Report on R&D
Autonomous UAV Navigation in Unknown Indoor Environment using Onboard Light Weight Camera

as part of

TRADR* Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response

Devvrat Arya†
Matrikel Nr.: 9022608

B-IT Master Studies in Autonomous Systems

Bonn-Rhein-Sieg University of Applied Sciences

Advisors: Dr. rer. nat. Björn Kahl‡
Rainer Worst§

January 15, 2015

*TRADR is an EU-funded Integrated Project in the FP7 Programme ICT: Cognitive Systems, Interaction, Robotics (grant nr. 609763), running from November 2013 to December 2017. www.tradr-project.eu

†devvrat.arya@smail.inf.h-brs.de

‡bjoern.kahl@h-brs.de, Bonn-Rhein-Sieg University of Applied Sciences

§rainer.worst@iais.fraunhofer.de, Fraunhofer IAIS

Declaration

I, **Devvrat Arya**, hereby declare that this work has not previously been submitted to this or any other university, and that unless otherwise stated, the content is entirely my own work.

Date	Devvrat Arya

Abstract

The field of aerial vehicles is nowadays a popular and fast evolving one. In recent years, robotics community has shown an increasing interest in autonomous flying robots as they can well implement many flight missions in more challenging environments, with lower risk of damaging itself and its surroundings. Miniature aerial vehicles (MAVs) that can fly autonomously in unstructured indoor GPS-denied environments such as homes and offices would be useful for exploration, rescue, surveillance and also for entertainment. These aerial vehicles need to be small and light weight which restricts them with limited payload and power, so a low power consuming long range sensor is required instead of power consuming LIDARs or Lasers.

In this report we propose a system that enables a low cost quadrocopter coupled with a ground based laptop to fly autonomously in a previously unknown and GPS denied environments such as indoor corridors or hallways. The proposed method is simple and based on onboard robust monocular camera. The aim is to find the vanishing point by using least computationally expensive image processing technique and make the drone fly towards the vanishing point. We also discussed the difficulties in achieving fully autonomous UAV flight, highlighting the differences between ground and aerial robots that make it difficult to use algorithms that have been developed for ground robots. The primary contribution is the development of a basic indoor navigation szstem for UAV. We also show experimental results that validates the UAV's ability to fly in unknown corridors, and demostrate the robustness of the algorithm implemented.

Contents

List of Figures

1 Introduction

1.1 Motivation

Natural disasters have continuously threatened human life and livelihood worldwide. Although we cannot avoid the occurrence of natural disasters, we can attempt to minimize the impact of such disasters on human life and livelihood. The first priority is to recognize the scope and extent of the disaster damage over the affected area in order to most effectively begin rescue and disaster recovery activities.

Consider the disaster scene in figure 1. Sending rescue personnel into the building to search for survivors puts them in grave danger. Without knowing what awaits them inside the building, it is very difficult to make good decisions about where it is safe to venture and where to look for survivors. If instead, the building could be searched by a robot, the risks taken by the rescue workers would be greatly diminished. There are many situations where it is dangerous and difficult for humans to acquire sensing information and where robots could be of use.

Figure 1: Collapsed building

[Photo credit : http://thumbs.dreamstime.com/]

Functioning in the collapsed building, requires a robot to traverse through cluttered,obstacle strewn and uneven environment. Over the years, researchers have tackled these problems and designed a number of ground robot systems as shown in figure 2. capable of traversing rough terrain. There has been a great progress toward this goal however it is still an active area of research and no matter how far the field advances, there will always be some terrain which a ground robot is simply not physically capable of climbing over. Many researchers have therefore proposed the use of Micro Air Vehicles (MAVs) as an alternative robotic platform for rescue tasks and a host of other applications.

Figure 2: Ground robots that can traverse through rough surfaces

[Photo credit: (a)DARPA Learning Locomotion Project at MIT, (b) NIST]

Compared with ground vehicles, the main advantage of flying devices is their increased mobility. This directly raises the question of how to equip them with autonomous navigation abilities. Aerial vehicles need to be small and lightweight, so a passive, low-power long-range sensor is preferable than power-consuming sensors such as LIDAR or Hokuyu.

Most of the UAVs designed today are enabled by GPS and IMU[1](Inertial measurement unit) that makes UAVs capable to fly in outdoor environments without human intervention. Only few approaches have been developed so far for indoor environments where the system can not rely of GPS and therefore it has to use exteroceptive sensors for navigation like RGB/RGB-D cameras.

For ground vehicles, payload is not a major problem, so they can be equipped with heavy sensors like lasers. Several effective systems for indoor and outdoor navigation of ground vehicles are currently available[1, 2]. In general navigation algorithms applied on ground robots could in principle be applied on flying robots, however this transfer is not simple and straightforward for various reasons such as limited payload, computational limitations, dynamics etc. These are described latter in the key challenges with UAVs.

Thus applying ground navigation systems on aerial robots has to fulfill these stringent constraints efficiently.[3] Because of the fast dynamics of a flying vehicle compared with a ground one all the quantities necessary to stabilize the vehicle should be computed within a short time and with an adequate level of accuracy.

Although there have been significant advances in developing accurate, drift-free SLAM algorithms for large-scale environments, these algorithms have focused almost exclusively on ground or underwater vehicles. However applying the same techniques on the aerial robots have not been as successful due to the combination of limited payloads for sensing and computation, along with the fast, unstable dynamics of the air vehicles as mentioned above.

[1]An inertial measurement unit, or IMU, is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes, sometimes also magnetometers.

1.2 Objective

Our aim is to design and implement a navigation system for an UAV that has the task to explore a corridor within a building. We will rely only on perspective cues from a stream of images taken by the on-board light weight camera, requires little computational power and can be used by a controller directly without building a 3D model[4]. UAV would first fly autonomously to one end of the corridor and then to the other end. The system should only be dependent on on-board exteroceptive sensors like a light weight camera for autonomous navigation, requires minimum computation and is less power consuming. This can help in increasing the flight time for UAV or can save power for UAV for other processes like planning, control etc.

Experiments will be performed on an inexpensive drone like the Parrot AR-Drone 2.0, which can be controlled by a ground station running ROS framework. Once the approach is successful, this approach will be implemented on AscTec drone which has high end processors and capable of doing calculations onboard.

1.3 Structure

- Section 2 gives a background about the concepts relevant to this work.
- Section 3 covers the related work on UAV navigation for both indoor and outdoor environment. This section also cover some techniques which have already been successfully applied for UAV navigation using only onboard sensors.
- Section 4 describes the problem statement and under the task description subsection we discuss the work done in this research.
- Section 5 gives details regarding the experimental setup. This covers the drone we used, its sensors and other properties.
- Section 6 is the main section which elaborates our proposed architecture, environment model, algorithms used for vanishing point detection and the controller used.
- Section 7 shows the evaluation of the system. Here we have provided experimental results validating our design and capabilities of our system.
- Section 8-9 gives the conclusion and future work needed for this R&D.

2 Background

This section gives brief description/definitions of concepts relevant for this work with respect to flying robots/drones.

2.1 Unmanned Aerial Vehicle(UAV)

"An unmanned aerial vehicle (UAV), commonly known as a drone and also referred to as an unpiloted aerial vehicle and a remotely piloted aircraft (RPA) by the International Civil Aviation Organization (ICAO), is an aircraft without a human pilot aboard[5]." Unmanned Aerial Vehicles (UAVs) have been referred to in many ways: RPVs (remotely piloted vehicle), drones, robot planes, and pilot-less aircraft are a few such names. UAVs can be

- remote controlled aircraft (e.g. flown by a pilot at a ground control station)
- can fly autonomously based on pre-programmed flight plans or more complex dynamic automation systems.

We are interested only in autonomous flying robots. Because of the increased mobility of the flying robots over ground robots, this gives them an added advantage. Therefore, UAVS that can autonomously operate in indoor environments are envisioned to be useful for a variety of applications that include surveillance and search and rescue[6].

Key challenges with UAVs

In the ground robotics domain, combining wheel odometry with sensors such as laser range-finders, sonars, or cameras in a probabilistic SLAM framework has proven very successful[11]. Many algorithms exist that accurately localize ground robots in large-scale environments.UAVs face a number of challenges that make developing algorithms for them far more difficult than their indoor ground robot counterparts. The requirements and assumptions that can be made with flying robots are sufficiently different that they must be explicitly reasoned about and managed differently.

Limited Payload UAVs can carry only a limited payload, which is a substantial obstacle in achieving their full autonomy. As the payload of aerial robots is limited so the sensors are limited on aerial robots. This constraint eliminates use of popular sensors such as SICK laser scanners, large-aperture cameras and high-fidelity IMUs. Therefore UAVs rely only on lightweight Hokuyo laser scanners, micro cameras and lower-quality IMUs. These sensors have limited ranges and field of views compared to sensors used for ground robots. Also the readings from these sensors are noisy.

Limited Onboard Computation The navigation algorithms are computationally demanding even for powerful desktop computers and therefore cannot be used on today's small embedded computer systems which are mounted onboard UAVs. Another way to deal with the limited onboard computation issue is to transmit the UAV sensory readings to a ground station which performs the computationally demanding sensor processing and localization algorithms[7, 8, 9]. However the architecture based on ground system is not fully autonomous due to the dependency on ground system for computation which brings network dependencies such as communication bandwidth then becomes a bottleneck that constrains sensor options. For example, camera data must be compressed with lossy algorithms before it can be transmitted over wireless links, which adds noise and delay to the measurements. The delay is in addition to the time taken to transmit the data over the wireless link. The noise from the lossy compression artifacts can be particularly damaging for feature detectors that look for high frequency information such as corners in an image.

Fast Dynamics The helicopterâs fast dynamics result in a host of sensing, estimation, control and planning implications for the vehicle. Filtering techniques such as Kalman Filters are often used to obtain better estimates of the true vehicle state from noisy measurements. Smoothing the data generates a cleaner signal, but adds delay to the state estimates. While delays generally have insignificant effects on vehicles with slow dynamics, the effects are amplified by the MAVâs fast dynamics.

Indirect Relative Position Estimates Flying robots are unable to measure odometry directly, as they do not maintain physical contact with the ground or surroundings. Alternatively, this can be computed by double integrating the accelerations for which light weight MEMs IMUs are used however readings through these sensors are unstable and biased which results in large drift. Therefore, UAV must get the relative position of the vehicle using exteroceptive sensors, and computing the vehicleâs motion relative to reference points in the environment.

Unsteady motion Ground robots are inherently stable, in the sense that by issuing a zero-velocity command results in the robot to smoothly decelerate until it stops. The same does not apply for flying robots that need to be actively stabilized, even when they are already in the desired location[10]. As a result, it may pick up speed or oscillate, degrading the sensor measurements further.

3D Motion UAVs operate in 3D environment. For a ground robot it is sufficient to estimate a 2D map of the environment but for air vehicles 2D projection or cross section of 3D environment

can change drastically with height and attitude, as obstacles suddenly appear or disappear. However, if we explicitly reason about the effects of changes due to the 3D structure of the environment, we have found that a 2D representation of the environment is surprisingly useful for MAV flight.

2.2 Different architectures for UAVs

Based of processing unit of the UAV, autonomous flying robots can further be designed using two architecture models.

- A basic UAV architecture where total computation is performed onboard. In this model, all the processing is done on the drone on board, UAV only communicate with the external system or the supervisor to send the necessary information through wireless network. This architecture results in a closed-loop UAV system that can operate without communications to any ground controller. UAV is equipped with many sensors and high-end processors capable to perform heavy computation, however this increases the payload and these processors consume power which results in reduced flight time.

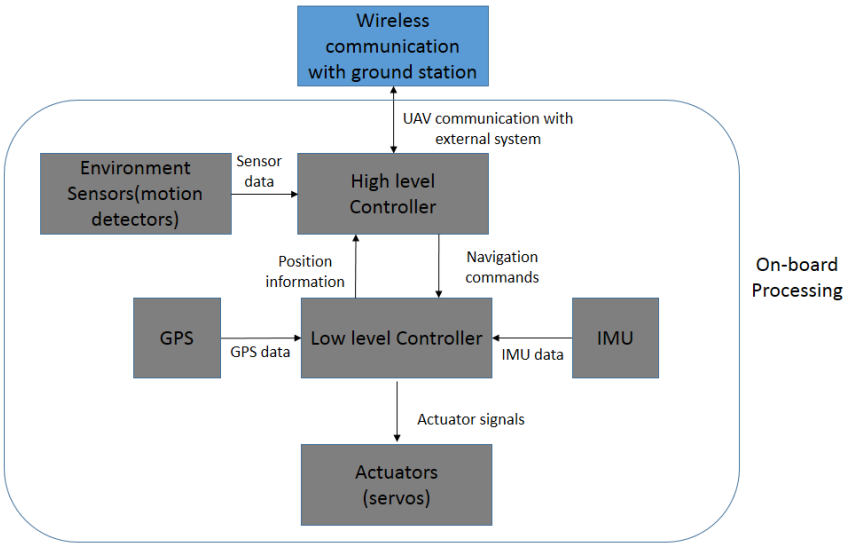


Figure 3: Basic drone architecture with all computation onboard

- The second model commonly used by researchers of UAVs is in which the autonomy algorithms(navigation algorithms) are run on an external system(desktop computers) and navigation commands are sent to the vehicle from the controller running on the ground system. This model is used to overcome the limited onboard computation problem as described above.

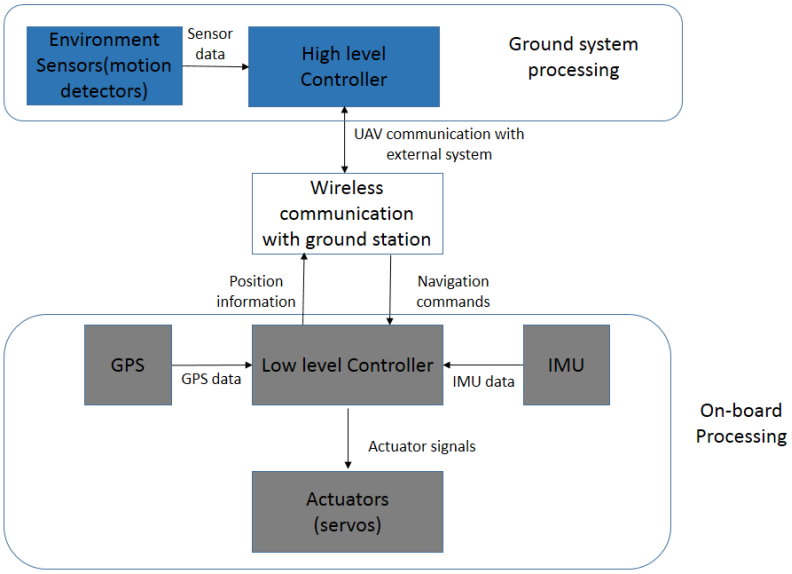


Figure 4: Basic drone architecture with computation on ground system

Many commercial UAV manufactures provides the flexibility to program their UAVs for research purposes and hence provides APIs to control the air vehicle. In such cases the low level controller on UAV drives the actuators based on the navigation commands given by the ground system. Typically these low level controllers uses simple proportional-integral-derivative (PID) controller to map the given commands to actuator motion.

2.3 UAV navigation

As for ground vehicles, the main task for an autonomous flying robot consists in reaching a desired location in an unsupervised manner, that is, without human interaction. In the literature, this task is known as navigation or guidance[10]. Several effective systems for indoor and outdoor navigation of ground vehicles are currently available. However the these algorithms or systems cannot be directly applied to flying robots.

One major difference between ground and aerial robots is ground robots are inherently stable, in the sense that by issuing a zero-velocity command results in the robot to smoothly decelerate until it stops. The same does not apply for flying robots that need to be actively stabilized, even when they are already in the desired location. Another problem in porting navigation systems is because of the fast dynamics of a flying vehicle compared with a ground one all the quantities necessary to stabilize the vehicle should be computed within a short time and with an adequate level of accuracy.

2.4 Closed loop UAV navigation system

Similar to most of the navigation techniques for ground robots, UAV navigation system is also closed loop therefore the output of every command i.e the current state of the UAV is feed to the system to calculate the next command. For any general navigation system, we need goal position and current state. Figure 5 explains the how closed loop behaviour works for UAV navigation.

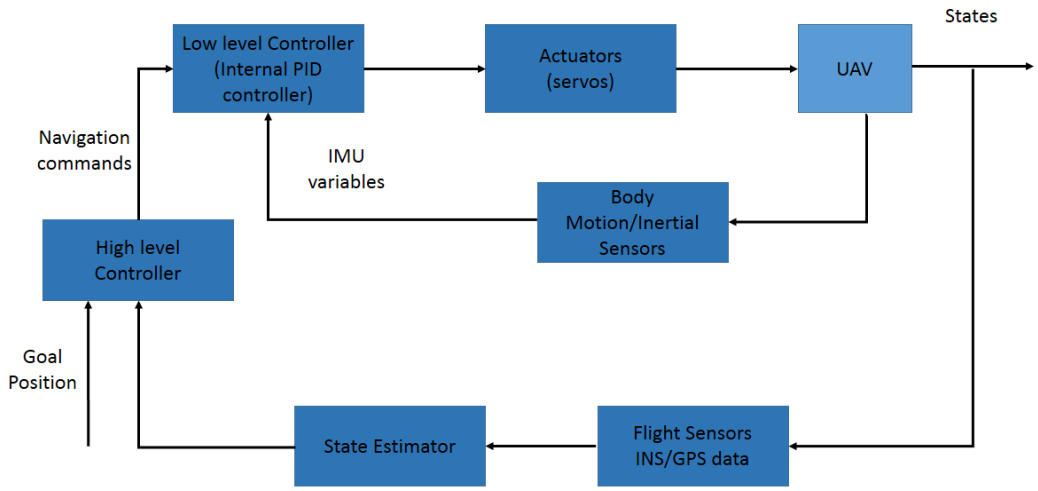


Figure 5: Typical UAV navigation control system

3 Related work

In the recent years, flying platforms received an increasing attention from the research community. Many authors focused on the modeling and on the control of these vehicles[12, 13, 14],with a particular emphasis on small and unmanned aerial vehicle[15]. Many capabilities have been developed for autonomous operations in outdoor environments, including high-speed flight through cluttered environment, helicopter acrobatics, autonomous landing and terrain mapping[16], coordinated tracking and planning of ground vehicles[17], etc.

There are two aspects of navigation, outdoor and indoor navigation. Both navigation scenarios have different constraints hence we need different approaches to tackle these problems. Research so far has produced a number of navigation systems with a wide range of capabilities when operating in outdoor environments. For example vehicles have been developed that can perform high speed flight through cluttered environments[18], or even acrobatics[19]. High level capabilities such as autonomous landing, coordinated tracking, terrain mapping[16], planning of air and ground vehicles[17], and multi-vehicle coordination [22, 23] are already developed by the researchers. These researches are challenging in their own right and now researchers are trying to take this work for indoor navigation. But these systems for outdoor navigation typically take advantage of GPS measurements for state estimation and localization.

While some authors[27, 28] have demonstrated indoor flight using GPS simulated from motion capture systems, we seek to develop flying robots that are able to operate autonomously while carrying all sensors used for localization, control and navigation onboard.

In our work, we focus on flying robots that are able to operate autonomously in indoor environment. Most indoor environments and many parts of the urban canyon remain without access to external positioning systems such as GPS. Also for indoor environment collision avoidance cannot be achieved based on GPS-data alone. Too many unforeseen obstacles may cross the way path of the MAV. Especially in autonomous use, reliable obstacle avoidance is indispensable for ensuring the MAVâs survivability. Therefore autonomous MAVs today are thus limited in their ability to fly through these indoor areas. Traditionally, unmanned vehicles operating in GPS-denied environments can rely on dead reckoning for localization or motion detectors.

3.1 Non Vision sensors based indoor UAV navigation

Because of constraints associated with the use of GPS, different sensors have been used for indoor navigation. Kumar and Ghose[24] and Kwag and Kang[25] implemented radar based navigation and obstacle avoidance. Saunders [26] used a forward looking laser range finder for path planning. Achtelik et al. [29] used a laser range finder and a stereo camera for quadrotor navigation, with the eventual goal of combining the two complementary sensors. However, these approaches lack in heavy weight or high power consumption. Custom-built quadrotors are used

in order to support heavy sensors or additional battery weight necessary to support active, power-hungry sensors. But for most miniature flying platform it is important to keep weight light and to save energy wherever possible. Therefore, having a lightweight sensor with a small power-consumption is of great interest.

3.2 Vision sensors based indoor UAV navigation

For UAVs, we need light weight and less power hungry sensors, both targets are reached using a camera. Since it is a passive sensor, power requirements are low while having a light weight and offers long-range sensing. These arguments declare the camera to one of the most suitable sensors for navigation algorithms. Therefore recently, there is an interest in building UAVs that only uses vision sensor to fly in indoor environments. Many successful vision based UAV navigation systems exist for outdoor navigation which also use vision sensors along with other devices and sensors. For example Tempelton [30] demonstrate how to use vision for outdoor terrain mapping and autonomous landing, Tournier [31] and Bourquardez [32] used vision to estimate and stabilize the current pose of a quadrotor. However for indoor environment, a perfect solution is yet to be developed.

Navigation in known and unknown environment using vision sensors

The navigation problem can further be classified in two categories based on the environment considered such as known or unknown environment. The navigation approach for both type of environment is very different. There has been great work done on vision based navigation techniques for both type of environments.

- **Known environments:** When the knowledge or map of the environment is available, such environment is considered as known environment. Soundararaj, Prasanth and Saxena [33] and Courbon [34] used vision to fly in known environments, however their method does not apply to scenarios where full visual databases are not available. Mori[35] used markers to stably hover a co-axial helicopter and go from one marker to another.
  There have been numerous efforts to fly helicopters autonomously indoors using monocular camera sensors. Some authors have demonstrated autonomous flight in limited indoor environments, their approaches have been constrained to environments with specific features, and thus may not work in unknown or featureless environment. For example [36] performed visual servoing over known Moire patterns to extract the full 6 degree-of-freedom state of the vehicle for control, while [37] detects lines in a hallway, and [38] tracked edges in office environments with known structure.

- **Unknown environments:** When there is no complete knowledge about the environment is available, in such unknown environment navigation task has more challenges. Any vision related approach primarily depends on the features from the environment to perform analysis. Celik [39] properties of the environment for UAV path-planning, this technique only applies to situations where there are strong feature points that could be tracked from frame to frame, and would not apply in many indoor environments that are devoid of track-able features (e.g., walls). use vision to reconstruct 3D.
  Vision based techniques can also be used to estimate an aircraft position through matching a sequence of onboard images to a geo-referenced image. Such approach can be found in [40] which uses aerial image matching for aircraft position estimation. One can also build a map of the environment using Visual SLAM [41, 42], in which the map is built using images captured from a camera.

Some researchers customize the UAV and put an extra micro controller to assist the computation for the UAV. [44]. This additional controller is in charge of path planning and control tasks.

Figure 6: Low cost quadrotor AR.Drone version 1.0 and 8-bit microcontroller. [44]

4 Problem formulation and Task description

4.1 Problem formulation

Image features are locations in the image which are distinctly recognizable such that finding the correspondence between the same features in different images is possible. In general image features can be any visually distinct pattern in the image, however for our purposes we seek to find "edge-features" which are characterized by strong intensity gradients in two directions. Feature detection is a basic image processing primitive underlying many computer vision algorithms, and there are many options available which have different performance and computational characteristics. Although SIFT [47] and SURF [48] features are popular choices for visual feature detectors due to the ease with which they can be corresponded, computing them fast enough for our purposes on modern hardware remains computationally infeasible given the control requirements of our quadcoptors discussed in section 2.1

Many of the vision sensor based navigation techniques explained in the previous section 3.2 involve optical flow based algorithms [2][4]. Stabilizing controllers based on optical flow were presented in [15], and similar methods are integrated in commercially available hardware [43]. Such systems however are subject to drift over time, and hence not suited for long-term navigation. Also 3D modeling or optical based vision techniques require complex computation and rely on high-end processing unit. Therefore vision based navigation systems for UAVs often use remote ground system for computation. Our aim is to implement a vision method which uses less computation and can be run onboard using lesser resources than heavy vision techniques.

However there are some vision based techniques available which are less complex, faster and can work on embedded system framework similar to used in UAVs. For example methods involving extracting features like edges or corners could be used for developing navigation algorithms. Such navigation methods have been successfully applied on ground robots, applying such algorithms on UAVs with some modifications with respect to UAVs constraints may result in a better navigation system for UAVs.

4.2 Task Description

Navigation problem comprises of several smaller sub tasks like mapping, localization, path planning and obstacle detection. These are high level sub tasks. In the research presented in this report, we sought to tackle navigation problem on a smaller aspect limited to corridors such as shown in figure 7 below. The developed navigation system should be designed considering the problems discussed in 4.1 and is suitable to run on embedded hardware of UAV. This will allow onboard processing for UAV and make it independent of ground station for computations. For

[2]Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene.

this research we are not considering obstacle avoidance. This system should integrate sensing, planning, and control to enable a MAV to autonomously explore indoor environments. We seek to do this using only onboard sensing and without prior knowledge of the environment.

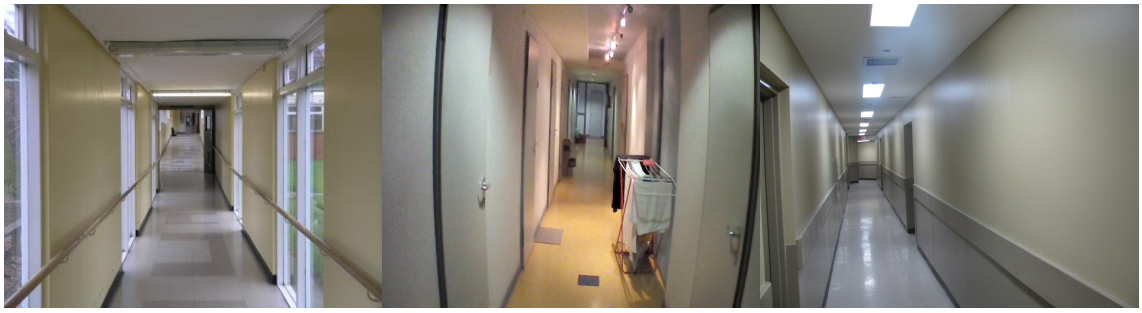


Figure 7: Corridor sample images

5 Experimental setup

This section gives a description about the equipment used in this research.

Our primary platform is the Parrot AR.Drone 2.0 quadrotor (Figure 8). We will first implement our approach on the inexpensive toy, and if successful we will further implement this approach on Asctec Pelican quadrotor. Parrot AR. Drone 2.0 is currently available to the general public and contains two cameras (one facing forward, the other horizontally downwards), a sonar height sensor, and an onboard computer running proprietary software for communication and command handling.

Commands and images are exchanged via a WiFi ad-hoc connection between our host machine and the AR.Drone. We are processing the images from the front camera, which has a wide angle lens, specifically 92°. The resolution is 1280 X 720 pixels for recording, 640 X 480 pixels for WiFi streaming and this frames are sent at a speed of 30fps. Drone has an onboard 1GHz 32 bit ARM Cortex A8 processor. Ar drone 2.0 is also equipped with 3 axis gyroscope 2000°/second precision, 3 axis accelerometer +-50mg precision, 3 axis magnetometer 6°precision, Pressure sensor +/- 10 Pa precision, Ultrasound sensors for ground altitude measurement and 60 fps vertical QVGA camera for ground speed measurement with an opening angle of 63°and resolution of 320 X 240 pixels.

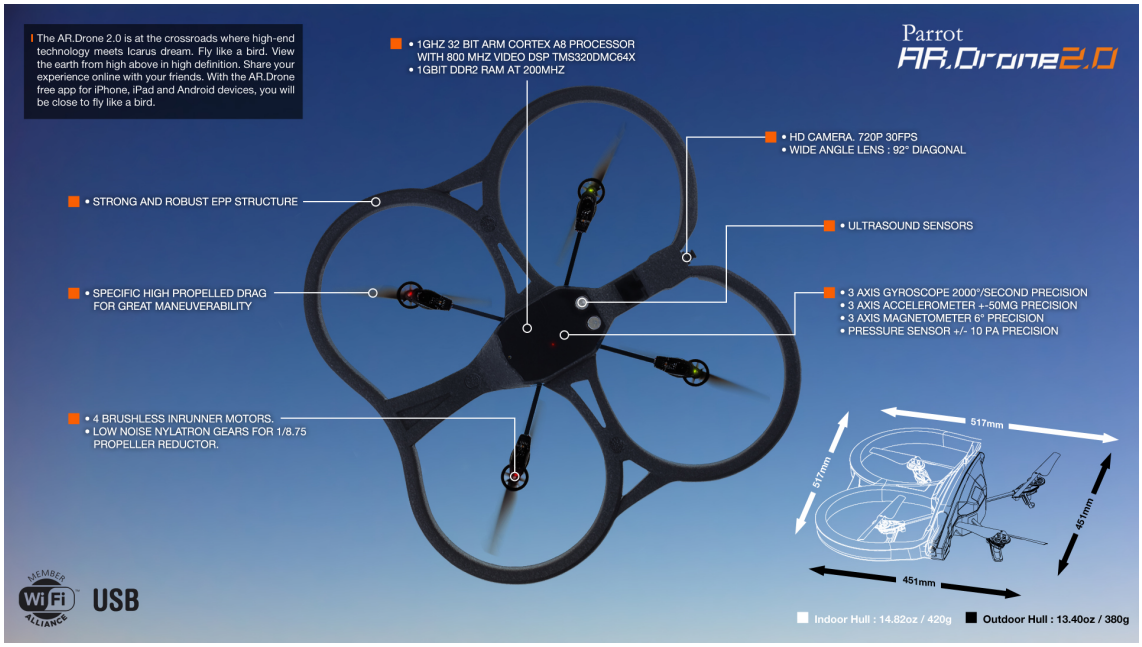


Figure 8: Parrot AR.Drone 2.0

©http://www.parrot.com/media/gallery/ressources/

Our algorithms run on the host machine –a Lenovo Y560 (Intel®Core i7 CPU Q 720 @ 1.60GHz x 8, 4GB RAM), running Ubuntu 12.04.

Forward Camera	HD, 720p 926°diagonal viewing area
Bottom Camera	QVGA, 320x240 646°diagonal viewing area
Computational Resources	1 GHz ARM Cortex-A8 CPU 800 MHz Video Digital Signal Processor 256 MB (1 Gbit) DDR2 RAM
Networking	802.11n WiFi
Sensors	3-axis gyroscope (2000 degree/second) 3-axis accelerometer (+/- 50 mg precision) 3-axis magnetometer (6 degree precision) Pressure sensor (+/- 10 Pa precision) Ultrasound altitude sensor

Table 1: AR.Drone 2.0 technical specifications [50].

In spite of the high frame rate, the speed can drop in case the drone is destabilized, because the stabilization procedure has higher priority than the video broadcasting.

The drone is supplied with its own software, which is not designed to be modified, but different drivers are implemented in ROS to communicate with the drone via WiFi for example ardrone_autonomy.

As mentioned in the objective of this report 1.2, the final target Asctec Pelican is a sophisticated drone which has powerful processors and equipment. The specifications of this drone are as follows :

Figure 9: AscTec Pelican

©http://www.asctec.de/en/uav-uas-drone-products/asctec-pelican/

Technical Data â AscTec Pelican	
UAV Type	Quadcopter
Onboard computer	Up to 3rd Generation Intel®Core i7 processor
Size	651 x 651 x 188 mm
Max. take off weight	1,65 kg
Max. payload	650 g
Flight time incl. payload	16 min.
Wireless communication	2,4 GHz XBee link, 10 - 63 mW, WiFi (optional)

Asctec Pelican also has mount for Laser scanner and comes with 2 HDR cameras. Some researchers also added Kinect to the Pelican to get 3D data.

6 Approach

6.1 Overview

In our approach we will implement a basic navigation system based on vanishing point algorithm using computer vision image processing techniques. For images collected by the camera of quad-rotor helicopter, the system executes the process of preprocessing of image and extracting straight lines. Then system obtains the position of vanishing point and regards it as destination point and finally controls the autonomous navigation of the quad-rotor helicopter by a controller. All algorithm development is in C++ and python using ROS[3] as the interfacing robotics middleware. We utilize the OpenCV library for features extraction and tracking.

6.1.1 What is a vanishing point?

As things get further away from us, they seem smaller. When they get far enough away, distances become ever tinier and so form a single point called vanishing point. In one-point perspective, all the horizontal lines go straight across, while the lines that move away from us the sides of boxes, the road we are on, or the railway lines in front of us all converge towards the center of the picture [45].

So the vanishing point contains much three-dimension information of the real environment, which can help to understand the real environment especially containing many parallel lines. The position estimation of vanishing point is widely used in applications such as navigation of robot, three-dimensional reconstruction, and camera calibration, and it has already become the hot research topic nowadays.

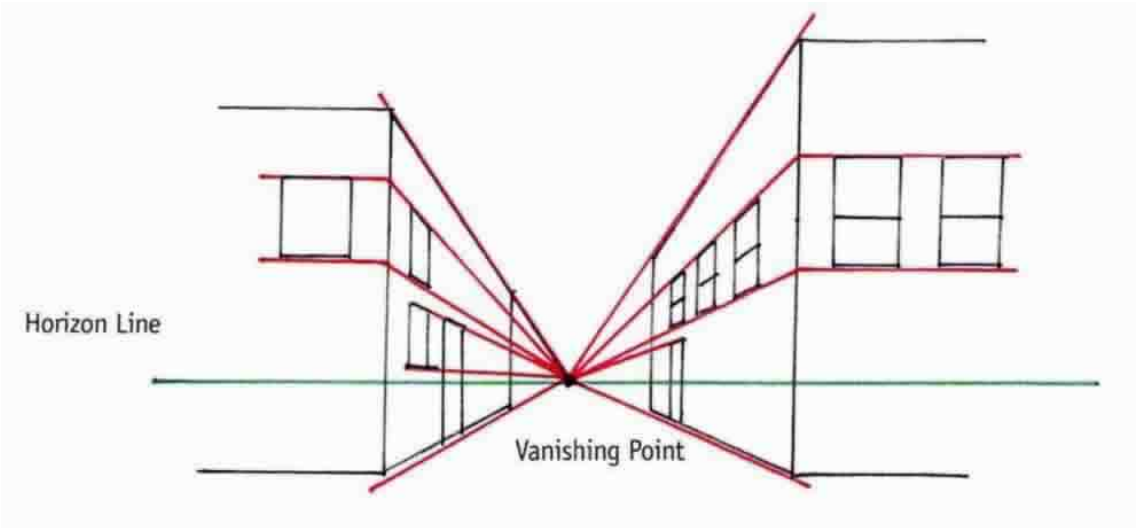


Figure 10: Example image of Vanishing Point

[3] http://www.ros.org

6.1.2 Component view of the architecture

Figure 11 shows the component view of our proposed system architecture.

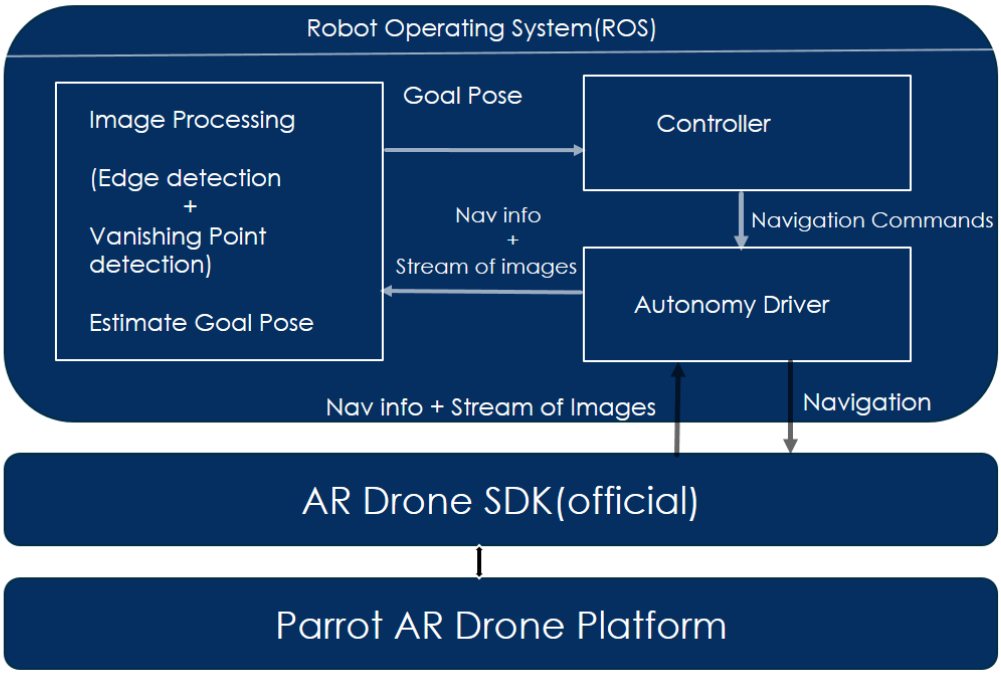


Figure 11: Component view of the Architecture

The system consists of three main modules: AR Drone Autonomy Driver, Image Processing and Controller. ROS is running on a ground system which has a ROS package AR Drone Autonomy Driver that receives navigation information and raw image from AR Drone via wireless network. The AR Drone SDK driver bundled in this ROS package provides the APIs to communicate with the AR drone. The image processing component then computes the vanishing point from the image using OpenCV(Open source computer vision) library and sends the goal position along with yaw angle to the controller. Controller then generates the linear and angular velocities based on the input from image processing component. These velocities are then fed to AR Drone internally by bundled AR Drone SDK in ardrone_autonomy package. A general idea about these three main components is given below.

6.1.2.1 AR Drone Autonomy Driver

This component of the system architecture is very important as this provides ROS interface to communicate with AR Drone. Ardrone autonomy is a ROS driver for Parrot AR Drone quadrocopter. This package handles the interface of navdata(navigation data) messages, video feeds, and control commands between ROS and the AR.Drone. This allows the use of many existing

ROS packages in localizing and controlling the quadcopter.

This driver is based on official AR Drone SDK[4] version 2.0.1. The driver supports both AR-Drone 1.0 and 2.0. "ardrone_autonomy" is a fork of AR-Drone Brown driver. This package has been developed in Autonomy Lab of Simon Fraser University by Mani Monajjemi. Thanks to the supplied open source SDK, several control parameters of flight can be set via its API which also provides access to the sensorsâ data and images from the cameras. The bundled AR-Drone SDK has its own build system which usually handles system wide dependencies itself.

Ardrone_autonomy package can be downloaded from github.
Git source : `https://github.com/AutonomyLab/ardrone_autonomy.git` (branch hydro-devel).

Driver Update Frequencies: The driver can operate in two modes: real-time or fixed rate. When the realtime_navdata parameter is set to True, the driver will publish the received information instantly. However when it is set to False, the driver will cache the most recent received data, then it will publish that at a fixed rate, configured by looprate parameter. The default configuration is: `realtime_navdata=False` and `looprate=50`.

This package provides interface to the AR drone navigation data, IMU data, camera driver, coordinate frames and send commands to the AR-drone[46].

Navigation information received from the drone is published to the ardrone/navdata topic. The message type is `ardrone_autonomy::Navdata`. Navdata is a custom ROS message and provides information about drone status like taking off, landing,flying, emergency, turning etc., battery percentage, altitude, rotation in x,y,z, timestamp and a lot more.

In order to fly the drone after takeoff, we can publish a message of type `geometry_msgs::Twist` to the `cmd_vel` topic.

6.1.2.2 Image Processing

In this component we are processing the images of the droneâs front camera. The frames are analyzed based on perspective clues. We aim to extract lines from these images and find vanishing point. The image processing component gives the x,y pixel coordinates of the vanishing point. These coordinates needs to be transformed in to world coordinate with respect to camera frame and thus calculate the goal position for the drone and the yaw angle is calculated based on how far the VP is from the image center. This component is explained in detail latter in section 6.3.

[4] https://projects.ardrone.org/

6.1.2.3 Controller

The purpose of the controller is to produce the control inputs which move the quadcopter from its current pose to a desired pose. The controller receives the estimated pose from the image processing module and sends the flight commands to the AR.Drone via ardrone autonomy.
We want to maintain at zero the horizontal distance between the vanishing point and the center of the image. We implemented a PID controller in order to achieve this. The drone needs to move towards the VP and once it reaches the end, it should come back. The controller gets the yaw angular velocity and linear velocity in x axis as input from the image processing component while the drone is flying forward with a constant linear velocity. This component is explained in detail latter in section 6.5.

6.2 Environment Model

This section elaborates our environmental model with respect to UAV camera which represents 3D world in to 2D image plane.

6.2.1 System Model of AR Drone and camera

To detect vanishing point from 2D image of the 3D space, we need to understand system model and coordinate system of the quadrocopter.

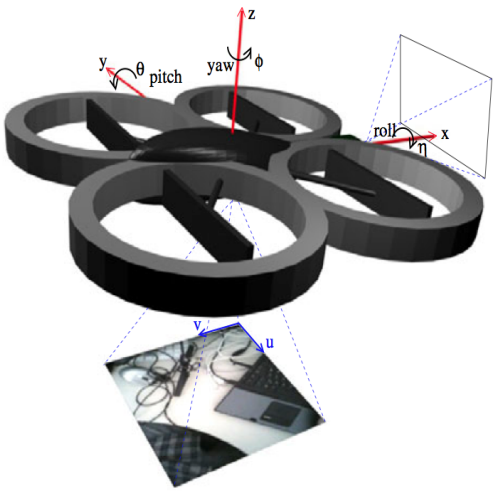


Figure 12: AR.Drone coordinate system

The coordinate frame follows the standard set by the `ardrone_autonomy` package. As shown in Figure 12, the coordinate frame is right-handed, with positive x as forward, positive y as left,

and positive z as up. In terms of rotation, a counter clockwise rotation about an axis is positive. The heading value ranges from -180 degrees to 180 degrees, with 0 centered along the x-axis.

6.2.2 Camera calibration

We need to understand the image formation and camera geometry in a more detail to determine the relationship between what appears on the image (or retinal) plane and where it is located in the 3D world. This relationship is understood by understanding how one calibrates a camera.[5]

We have a three dimensional coordinate system whose origin is at the centre of projection and whose Z axis is along the optical axis, as shown in figure 13. This coordinate system is called the *standard coordinate system of the camera*. A point M on an object with coordinates (X, Y, Z) will be imaged at some point $m = (x, y)$ in the image plane. These coordinates are with respect to a coordinate system whose origin is at the intersection of the optical axis and the image plane, and whose x and y axes are parallel to the X and Y axes. The relationship between the two coordinate systems (c, x, y) and (C, X, Y, Z) is given by

$$x = \frac{Xf}{Z} \qquad \texttt{and} \qquad y = \frac{Yf}{Z} \tag{1}$$

This can be written linearly in homogeneous coordinates as,

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

where $s \neq 0$ is scale factor.

[5] Camera calibration is the process of finding the true parameters of the camera that took your photographs.

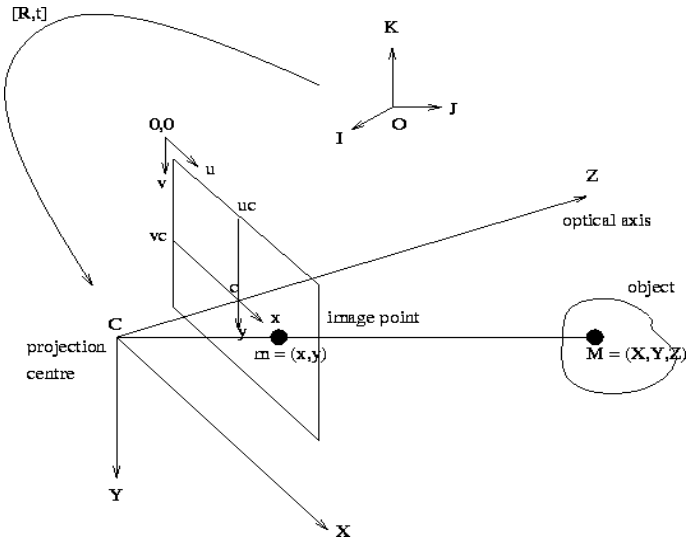


Figure 13: The coordinate systems involved in camera calibration.

[Photo credit: University of Edinburgh, School of informatics]

But, the actual pixel coordinates (u, v) are defined with respect to an origin in the top left hand corner of the image plane (image $(0, 0)$), and will satisfy

$$u = u_c + \frac{x}{\texttt{pixel width}} \qquad \texttt{and} \qquad v = v_c + \frac{y}{\texttt{pixel height}} \tag{3}$$

(x, y) are obtained from eq. 1.

Overall, we can express this transformation from 3D world coordinates to image pixel using a 3 X 4 matrix which is done by substituting equation (1) in to equation (3) and multiplying through by Z. The resultant is,

$$Zu = Zu_c + \frac{Xf}{\texttt{pixel width}}$$
$$Zv = Zv_c + \frac{Yf}{\texttt{pixel height}}$$

In other words, the image coordinates of a 3D point is standard camera frame can be obtained by this relationship

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \frac{f}{\texttt{pixel width}} & 0 & u_c & 0 \\ 0 & \frac{f}{\texttt{pixel height}} & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

where the scaling factor s is equivalent to Z (linear distance of 3D point). In short hand notaion, this is written as

$$\tilde{\mathbf{u}} = \mathbf{P} \cdot \tilde{\mathbf{M}}$$

where $\tilde{\mathbf{u}}$ represents the homogeneous vector of image pixel coordinates, P is the perspective projection matrix, and $\tilde{\mathbf{M}}$ is the homogeneous vector of world coordinates. Thus, a camera can be considered as a system that performs a linear projective transformation from the projective space $\mathcal{P}^3$ into the projective plane $\mathcal{P}^2$.

6.2.2.1 Intrinsic Parameters

The intrinsic parameters of the camera are addressed by the first matrix from Equation (4), denoted as $\mathbf{P}$. With these parameters, a relationship can be established between the pixel coordinates in the image and its position in the camera reference frame. There are five camera parameters, namely the focal length f, the pixel width, the pixel height, the parameter u_c which is the u pixel coordinate at the optical centre, and the parameter v_c which is the v pixel coordinate at the optical centre. However, only four separable parameters can be solved for as there is an arbitrary scale factor involved in f and in the pixel size. Thus we can only solve for the ratios $\alpha_u = f/\text{pixel width}$ and $\alpha_v = f/$ pixel height. The parameters $\boldsymbol{\alpha}_\mathbf{u}, \boldsymbol{\alpha}_\mathbf{v}, \mathbf{u_c}$ and $\mathbf{v_c}$ do not depend on the position and orientation of the camera in space, and are thus called the intrinsic parameters.

6.2.2.2 Extrinsic Parameters

In general, the three dimensional world coordinates of a point will not be specified in a frame whose origin is at the centre of projection and whose Z axis lies along the optical axis. Some other, more convenient frame, will more likely be specified, and then we have to include a change of coordinates from this other frame to the standard coordinate system. Thus we have

$$\tilde{\mathbf{u}} = \mathbf{P} \cdot \mathbf{K} \cdot \tilde{\mathbf{M}}$$

where K is a 4×4 homogeneous transformation matrix:

$$K = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0_3^\top & 1 \end{bmatrix}.$$

The top 3×3 corner is a rotation matrix $\mathbf{R}$ and encodes the camera orientation with respect to a given world frame, the final column is a homogeneous vector $\mathbf{t}$ capturing the camera displacement from the world frame origin. The matrix K has six degrees of freedom, three for the orientation, and three for the translation of the camera. These parameters are known as the extrinsic camera parameters.

Thus extrinsic parameters, define the camera position and orientation with reference to the world reference system.
The 3×4 camera matrix $\mathbf{P}$ and the 4×4 homogeneous transform $\mathbf{K}$ combine to form a single 3×4 matrix $\mathbf{C}$, called the **camera calibration matrix**.

6.2.3 Model of Vanishing Point

A group of parallel lines in three-dimension (3D) space can be mapped into some intersection lines in two-dimension (2D) image and the intersection point formed by these intersection lines is called vanishing point [49].
From Figure 13, equation (1) and equation (3) we know the the mapping relationship between point $A(x_0, y_0, z_0)$ in the three-dimension space and point $B(x'_0, y'_0, z'_0)$ in the two-dimension image, and we can obtain,

$$(x'_0, y'_0, z'_0) = \left(\frac{x_0 f}{z_0}, \frac{y_0 f}{z_0}, f \right) \tag{5}$$

where f represents focal length of visual system.

The indoor flight environment for quad-rotor helicopter navigation is shown as Figure 14, and the vanishing point can be obtained by the above calculation, and then it is set as the destination point navigation for quad-rotor helicopter in the indoor environment.

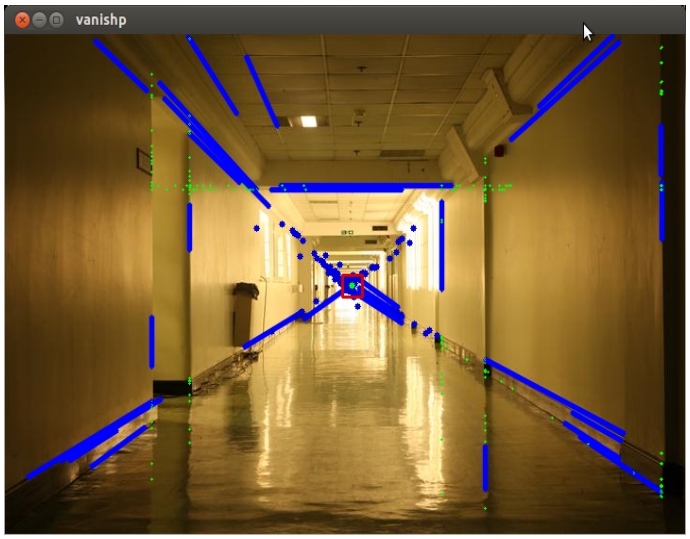


Figure 14: Corridor vanishing point

6.3 Algorithms for VP detection

This section elaborates our image processing approach for vanishing point detection in detail.

In this section, we review the methods applied to find the vanishing point(VP). We are processing the images of the droneâs front camera. The frames are analyzed based on perspective clues. We try to find the VP using two methods based on these perspective clues. Latter we compare the results from the two methods and more robust approach is selected which is used for UAV navigation.

6.3.1 Method 1: VP based on edge detection

6.3.1.1 Straight lines extraction

Recall from the overview of the approach 6.1, the idea of using line detection in the images in order to find the VP. Detecting the parallel lines existing in the environment is the premise for calculating the vanishing point, so this procedure is very important. It contains the below process: preprocessing of image, deleting noise interference, edge extracting using Canny operator, and extracting lines by RHT(Randomized Hough Transform) method.

6.3.1.1.1 Preprocessing of image: Preprocessing the image is required for image enhancement. Image enhancement is basically a conversion of image quality to a better and more understandable level for feature extraction or image interpretation. Typical image enhancement techniques include grayscale conversion, histogram conversion, color composition, color conversion, etc. which are usually applied to the image output for image interpretation. Gray scale conversion is one of the simplest image enhancement techniques. In Opencv grayscale conversion is performed using following function :

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

After converting the image to grayscale, we need to smoothen the image, also called blurring. There are many reasons for smoothing for ex. to reduce noise and capture important patterns in the data.
To perform a smoothing operation we will apply a filter to our image. The most common type of filters are linear, in which an output pixel's value (i.e. $g(i,j)$) is determined as a weighted sum of input pixel values (i.e. $f(i+k,j+l)$) :

$$g(i,j) = \sum_{k,l} f(i+k,j+l)h(k,l)$$

$h(k,l)$ is called the kernel, which is nothing more than the coefficients of the filter. [52]
We will use the simplest kind of filter known as blur which uses Normalized Box filter. This filter is the simplest of all. Each output pixel is the mean of its kernel neighbors (all of them contribute with equal weights). The kernel is :

$$K = \frac{1}{K_{width} \cdot K_{height}} \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & 1 & 1 & ... & 1 \\ . & . & . & ... & 1 \\ . & . & . & ... & 1 \\ 1 & 1 & 1 & ... & 1 \end{bmatrix}$$

Figure 15 shows the result of applying preprocessing to the input image.

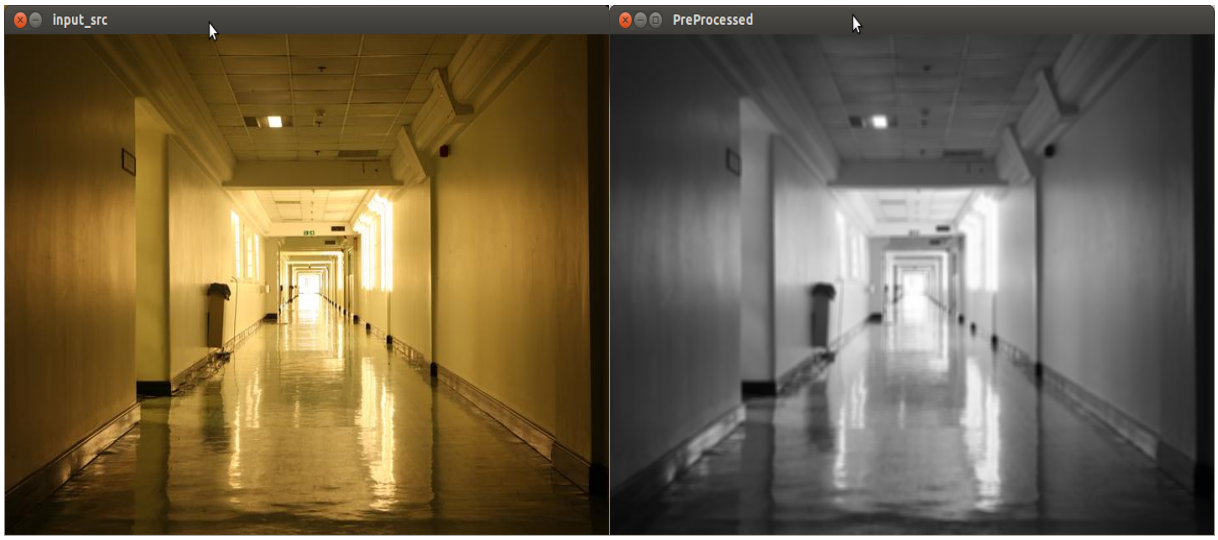


Figure 15: Result of preprocessing the image

6.3.1.1.2 Canny Operator The Canny Edge detector was developed by John F. Canny in 1986. Also known to many as the optimal detector. Canny algorithm aims to satisfy three main criteria: [53]

- Low error rate: Meaning a good detection of only existent edges.
- Good localization: The distance between edge pixels detected and real edge pixels have to be minimized.
- Minimal response: Only one detector response per edge.

Its principle in nature is to execute smooth image operation by using Gaussian function and calculating max derivative through first order differential operator. The process of edge detection is used to delete much useless information. In fact, it is used to detect points having obvious brightness variation, because these points include much information such as discontinuous depth and surface direction, change of material attribute, and scene illumination. This operator includes below 4 procedures.

- **Gaussian Filter De-noising:** Gaussian blur is a filter for image penumbra, and it calculates the conversion for each pixel in the image by convolving it with a Gaussian kernel and then summing them all to produce the output array.
  In the two-dimensional space, it can be expressed as:

$$G(u,v) = \frac{1}{2\pi\sigma^2} e^{\frac{-(u^2+v^2)}{2\sigma^2}} \tag{6}$$

where $\boldsymbol{\sigma}$ is the standard deviation of the above normal distribution, and it is used to control the smoothness of Gaussian filter parameters. The greater the width of the kernel, the more smoothing (i.e. noise reduction) is achieved. However, larger kernels result in a greater error in the edge location.

- **Gradient Operator Sobel:** This step finds the intensity gradient is of the image. This is achieved by taking the gradient of the image in the horizontal and vertical directions with the procedure analogous to Sobel operators[6]. It is a first order edge detection based on the use of first-order image derivatives. Adding the magnitude of these components gives the measure of the âedge strengthâ. The masks used by Canny edge detector in x and y directions:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

The edge strength or pixel gradient is defined as

$$|E(x,y)| = |\bar{G_x}(x,y)| + |\bar{G_y}(x,y)| \tag{7}$$

The first derivative of one two-dimension equation can be regarded as the gradient, and its vector can be expressed as,

$$\bar{G}(x,y) = \begin{bmatrix} \bar{G}_x \\ \bar{G}_y \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix} \tag{8}$$

The magnitude $|\bar{G}|$ and orientation θ of the image gradient are thus given by:

$$|\bar{G}(x,y)| = \sqrt{\bar{G}_x^2 + \bar{G}_y^2} \tag{9}$$

$$\theta = \texttt{arctan}\left(\frac{\bar{G}_y}{\bar{G}_x}\right) \tag{10}$$

Once the edge direction is known, we approximate it to an edge direction that can be traced in a digital image. Considering an arbitrary pixel, the direction of an edge through this pixel can take one of only four possible values -0°(neighbours to east and west),

[6]Sobel operator is an example of convolution over the image. The Sobel kernel implements differentiation in one direction and (approximate) Gaussian averaging in the other. The advantage of this is that it smooths the edge region, reducing the likelihood that noisy or isolated pixels will dominate the filter response.

90°(neighbours to north and south), 45°(neighbours to north-east and south-west) and 135°(neighbours to north-west and south-east). Accordingly, we approximate the calculated θ by whichever of these four angles is closest in value to it.

- **Control of Gradient Value:** This is also known as Nonmaximum suppression. After the edge directions are known, nonmaximum suppression is applied. This will give a thin line in the output image. This works by tracing along the edge in the edge direction and suppressing any pixel value (i.e. set it equal to zero) that is not considered to be an edge. In other words, The system compares the gray level and direction of current pixel with the pixel gradient value whose position is two pixels prior to current pixel, if it is smaller than the pixel gradient value of prior pixel, and then it would be set as 0, which means it is not an edge.

- **Partition of Lag Threshold/Hysteresis:** The final step is to track along the remaining pixels that have not been suppressed and threshold the image to identify the edge pixels. Critical to the Canny method, however, is the use of two distinct thresholds â a higher value T_2 and a lower value T_1. The selection of each pixel is then determined according to the following criteria:

  - if $|E(x, y)| < T_1$, then the pixel is rejected and is not an edge pixel
  - if $|E(x, y)| > T_2$, then the pixel is accepted and is an edge pixel
  - if $T_1 < |E(x, y)| < T_2$, the pixel is rejected except where a path consisting of edge pixels connects it to an unconditional edge pixel with $|E(x, y)| > T_2$.

  The system begins this process with a large threshold, firstly marks these luminance gradients having high edge probability, and then tracks them in the whole image by the calculated direction information. For the process of edge tracking, the system uses the low threshold to control tracking path and makes the tracking path finally return the beginning position along with the obscure part.

System stores the detected edge results in the form of binary image, and Figure 16 is the edge effect picture using the above edge detecting process. Experimental results show the basic edge information can be successfully extracted.

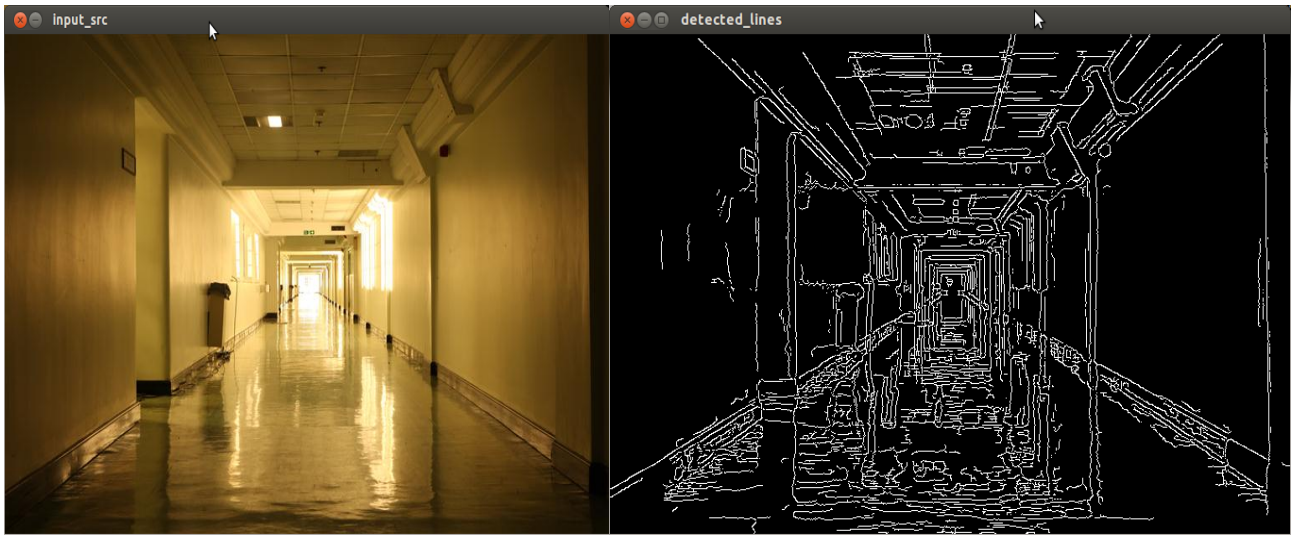


Figure 16: Detected edges using canny edge detector

6.3.1.2 Extracting lines by RHT

Sharp and long edges can be seen as lines. The Probabilistic Hough Transformation (PHT) is one of the most commonly used algorithms in perspective vision. The main idea used to detect straight lines is to establish a mapping between image coordinate space and parameter coordinate space. Many pixels in the image space are mapped into the parameter space, if these pixels are mapped into the same point in the parameter coordinate, and then the counter will be increased by 1. The points coordinate having largest counter value is corresponding to a straight line in the image space.
The algorithm is based on the parametric representation of a line:

$$\rho = x \cos\theta + y \sin\theta \tag{11}$$

where ρ is the perpendicular distance from the origin to the line and θ is the angle between the horizontal axis and the this perpendicular. The family of lines going through a given point (x_0, y_0) can be written as a set of pairs of (ρ_θ, θ). If for a given (x_0, y_0) we plot the family of lines that goes through it, we get a sinusoid. For instance, for $x_0 = 8$ and $y_0 = 6$ we get the following plot (in a plane θ - ρ)[54]:

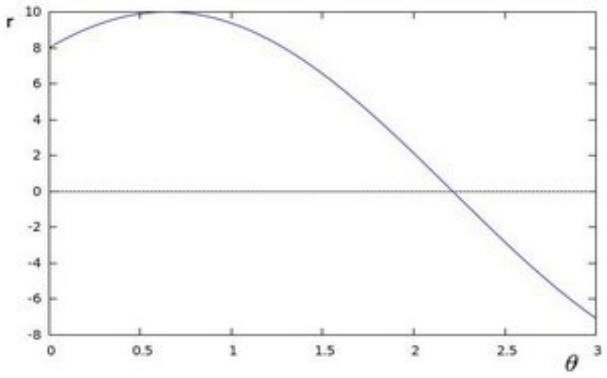


Figure 17: (ρ, θ) plot for points (x, y)

[Photo credit: Opencv documentation Hough Line Transform]

This set of lines can be represented as a sinusoidal, if $\rho > 0$ and $\theta \in (0, 2\pi)$. In Hough Transform, the same operation is done above for all the points in an image. If the curves of two different points intersect in the plane (θ - ρ), that means that both points belong to a same line. The algorithm searches intersections of sinusoidal curves. If the number of curves in the intersection is more than a threshold, then the pair of (ρ_θ, θ) is considered to be a line on the image. For example, as shown in figure 19, the three plots intersect in one single point (0.925, 9.6), these coordinates are the parameters (θ, ρ) or the line in which corresponding points to the plots lay.

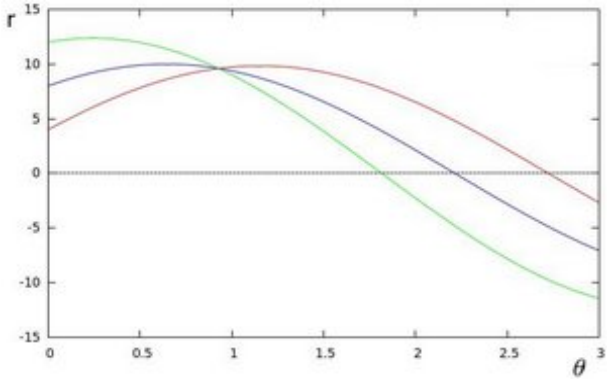


Figure 18: (ρ, θ) plots for 3 points (x, y) intersecting at one single point

[Photo credit: Opencv documentation Hough Line Transform]

In general, by Hough Transform a line can be detected by finding the number of intersections between curves. The more curves intersecting means that the line represented by that intersection have more points. In general, we can define a *threshold* of the minimum number of intersections needed to detect a line.

In our approach we are using RHT, the algorithm takes a random subset of points for line detection, thus optimizing the procedure. RHT method uses many-to-one mapping so as to avoid huge amount of calculations in the traditional Hough transformation (HT) method using one-to-many mapping. The system uses dynamic list structure and allocates corresponding memory space to these parameters only accumulated by the many-to-one mapping.[51]

Resultant image after lines extraction.

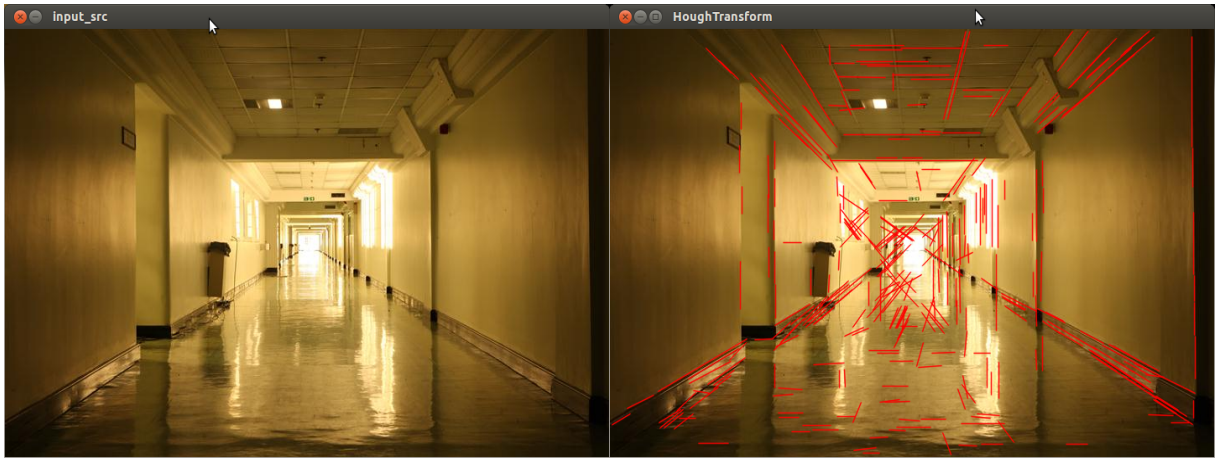


Figure 19: Results of Hough Line Transform

6.3.1.3 Deleting Unreasonable Straight Line

In an indoor environment, the detected lines from Hough Transform also include many vertical and horizontal edges. Since these lines do not converge to the VP and are not useful in detecting it, they are filtered in order to have a more accurate VP detection. In order to make the detection robust, we separate out the lines based on their orientation angle: we neglect the lines which are approximately horizontal and vertical. Most of the remaining lines will be the lines from walls cross section. so the most dense area of crossing points of these lines will be the vanishing point in the corridor. Figure 20 shows the fundamental behind deleting the unreasonable lines. Lines which have slope between $\theta_1 = \pm 10°$ or $\theta_2 = \pm 10°$ are deleted. These parameters can be tuned.

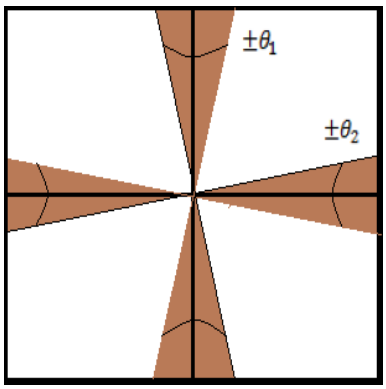


Figure 20: Unreasonable lines with slope between $\theta_1 = \pm 10°$ and $\theta_2 = \pm 10°$

6.3.1.4 Locating the Vanishing Point

After deleting undesirable lines, the number of lines in the line set is M, so we look for the region in the image which has the highest density of pairwise line intersections. To do this, we

divide the image plane into a 22×33 grid G and count how many line intersections fall into each grid element.

Let l_i is one of the lines $\in M$. This l_i can be represented by two parameters, m_i, b_i as $y = m_i x + b_i$. Consider another line $l_j \in M$, and the intersection point of l_i and l_j is (x_{ij}, y_{ij}). If the lines l_i and l_j do not intersect, let $(x_{ij}, y_{ij}) = (\infty, \infty)$. But if they intersect, then we can get the cross point of two lines.

We calculate all the intersections by the above equations, where is the number of all intersection points. Those intersection points whose coordinate does not exist in the range of $(0, 0)$ to (h, w) to are undesirable ones, and they should be deleted, where h and w are the height and weight of the image. Then we calculate the number of lines intersection falling in each grid element. Consider (a^*, b^*) be the grid element which has maximum number of intersections and $G(a, b)$ number of lines intersection falling in each grid element where $a \in [0, 22)$ and $b \in [0, 33)$. The grid with the maximum number of intersections is:

$$(a^*, b^*) = \texttt{argmax}_{ab} G(a, b) \tag{12}$$

and vanishing point is given as:

$$(x_{vp}, y_{vp}) = ((a^* + 0.5) * w/33, (b^* + 0.5) * h/22) \tag{13}$$

Firgure 21 shows the division of image in to grids of size 22×33 and the result of extracted vanishing point using edge detection method.

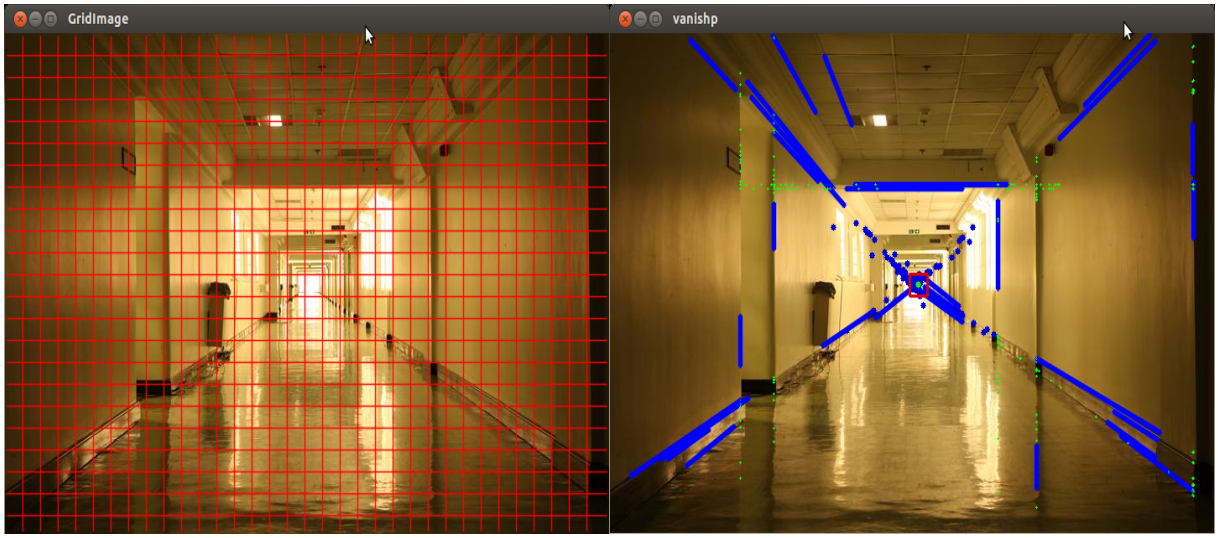


Figure 21: Divided image into a 22×33 grid and located vanishing point

We keep the track of vanishing point detected in previous frame and the output point is the current vanishing point half plus previous vanishing point half.

$$(x_{vp}, y_{vp}) = (current_{vp}.x/2 + previous_{vp}.x/2, current_{vp}.y/2 + previous_{vp}.y/2) \tag{14}$$

This can be further optimized by keeping track of last few output points and calculate average of all previous and latest detected point as the output of the current frame.

6.3.2 Method 2: Image density clustering

The second method used to find the vanishing point is based on the principle that lines those move away from us converge towards the center of the picture. In other words, density of points of intersection of lines is maximum towards the center. Considering this fact, we came up with another approach to find the VP.

6.3.2.1 Preprocessing the image

In this method we first convert the image in to grayscale. Then we apply the Sobel operator(explained in section 6.3.1) in both x and y direction to find out the edges. The output of Sobel operator is show in figure 22

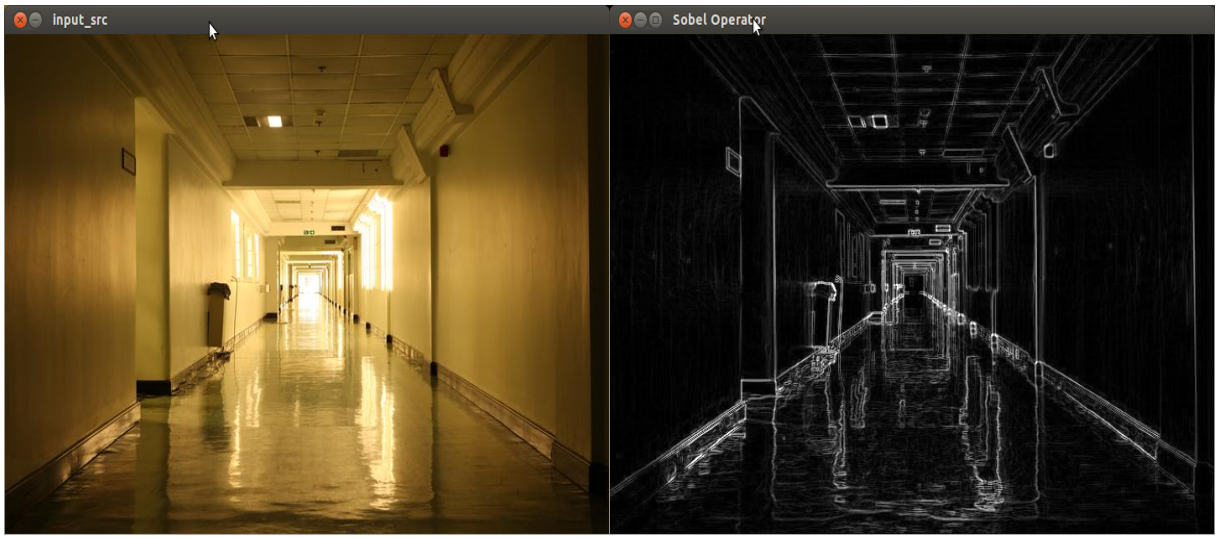


Figure 22: Output of sobel operator

6.3.2.2 Locating the Vanishing Point

As we can see from the output the density of pixels is more towards the center of the image. Considering, VP to be the point of maximum density, the aim is now to find that point. To achieve this, we started scrolling through the image from left to right if $w > h$ or top to bottom if $h > w$, where h and w represents height and width of image. We take a square box of side size equal to the one smaller between h and w and traverse through the image in the direction selected earlier. We keep track of density of points in each box and after the complete scroll we take the box with maximum density. This process is repeated 25 times and each time the

scrolling area is reduced to the box selected from the last scroll. This way we achieve the point of maximum density at the end of 25 iterations. Figure 25 shows the output of this approach.

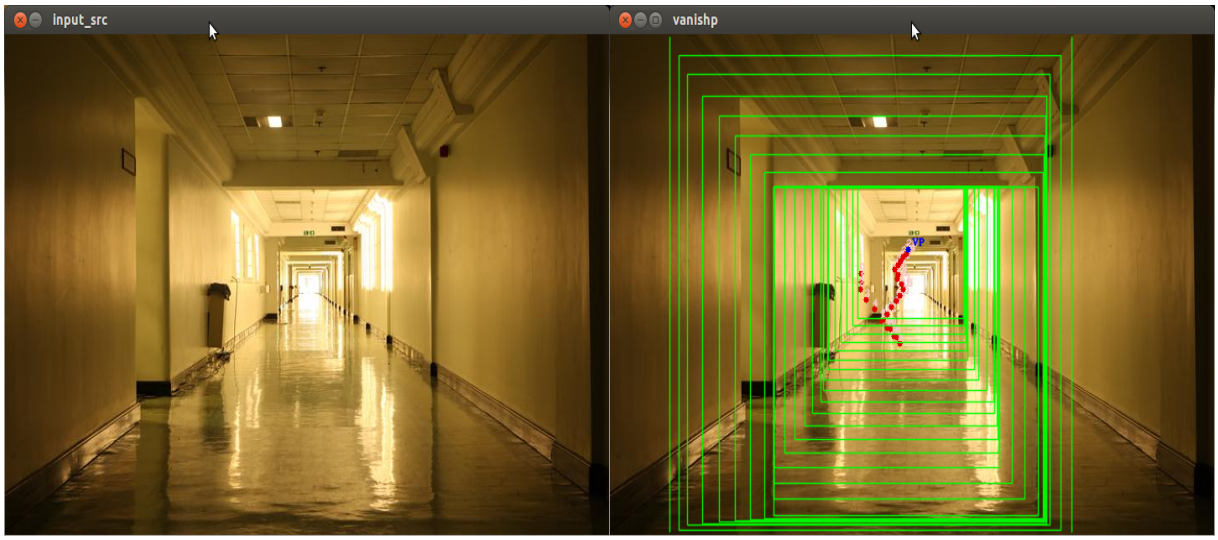


Figure 23: Output of sobel operator

Each red dot number $\in [1, 25]$ represents the center of the box with maximum density after each iteration. The dot in blue color is the output of the last iteration and its is the vanishing point of the image.

6.3.2.3 Integral Image to find densities

The problem in this approach was the calculation of the density of each box. As calculating intensity of each pixel in a box and then summing them is a time consuming process. To solve this problem, we use the concept of Integral Image.
An integral image helps you rapidly calculate summations over image subregions. Every pixel in an integral image is the summation of the pixels above and to the left of it. We can construct the integral image of a given image with only one pass over of the given image. Value s of a pixel (x, y) which is the sum of all the pixel values above, which is is calculated based in below equation,

$$s(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) + s(x - 1, y - 1) \tag{15}$$

Consider the given image below and its integral image:

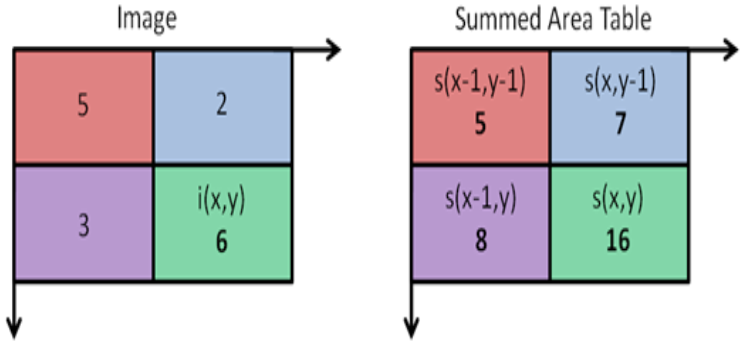


Figure 24: Integral image concept

[Photo credit: https://computersciencesource.wordpress.com]

On the left we have the given image, with its corresponding pixel values. On the right we have the images corresponding Summed Area Table. Integral image adds one extra row and column on top and left with values as O as shown in figure below.

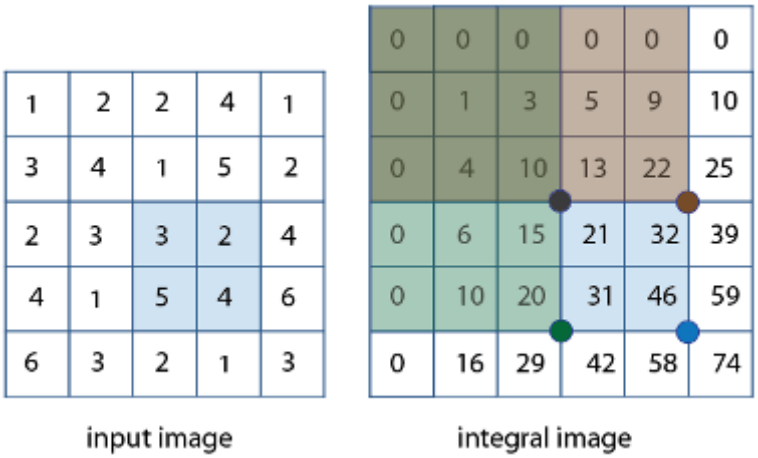


Figure 25: Example of integral image

[Photo credit: https://www.mathworks.com]

The advantage of Integral Image is the task of calculating the sum of pixels in some rectangle which is a subset of the original image can be done in constant time. This process have now $O(1)$ complexity. So to calculate the sum, we need 4 values from the Summed Area Table and then add or subtract them for the correct value of the sum of the pixels within that region. To do this, we use this equation:

$$i(x\prime, y\prime) = s(\texttt{left top}) + s(\texttt{bottom right}) - s(\texttt{top right}) - s(\texttt{bottom left}) \qquad (16)$$

6.4 Method selection for VP

Method 2 which uses Image density as a criteria for calculating the VP is faster in comparison to edge detector method and does not need undistortion of the input image, however it is unreliable

in many cases where we have objects lying around in the floor and also when no vanishing point exist. Figure 27 shows the elapsed time for both methods for the same set of frames.

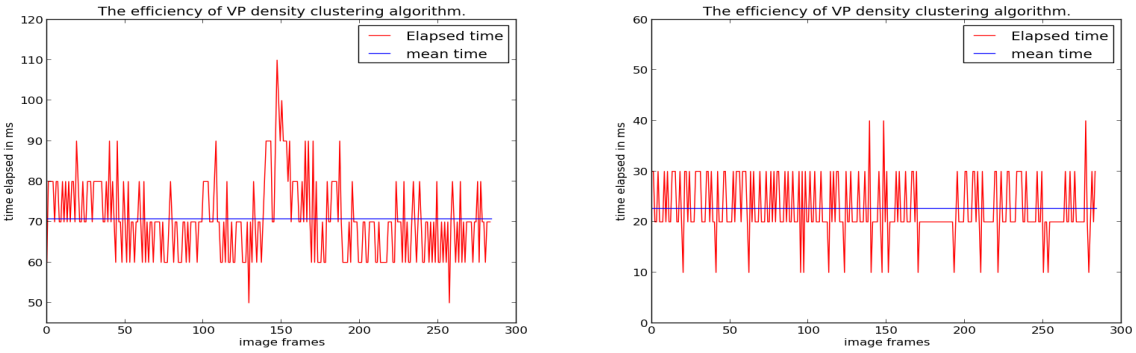


Figure 26: Time elapse by Edge detector method (left) and Density cluster(right)

VP detection based on density clustering is very fast in comparison with edge detection technique. Figure 27 shows the output of both methods when no Vanishing Point exists.

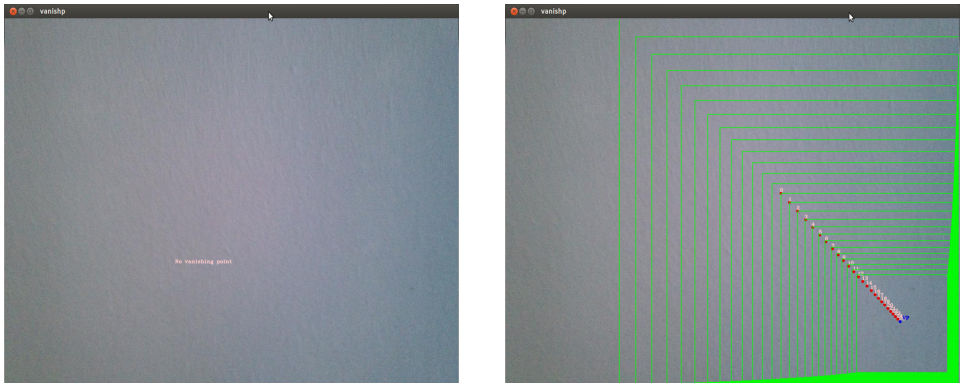


Figure 27: Output of Edge detector(on left) and Density cluster(on right) when no vanishing point exists

As we can see the density clustering technique gives us false result which needs to be improved in future. Therefore for this research we will use edge detection method for vanishing point detection.

6.5 Controller

The implementation of quad-rotor navigation is composed of image data collection and receiving control command, which forms a closed-loop control system. Besides, the control command calculated by PID control forms another closed-loop control processing as figure 28 shows.

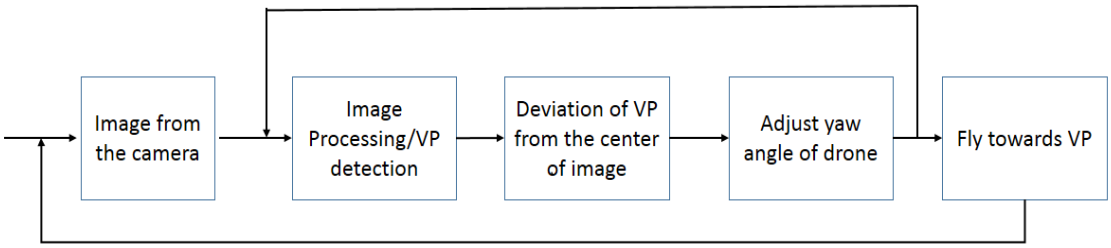


Figure 28: Quad-rotor navigation control.

The image processing output is the VP position x, and the reference value x_r is equal to half of the image width, 320 pixels. The control input is the yaw angular velocity while the drone is flying forward with a constant linear velocity. Vanishing point is the destination point for the quadrotor. The measured error $e(t)$ between a given setpoint $w(t)$ and the measured output of the system $y(t)$ is to be minimized over time. The goal is to quickly reach the desired setpoint and hold it without oscillating around it. A PID controller is used in our approach to directly control the quadrocopter. It is based on three separate control mechanisms, the control signal is weighted sum of all three terms:

- the **proportional** part depends on the current error $e(t)$.
- the **integral** part depends on the accumulated past error

$$\int_0^t e(\tau)\, d\tau$$

  .
- the **derivative** part depends on the predicted future error, based on the derivative of the error with respect to time $e(t)$

The PID controller now calculates the system input values according to

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d e(t) \tag{17}$$

where K_p, K_i and K_d are tunable parameters that typically are determined experimentally by the means of trial-and-error. The effect of these three parts of a PID-controller is explained below.

The Proportional Term

The proportional part is always required, and is the part responsible for reducing the error: the bigger the error, the stronger the control signal. In real-world systems however, a purely proportional controller causes severe overshoot, leading to strong oscillations.

The Derivative Term

The derivative part has the effect of dampening occurring oscillations: the higher the rate of

change of the error, the more this term contributes towards slowing down this rate of change, reducing overshoot and oscillations.

The Integral Term

The integral part is responsible for eliminating steady-state errors: for a biased system requiring a constant control input to hold a state, a pure PD-controller will settle above or below the setpoint. Depending on accumulated past error, the integral term compensates for this bias - it however needs to be treated with caution as it may increases convergence time and cause strong oscillations.

In our approach, for angular and linear velocities we use two different controllers. Angular velocity is given based on latest vanishing point found, however the linear velocities consider the latest velocities calculated by the controller based on the vanishing point and the current linear velocities.

For the safety of the hardware and environment, vanishing point driven velocities for the drone can be interrupted by the joystick anytime. As soon as the command is issued by the joystick, vanishing point driven velocities are suppressed by the joystick controller velocities. This functionality can be enabled or disabled by a specific button on the joystick which can be set according to user preference.

Figure 29: Joystick Controller

Table 2: Joystick settings in our experiment

Button/Axes	**Function**
Button 1	Emergency
Button 2	Land
Button 3	Takeoff
Button 5	Increase altitude
Button 6	Decrease altitude
Left Axes Forward/Backward	Pitch
Left Axes Left/Right	Roll
Right Axes Forward/Backward	Altitude
Right Axes Left/Right	Yaw
Button 9	Enable/Disable Vanishing Point Velocities
Button 10	Initialize joystick

These settings can be changed in the launch file of the joystick ROS node. In our experiment, once all the ROS nodes are running, the joystick is initialized by a joystick button. User presses

the Takeoff button, as soon as the drone takeoff, the vanishing point node start sending the directions. Once the user presses the vanishing point driven velocities enable button, the drone starts navigating towards the vanishing point autonomously. Figure 30 shows the state diagram of the drone.

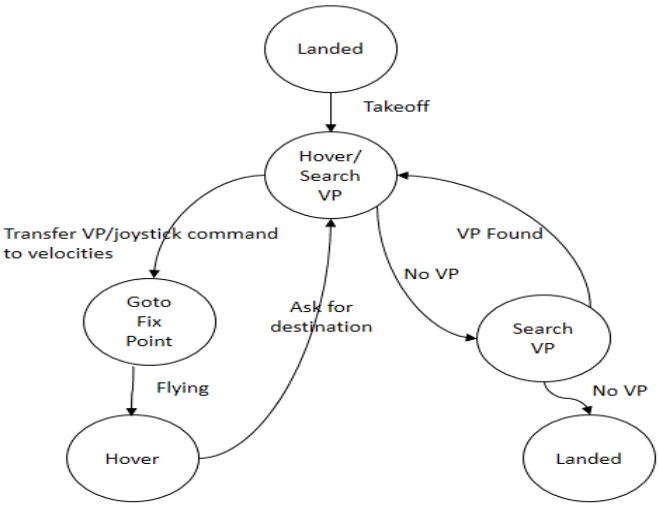


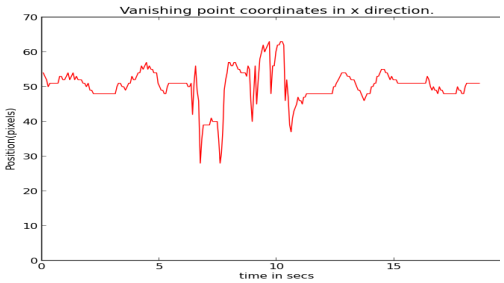
Figure 30: State diagram of drone

7 Evaluation

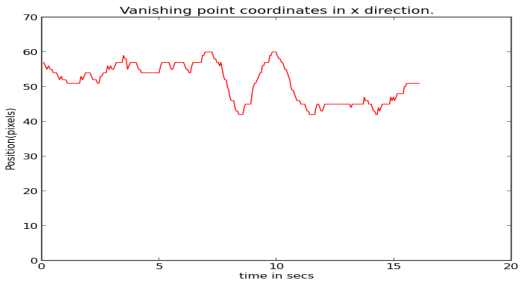
In this section we are going to present the experimental results of the detection and tracking methods based on real data sets.

7.1 VP Detection

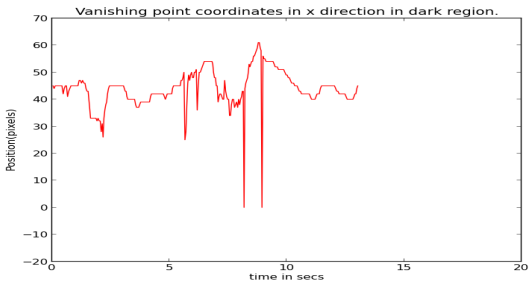
We tested the edge detection based VP method in several corridors, for example, narrow, broad, corridors with obstacles and dark corridors. For edge based detection, we also tried to use Laplace filter on a dataset of stored images from drone. Though the results from Laplace filter were better than Sobel or Canny but the computation time using Laplace is very high. So we used Canny operator in our implementation. The results shown in figure 31 displays the variance in the position of vanishing point in horizontal direction. The two graphs are the results from the same environment but with varied motion. Figure 31a shows a huge variance in VP at some points for example at time 6 secs, because we tilted the drone (roll) to a huge angle or moved the drone to extreme right or left to test the robustness of the algorithm however in figure 31b, the VP is quite stable as this was smooth flight without sharp maneuvers. These results are from the images taken from the drone camera but off flight.



(a) with sharp maneuvers



(b) without sharp maneuvers

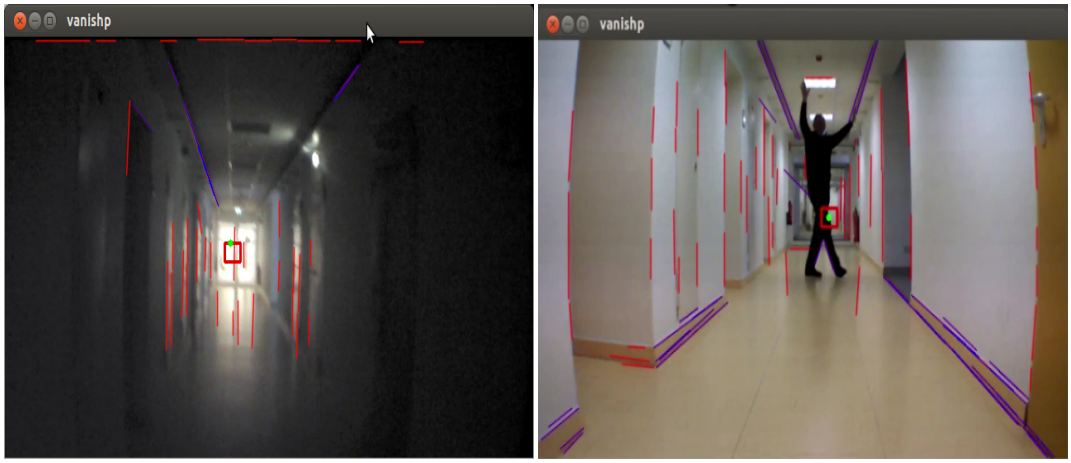


(c) [Dark Corridor(bad illumination)

Figure 31: Variance in VP position

From the above results, we can see the VP detection fails when there is huge rolling angle, to prevent these sharp maneuvers we need to make our algorithm and controller more robust to handle such problem. However the results on a flight without sharp maneuvers are are satisfactory and

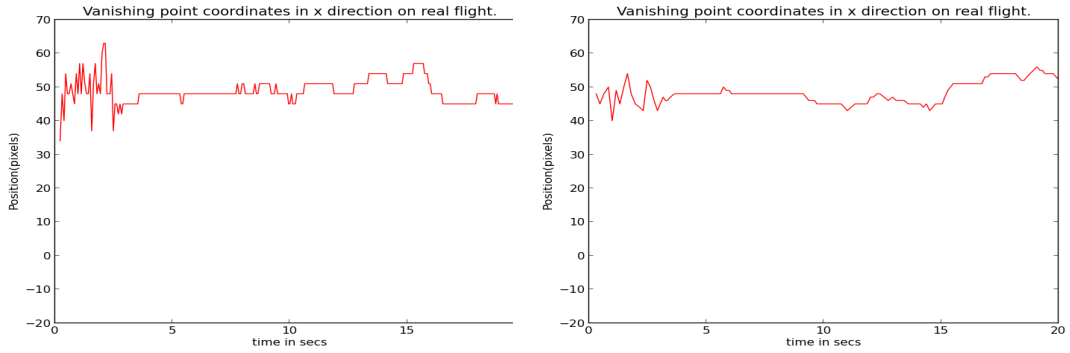
vanishing point is stable. Figure 32a shows a dark corridor example and figure 31c displays the results of the vanishing point detection in this place. In this case, vanishing point jumps very often and is unstable due to lack of features. In future, algorithm needs to be updated to handle similar situations. The current system does not consider obstacles, so if there are enough lines to calculate the vanishing point, drones keeps flying towards it as shown in figure 32b.



(a) Dark corridor example (b) VP with obstacle

Figure 32: Different scenarios of corridor for testing

Figure 33 shows the variance in position of vanishing point on real flight in different corridors. In the beginning of the flight, drone tries to align itself to bring the vanishing in the center of the image, therefore we can see the drifts at the beginning.



(a) corridor 1 (b) corridor 2

Figure 33: Vanishing point horizontal position on real flight in different floors

7.2 Controller

Here we evaluate our controller. We make the auto mode enable for the drone only once the drone is stable after the take off. If the drone is not stable at the beginning, the system fails in the auto mode. The controller publishes the velocities at 50Hz based on the vanishing point

found. We evaluate the controller based on the smoothness of trajectory followed by the drone at linear velocities given to it by the controller. The linear velocities are x(pitch) and y(roll) for the drone. Figure 34 shows that the velocities given are satisfactory considering the dynamics of the drone.

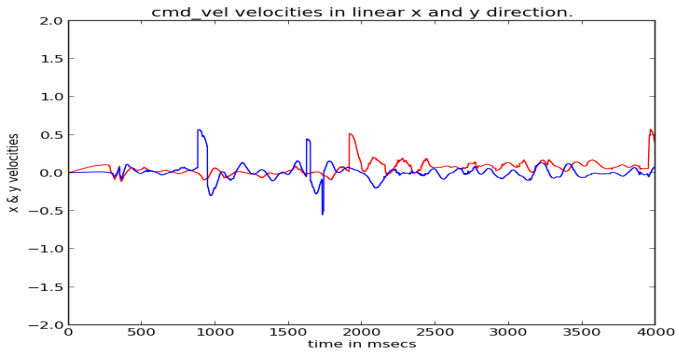


Figure 34: Linear roll and pitch velocities on real flight

Figure 35 shows the angular velocities. This also shows the stability in vanishing point detection except at certain point for ex at 1627 ms however drone drone recovered its direction as shown in the figure.

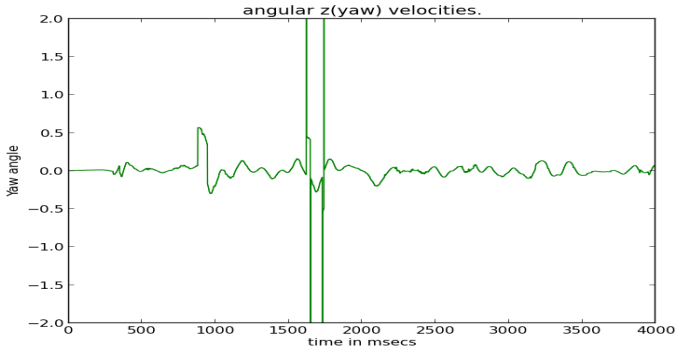


Figure 35: Angular velocities of the drone during flight

To test the PID values for the controller we used classical tuning method Ziegler-Nichols[53] closed-loop tuning method to tune the parameter. From the analysis by providing different PID values, we observe that only the proportional controller is required to calculate the angular velocity and PD for linear velocities, integral term made the system unstable. After a lot of tuning, we found following values of gains for Linear and angular velocities.

Table 3: K_p values for Linear x,y,z and angular z

Velocity Type	K_p	K_i	K_d
Linear x	0.50	0.0	0.50
Linear y	0.50	0.0	0.50
Angular z	0.40	0.0	0.0

Figure 36 shows the stable drone in takeoff state using the above stated PID values.

Figure 36: Drone in Take off state

The results stated above can be verified through the videos provided along with the resources. These videos show the flying drone under different scenarios for example obstacle in view, two different corridors and third person video of the flying drone.

8 Conclusion

We presented a system design and methodology that enables autonomous navigation with real-time performance on a mobile processor using only on-board sensors. In this work we propose two vanishing point based autonomous flying methods. We developed two methods to control a quadrotor UAV in a corridor, using images from its front camera. These proposed approaches can be used for UAV based first time responders in disaster ruined buildings.

The proposed methods are simple and based on onboard robust monocular camera. The aim is to find the vanishing point by using least computationally expensive image processing technique and make the drone fly towards the vanishing point. Out of the two image processing approaches proposed, one is based on edge detection principle. The other method is based on density clustering of the pixels in a Sobel output image. Latter stated method is faster in comparison to the former, however this still needs to be worked upon and made more robust in cases of no vanishing point. The edge based method is more robust and reliable, however in our implementation, vanishing point is not stable sometimes but gives satisfactory results to fly the drone.

We developed a PID controller which stabilize the VP to the center of the image, and make drone fly towards it. The controller listens to autonomous commands from vanishing point and also can be overridden by joystick velocities in case of joystick interrupt. The selected approach was tested in different indoor environment, we evaluated the our implemented approach based on stability, robustness and reliability to find the correct vanishing point. The experiments showed that this system is sufficient for simple corridors with out dynamic obstacles, with day light illumination and environment has enough edges visible to calculate the vanishing point.

9 Future work

While the system that we have developed is very basic, indoor flight is by no means a solved problem. As it stands right now, the system provides a functionality that would enable a UAV to fly autonomously in a corridor using only the perspective clues from the front camera images. However, there is still a tremendous amount of work to be done, to improve the robustness and autonomy capabilities of this indoor system.

- **Edge based VP detection:** The detection of vanishing point through edge detection approach still gives some bad results. Vanishing point is sometimes found on the edge which is not accepted for drone to fly. This needs to be improved and made more reliable. As stated before this system is very basic, it does not consider dynamic obstacles. Researchers have been working to make a complete robust navigation system for UAVs as like for ground robots, in future, this approach can be improved to consider dynamic obstacles and can help in developing better navigation system.
- **Density based VP detection:** The second approach based on density clustering seems promising as it is computationally very less demanding and fast in comparison to other techniques including edge based detection. Currently this approach fails when there is any obstacle in the path and when there is no vanishing point, it gives faulty results. Further work is needed to make this approach useful for drone flying.
- **Onboard Computation:** The current system setup performs most of the computation offboard at the ground-station. This is a major limitation of the system. Using offboard computation limits the effective range of the vehicle since the bandwidth required to send the sensor data to the ground-station is more than long-range commercially available wireless links currently provide. We need to test this approach on Asctec Pelican which was the primary aim of this research. While the above stated approaches still need to be improved and we still need to find the compatible software interface to integrate ROS, opencv and Asctec Pelican actuators.

References

[1] Carmen. http://carmen.sourceforge.net/

[2] ROS (Robot Open Source) http://www.ros.org

[3] Slawomir Grzonka, Giorgio Grisetti, Wolfram Burgard, "A Fully Autonomous Indoor Quadrotor", IEEE Transactions on Robotics, vol. 28, NO. 1, February 2012

[4] Bills, C. ; Chen, J. ; Saxena, A., "Autonomous MAV flight in indoor environments using single image perspective cues", Robotics and Automation (ICRA), 2011 IEEE International Conference on 9-13 May 2011, pages = 5776 - 5783.

[5] http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[6] S. Bouabdallah, C. Bermes, S. Grzonka, C. Gimkiewicz, A. Brenzikofer, R. Hahn, D. Schafroth,G.Grisetti,W. Burgard, and R. Siegwart, "Towards palm-size autonomous helicopters", Int. Conf. Exhib. Unmanned Areal Veh., Dubai, UAE, 2010.

[7] M. Bloandsch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments", in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), May 2010, pp. 21-28.

[8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments", in SPIE Unmanned Systems Technology XI, 2009.

[9] T. Krajnk, V. Vonasek, D. Fiser, and J. Faigl, "AR-Drone as a Robotic Platform for Research and Education", in International Conference on Research and Education in Robotics. Prague: Springer, 2011.

[10] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor", IEEE Trans. on Robotics, vol. 28, no. 1, pp. 90-100, 2012.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. "Robust Monte Carlo localization for mobile robots". Artificial Intelligence,pg 99-141, 2000.

[12] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback", in Proc. IEEE Int. Conf. Robot. Autom., 2002.

[13] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor", in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2007.

[14] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments", Int. J. Micro Air Veh., vol. 1, no. 4, 2009.

[15] S. Bouabdallah, C. Bermes, S. Grzonka, C. Gimkiewicz, A. Brenzikofer, R. Hahn, D. Schafroth,G.Grisetti,W. Burgard, and R. Siegwart, "Towards palm-size autonomous helicopters", presented at the Int. Conf. Exhib. Unmanned Areal Veh., Dubai, UAE, 2010.

[16] T. Templeton, D.H. Shim, C. Geyer, and S.S. Sastry. "Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft", In Proc. ICRA, pages 1349-1356, 2007.

[17] A. Bachrach, A. Garamifard, D. Gurdan, R. He, S. Prentice, J. Stumpf, and N. Roy. Coordinated tracking and planning using air and ground vehicles. In Proc. ISER, 2008.

[18] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli. "Flying Fast and Low Among Obstacles". In Proc. ICRA, pages 2023-2029, 2007.

[19] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. "Learning for control from multiple demonstrations", In Proceedings of the 25th International Conference on Machine Learning, pages 144-151, New York, NY, USA, 2008. ACM.

[20] Templeton, D.H. Shim, C. Geyer, and S.S. Sastry. Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft. In Proc. ICRA, pages 1349-1356, 2007.

[21] A. Bachrach, A. Garamifard, D. Gurdan, R. He, S. Prentice, J. Stumpf, and N. Roy. Coordinated tracking and planning using air and ground vehicles. In Proc. ISER,2008.

[22] T. Furukawa, F. Bourgault, B. Lavis, and H.F. Durrant-Whyte. Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In Proceedings of IEEE International Conference on Robotics and Automation, pages 2521-2526,May 2006.

[23] J. Tisdale, A. Ryan, Z. Kim, D. Tornqvist, and J.K. Hedrick. A multiple UAV system for vision-based search and localization, 2008.

[24] B. Ajith Kumar and D. Ghose, Radar-assisted collision avoidance, guidance strategy for planar flight, Aerospace and Electronic Systems, IEEE Transactions on, vol. 37, pp. 77-90, Jan 2001.

[25] Y. Kwag and J. Kang, Obstacle awareness and collision avoidance radar sensor system for low-altitude flying smart uav, Digital Avionics Systems Conference, 2004. DASC 04. The 23rd, vol. 2, pp.12.D.2,121-10 Vol.2, 24-28 Oct. 2004.

[26] J. B. Saunders, O. Call, A. Curtis, A. W. Beard, and T. W. Mclain, Static and dynamic obstacle avoidance in miniature air vehicles, in Proceedings of the Infotech@Aerospace Conference, pp. 2005-6950, 2005.

[27] J.P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. Control Systems Magazine, IEEE, 28(2):51-64, 2008.

[28] G.M. Hoffmann, H. Huang, S.L. Waslander, and C.J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In Proc. of GNC, Hilton Head, SC, August 2007.

[29] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments, in SPIE Unmanned Systems Technology XI, 2009.

[30] T.Templeton, D. H. Shim,C.Geyer, and S. S. Sastry, Autonomous vision based landing and terrain mapping using an MPC-controlled unmanned rotorcraft, in Proc. IEEE Int. Conf. Robot. Autom., 2007, pp. 1349-1356

[31] G. P. Tournier,M.Valenti, J. P. How, and E. Feron, Estimation and control of a quadrotor vehicle usingmonocular vision andmoire patterns, in Proc. AIAA Guid., Navigat. Control Conf. Exhibit., 2006, pp. 21-24.

[32] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle, IEEE Trans. Robot., vol. 25, no. 3, pp. 743-749, Jun. 2009.

[33] S. Soundararaj, A. Sujeeth, and A. Saxena, Autonomous indoor helicopter flight using a single onboard camera, in IROS, 2009.

[34] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet, Visual navigation of a quadrotor aerial vehicle, in IROS, 2009.

[35] R. Mori, K. Hirata, and T. Kinoshita, Vision-based guidance control of a small-scale unmanned helicopter, in IROS, 2007.

[36] G.P. Tournier, M. Valenti, J.P. How, and E. Feron., Estimation and control of a quadrotor vehicle using monocular vision and moire patterns., In Proc. of AIAA GNC, Keystone, Colorado, 2006.

[37] N.G. Johnson., Vision-assisted control of a hovering air vehicle in an indoor setting., Masters thesis, BYU, 2008.

[38] C. Kemp. Visual Control of a Miniature Quad-Rotor Helicopter. PhD thesis, Churchill College, University of Cambridge, 2006.

[39] K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani, Monocular vision slam for indoor aerial vehicles, in IROS, 2009.

[40] D. G. Sim, R. H. Park, R. C. Kim, S. U. Lee, and I. C. Kim, Integrated position estimation using aerial image sequences, IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 1, pp. 1-18, 2002.

[41] M. Cummins and P. Newmann, Fab-map: Probabilistic localisation and mapping in the space of appearance, International Journal of Robotics Research, vol. 27, no. 6, pp. 647-665, 2008.

[42] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard, Visual slam for flying vehicles, IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1088-1093, 2008.

[43] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, MAV navi- gation through indoor corridors using optical flow, in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2010

[44] J. Jimenez and A. Zell, "Framework for autonomous onboard navigation with the a.r.drone", in Proceedings of the International Conference on Unmanned Aerial Systems (ICUAS), 2013.

[45] http://drawsketch.about.com/od/drawingglossary/g/vanishingpoint.htm.

[46] https://github.com/AutonomyLab/ardrone_autonomy/blob/master/README.md

[47] David G. Lowe. Distinctive image features from scale-invariant keypoints. Interna- tional Journal of Computer Vision, 60:91â110, 2004.

[48] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Proceedings of European Conference on Computer Vision, pages 404â417, 2006.

[49] F. Schaffalitzky and A. Zisserman, âPlanar grouping for automatic detection of vanishing lines and points,â Image and Vision Computing, vol. 18, no. 9, pp. 647â658, 2000.

[50] Pierre-Jean Bristeau, Franois Callou, David Vissire, and Nicolas Petit. The navigation and control technology inside the ar.drone micro uav, 2011.

[51] http://docs.opencv.org/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[52] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

[53] https://controls.engin.umich.edu/wiki/index.php/PIDTuningClassical

[54] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html