5):

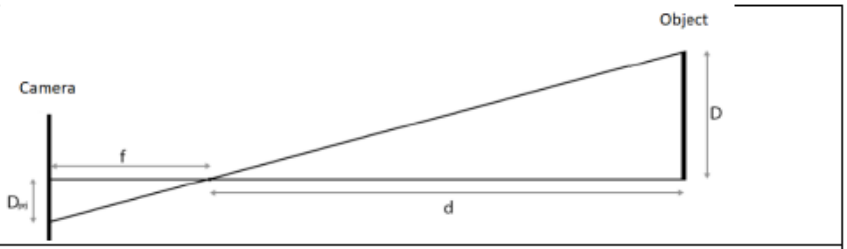


Figure 5. Image captured through a thin lens.

Where:

- f: camera lens focal (mm)
- d: distance between camera and object (mm)
- D: size of object (mm)
- D_{prj}: size of object in image (pixel)

The relationship between quantities is as follows:

$$d(mm) = f(mm).\frac{D}{D_{prj}} \quad (2)$$

$$D_{prj}(mm) = D_{prj}(pixel) * pixel\_to\_mm\_ratio \quad (3)$$

$$pixel\_to\_mm\_ratio = D_{prj}(pixel)/D_{mm} \quad (4)$$

In order to calculate the velocity of an object, it is necessary to obtain at least 2 frames has the same object and follow the steps (see Fig. 6):

a. Determine the position of the object in the image and find the box center coordinates in the first frame.
b. Determine the position of the object in the image and find the box center coordinates in the second frame.
c. Calculate the differences in position of box center between 2 frames.
d. Calculate the time difference between 2 frames.
e. Calculate the velocity by taking the ratio of the change in position of box center and time difference between 2 frames.

3. Literature Review

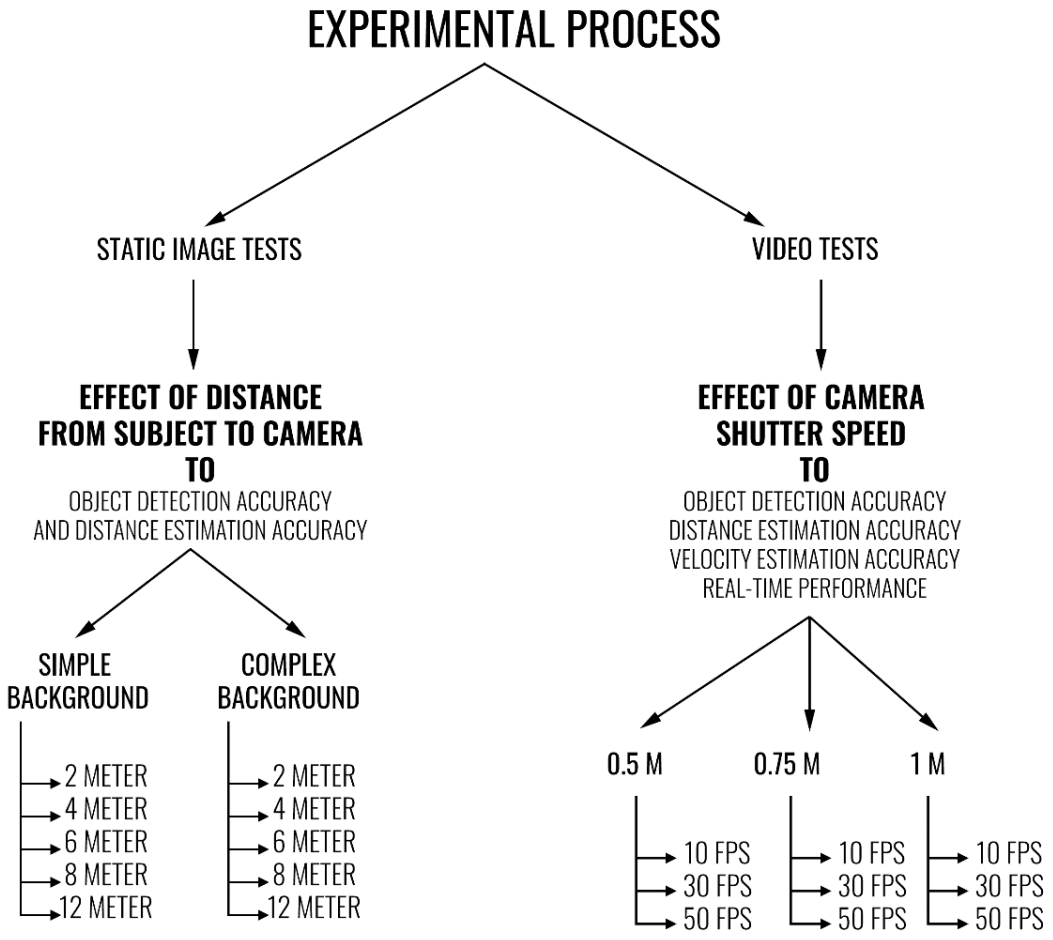
Introduction

The training of artificial neural networks means finding the appropriate set of weights and deviation coefficients for each neuron in the network so that the final output is as closed as possible to the input references (i.e. expectation of the trainer), hence minimizing the objective function (cost function). To train neural networks we use back-propagation techniques in conjunction with optimization algorithms such as Gradient Descent [6].

To solve object detection and identification problems in computer vision, we use a convolutional neural network. The convolutional neural network has a different architecture than a normal neural network. The latter transforms input through a series of hidden layers. Each layer is a set of neurons and fully connected with the neurons on the previous layers. The final layer will represent the prediction of the network. Meanwhile, the convolution neural network (CNN) has neurons that are not fully connected to each neuron of the next layer, but only to a small region (some neurons). This section of the CNN is called the feature extractor, its function is to detect and extract different features in the image. Finally, a fully-connected flatten output layer will show the prediction result [7].

From the fundamental convolutional neural network AlexNet in 2012, there have been different convolutional neural network architectures developed so far (for example SSD and YOLO as shown in Fig. 3). Each network architecture has its own advantages in processing speed or resource consumption or accuracy (see Fig. 4) while the To investigate the program's performance on object detection accuracy, distance estimation accuracy and velocity estimation accuracy, we perform 2 types of test: static image and video.

The experimental flow chart is as follow:



This test focuses on evaluating the ability of the program to detect objects for still images in order to obtain the optimal distance between the camera and the object where the program gives the best results. (lowest error in both vertical and horizontal direction)

Experiment process: (see Fig. 9 – 11)

a. Take photos of a person at distances d = 666f (≈ 2 m), 1,333f (≈ 4 m), 2,000f (≈ 6 m), 2,666f (≈ 8 m), 4,000f (≈ 12 m).

b. At each distance, take pictures at horizontal positions equal to 0.5 m from the center to the left and right.

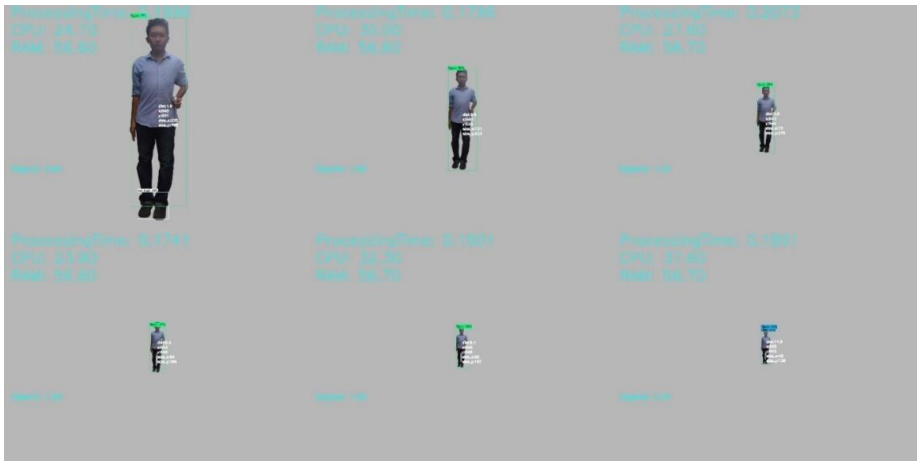
c. Calibrate the program's parameters at a distance of d = 666f (≈ 2 m) so that the distance estimation of the program is best.

d. Repeat step c for the remaining cases.

e. Make conclusions about the effects of object position in horizontal and distance to program's detection accuracy.

f. Make conclusions about the program's distance estimation accuracy at different distances.

The subject is a human photo (Fig.9). A simple grey background (Fig.10) is generated using Photoshop to compare the performance of the program with real life background (Fig .11) (complex background).



4. Methodology

Camera Calibration and code explanation

Any camera that you will be using, it needs to be calibrated as well. In case of using a raspberry pi on the drone, it is better to use the raspberry camera, because of its lightweight and easy connection to the Pi

For calibrating the camera you will be required OpenCV Library, it is one of the most widely used packages for video detection, motion detection, image recognition, and deep learning.

Follow the codes one by one to install and compile OpenCV 4.1.2 on Raspbian OS

```
$ chmod +x *.sh
$ ./download-opencv.sh
$ ./install-deps.sh
$ ./build-opencv.sh
$ cd ~/opencv/opencv-4.1.2/build
$ sudo make install
```

The full description of code lines can be found in the here

After installing OpenCV, do a test run by using below codes

```
$ wget "https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna_%28test_image%29.png" -O
lenna.jpg
$ python2 test.py lenna.jpg
$ python3 test.py lenna.jpg
```

Now, for detecting an Aruco marker by a Raspberry Pi camera Follow the procedure:-

1. Download the checkboard image and take a printout
2. Now you have to hold that checkboard printout image and take pictures using the raspberry pi camera.

For taking images, run save_images.py script:

```
$ python save_images.py
```

(Press spacebar for saving images) **The images will be saved in the same folder where the (camera_test.py) python script is located** Take almost 30 to 35 images of the checkboard and save in Raspberry Pi memory.

3. Images processing

After images are collected, they are processed by cameracalib.py file.

Command to run

```
cameracalib.py <folder> <image type> <num rows> <num cols> <cell dimension>
```

there folder - path containing images files, default "./camera_01" image type - the type of these images, default "jpg" num_rows - number of rows, default 9 num_cols - number of cols, default 6 cell dimension - dimension of cell in mm, default 25

Code explained

1. Do not process if the number of images is less than 9

```
if len(images) < 9:
    print("Not enough images were found: at least 9 shall be provided!!!")
    sys.exit()
```

2. For each image in folder find chessboard corners with findChessboardCorners method and draw them with drawChessboardCorners method from OpenCV. At the same time, store object points (ids of eacht image) and corners in an array:

```
3.  nPatternFound = 0
4.  imgNotGood = images[1]
5.
6.  for fname in images:
7.      if 'calibresult' in fname: continue
8.      #-- Read the file and convert in greyscale
9.      img     = cv2.imread(fname)
10.     gray    = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
11.
12.     print("Reading image ", fname)
13.
14.     # Find the chess board corners
15.     ret, corners = cv2.findChessboardCorners(gray, (nCols,nRows),None)
16.
17.     # If found, add object points, image points (after refining them)
18.     if ret == True:
19.         print("Pattern found! Press ESC to skip or ENTER to accept")
20.         #--- Sometimes, Harris cornes fails with crappy pictures, so
21.         corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
22.
23.         # Draw and display the corners
24.         cv2.drawChessboardCorners(img, (nCols,nRows), corners2,ret)
25.         cv2.imshow('img',img)
26.         # cv2.waitKey(0)
27.         k = cv2.waitKey(0) & 0xFF
28.         if k == 27: #-- ESC Button
29.             print("Image Skipped")
30.             imgNotGood = fname
31.             continue
32.
33.         print("Image accepted")
34.         nPatternFound += 1
35.         objpoints.append(objp)
36.         imgpoints.append(corners2)
37.
38.         # cv2.waitKey(0)
```

```
    else:
        imgNotGood = fname
cv2.destroyAllWindows()
```

42. if any pattern was found, calculate distortion coefficient and camera matrix with calibrateCamera method

```
if (nPatternFound > 1):
    print("Found %d good images" % (nPatternFound))
    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1],None,None)

    # Undistort an image
    img = cv2.imread(imgNotGood)
    h,  w = img.shape[:2]
    print("Image to undistort: ", imgNotGood)
    newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))

    # undistort
    mapx,mapy = cv2.initUndistortRectifyMap(mtx,dist,None,newcameramtx,(w,h),5)
    dst = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)

    # crop the image
    x,y,w,h = roi
    dst = dst[y:y+h, x:x+w]
    print("ROI: ", x, y, w, h)

    cv2.imwrite(workingFolder + "/calibresult.png",dst)
    print("Calibrated picture saved as calibresult.png")
    print("Calibration Matrix: ")
    print(mtx)
    print("Disortion: ", dist)
```

67. Save results to txt files

```
filename = workingFolder + "/cameraMatrix.txt"
np.savetxt(filename, mtx, delimiter=',')
filename = workingFolder + "/cameraDistortion.txt"
np.savetxt(filename, dist, delimiter=',')
mean_error = 0
for i in xrange(len(objpoints)):
    imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs[i], mtx,
dist)
    error = cv2.norm(imgpoints[i],imgpoints2,
cv2.NORM_L2)/len(imgpoints2)
    mean_error += error
print("total error: ", mean_error/len(objpoints))
```

Calibration computes a set of camera indicators (camera matrix, distortion) for a specific camera and should be done only once unless the camera's optic characteristics change.

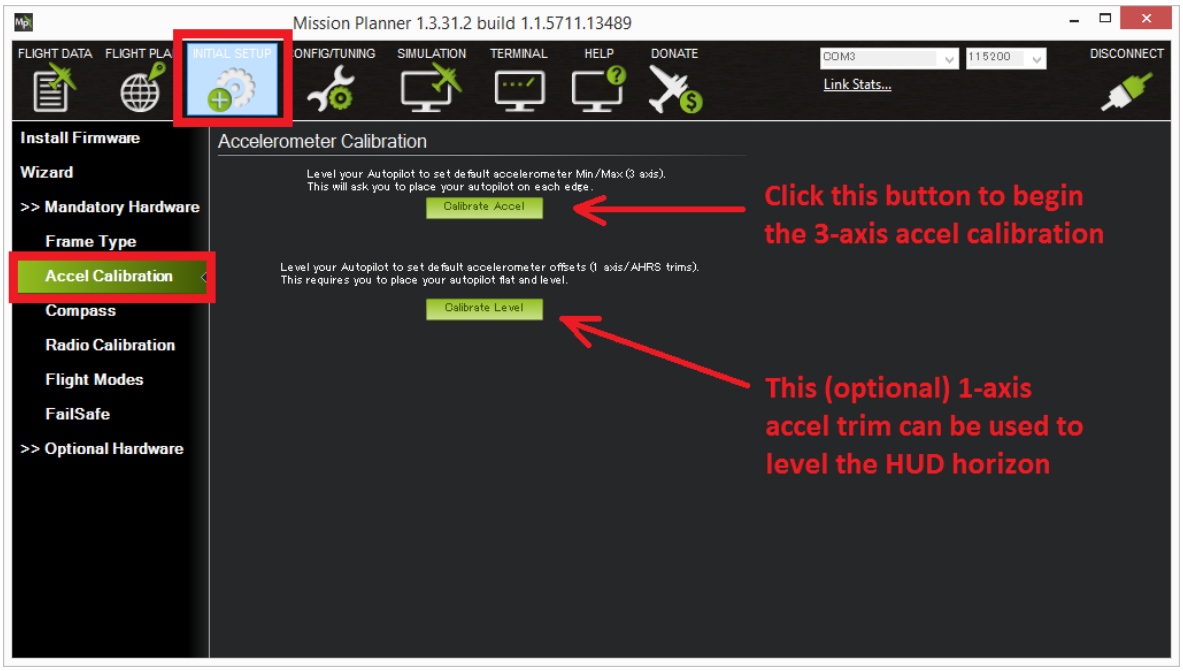
Drone part

Select the drone as per your requirement, after selecting the type of drone-based on the motors or propellers, weather it is quadcopter or hexacopter it has to be calibrated. It can be calibrated with the help of mission planner software To calibrate follow the steps one by one: -

1. Download and install the mission planner software from here (https://firmware.ardupilot.org/Tools/MissionPlanner/).
2. Click on INITIAL SETUP > Install Firmware > (based on your number of motors in drone select the ArduCopter version)
3.
4. After connecting from drone pixhawk to your laptop or pc click on CONNECT in mission planner
5.
6. Go to INITIAL SETUP click on Frame type (select the frame according to your drone).
7.
8. Followed to that in INITIAL SETUP click on mandatory hardware for calibrating: -
9. i. Accelerometer Calibration
10. ii. Compass Calibration
11. iii. Radio Calibration
12.

i. Accelerometer Calibration: For a drone to let know all the sides and directions, accelerometer calibration is required, it helps the drone to identify which side is left, right front and back.

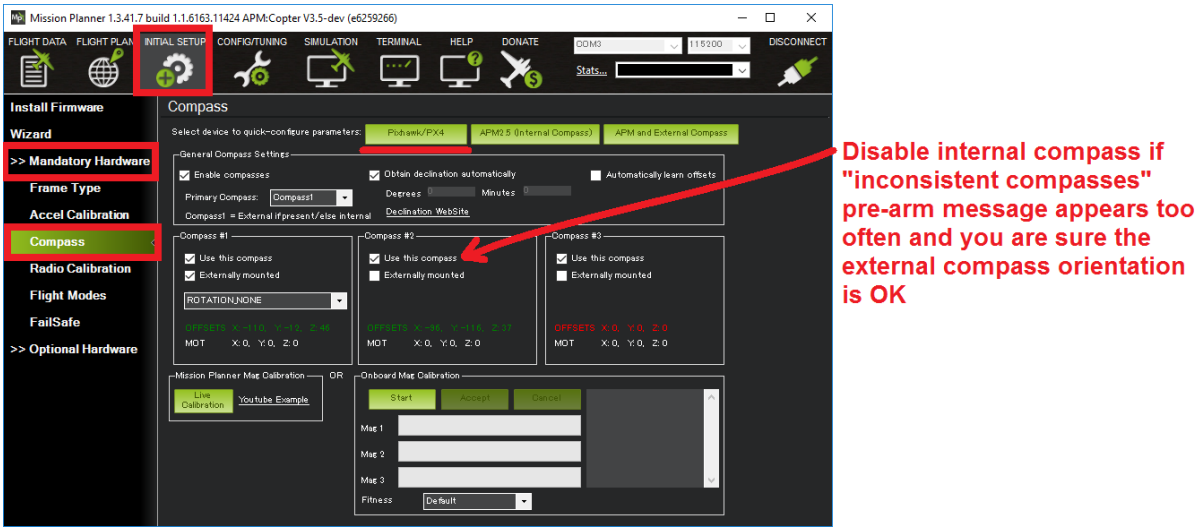
1. Click Accelerometer Calibration to start the calibration, then click Calibrate Accel.
2. Go according to the Guidelines of Calibrate Accel of setting sides of your drone. The setup will take around 5 to 10 mins.
3. In case required full instructions, please visit the website:- https://ardupilot.org/copter/docs/common-accelerometer-calibration.html



https://ardupilot.org/copter/docs/common-accelerometer-calibration.html

ii. Compass Calibration: - It is for the drone to understand all the directions.

1. To perform the onboard calibration of the compass on the drone:
2. Click on Compass in Mandatory hardware
3. Select the device Pixhawk/PX4, then click start, you will hear a beep sound once per second. This means the time has started to rotate the drone.
4. Hold the drone and rotate it in all the directions like a full rotation from the front, back, left, right, top and bottom and keep rotating until you hear three rising beeps, which means your calibration is successful. If you hear an unhappy failure tone, start the procedure again from step 3.
5. In case required instruction in detail, please visit the website: https://ardupilot.org/copter/docs/common-compass-calibration-in-mission-planner.html



https://ardupilot.org/copter/docs/common-compass-calibration-in-mission-planner.html

iii. Radio Calibration It is the calibration between controller and drone, to set the proper connection of the transmitter and the receiver in both the devices.

• Click on Radio Calibration in mandatory hardware, then click Calibrate Radio on the bottom right, press "OK", when prompted to check the radio control equipment, is on, the battery is not connected, and propellers are not attached. https://ardupilot.org/copter/docs/common-radio-control-calibration.html

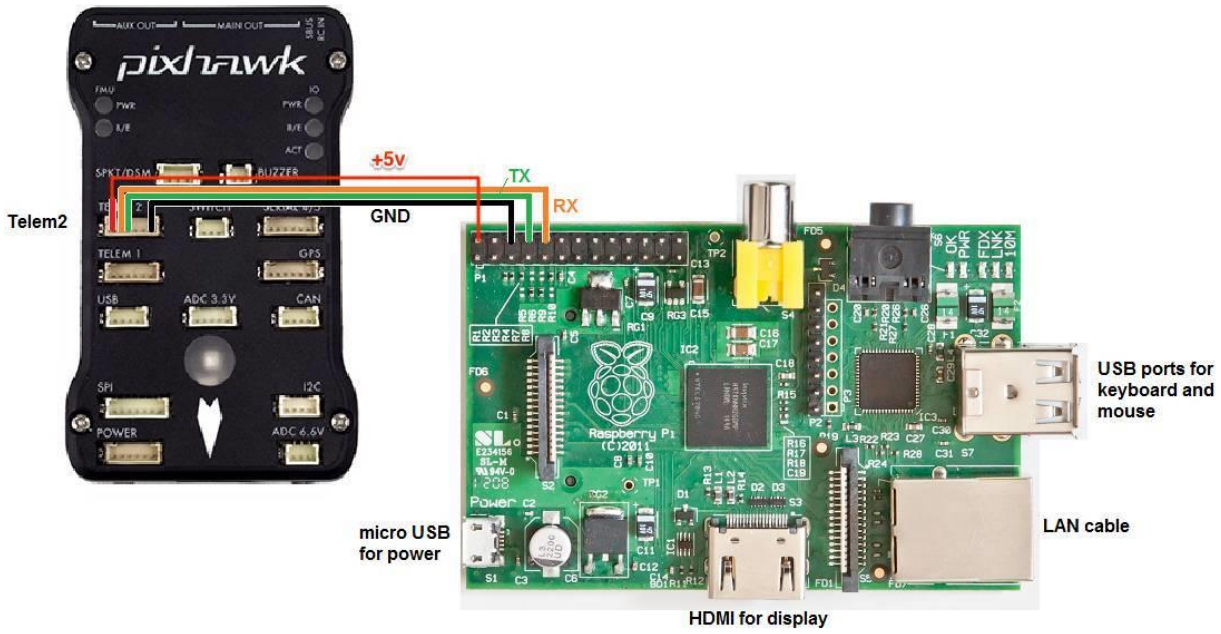
• Follow the complete instruction in detail given on the website: https://ardupilot.org/copter/docs/common-radio-control-calibration.html Select Click when Done • A window will appear with the prompt, "Ensure all your sticks are centered and the throttle is down and click ok to continue". Move the throttle to zero and press "OK". • Mission Planner will show a summary of the calibration data. Normal values are around 1100 for minimums and 1900 for maximums.

https://ardupilot.org/copter/docs/common-radio-control-calibration.html

Raspberry Pi to Pixhawk connection

Hardware Communication

Communication is done through the Telem2 telemetry port of Pixhawk. Ground, Rx (receiver), Tx(transmitter) and +5V should be connected to 4 pins of Raspberry Pi.



The required packages should be installed

```
sudo apt-get update    #Update the list of packages in the software center
sudo apt-get install python3-dev python3-opencv python3-wxgtk3.0 libxml2-dev python3-
pip python3-matplotlib python3-lxml
sudo pip3 install future
sudo pip3 install pymavlink
sudo pip3 install mavproxy
```