Object Tracking Using Quadcopter

Kalpesh Patil
Electrical Engineering
IIT Bombay

ABSTRACT

Object tracking is very important component of the computer vision system. It has multiple applications in video surveillance, navigation, 3D image reconstruction, robotics etc. It has attracted many researchers in this field. This project was aimed to continuously track an object using a movable camera mounted on a quadcopter and instruct it to move towards the object. This paper will give a brief idea about algorithms used for object tracking, hardware setup of movable camera on quadcopter and softwares required for this task.

Keywords: Object tracking, CamShift, Gimbal, Quadcopter, Lidar

INTRODUCTION

Problem Statement

The aim of the project was to track the location of the object continuously with the help of a movable camera attached on the quadcopter and instruct it to move accordingly. A 2-axis gimbal is used for carrying out the motion of camera. For the simplicity the problem statement was divided into three parts:

1. To track object in video feed
   An object will be selected on computer screen with the help of drawing a box using mouse around the object. Tracking algorithm will run on computer and track the position of the object in video feed.
2. To carry out motion of the camera and try to bring/keep the target object always at the centre of the frame
3. To instruct quad copter to move in desired location by desired amount of distance and follow the object to be tracked

Application and Motivation

There has been a lot work done on object tracking with the help of stationary camera. But this kind of tracking is not useful in the scenarios where object goes out of the frame of the camera. Therefore tracking should be done by a movable camera. Nowadays quadcopters are used for various applications. If autonomous object tracking and following is achieved by quadcopter, it will enhance their existing applications to the next level as well as open new arenas for quadcopters. It finds its application in aerial photography, traffic surveillance, navigation etc.

Solution and Contribution of Project

This project mainly contribute in two ways for existing problem.

1. Off board image processing
   Most of the object tracking quadcopters present today use on board computer for image processing and estimating position of the object. In this project we have provided video feed wirelessly to the off board system which directly generates the command after processing the input. This will allow users to use their own tracking algorithm as well as enables us to use computationally heavy algorithms.

2. Tracking using movable camera
   We have used gimbal for controlling the motion the camera. The use of moving camera will allow us to keep focus on the object continuously. We can theoretically have a very large total field of view due to possible rotations of the camera.

BACKGROUND AND LITERATURE REVIEW

Object Tracking Algorithms

Object tracking was very crucial part of the project. At present there are plenty of algorithms available for us to implement, but each of them has its own merits and demerits. Object tracking basically involves detection of object in each frame and predict its motion. There are various kinds of methods which are used for solving both the purposes. There are many literature surveys available on the internet comparing results of various tracking algorithms against the standard databases. Here I've enlisted few commonly occurring algorithms along with brief description. References are mentioned at the end of this report.

1. TLD (Track Learn and Detect)
   It is by far the most celebrated object tracking algorithm. It explicitly decomposes the long-term tracking task into tracking, learning and detection. The tracker follows the object from frame to frame. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future. Specially developed learning method (P-N learning) is used to estimates the errors by a pair of "experts": (i) P-expert estimates missed detections, and (ii) N-expert estimates false alarms. The learning process is modelled as a discrete dynamical system and the conditions under which the learning guarantees improvement are found.
2. Mean Shift
   Region of interest (object to be tracked) is defined in first frame. Then the mean shift algorithm is applied to separate the tracked object from the background in colour space. It shifts data to the local maximum of probability density function. Aryabhatta coefficient is used to calculate the difference between two distributions.
3. CamShift
   The Continuously Adaptive Mean Shift Algorithm (CamShift) is an adaptation of the Mean Shift algorithm. The primary difference between CamShift and the MeanShift algorithm is that CamShift uses continuously adaptive probability distributions (that is, distributions that may be recomputed for each frame) while Mean Shift is based on static distributions, which are not

updated unless the target experiences significant changes in shape, size or colour. Target distribution to be tracked is the object. Colour probability distribution at centre of the Mean Shift search window is calculated. Mean shift is iterated to find the centre and then window size is set accordingly. We have used this algorithm for our purpose. More details are given later.

4. Compressive Tracking
   Most of the online learning tracking algorithms have issues of lack of enough data to learn and drifting from original target due to learning false data. Compressive Tracking is an appearance model based on features extracted from multi scale image feature space with data independent basis. A very sparse measurement matrix is adopted to efficiently extract the features for the appearance model. Samples of foreground targets and the background are compressed using the same sparse measurement matrix. The tracking task is formulated as a binary classification via a naive Bayes classifier with online update in the compressed domain.
5. STRUCK (Structured Output Tracking with Kernels)
   Other approaches treat the tracking problem as a classification task and use online learning techniques to update the object model. However, for these updates to happen one needs to convert the estimated object position into a set of labelled training examples, and it is not clear how best to perform this intermediate step. By explicitly allowing the output space to express the needs of the tracker, STRUCK is able to avoid the need for an intermediate classification step. This method uses a kernelized structured output support vector machine (SVM), which is learned online to provide adaptive tracking. To allow for real-time application, it uses a budgeting mechanism which prevents the unbounded growth in the number of support vectors which would otherwise occur during tracking
6. Simple colour based object detection and tracking
   This method is the simplest amongst all. Given the range in HSV colour space it thresholds the image in every frame and detects the object. This is how location of the object is tracked. Applying filter above this gives better results. Although this lacks in adopting changing features of the object as well as differentiating object by features other than colour, it can be used for tracking mono coloured objects
7. Particle Filter based tracking
   Particle filter-based tracker maintains a probability distribution over the state (location, scale, etc.) of the object being tracked. Particle filters represent this distribution as a set of weighted samples. Each particle is a guess representing one possible location of the object being tracked. The set of particles contains more weight at locations where the object being tracked

is more likely to be. This weighted distribution is propagated through time using a set of filtering equations and particle with the highest weight or the weighted mean of the particle set are taken as the target at each time step.

DESIGN AND METHODOLOGY

Algorithm Outline

The final task of object tracking was divided into three subtasks; tracking in video feed, bringing object to the centre and moving quadcopter above the object. Since all three tasks are independent, we can try different algorithms for each one of them.

Algorithms for tracking in video feed are already discussed. We will use tilt angle of the gimbal to bring the object on X axis. Similarly yaw movement of quadcopter will try to bring the object on Y axis simultaneously. Set points for the quadcopter motion will be generated. All three motion will happen parallel to each other.

There will be three loops running simultaneously

1. Gimbal tilt loop which tries to bring the object on X axis
2. Quadcopter yaw loop which tries to bring the object on Y axis
3. Set point loop which will generate set points for the quad to move

Overall Setup Overview

List of major hardware and their brief description (detailed explanation along with pictures and circuit connections is given later)

i. Quadcopter
All the below mentioned components were installed on an already available quadcopter frame. It will be used for demonstrate tracking

ii. SJ4000 camera for capturing image
The camera attached to the gimbal below the quadcopter to capture video feed

iii. Tarot t2D gimbal
A gimbal is a device which allows independent motion of the device connected to it irrespective of the motion of the parent device where base of the gimbal is connected. In short it will allow us to control motion of the camera without interfering with quadcopter motion

iv. NavStik autopilot
The autopilot controller developed by NavStik having all necessary sensors was used as controller to the quadcopter as well gimbal motion.

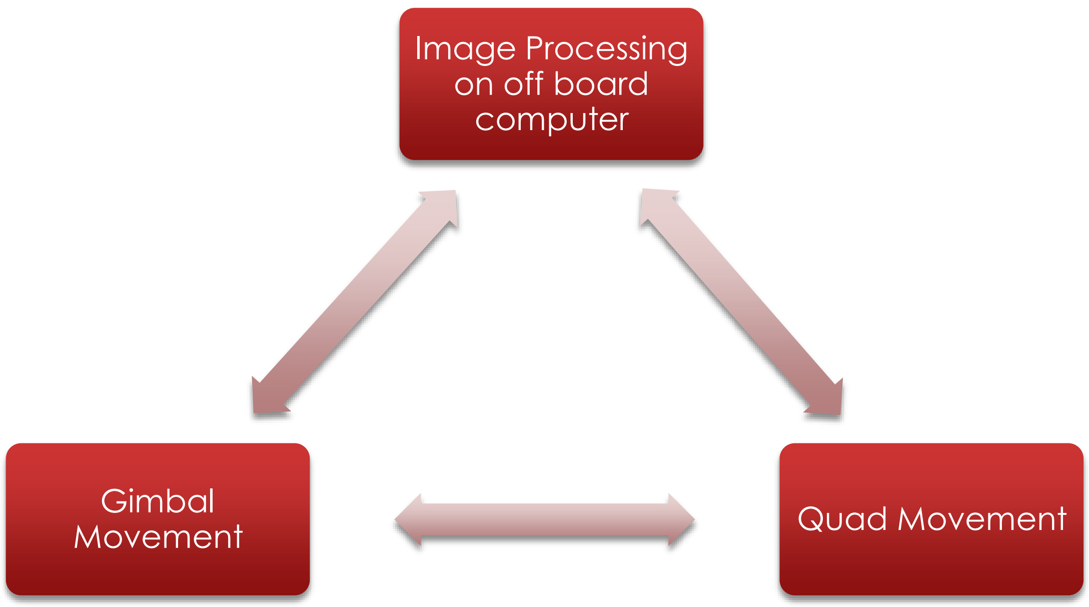
v. Interfacing board
Interfacing board is used for making connection from ESCs, gimbal and other sensors and devices to NavStik

vi. ZigBee modules
They are used for serial wireless communication

vii. Lidar
This a sensor used for measuring distances. It uses Laser beam instead of sound waves as in Sonar

viii. AV Tx-Rx
5.8 GHz AV transmitter-receiver pair was used to send video feed wirelessly from quadcopter to the ground

Major Blocks



There are three major components in the project with interconnection of data flow amongst themselves.

a. Image Processing on off board computer
This block of the project is supposed to get input from video feed of the camera. It will track object in each frame and generate X-Y coordinates of the object in the image frame. Using an algorithm those coordinates will be

used to calculate amount rotation required for gimbal to bring the object to the centre and also set points for the quadcopter motion

b. Gimbal Movement
   It will get instructions from the object tracking code and implement the required motion to bring the object to the centre
c. Quadcopter movement
   Along with gimbal, quadcopter will also receive instructions from object tracking code and implement the desired motion

(b) and (c) will result in motion of gimbal and quadcopter respectively which will change the position of the object in the video feed. Therefore these two blocks provide kind of a feedback to the object tracking code.

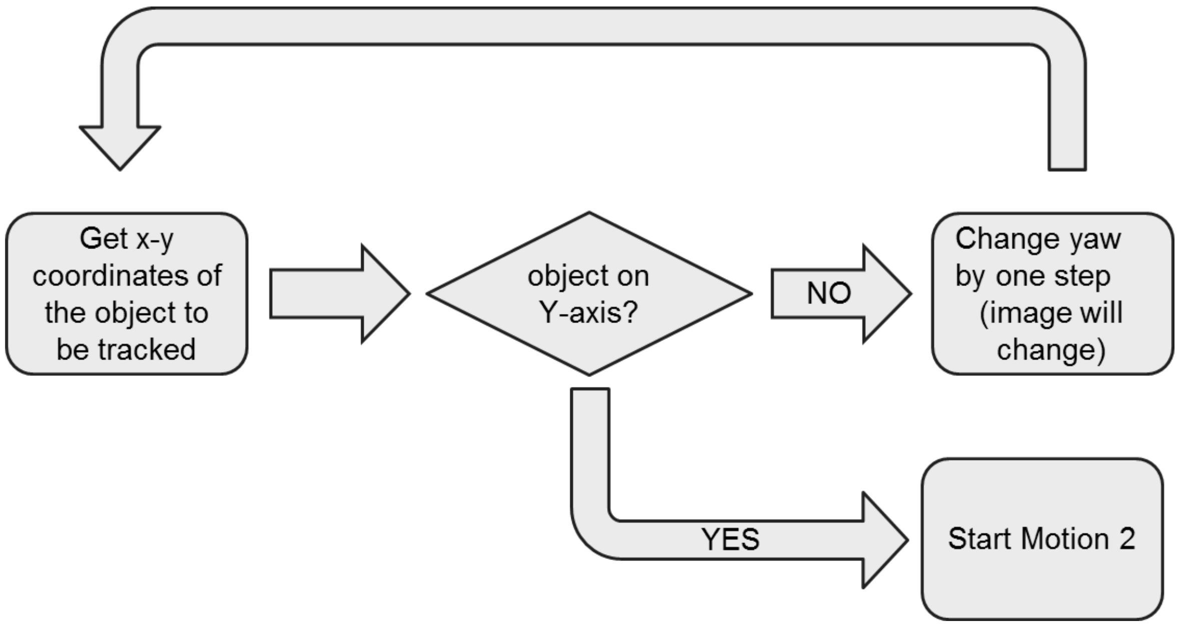
Algorithm Details

Initial attempts

1. Using tilt and roll of gimbal
   Initially we had decided to use to tilt and roll both inputs of the gimbal to bring the object at centre of the frame. Depending upon the amount of tilt and roll required and current height of the quadcopter we could have generated X-Y set points for the motion of the quadcopter. This requires simple trigonometric calculations and little knowledge of 3D vectors (projections of 3D objects on 2D plane). But the expressions we got were complex and computationally a bit harder. Also rotation of object along both the axes would have made tracking harder. The motive behind using this algorithm was to bring the object on X axis using tilt and on Y axis using roll. Even after knowing the angles rotated by gimbal it was hard to compute set points for the quadcopter. Therefore we had to give up on the idea of using roll for bringing the object on Y axis. Simple yaw movement of the quadcopter can also bring the object on Y axis.

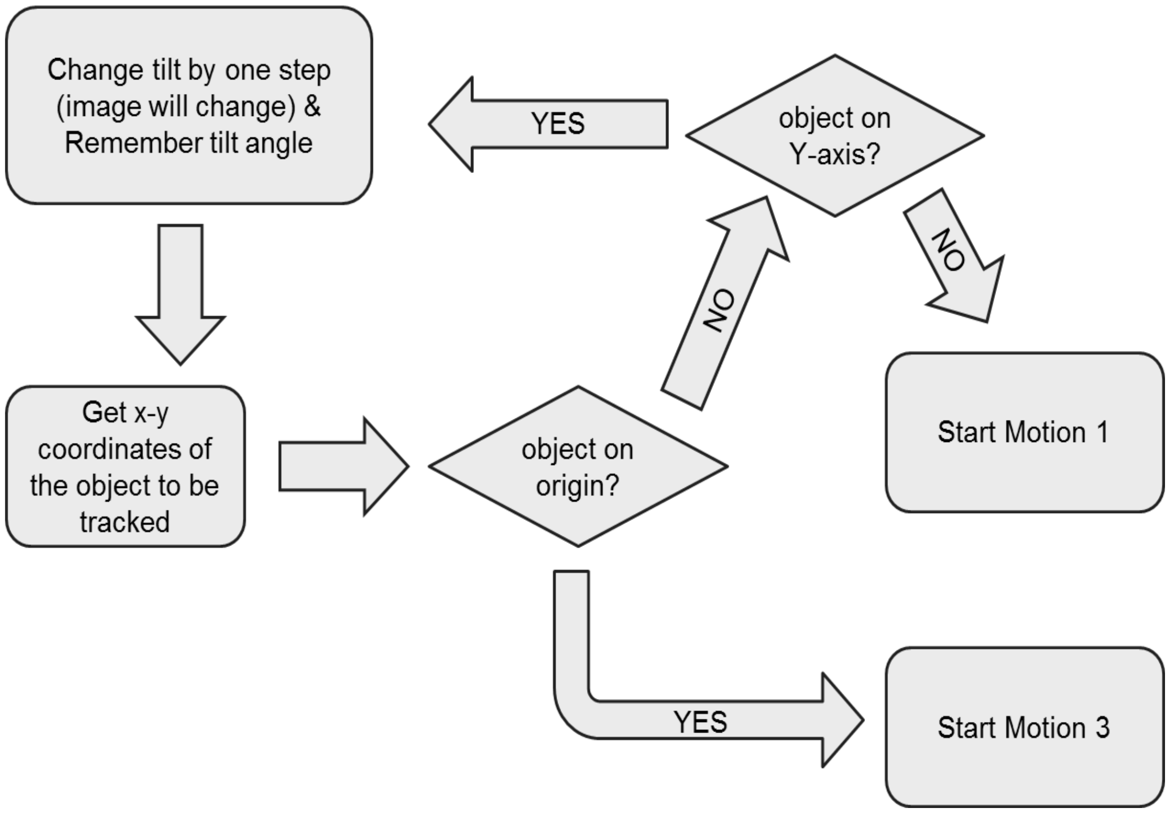
2. Dividing motion in three consecutive motions
   Here we had decided to use three motions starting one after the other for achieving our target. Motion 1 was about yaw movement of the quadcopter which would have brought the object on Y axis. Motion 2 was about tilt of the gimbal which would have brought the object on X axis. Motion 3 was about quadcopter motion which will make quadcopter hover above the object. The flow charts for all the three motions are as follows:

Motion 1:



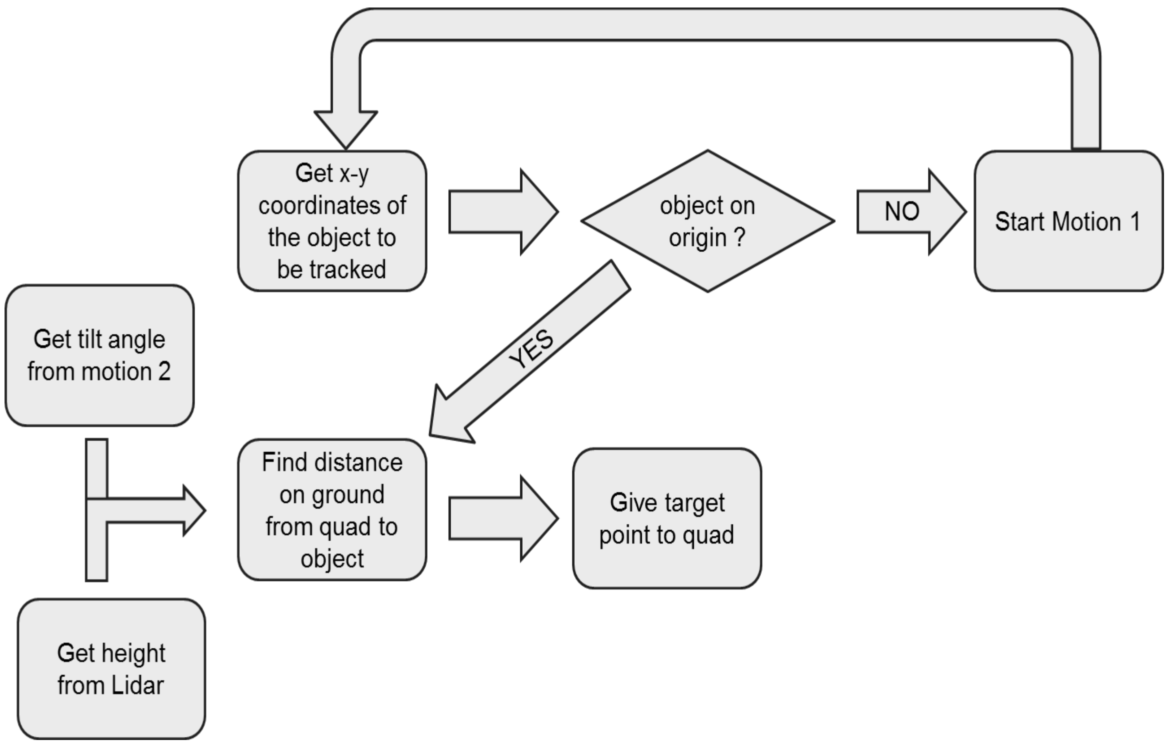
Get x-y coordinates of the object to be tracked
object on Y-axis?
NO
Change yaw by one step (image will change)
YES
Start Motion 2

Motion 2:



Change tilt by one step (image will change) & Remember tilt angle
YES
object on Y-axis?
NO
NO
Start Motion 1
Get x-y coordinates of the object to be tracked
object on origin?
YES
Start Motion 3

Motion 3:

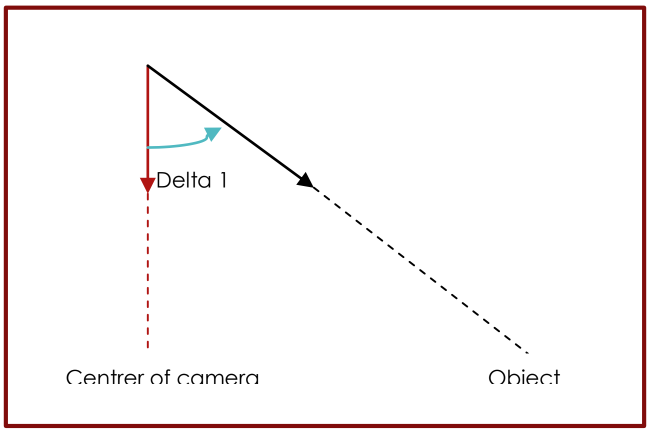


Get x-y coordinates of the object to be tracked
object on origin ?
NO
Start Motion 1
YES
Get tilt angle from motion 2
Find distance on ground from quad to object
Give target point to quad
Get height from Lidar

The problem with this algorithm was waste of time caused by waiting for the previous motion to end. All three motions are independent therefore they can run independent of each other. Therefore all these loops should run parallel to each other.

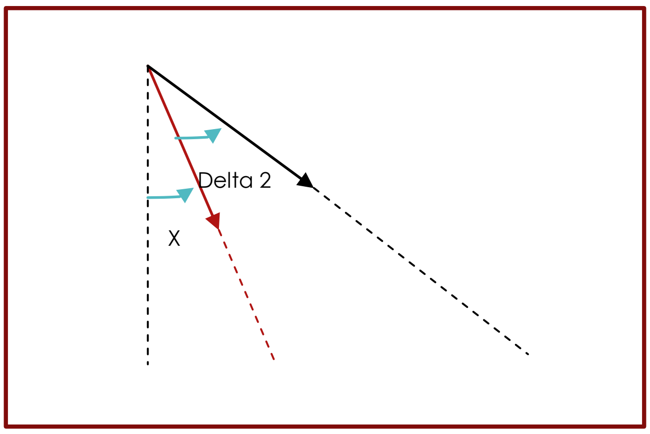
3. Using constant step sizes for the gimbal
   Even after making all three loops run parallel, we were assuming that we were still giving constant step for increment/decrement of the tilt angle, yaw angle and quadcopter set point. This makes the motion very slow. Therefore we had to come up with some method which would have generated inputs for the motions which were proportional to the error. To calculate error, we must know our current position values. Current yaw position and quadcopter position can be obtained easily, thus error in yaw and X-Y coordinates could have been calculated. But it wasn't possible to get current tilt angle position of the gimbal. We could not get feedback from gimbal. If feedback is not obtained from gimbal, we don't know how much angle has been covered up so far and cannot calculate other set points based on that angle values. The following example will explain the problem in brief. For simplicity we will assume that object is on Y axis and yaw is not required. Assume that we had kept our camera downward facing initially, which corresponds to certain PWM value. You can see the object in the image and depending upon its Y

coordinate we calculated angle increment required for gimbal, say Delta 1. We will give "initial angle 1 + delta 1" as tilt input for the gimbal.



Frame 1 (initial)

Till the next frame arrives gimbal would have been moved slightly and position of the object would have been changed in image frame, therefore new delta has to be calculated. We don't know how much gimbal has moved (gimbal doesn't provide current angle). Therefore when we are giving set point we can't use the tilt information.



X: Unknown angle by which gimbal has rotated

Frame 2 (after gimbal has rotated)

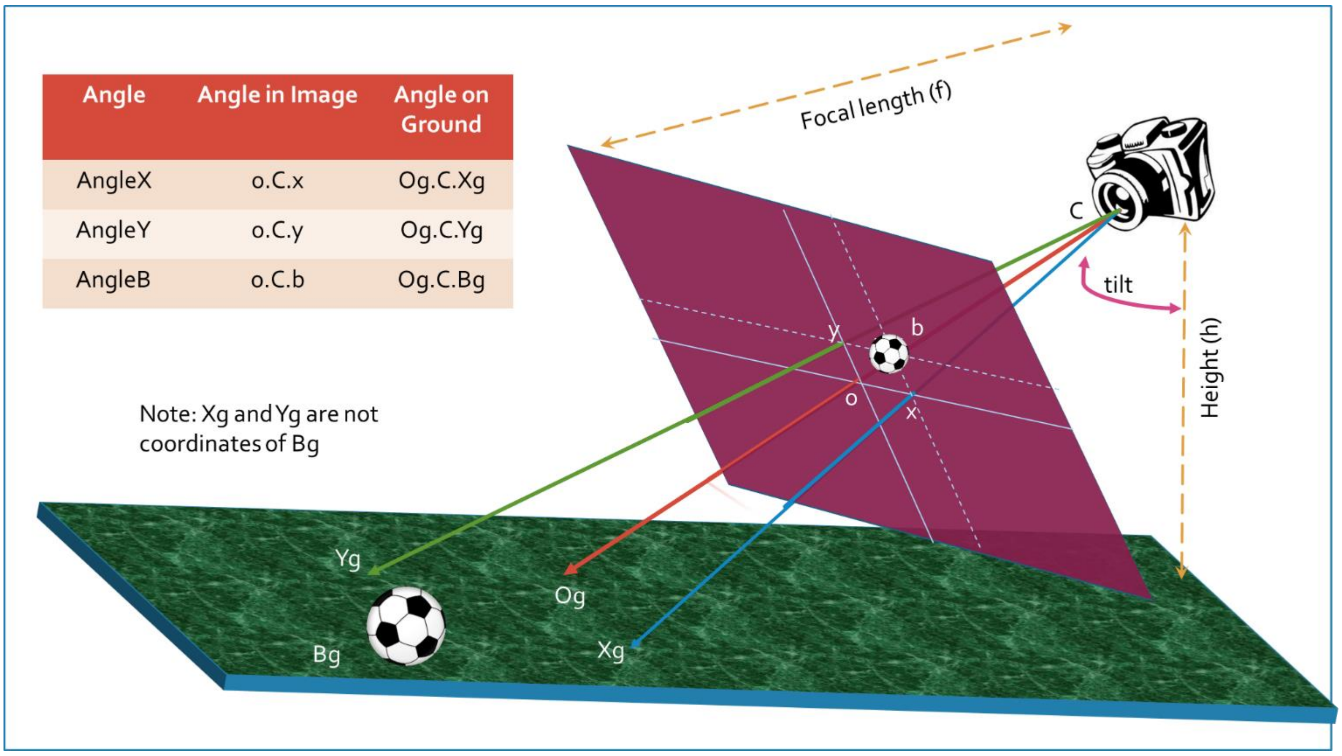
This makes it impossible to give correct set point for quad movement. Since we don't know the current angle, we won't be able to find correct angle to give input for calculating Y set point. The only way to make sure to know current angle was to use constant step size increment/decrement for gimbal. This makes sure that gimbal has been rotated completely till the next frame arrives and thus the angle we are using in code will be equal to the physical angle of the gimbal. We tried with various step sizes. This made the whole process of bringing the object very slow and inefficient. Although we could have got exact set points for quadcopter motion using this approach but it was very slow and hence dynamic tracking would have been impossible.

Improved Algorithm

We found out a way to find heading direction for the quadcopter. Once we know the heading direction, set point for the motion can be generated in steps. Thus instead of using steps for gimbal motion, we decided to use steps for quadcopter motion. We also found out way to convert X-Y coordinates in image to tilt angle and yaw angle.

This algorithms uses angles formed in camera as representative of the actual angles. The angles in image will converge to zero as actual angles converge to zero. Therefore input proportional to this angles can be given for the tilt and yaw. Since gimbal has its own PID controller, it will quickly achieve the given set point. The following image illustrates the concept of angles formed in image.

NOTE: Here we are assuming simple pinhole model for the camera (image forming at the focal length of the lens).



Angle	Angle in Image	Angle on Ground
AngleX	o.C.x	Og.C.Xg
AngleY	o.C.y	Og.C.Yg
AngleB	o.C.b	Og.C.Bg

If we could get feedback from gimbal (tilt), the following table can be used to calculate set points

Quantity	Value
angleX	atan(x*p/f)
angleY	atan(y*p/f)
Tilt	From gimbal feedback
Y set point	h*tan(tilt + angleY)
X set point	h*tan(angleX)/cos(tilt)
Yaw set point	angleX

h:	height
p:	pixel size
f:	focal length
x:	X coordinate in image
y:	Y coordinate in image

We need "tilt" only for X and Y set points. If we don't get feedback, we can use some other technique to give the set point. Above mentioned values are tested for tilt and yaw and they have produced positive results. The technique mentioned below for the quadcopter motion has not been tested.

Assuming an effective closed loop controller for tilt and yaw, most of the time object is going to remain at the centre of the image frame. We can generate a set point step for quadcopter motion which will depend upon X and Y coordinates of the object in image. When object is at the centre, we know quad has to move in straight direction. This movement of quad will create an offset which will be immediately removed by the motion of the gimbal and yaw of the quadcopter. This is how we will initiate quadcopter motion. Next part is to terminate its motion. Motion of the quadcopter will be terminated when it hovers exactly above the object. For that two criteria should be satisfied simultaneously:

1. Object at the centre of the image frame
2. Camera pointing exactly downwards

Since we are not getting any feedback from gimbal, we won't realise when camera is pointing downwards. Therefore a marker can placed below the quadcopter, which will come into view of camera when it is looking downwards. This will work as a flag to terminate the quadcopter motion. We can get this done by continuously looking for the flag in the desired pixel area of the frame. This is how motion of quadcopter is going to terminate. All these things are going to happen in a single closed loop which will eventually generate smooth motion for the quadcopter. For the set point generation we can use different methods. We can use a hybrid method which will generate constant set point when object is at the centre and will generate an offset above the constant step when it is not at the centre. The offset will be determined by X-Y coordinates in image. X set point also be modified by directly using angleX. There are plenty of possibilities to take this forward.

HARDWARE SETUP

Gimbal tarot t2D

It is a two axis motion controller specifically designed GoPro Hero 3 camera. It properly balances the mentioned camera. Datasheet for gimbal is easily available on the internet. It consists of brushless motors which control the tilt and roll angles of the gimbal. The specifications are as follows:

- Power Supply
  It can run on 7.4V – 14.8V DC supply, but 12V was recommended. We connected it directly to the main three cell LiPo battery
- Control inputs
  All three control inputs take PWM signals
  1. Mode selector
     It can switch between velocity mode (1000 PWM) and position mode (2000 PWM). The default mode is velocity mode, if left unconnected.
  2. Tilt input
     This similar to pitch in default conventions. The range is from -135° (1000 PWM) to 90° (2000 PWM). One should note that 1500 PWM will correspond to front facing camera while 1200 will correspond to downward facing camera
  3. Roll input
     Range is from -45° to 45° for the range of PWM as in tilt. Here also 1500 PWM signal will correspond to centre.

A mechanical joint was created already to attach it below the quadcopter. The tarot t2d gimbal which we were using was specifically designed to balance the weight of GoPro Hero 3 camera (SJ4000 is similar to original GoPro Hero 3 in terms of physical aspects).

- Precautions to be taken while using gimbal with camera
  1. Don't use gimbal without camera. It will unnecessarily spoil motors of gimbal. Also make sure that camera is properly situated in the given slot and weight is balanced properly.
  2. Don't let the base of gimbal move when it is calibrating to find initial orientation (yellow light is glowing). Blue light indicates calibration is over
  3. Don't forget to connect grounds for gimbal pins.

SJ400 camera

The camera provides multiple options for resolution and fps. We should use video with lesser resolution for faster image processing

Camera model: SJ4000

Focal length: 2.99 millimetre

Sensor: N201405270ZV02

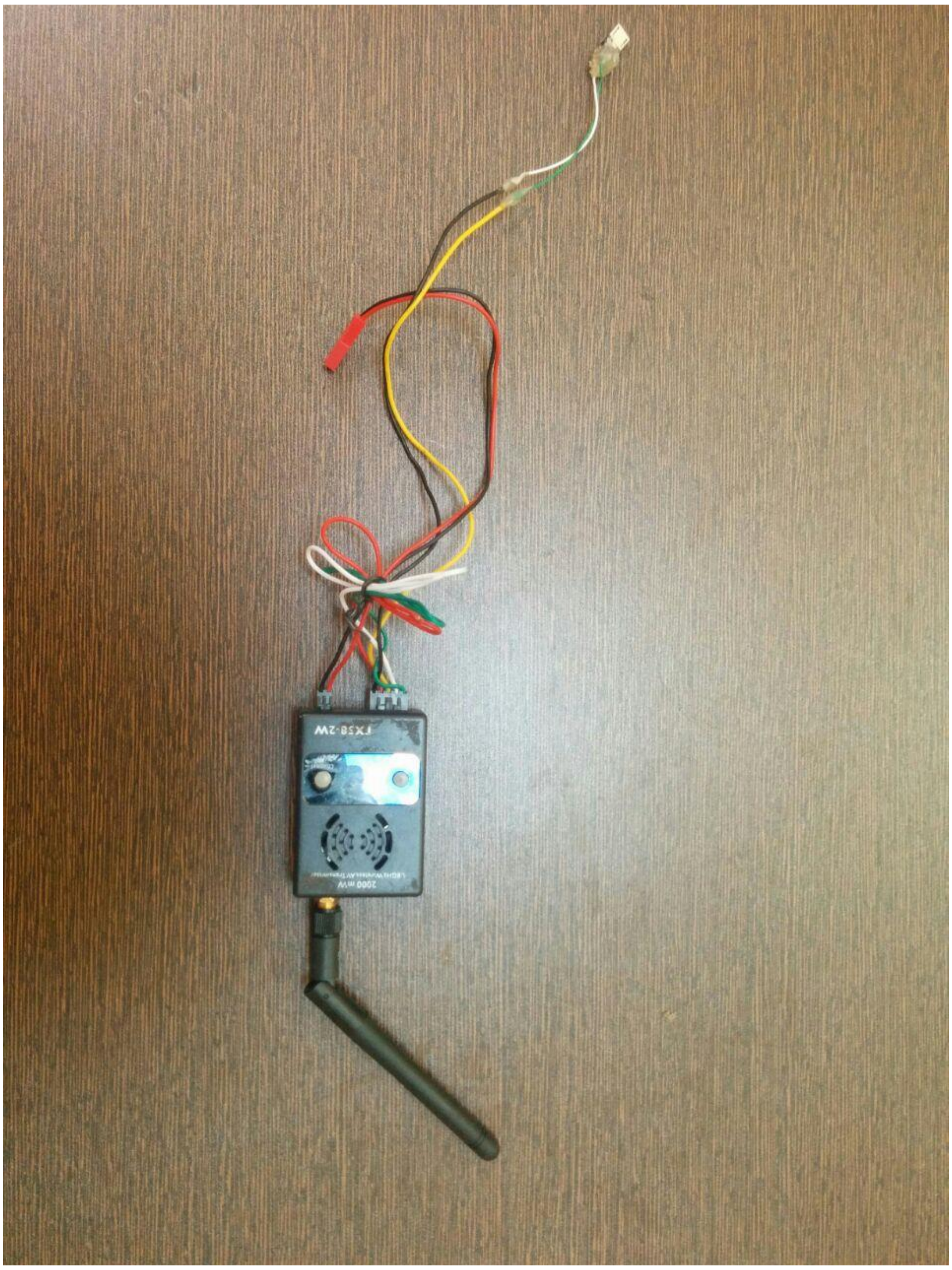
Pixel size: 2.2 micrometre

NOTE: Camera should be kept in TV mode for wireless video transmission

Video transmission assembly

It consisted of three parts and intermediate cables. We used a 5.8 GHz AV transmitter-receiver pair for sending video feed from on board camera to off board computer for processing. A micro USB connecter was made to connect transmitter to the camera. Video feed was taken from receiver on computer using an intermediate device called EasyCap. The output of the receiver is in some different video format. Therefore you need convert it to PAL using settings in VLC, then only you will be able to use the video feed from receiver in OpenCV functions. Whenever transmitter is connected using EasyCap, the format has to be changed to PAL. The alternative for this is to make some changes in OpenCV libraries which is more complicated than changing the format using VLC.

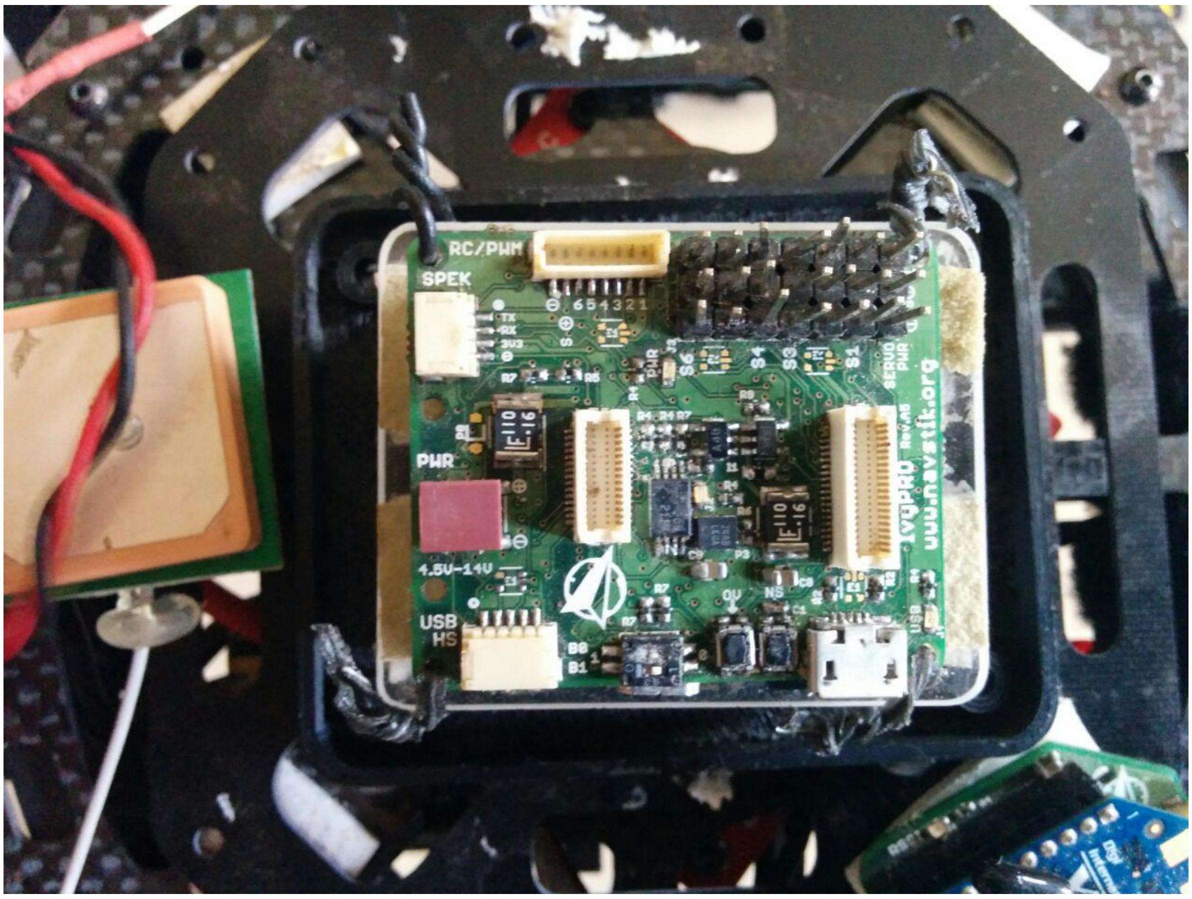
Make sure glue bulge formed on microUSB pin after soldering with transmitter wires is small enough to be accommodated in the slot for camera on Gimbal. Also it should be thick enough to maintain connections even after collisions with surrounding

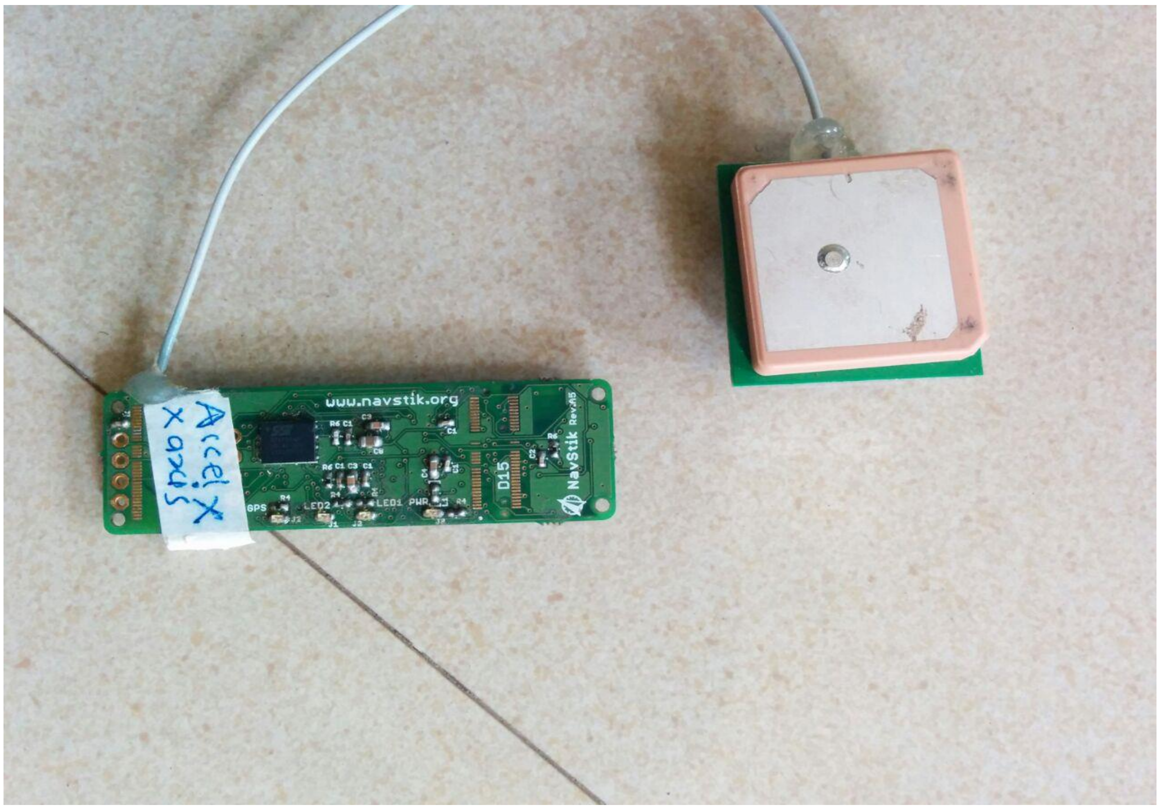


NavStik autopilot with interfacing board

NavStik autopilot works as the brain of the quadcopter and controls all the devices connected to it. There are several in built sensors present in NavStik autopilot chip like barometer, magnetometer, gyroscope and accelerometer. NavStik also provide slot for connecting GPS antenna. GPS module is already installed on NavStik. GPS antenna should be connected/soldered to use GPS.

We used pro-board as interfacing board to connect ESCs, gimbal, Lidar, XBee etc. devices to NavStik. We used GPIO pins 1 to 4 for connections of ESCs. 5th and 6th pin were used as PWM output pins for roll and tilt input of gimbal respectively. Also Lidar was tested using one of the GPIO pins (removing gimbal)


RC/PWM
SPEK
PWR
4.5V-14V
USB HS
lvyPRO
www.navstik.org

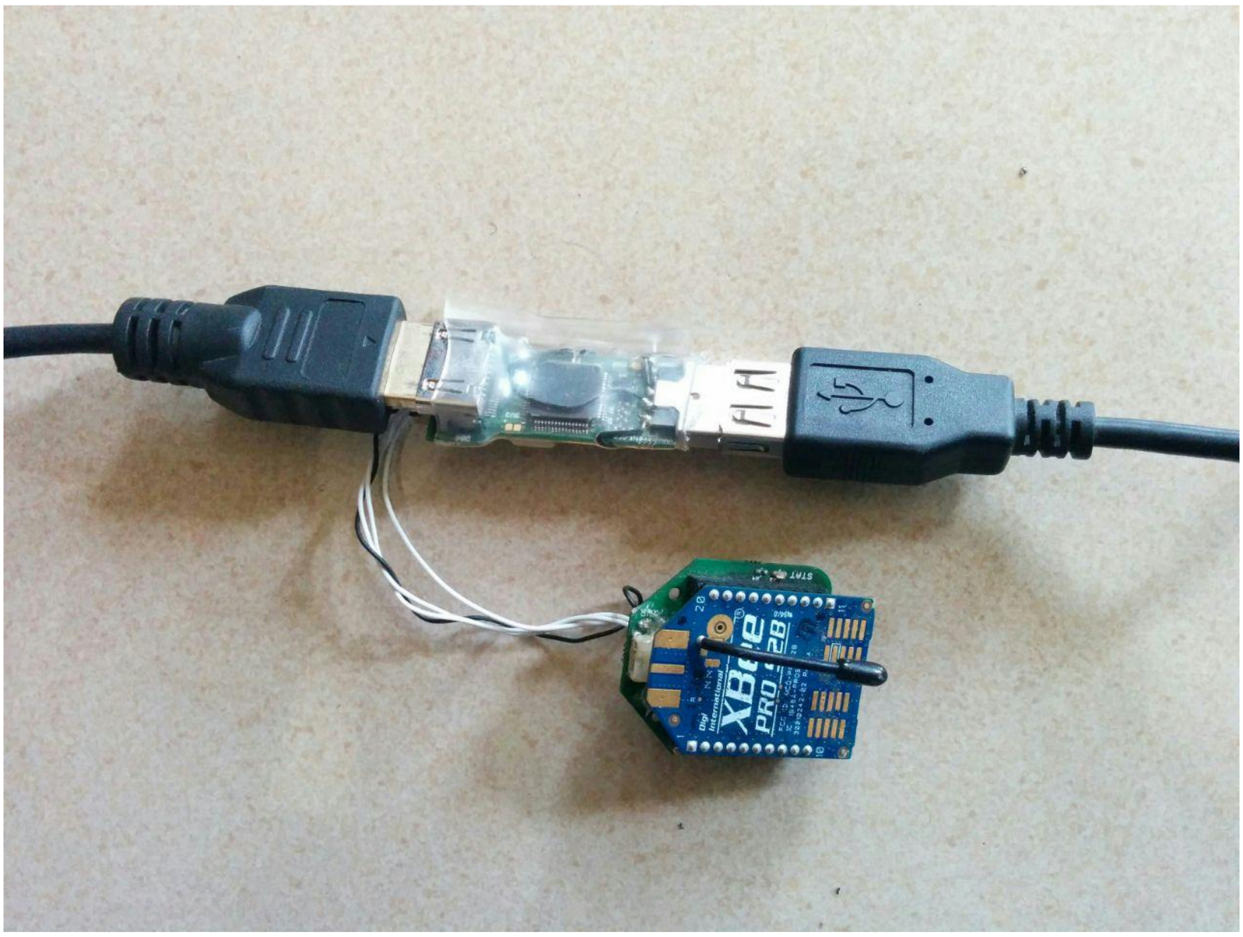

Accel X axis
www.navstik.org
NavStik

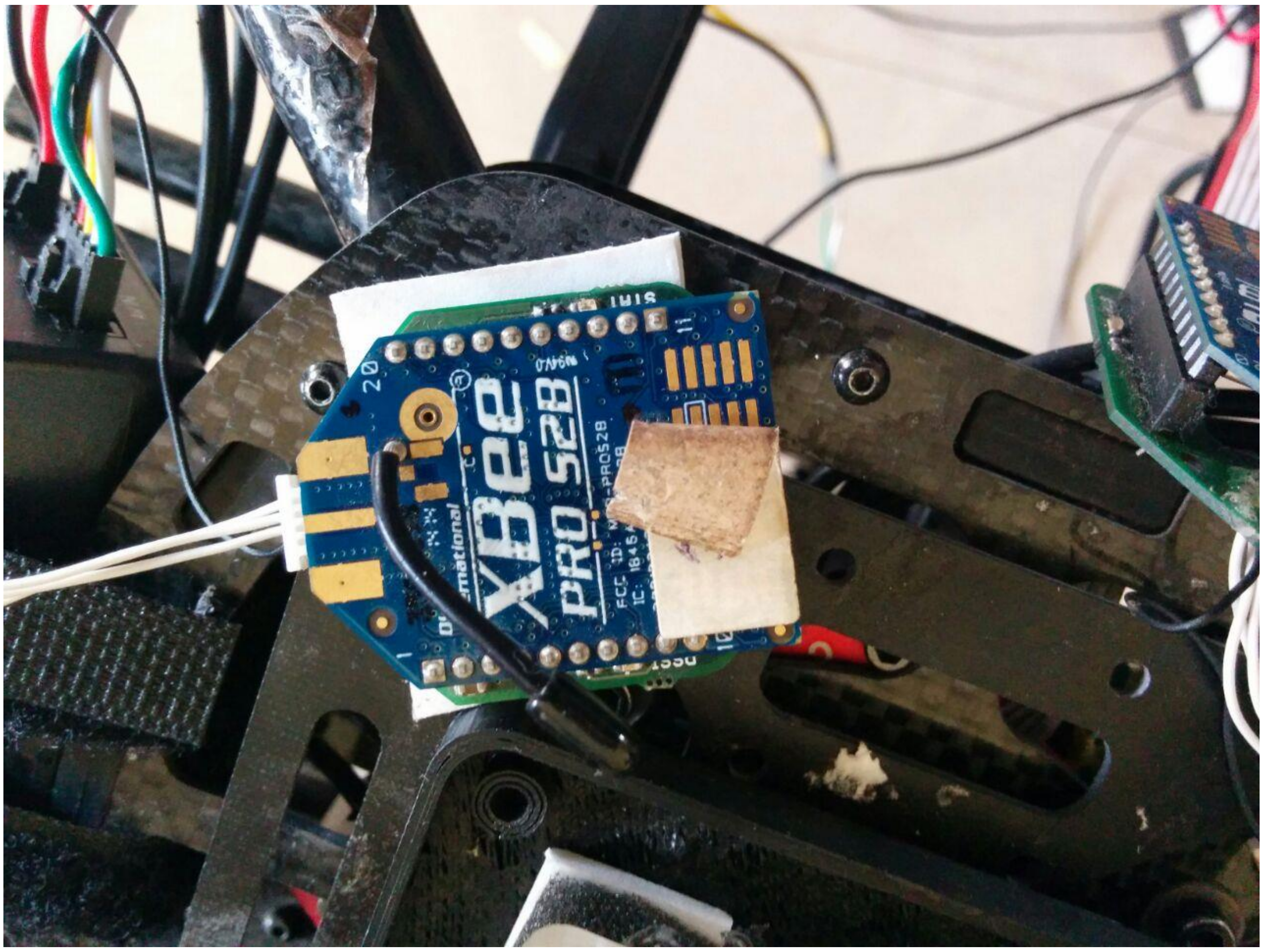
XBee modules and IvyGS board

XBee modules are used for wireless communication. We used mavlink protocol for sending commands and data from off board computer to quadcopter and vice versa. Before using XBee modules, they should be paired and set to the required baud rate which we are using for sending or receiving messages. Otherwise messages will not be sent or received. This is done using XCTU software. A separate blog has been written for detailed information on configuring XBee modules

IvyGS board acts as interface for connecting XBee module to the computer. Also it allows user to access terminal of NavStik, known as “nsh” terminal using HDMI cable.

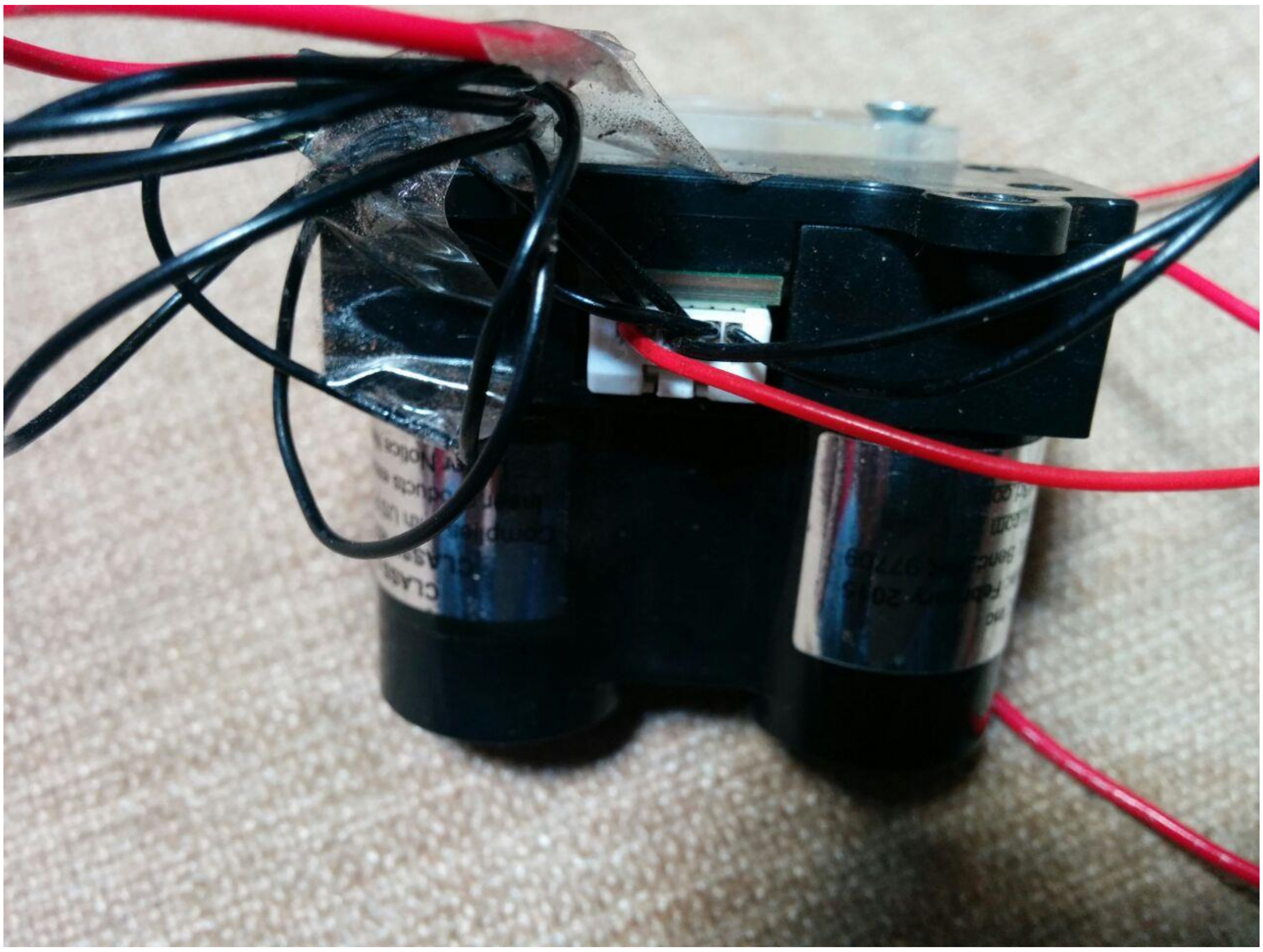
CAUTION: IvyGS board is very delicate and should be used with proper care while connecting it to USB slot in computer. An USB extension can be used to avoid potential damage





Lidar-lite

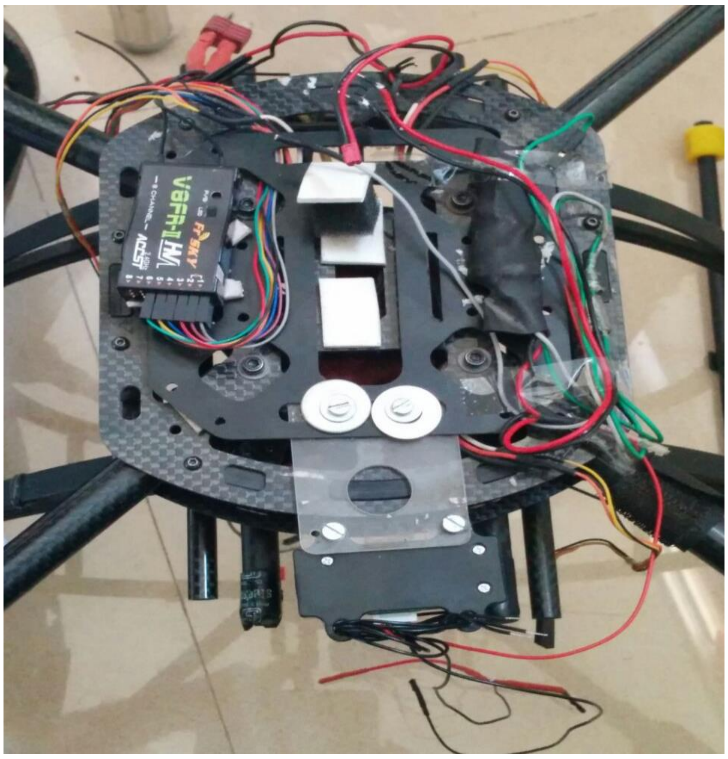
Lidar is a device used for calculating distances. It works on the same principle of Sonar, but uses laser beam instead. We had decided to use it for finding height of the quadcopter (we ended up not using the Lidar and used inbuilt barometer). The Lidar should be powered with IC 7805 for its required value of 5V. It can generate output in two different forms, I2C and PWM. We used PWM output terminals of Lidar. The PWM output pin should be divided into two terminals, say +ve and –ve. The +ve terminal can be connected directly to the GPIO pin and –ve terminal should be connected using a resistor to ground of GPIO pin set. Some changes in firmware has to be made to accommodate Lidar (instructions are given on NavStik wiki page for connecting Sonar. Lidar can be incorporated similarly if external connections are made properly). After that proper calibration is required for converting PWM signal to height.



Turnigy 9XR RC

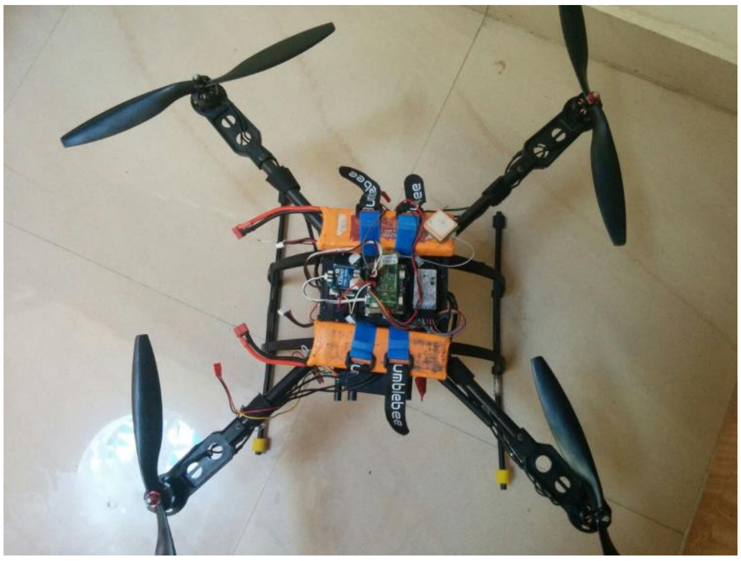
This is used for manual control of the quadcopter. Before using, it should be calibrated in NavCon/QGC.

V8FR Radio receiver



Pin 1 to 6 were used for quadcopter control. Pin 7 was used for mode control of Gimbal (which was kept at 1000 PWM to set it in position mode)

Quadcopter Assembly



Larger 12" propellers were used to lift the quadcopter. Two batteries were required to provide sufficient amount of current

SOFTWARE ARCHITECTURE

Object tracking code

Initially we had decided to use OpenCV implementation of TLD due to its versatility and robustness. Initial tests were conducted by connecting SJ4000 camera using USB cable to the computer which didn't have lag generated in processing due to sending mavlink packets wirelessly and lag in wireless video transmission. Therefore using TLD was the best option. The details of performances of different algorithms are summarised later. When we actually tested TLD on moving quadcopter, it wasn't able to track object efficiently. The lag was huge and processing was also quite difficult. We tried tweaking with the code here and there, still it wasn't able to produce the desired result. Although it is still the most famous algorithm for tracking using stationary camera, it is not much effective for tracking using the moving camera. The main difference is number pixels changing their values, which causes recomputation of many parameters required for learning. In stationary camera there is not much change in background even if object moves. But in case of moving camera background changes continuously until the object is brought to the centre of the image. Along with this, TLD was not able to track the object when it changes its shape suddenly. When we are using moving camera, the view of the object changes, thus the shape of object in frame also changes which makes it difficult for TLD to track objects. The detailed description of working of TLD and its implementation is given in codes submitted along with the folder.

Due to all the difficulties in tracking using TLD, we had to shift to an algorithm which has lesser complexity than TLD at the same ability to track objects even when angle of view is changed. Considering both the factors CamShift algorithm present on OpenCV was the best choice for tracking simpler mono coloured objects.

CamShift algorithm is an extension to the mean shift algorithm. The Mean Shift algorithm works well on static probability distributions but not on dynamic ones (changing size of the object). CamShift is based on principles of the Mean Shift but also a facet to account for these dynamically changing distributions. CamShift is able to handle dynamic distributions by readjusting the search window size for the next frame based on the zeroth moment of the current frames distribution. This allows the algorithm to anticipate object movement to quickly track the object in the next scene. Even during quick movements of an object, CamShift is still able to correctly track.

CamShift works by tracking the hue of an object. The video feed is converted to HSV space. The CamShift works in the following manner.

1. Initial location of the 2D search window was computed (by manually selecting the object)
2. The colour probability distribution is calculated for a region slightly bigger than the mean shift search window
3. Mean shift is performed on the area until suitable convergence. The zeroth moment and centroid coordinates are computed and stored.
4. The search window for the next frame is centred around the centroid and the size is scaled by a function of the zeroth movement.
5. Repeat step 2 to step 4.

In the OpenCV implementation of CamShift, the function used for scaling is proportional to the square root of the zeroth moment.

Once the location of the object is tracked, coordinates of the centroid are computed. These coordinates generate tilt and yaw set points as mentioned in the algorithm.

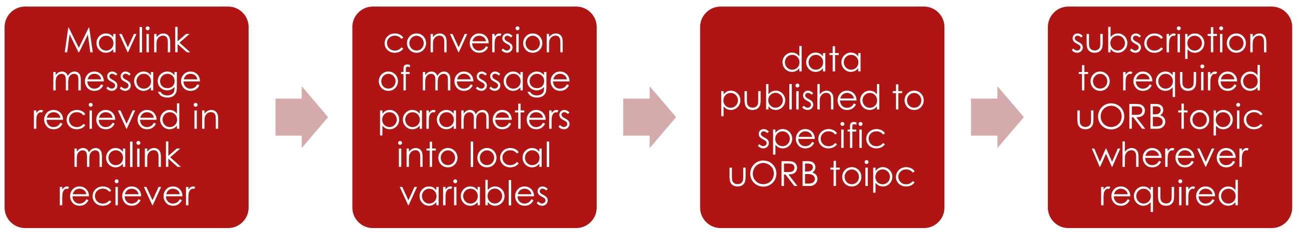
Firmware Changes

We used the latest "pandapilot_v4" firmware for NavStik. To control gimbal with NavStik, changes should be made in the firmware. We decided to use pin 5 and pin 6 for roll and tilt PWM inputs for gimbal. This needs isolation of these pins from default configuration. After that these pins are to be configured to PWM outputs in "pwm_output.c" file of the firmware. After initialisation, separate thread should be run for continuously receiving mavlink commands containing tilt and roll data from NavStik.

To incorporate Lidar in firmware, a separate file for driver should be created. Ideally pin 5 should have been used as PWM input from Lidar and pin configuration in firmware has to be changed accordingly. Z (vertical height) was being estimated properly after increasing weightage of Lidar with respect to barometer, but z-velocity was continuously drifting in one direction. Even after changing the estimator little bit we were able to achieve only oscillating z-velocity from the estimator. Due to all these difficulties in using Lidar data in velocity estimator we had to stop using Lidar and rely on barometer completely. The proper weights should be tuned in estimator for using Lidar.

Mavlink message packet for Gimbal

Mavlink has command for sending PWM values. We used “PWM_DO_SET_SERVO” mavlink command in “command_long” packet of mavlink to send PWM values for tilt and roll of the gimbal from the script running on off board computer to quadcopter. Before using mavlink data one should look for complete flow of information of the received mavlink packet in firmware.


Mavlink message recieved in malink reciever
conversion of message parameters into local variables
data published to specific uORB toipc
subscription to required uORB topic wherever required

External Softwares/packages required

XCTU

This software is used for configuring XBee modules. They have launched their latest GUI version for linux. The detailed information on configuring XBees using XCTU is given on the blogs uploaded

NavCon

This is used for monitoring quadcopter motion on ground. It is similar to QGC (Q Ground Control Station). This softwares provides facilities to calibrate gyroscope, magnetometer and accelerometer. It monitors data sent through various mavlink packets. It can also set and write permanently values of various parameters of quadcopter motion and other sensors.

OpenCV library

We used OpenCV 3.0 library for running various object tracking algorithms. It comes with large number of in built classes and functions which makes it easier to use for image processing and can be ported easily in our code. Compilation and linking instructions are easily available on internet

Logic Analyser

The software along with the hardware allows user to monitor PWM signals. It helps in debugging as well as checking our script before actual implementation with hardware

Minicom

This terminal based linux package helps in monitoring and sending data on various ports of the computer. “nsh” terminal of NavStik can be accessed from minicom. Also other devices connected via USB can be accessed through minicom.

QT 5.0

This is used as an IDE for creating various GUI based softwares. NavCon requires installation of QT 5.0 or more for its installation on computer.

NavStik Pandapilot_v4

This is firmware which is to be flashed into NavStik autopilot. Whatever changes you make in firmware will reflect in operation of NavStik firmware

Mavlink library

This is a library written in C++ used for wireless communication. Mavlink protocol should be followed for sending and receiving commands/data wirelessly.

RESULTS AND TESTS EVALUATION

Object tracking algorithms

Performance of different object tracking algorithms was tested against various parameters like occlusion, changes in shape and orientation, execution speed, fast movements etc. The table summarising the results is given below.

Algorithm	Occlusion	Change in shape or orientation	Robustness	Fast movement of object	Execution Speed
TLD	very good	good	good	very good	good
STRUCK	very good	very good	very bad	very bad	depends
Particle Filter	good	bad	bad	bad	bad
Compressive Tracking	bad	bad	bad	good	very good
CamShift (simple colour based)	very good	very good	very bad	good	very good

Test specifications:

C++ codes implemented using OpenCV library

Linux 14.04, 64 bit, Intel i5 system

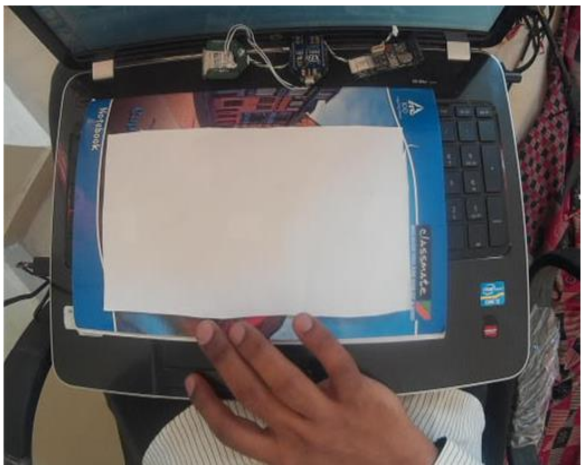
SJ4000 camera connected using USB

NOTE: Robustness is in the sense of ability to track different variety of objects (shapes, colour patterns, textures etc.)

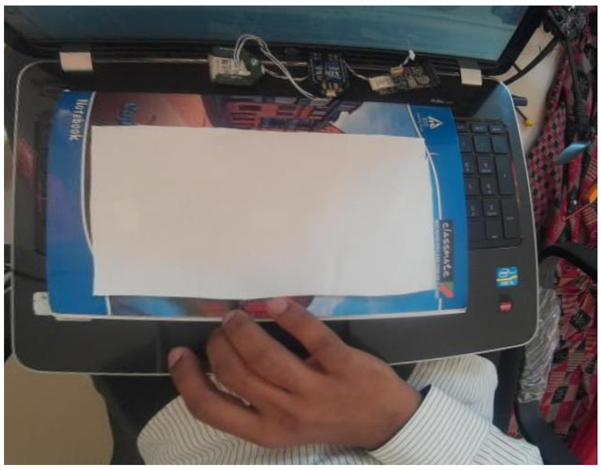
Undistortion Test

Image formed by wide angle cameras is generally distorted. Therefore it is necessary to remove distortion. Although the effect is not so significant when object size small, but it is recommended to remove the distortion.

To remove distortion, camera parameter matrix is first computed using standard chessboard technique. That matrix is fed into in built transformation function of OpenCV. The output is undistorted image. The images below are captured one after the other. One of them is processed while other isn't.



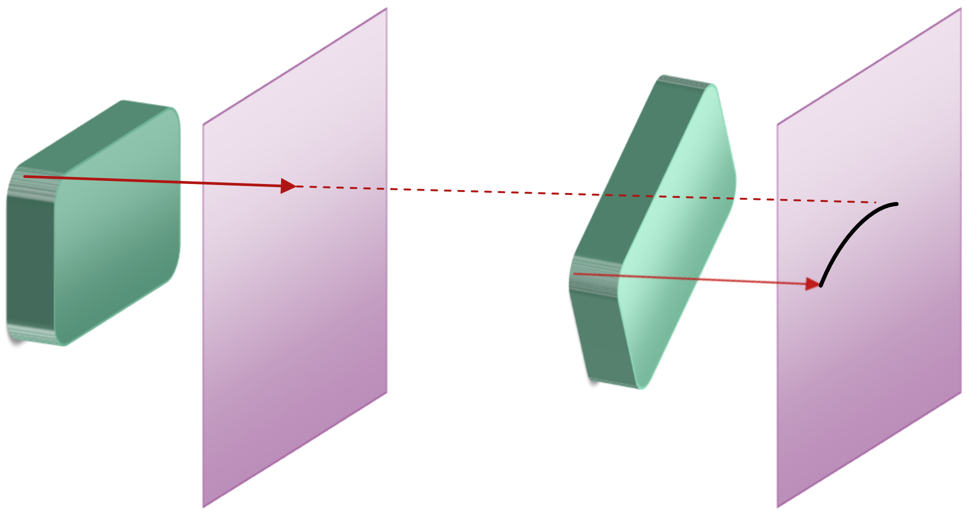
Original (distorted) image



Undistorted image

Gimbal Calibration

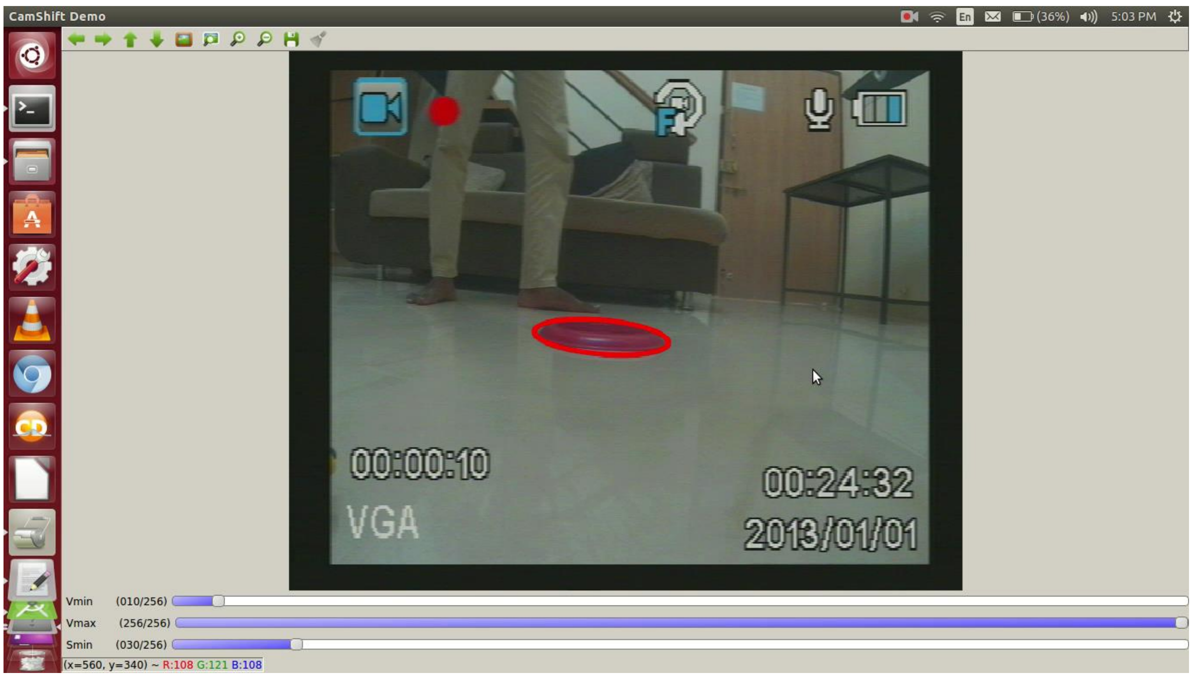
We had to map PWM values to the tilt angles of the gimbal. To make sure that gimbal is working properly as mentioned in data sheet, this test was conducted. A marker pen along the axis of rotation of gimbal was attached to one of the corners of the camera. Manual commands were given using "nsh" terminal of NavStik to set the PWM output of pins to the desired values. This caused rotation of camera and thus arcs were drawn by the marker on the paper attached perpendicular to the axis of rotation. Using simple geometric constructions angles corresponding to the given PWM values were calculated. Thus proportionality factor was found out. This was used wherever we wanted to convert angles to the PWM values



The black arc indicates the amount of rotation caused due to sending PWM commands. Red arrow is the marker attached to corner of the green camera. This picture illustrates the test for roll. Similar setup was made for tilt also.

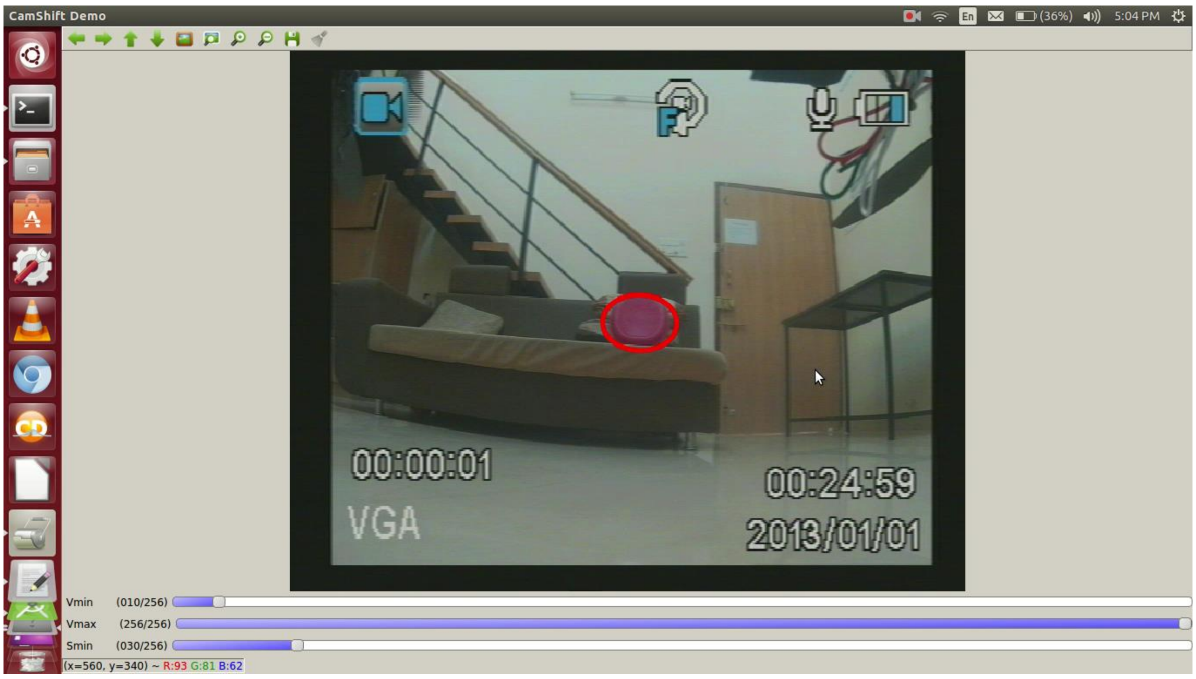
On ground gimbal tilt test

The test involved bringing object on X axis by changing the tilt of the gimbal. We used constant step size initially. In later stages we gave angles formed in image as set point for the tilt. The angles were exaggerating actual angle, therefore we had to use a proportional constant. We used only proportional logic (integral and differential terms were absent) therefore camera used to oscillate a bit near centre. This can be avoided by putting PID controller instead of only proportional logic.



```
Vmin    (010/256)
Vmax    (256/256)
Smin    (030/256)
(x=560, y=340) ~ R:108 G:121 B:108
```

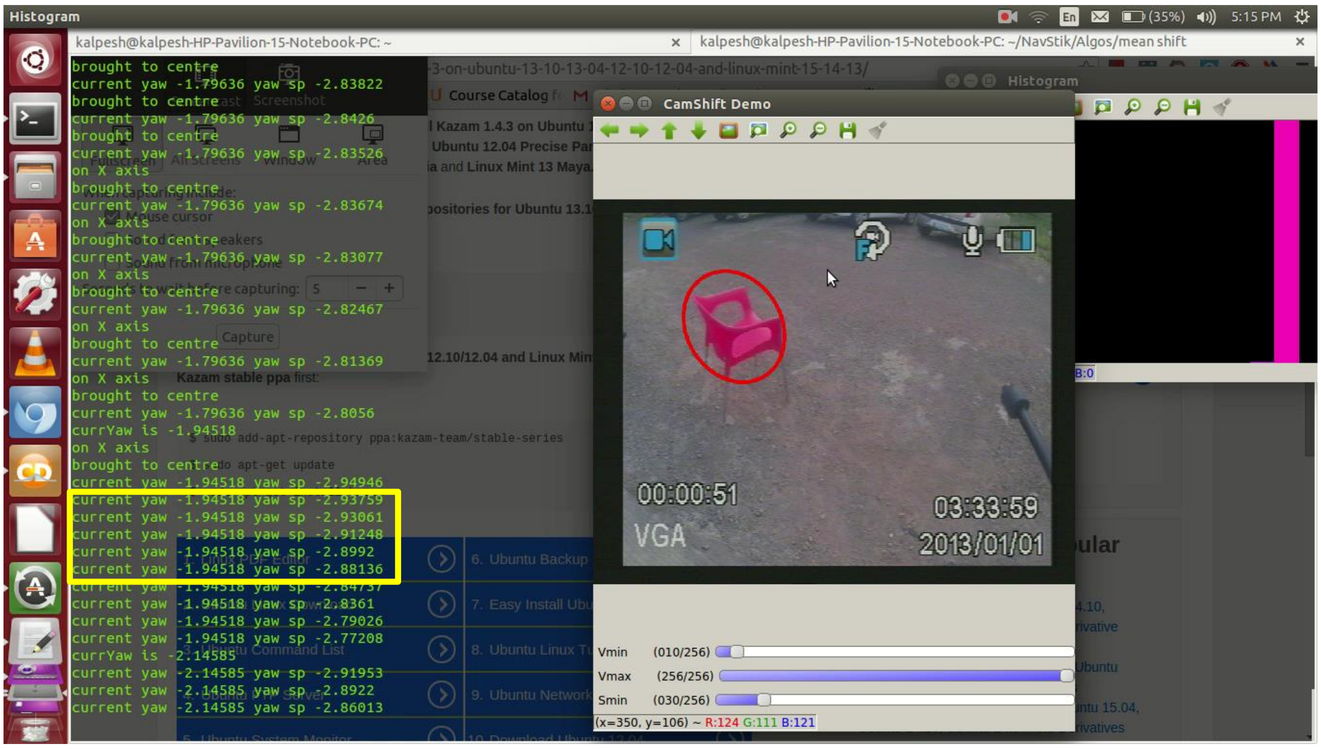
The plate is the object to be tracked. Camera is pointing towards the object.



Tilt of the gimbal is changed, focusing continuously on the object

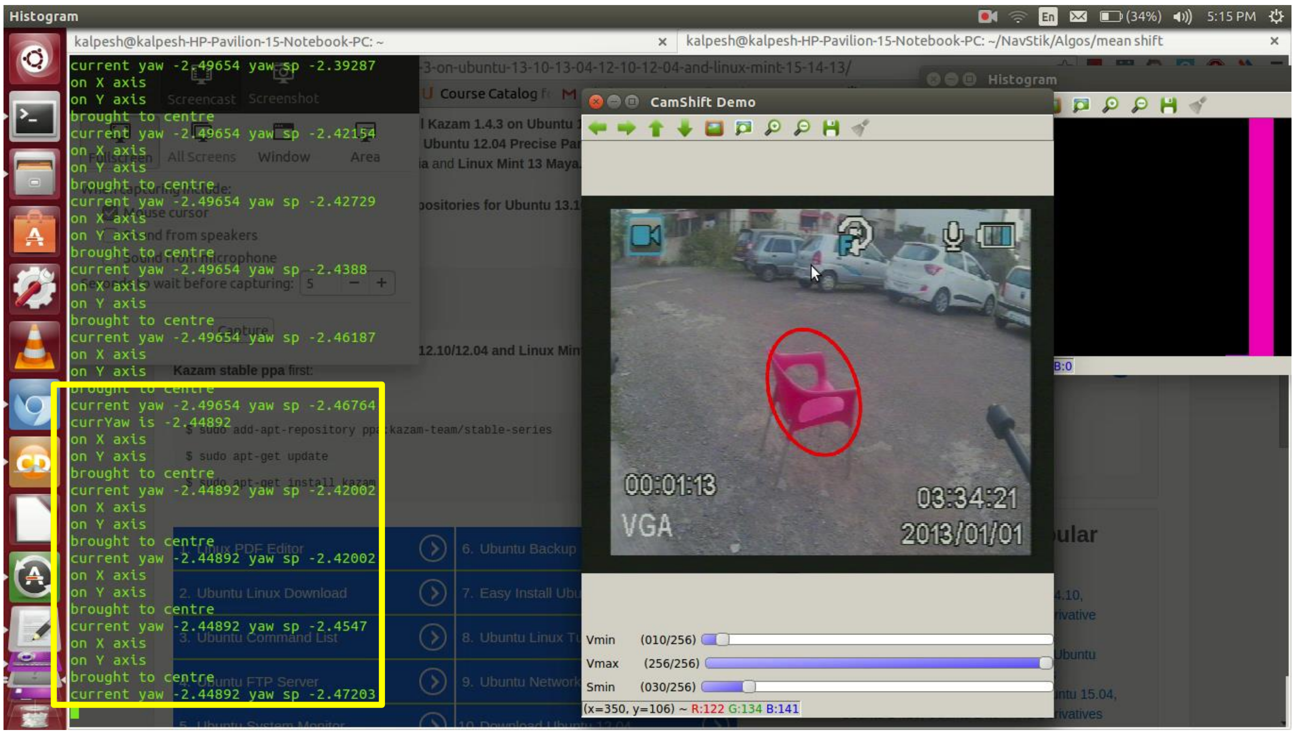
On ground quadcopter yaw test

The controller of NavStik is designed in such way to get absolute set point for yaw. Therefore we had to receive current yaw value and add or subtract the offset calculated from image to give the required absolute yaw set point. We implemented our algorithm and rotated quadcopter with hands till current yaw becomes equal to the yaw set point generated by the code. When both became nearly equal, object was at the centre


Histogram
kalpesh@kalpesh-HP-Pavilion-15-Notebook-PC: ~
kalpesh@kalpesh-HP-Pavilion-15-Notebook-PC: ~/NavStik/Algos/mean shift
CamShift Demo
00:00:51
VGA
03:33:59
2013/01/01
Vmin (010/256)
Vmax (256/256)
Smin (030/256)
(x=350, y=106) ~ R:124 G:111 B:121

```
brought to centre
current yaw -1.79636 yaw sp -2.83822
brought to centre
current yaw -1.79636 yaw sp -2.8426
brought to centre
current yaw -1.79636 yaw sp -2.83526
on X axis
brought to centre
current yaw -1.79636 yaw sp -2.83674
on X axis
brought to centre
current yaw -1.79636 yaw sp -2.83077
on X axis
brought to centre
current yaw -1.79636 yaw sp -2.82467
on X axis
brought to centre
current yaw -1.79636 yaw sp -2.81369
on X axis
brought to centre
current yaw -1.79636 yaw sp -2.8056
currYaw is -1.94518
on X axis
brought to centre
current yaw -1.94518 yaw sp -2.94946
current yaw -1.94518 yaw sp -2.93759
current yaw -1.94518 yaw sp -2.93061
current yaw -1.94518 yaw sp -2.91248
current yaw -1.94518 yaw sp -2.8992
current yaw -1.94518 yaw sp -2.88136
current yaw -1.94518 yaw sp -2.84737
current yaw -1.94518 yaw sp -2.8361
current yaw -1.94518 yaw sp -2.79026
current yaw -1.94518 yaw sp -2.77208
currYaw is -2.14585
current yaw -2.14585 yaw sp -2.91953
current yaw -2.14585 yaw sp -2.8922
current yaw -2.14585 yaw sp -2.86013
```

Current yaw is the absolute value of yaw determined by the sensor on NavStik. Yaw sp is the absolute set point generated by adding/subtracting the offset to the current value.



```
current yaw -2.49654 yaw sp -2.39287
on X axis
on Y axis
brought to centre
current yaw -2.49654 yaw sp -2.42154
on X axis
on Y axis
brought to centre
current yaw -2.49654 yaw sp -2.42729
on X axis
on Y axis
brought to centre
current yaw -2.49654 yaw sp -2.4388
on X axis
on Y axis
brought to centre
current yaw -2.49654 yaw sp -2.46187
on X axis
on Y axis
brought to centre
current yaw -2.49654 yaw sp -2.46764
currYaw is -2.44892
on X axis
on Y axis
brought to centre
current yaw -2.44892 yaw sp -2.42002
on X axis
on Y axis
brought to centre
current yaw -2.44892 yaw sp -2.42002
on X axis
on Y axis
brought to centre
current yaw -2.44892 yaw sp -2.4547
on X axis
on Y axis
brought to centre
current yaw -2.44892 yaw sp -2.47203
```

Histogram
kalpesh@kalpesh-HP-Pavilion-15-Notebook-PC: ~
kalpesh@kalpesh-HP-Pavilion-15-Notebook-PC: ~/NavStik/Algos/mean shift
CamShift Demo
00:01:13
VGA
03:34:21
2013/01/01
Vmin (010/256)
Vmax (256/256)
Smin (030/256)
(x=350, y=106) ~ R:122 G:134 B:141

When we rotated our quadcopter manually, till current yaw becomes approximately equal to the yaw set point. "brought to centre" was flag set for stopping yaw movement.

Gimbal tilt test during flight

The same test which was performed for gimbal on ground was also performed with flying quadcopter. We were manually controlling the quadcopter using RC and found out that even after changing the position of the quadcopter object was always being brought back to centre.

Gimbal is in position in which it is pointing slightly upwards to object kept in front of it.

Object was at the same position, but quadcopter was at some height. Now you can observe that camera is pointing slightly downwards which indicates motion of gimbal.

Position hold and Altitude hold

Proper gain tuning was done and quadcopter was kept in both the modes. NavStik has already developed facility to keep quadcopter in altitude hold as well as position hold. GPS was used for position hold while altitude was handled by barometer alone

NOTE: Videos for above mentioned tests are available except for the position and altitude hold and gimbal calibration

CONCLUSION AND FUTURE WORK

This project was aimed at creating autonomous object tracking quadcopter. Although it was difficult to achieve the final target from the scratch during the specified duration of few weeks, the project has definitely contributed significantly in that domain. This will help people working on this topic as well as projects along these lines. Our biggest achievement was to incorporate autonomously controlled gimbal on flying quadcopter with NavStik firmware. We were also able to achieve yaw movement of the quadcopter on ground.

Future Work

1. Achieving feedback from gimbal
   This could be the major milestone for achieving the target. Once we are able to get feedback from gimbal, a closed loop controller can be implemented easily for X-Y set points of the quadcopter. The expressions and methods for calculating those set points is mentioned in the project. Tarot has developed a software for windows which is used for monitoring gimbal movement and flashing its own firmware. The source code of the software isn't available, therefore we are not in position to use their protocol for our purpose. Gimbal is connected to computer using USB. If we somehow decode the protocol used in that serial communication, we can get the required angle information in our code. One must look for USB hacking of gimbal for this purpose.
2. Designing and testing algorithm for quadcopter movement
   If we are not able to get feedback from gimbal, we will have to design a new algorithm for quadcopter movement. We have plenty of options available to achieve this. One of the most feasible option is explained in detail earlier in this report.
3. Using both tilt and roll of gimbal
   This could also be an alternative to present algorithm. The problem with this algorithm was its difficulty in tracking objects. If we could design a better tracking algorithm, this can be considered for generating set points
4. Designing PID controller for tilt
   We have used only proportional logic for generating set points for tilt. We would like to design a full-fledged PID controller for generating tilt set points

Currently we are giving manually selected object as an input for tracking. An extension to this project can be autonomously detecting an object from learnt templates and take it as an input for further motion. Not only this, the project can be extended in various manners like tracking and following multiple objects, taking shots from different angles of the tracked object etc. Such projects will be contributing a lot to the fields like aerial photography, autonomous surveillance, vision based navigation etc. I hope this project helps all the contributors in future.

REFERENCES

Research Papers

1. People Tracking via a Modified CAMSHIFT Algorithm
   Fahad Fazal Elahi Guraya, Pierre-Yves Bayle and Faouzi Alaya Cheikh
2. AUTONOMOUS DETECTION AND TRACKING OF AN OBJECT AUTONOMOUSLY USING AR.DRONE QUADCOPTER
   Futuhal Arifin, Ricky Arifandi Daniel and Didit Widiyanto
3. An Efficient Moving Target Tracking Strategy Based on OpenCV and CAMShift Theory
   Dongyu Li
4. Tracking objects with fixed-wing UAV using model predictive control and machine vision
   Stian Aas Nundal, Espen Skjong
5. Robust Feature-based Object Tracking
   Bing Han, William Roberts, Dapeng Wu, Jian Li
6. Real-Time Compressive Tracking
   Kaihua Zhang, Lei Zhang and Ming-Hsuan Yang
7. Particle Filtering for Visual Tracking
   Pedram Azad
8. Tracking-Learning-Detection
   Zdenek Kalal, Krystian Mikolajczyk and Jiri Matas

Surveys

1. REVIEW AND EVALUATION OF WELL-KNOWN METHODS FOR MOVING OBJECT DETECTION AND TRACKING IN VIDEOS
   Bahadır KARASULU
2. Performance Comparison of Kalman Filter and Mean Shift Algorithm for Object Tracking
   Ravi Kumar Jatoth, Sampad Shubhra, Ejaz Ali
3. The Visual Object Tracking VOT2014 challenge results
4. An Experimental Comparison of Online Object Tracking Algorithms
   Qing Wang, Feng Chen, Wenli Xu and Ming-Hsuan Yang
5. Performance evaluation of object detection algorithms for video surveillance
   Jacinto Nascimento, Jorge Marques

6. A Superior Tracking Approach: Building a strong Tracker through Fusion
   Christian Bailer, Alain Pagani1 and Didier Stricker
7. A SURVEY ON MOVING OBJECT TRACKING IN VIDEO
   Barga Deori and Dalton Meitei Thounaojam

Codes

1. MO-TLD (Multi Object TLD)
   https://github.com/evilsantabot/motld
2. Compressive Tracking
   http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm
3. CMT
   http://www.gnebehay.com/cmt
4. STRUCK
   https://github.com/gnebehay/STRUCK
5. Open TLD
   https://github.com/gnebehay/OpenTLD/releases
6. Particle filter
   https://bitbucket.org/kschluff/particle_tracker