Autonomous UAV Navigation in Unknown Indoor Environment using Onboard Light Weight Camera

part of

TRADR - Teaming for Robot-Assisted Disaster Response

28th April 2015

Devvrat Arya
Masters in Autonomous System
Hochshule Bonn Rhein Sieg

Supervisors:

Dr. rer. nat. Björn Kahl
Hochshule Bonn Rhein Sieg
Sankt Augustin, Germany.
bjoern.kahl@h-brs.de

Mr. Rainer Worst
Fraunhofer- IAIS
Sankt Augustin, Germany.
rainer.worst@iais.fraunhofer.de

Contents

1. Introduction
2. Related Work
3. Problem handled
4. Setup and Platform
5. Approach
6. Experimental results
7. Future work

Motivation

- Recognize the scope and extent of the disaster damage over the affected area.
- Robots and UAVs can be of great help to analyze the scene.
- This can help to most effectively begin rescue and disaster recovery activities.

Motivation

- Functioning in collapsed building, requires robot to traverse cluttered and uneven environment.
- Ground robot can traverse rough terrain but in terrains with levels it is not physically capable.
- UAV is an alternative robotic platform for rescue tasks and a host of other applications.

Ground robots that can traverse through rough surfaces

Objective

Constraints with UAVs:

- Payload
- Computational limitations
- Fast dynamics

The objective of this R&D is to design and implement a basic navigation system for UAVs which uses only on-board light weight camera, requires minimum computation and less power consuming.

Related Work

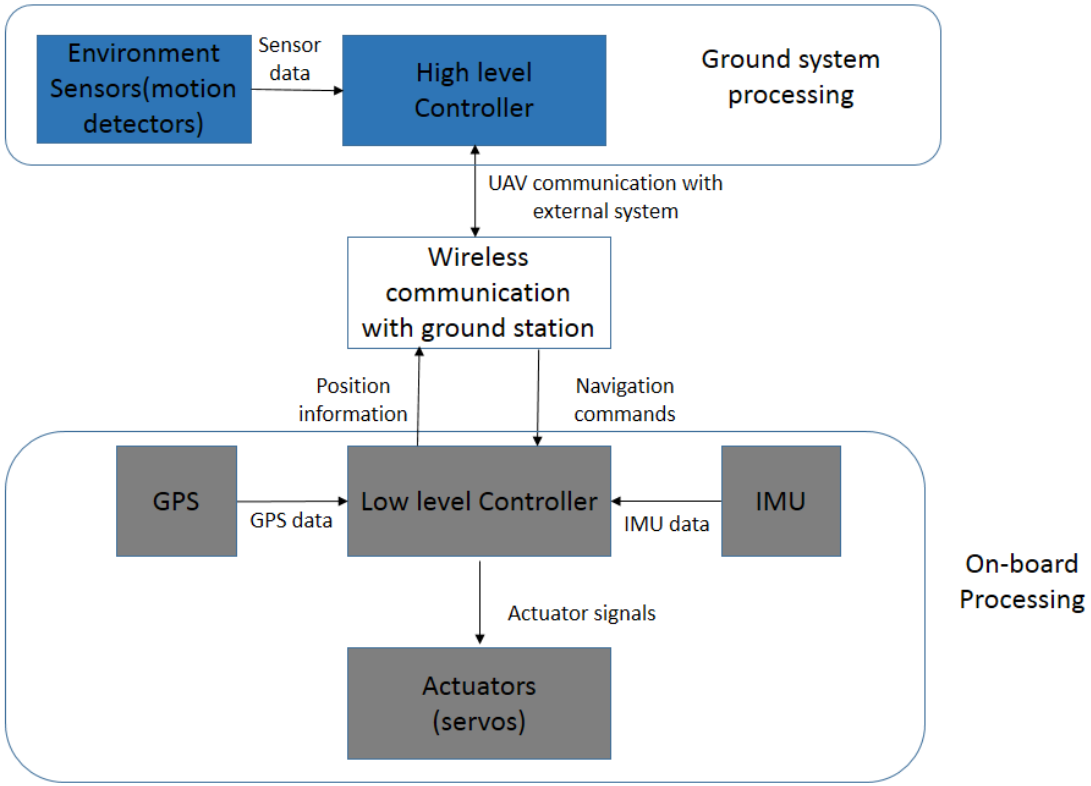
- Non Vision sensors based indoor UAV navigation:
  - Kumar & Ghose and Kwag & Kang implemented radar based navigation and obstacle avoidance.
  - Saunders used a forward looking laser range finder for path planning.

These approaches lack in heavy weight or high power consumption.

- Vision sensors based Unknown indoor UAV navigation:
  - Saxena and Courbon used vision to fly in known environments based on visual databases.
  - Chao, Yu Gu and Napolitano presented optical flow techniques for UAV navigation.
  - Conte and Patrick used inertial sensors and visual odometry.

These approaches would not apply in many indoor environments that are devoid of track-able features.

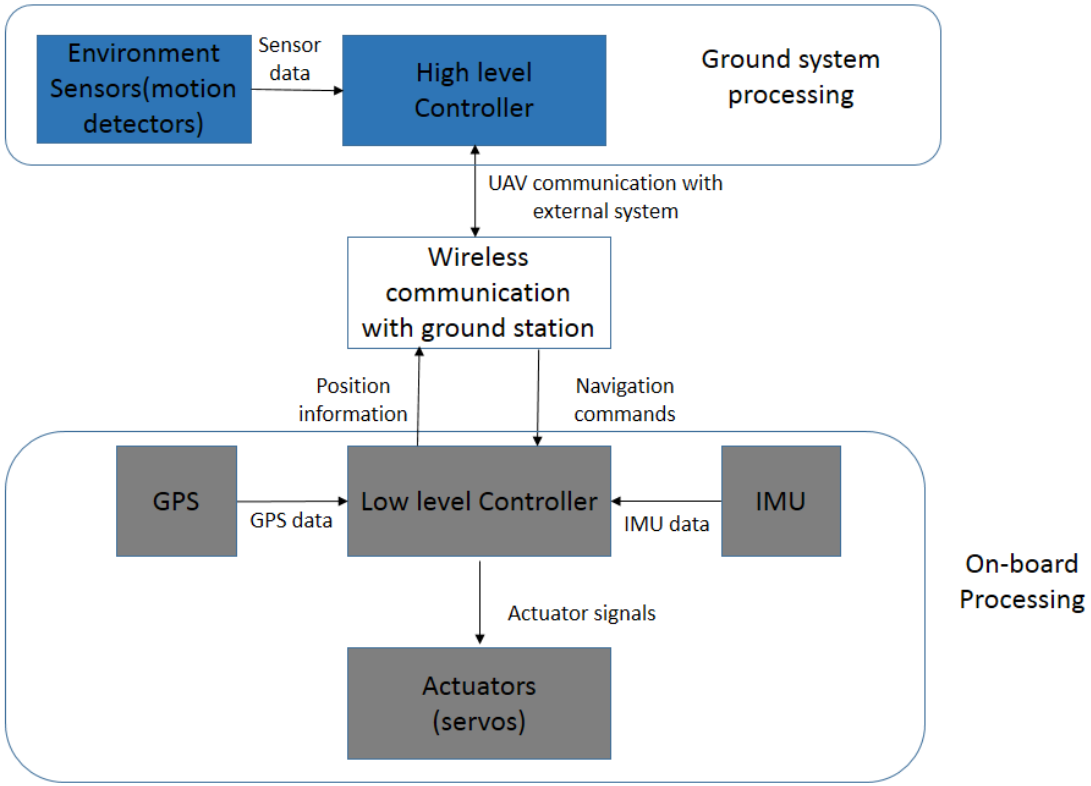
How to handle computational limitations?



Basic drone architecture with computation on ground system

How to handle computational limitations?

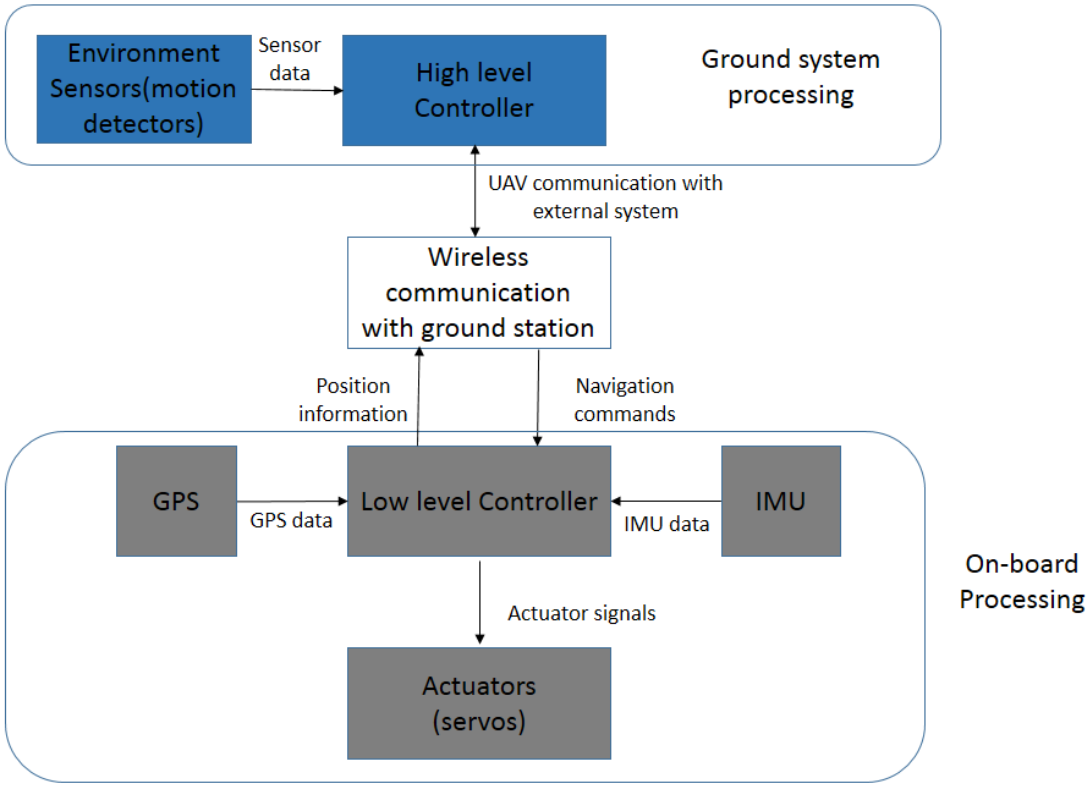
- Receive data from environment sensors(Motion trackers or camera) and UAV(IMU).



Basic drone architecture with computation on ground system

How to handle computational limitations?

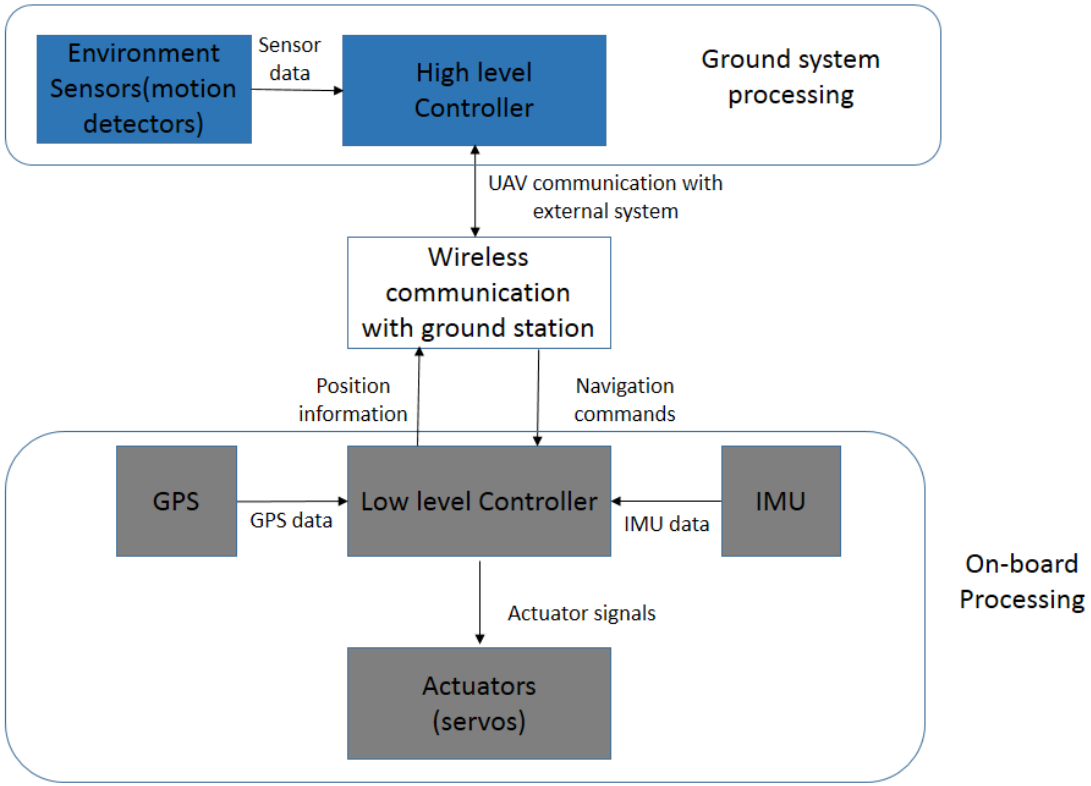
- Receive data from environment sensors(Motion trackers or camera) and UAV(IMU).
- Run autonomy algorithms(navigation algorithms) on external system(desktop computers).



Basic drone architecture with computation on ground system

How to handle computational limitations?

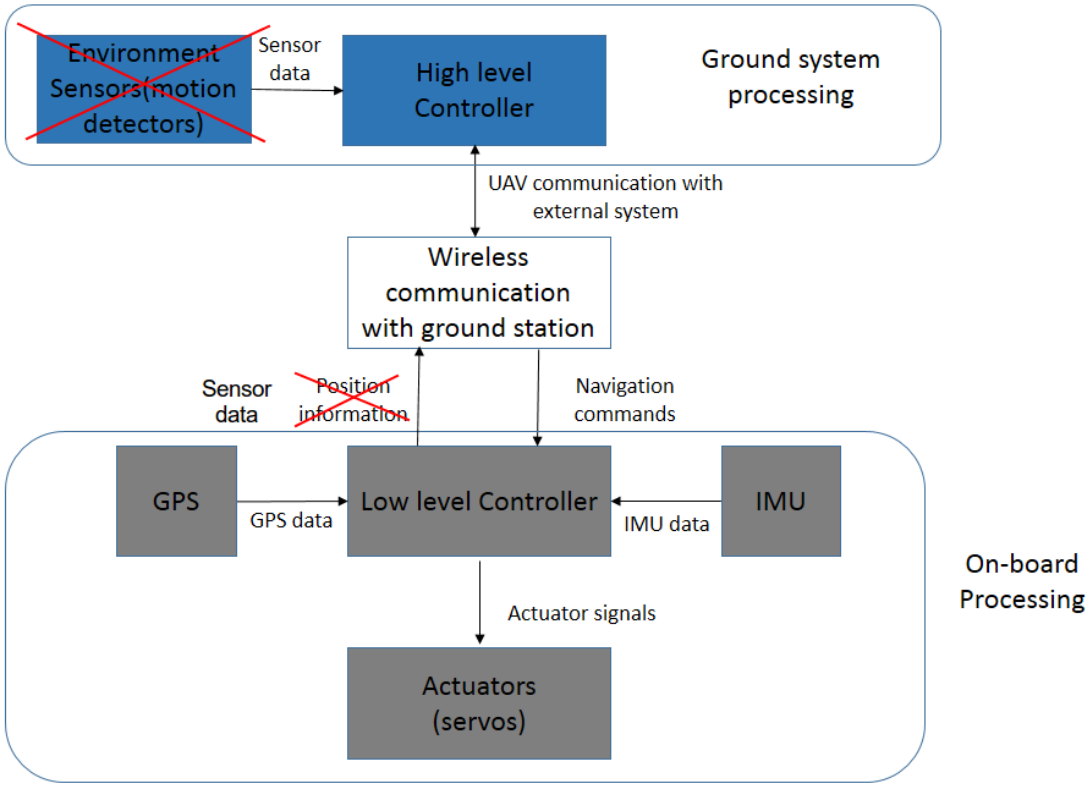
- Receive data from environment sensors(Motion trackers or camera) and UAV(IMU).
- Run autonomy algorithms(navigation algorithms) on external system(desktop computers).
- Send navigation commands back to UAV



Basic drone architecture with computation on ground system

How to handle computational limitations?

- Receive sensor data(image cues) from UAV.
- Run autonomy algorithms(navigation algorithms) on external system(desktop computers).
- Send navigation commands back to UAV



Basic drone architecture with computation on ground system

Problem handled

Design and implement a navigation system prototype for UAV which:

- enables a UAV to autonomously explore corridor environments
- uses only on-board sensors and works without prior knowledge of the environment
- less computationally expensive (algorithm able to run onboard)
- based on simple features extraction
- suited for long-term navigation.



Corridor sample images

Setup and Platform

Primary platform is the Parrot AR.Drone 2.0 quadrotor:

- first implement our approach on the Parrot A.R. Drone
- less computational capabilities on-board, therefore computation happens on ground system
- algorithm runs on host machine –Lenovo Y560 (Intel®Core i7 CPU Q720@1.60GHzx8, 4GB RAM), running Ubuntu 12.04
- commands and images are exchanged via WiFi between host machine and AR.Drone

If successful, implement the approach on **AscTec Pelican quadrotor**, *TRADR project target platform. AscTec Pelican has computational power similar to our host machine.*

Setup and Platform

AscTec Pelican

Technical Data	AscTec Pelican
UAV Type	Quadcopter
Onboard computer	Up to 3rd Generation Intel® Core i7 processor
Size	700 x 700 x 500 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
Sensors	HD Camera, Hokuyo laser scanner,...

Parrot AR.Drone 2.0

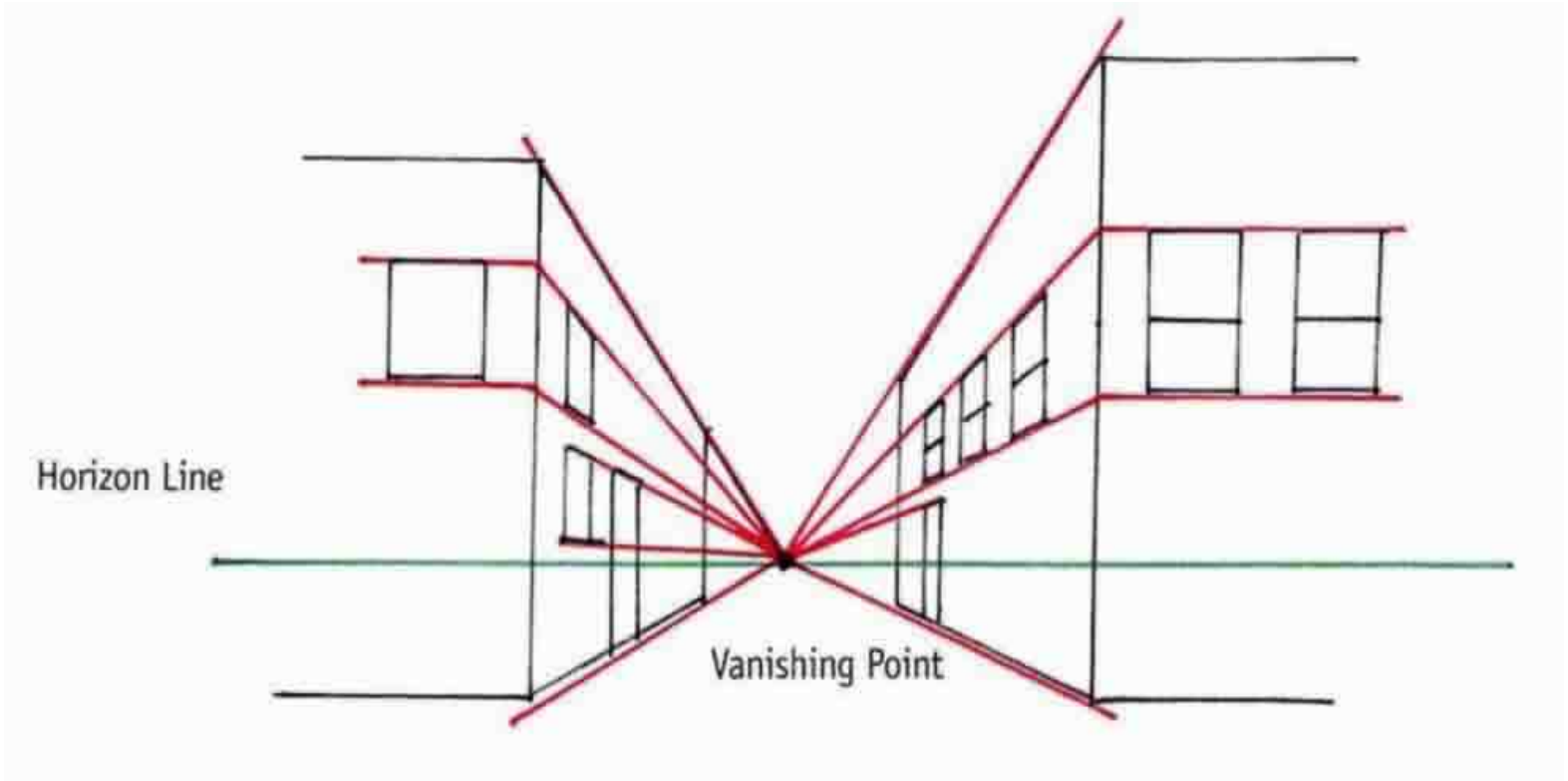
Technical Data	Parrot AR Drone 2
UAV Type	Quadcopter
Onboard computer	1 GHz ARM Cortex-A8 CPU
Size	670 x 670 x 125 mm
Max. take off weight	380grams
Max. payload	250grams(unstable)
Flight time incl. payload	15 min.
Wireless communication	802.11n WiFi
Sensors	Front HD 720p camera, Bottom VGA Camera,IMU,...

Approach Overview

- Indoor environments comprise of long straight parallel lines, and from UAVs perspective this has unique visual cues.
- Ends of the corridor are observed as vanishing points(VP) in images.
- In indoor environments, VP can be found consistently and hence can be used to locate the end of the corridor.
- If we have a VP, we command the drone to moves towards it.
- if no VP is found, drone rotates towards left until it founds a VP.

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.

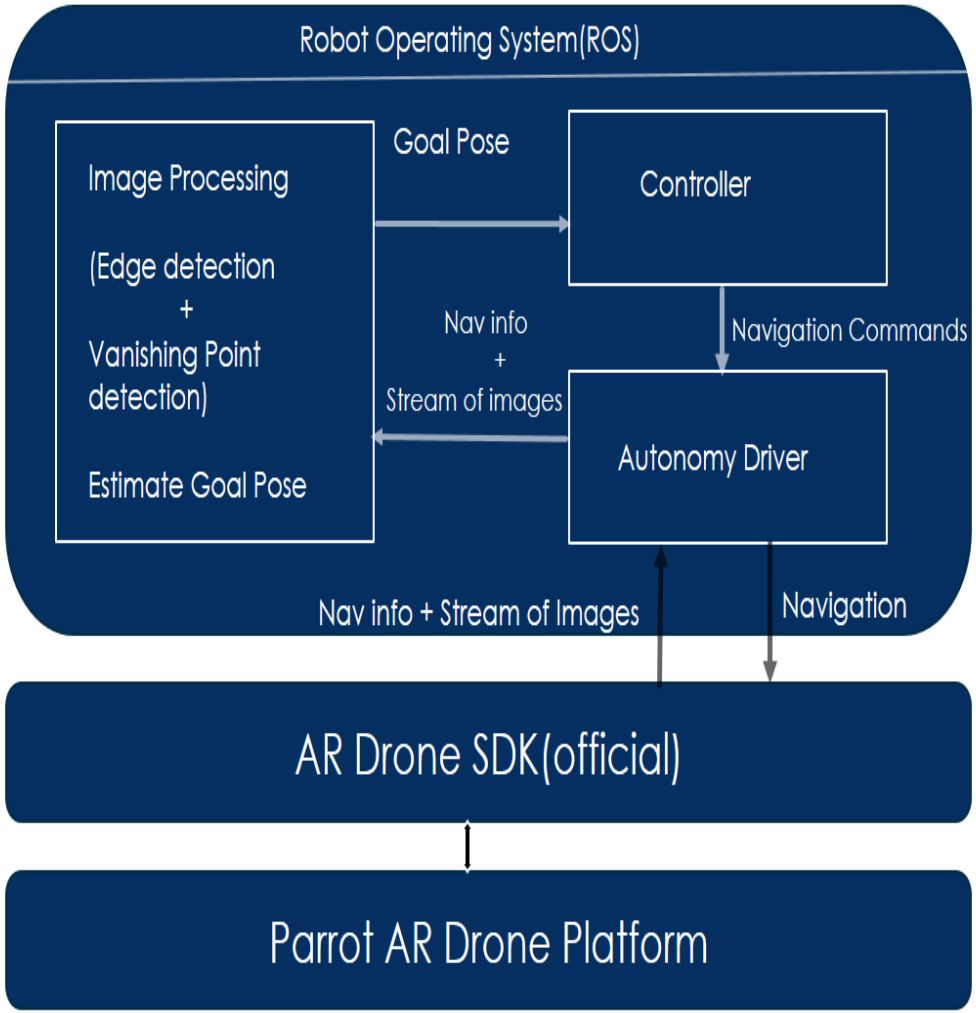


Example image of Vanishing Point

Design Architecture

Architecture consists of 5 main components:

- Parrot AR Drone Platform
- AR Drone SDK
  - provides APIs to communicate with AR drone.
- Autonomy Driver(ROS Package)
  - interface between ROS and the AR.Drone(navigation messages, video feeds and control commands)
- Image Processing*
- Controller*



Component view of the Architecture

Image Processing

Detect vanishing point from the perspective cues from stream of images received through AR Drone.

Implemented two methods of detecting vanishing point:

1. Classical VP based on edge detection
   - straight lines extraction from images
2. VP based on image density clustering
   - lines those move away from us converge towards the center of the picture.

Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
    1. preprocessing of image
    2. edge extracting using Canny operator
    3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
    1. preprocessing of image
    2. edge extracting using Canny operator
    3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Preprocessing of image

- Enhance image for more understandable level of feature extraction (Gray scale conversion).
- Smoothen the image to reduce noise (Normalized Box filter[1])

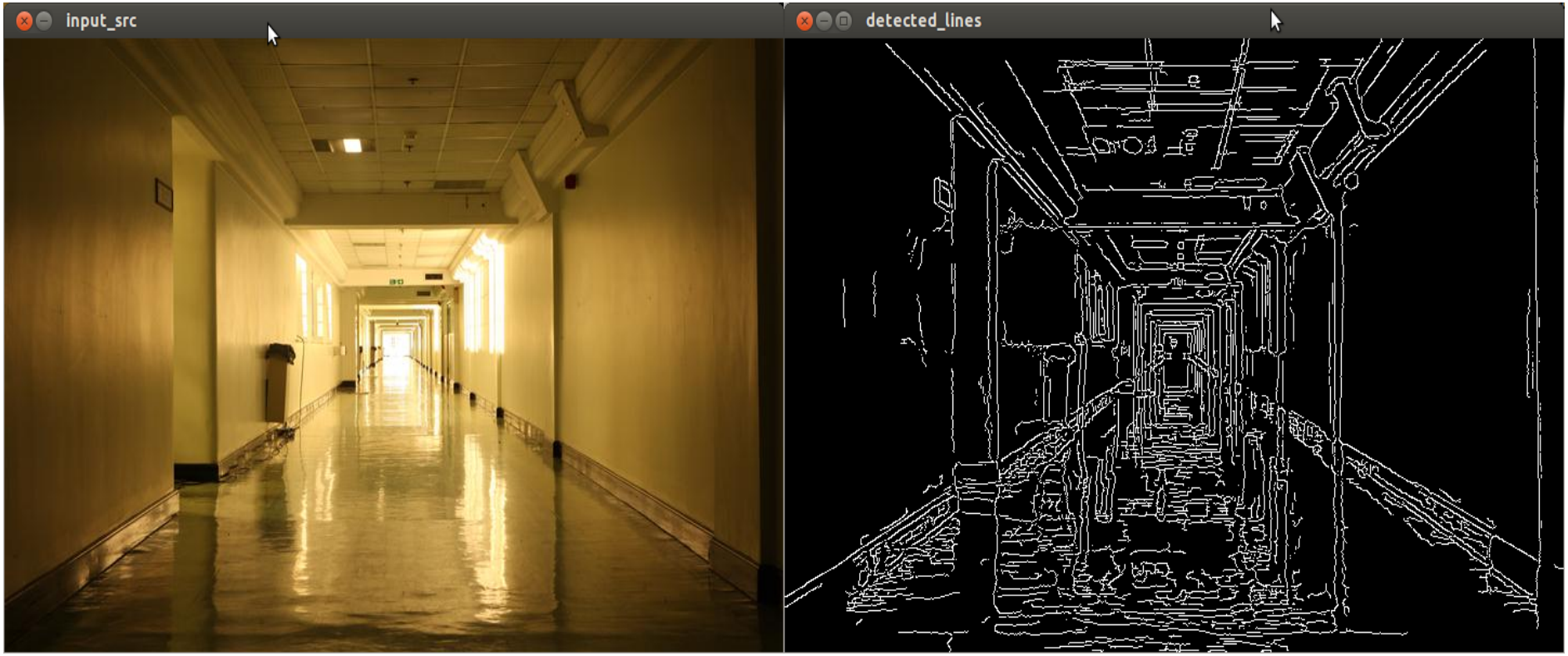
[1] Each output pixel is the mean of its kernel neighbors (all of them contribute with equal weights).

Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
   1. preprocessing of image
   2. edge extracting using Canny operator
   3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Output of Canny operator



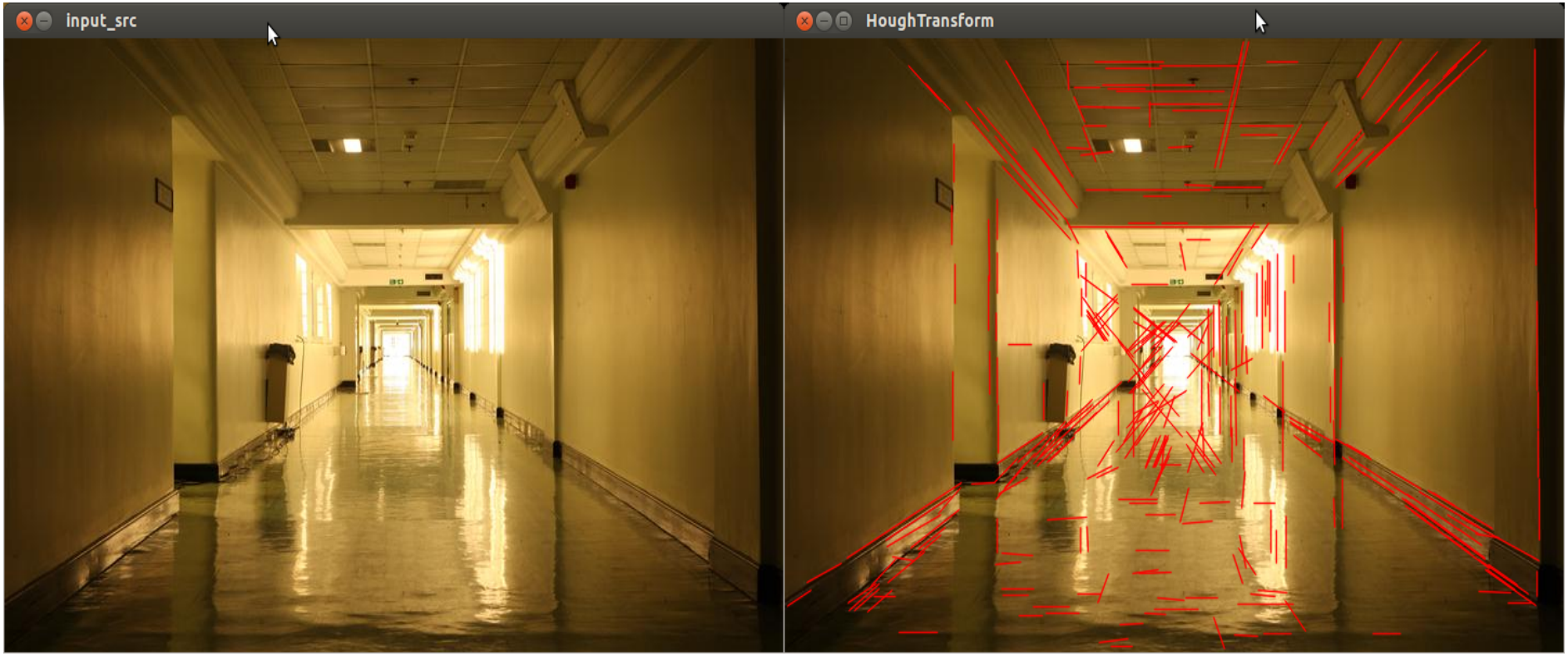
Detected edges using canny edge detector

Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
    1. preprocessing of image
    2. edge extracting using Canny operator
    3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Output of Probabilistic Hough Transform



Results of Hough Line Transform

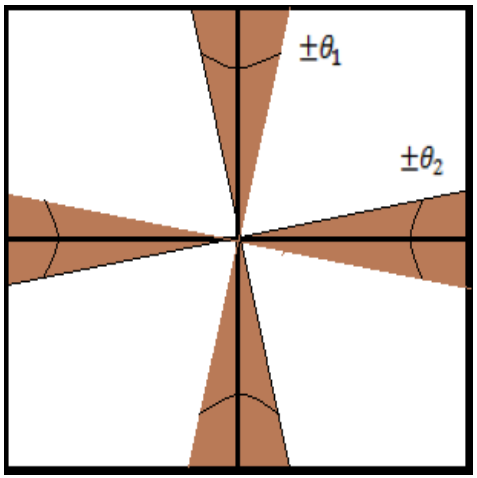
Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
    1. preprocessing of image
    2. edge extracting using Canny operator
    3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Deleting unreasonable straight lines

- Detected lines from Hough Transform include vertical and horizontal edges.
- These lines do not converge to VP, hence not useful.
- Such lines are filtered in order to have a more accurate VP detection.
- Lines which have slope between $\theta_1 = \pm 10^\circ$ or $\theta_2 = \pm 10^\circ$ are deleted.



Unreasonable lines with slope between $\theta_1 = \pm 10^\circ$ and $\theta_2 = \pm 10^\circ$

Method 1: Classical approach based on edge detection

Detecting the parallel lines in the environment is the premise for calculating the vanishing point.
Procedure for VP detection:

1. Straight lines extraction
    1. preprocessing of image
    2. edge extracting using Canny operator
    3. extracting lines by PHT(Probabilistic Hough Transform)
2. Deleting unreasonable straight lines
3. Locating vanishing point

Locating vanishing point

1. divide the image plane into a 22 x 33 grid G
2. calculate all intersections of the lines obtain from RHT
3. calculate the number of lines intersection falling in each grid element $G(x, y)$
4. consider $G(a, b)$ number of intersection lines falling in each grid element where $a \in [0, 22)$ and $b \in [0, 33)$.

The grid with the maximum number of intersections is:

$$(\mathrm{a}^*, b^*) = \mathtt{argmax}_{ab} G(a, b)$$

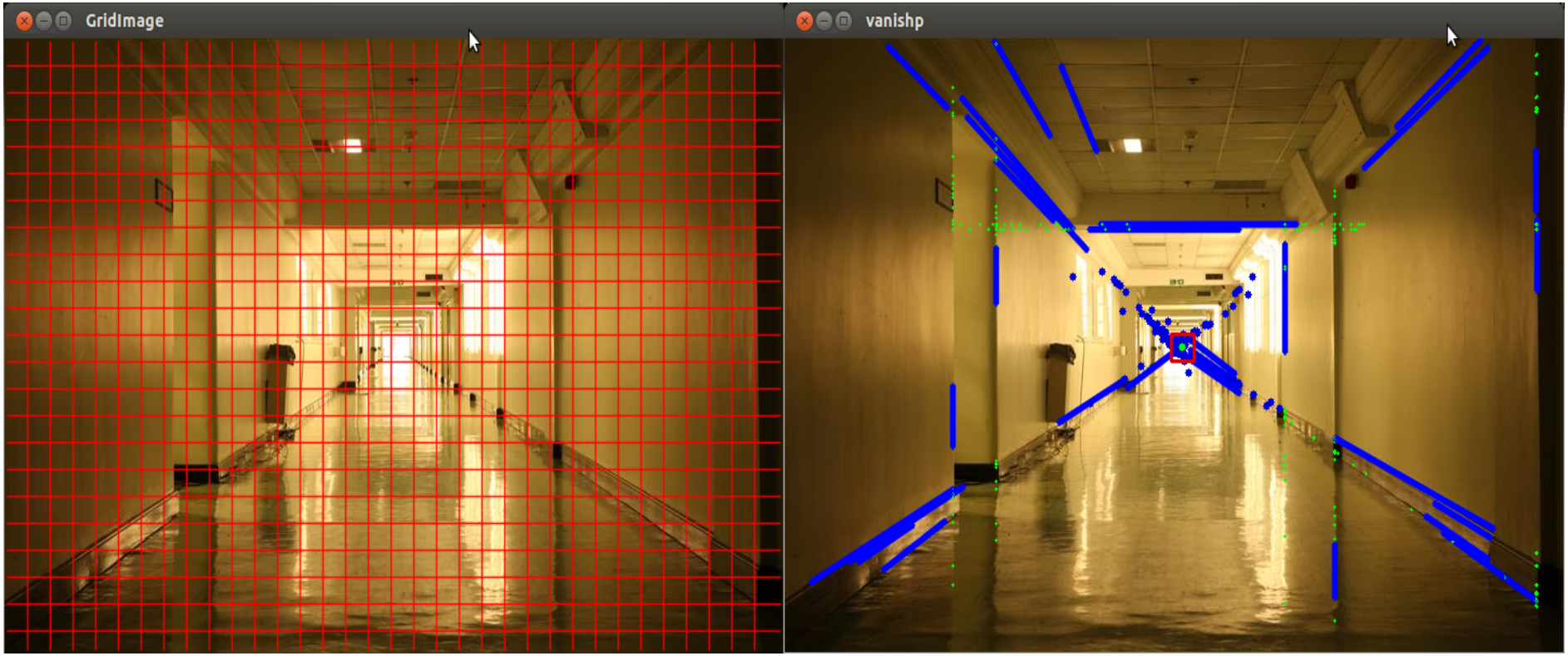
and vanishing point is given as:

$$(\mathrm{x}_{vp}, y_{vp}) = ((a^* + 0.5) * w/33, (b^* + 0.5) * h/22)$$

w,h - width,height of image

Locating vanishing point

Left figure shows division of image into a grid of 22×33 and right figure displays result of extracted vanishing point using edge detection method.



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

Method 2: VP based on image density clustering

Density of points of intersection of lines is maximum towards the center. Procedure for VP detection:

1. preprocessing of image
2. locate vanishing point

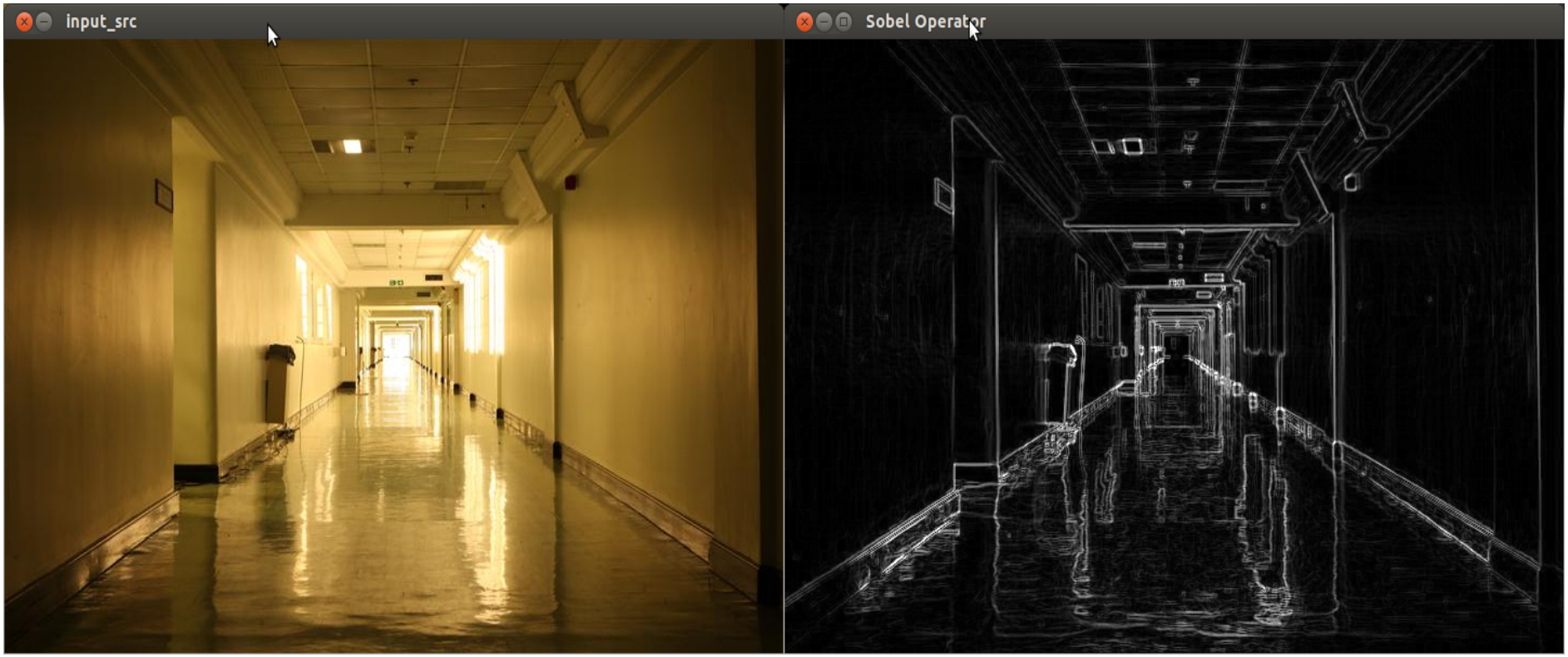
Method 2: VP based on image density clustering

Density of points of intersection of lines is maximum towards the center. Procedure for VP detection:

1. preprocessing of image
2. locate vanishing point

Preprocessing of image

- convert the image in to grayscale
- apply the Sobel operator in both x and y direction to find out the edges.



Output of sobel operator

Method 2: VP based on image density clustering

Density of points of intersection of lines is maximum towards the center. Procedure for VP detection:

1. preprocessing of image
2. locate vanishing point

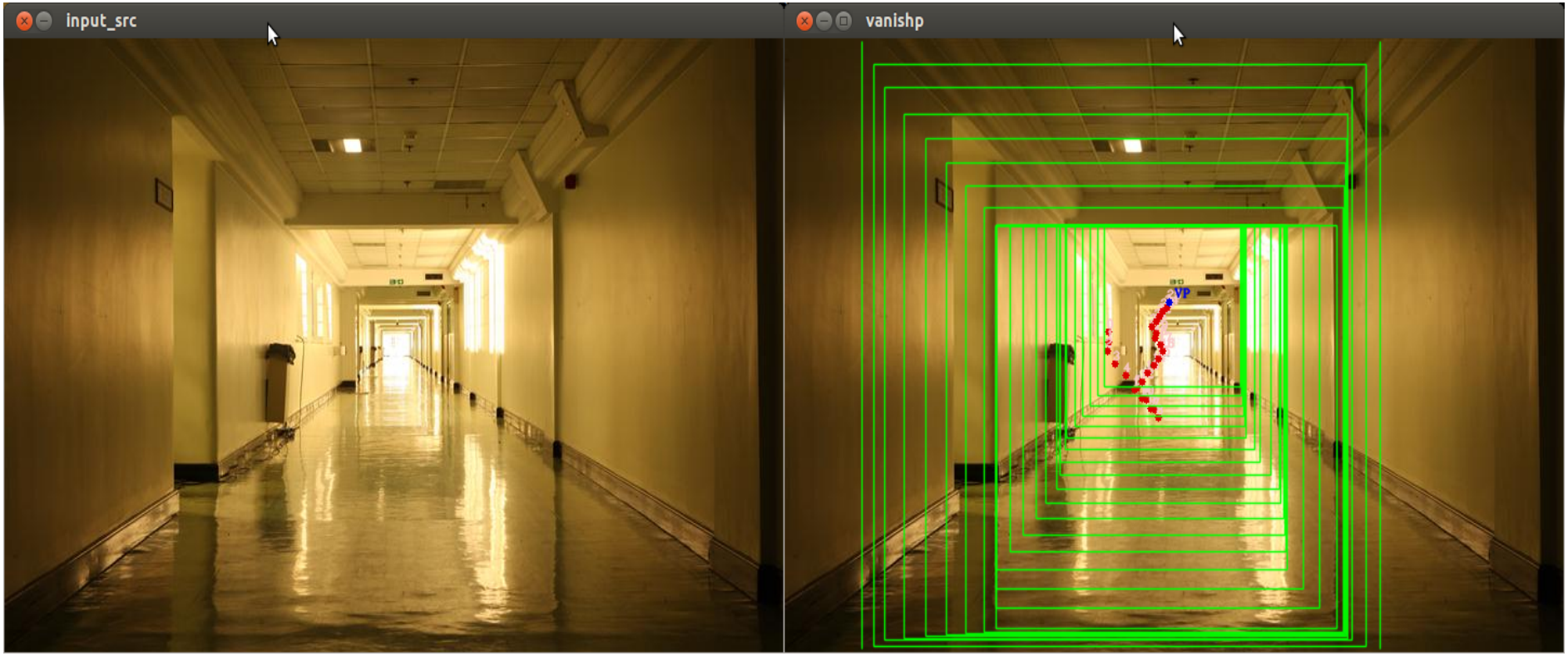
Locate vanishing point

VP is the point of maximum density, the aim is to find that point.

Procedure:

- Take a square box of side size equal to the one smaller between height(h) and width(w)
- Scroll through the image from left to right if w >h else top to bottom
- Keep track of density of points in each box and after the complete scroll, take the box with maximum density
- Repeat the process for 25 times and each time the scrolling area is reduced to the box selected from the last scroll.

Locating vanishing point



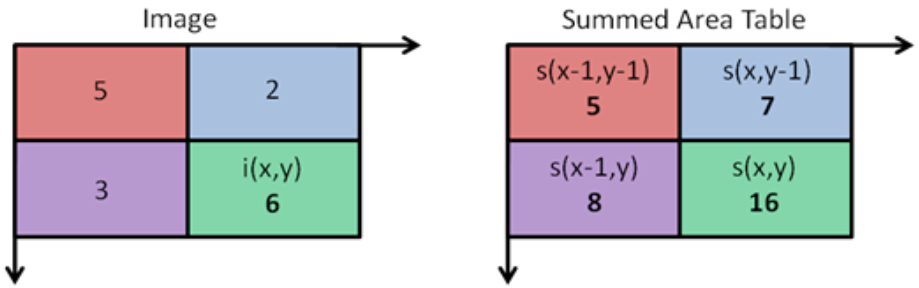
Output of density based vanishing point

Integral image to find densities

- Calculating intensity of each pixel in a box and summing them is a time consuming process.
- Integral image helps to 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 the given image.

Value s of a pixel (x, y) in output image is :

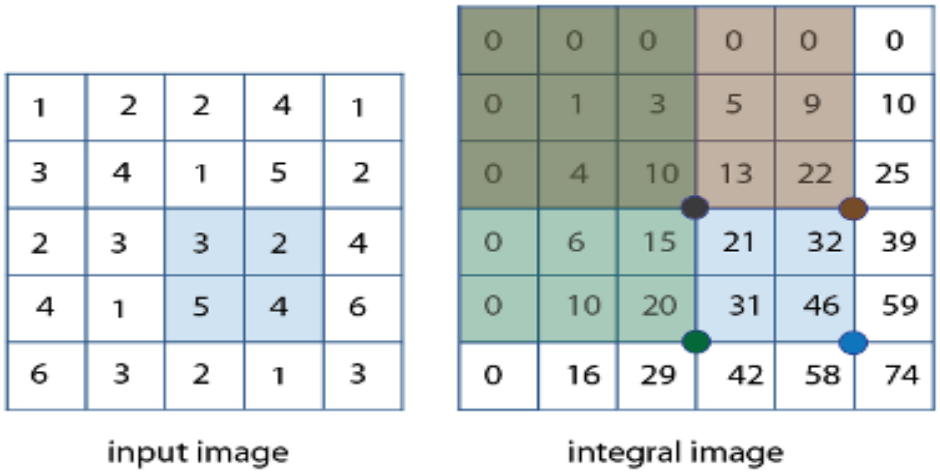
$$s(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) + s(x - 1, y - 1)$$



Integral image concept

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

Integral image to find densities



Example of integral image

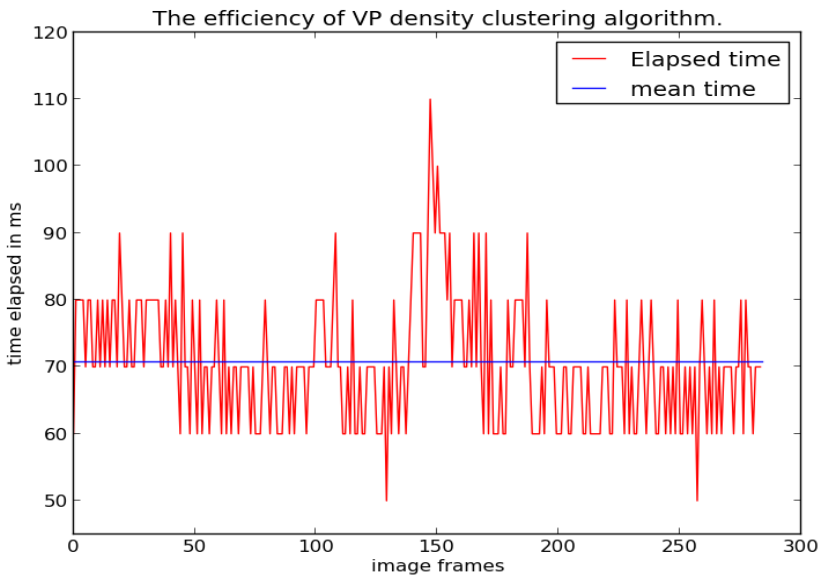
[Photo credit: https://www.mathworks.com]

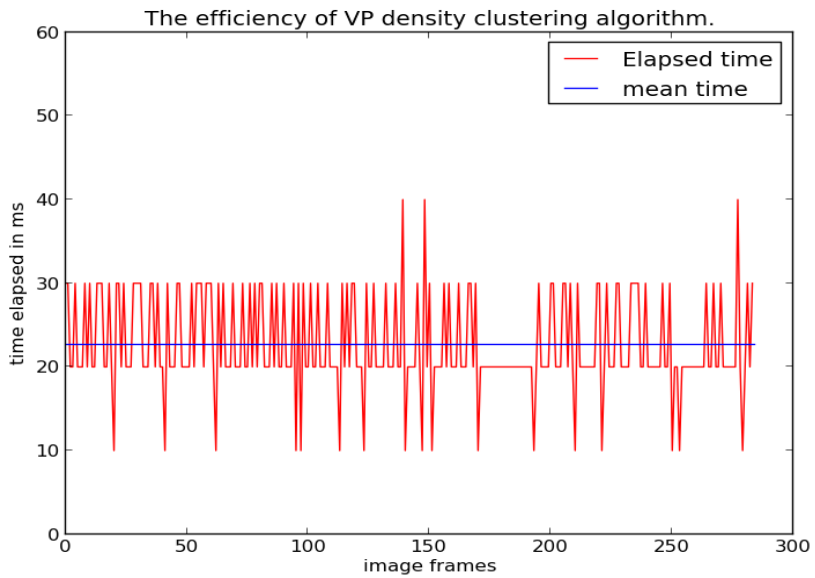
To calculate density of any sub region in image, we need only 4 values. This process have now $O(1)$ complexity.

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

Edge detection Vs Image density clustering

- VP detection based on image density is faster than edge detector method
- But very sensitive in case of noise or obstacles in the environment
- Therefore, we used VP detection based on edge detection.





Time elapse by Edge detector method (left) and Density cluster(right) over 250 sample images

Reliability of Vanishing Point detected

- Keep track of vanishing point detected in previous frame
- Resultant vanishing point is the latest vanishing point detected half plus previous vanishing point half.

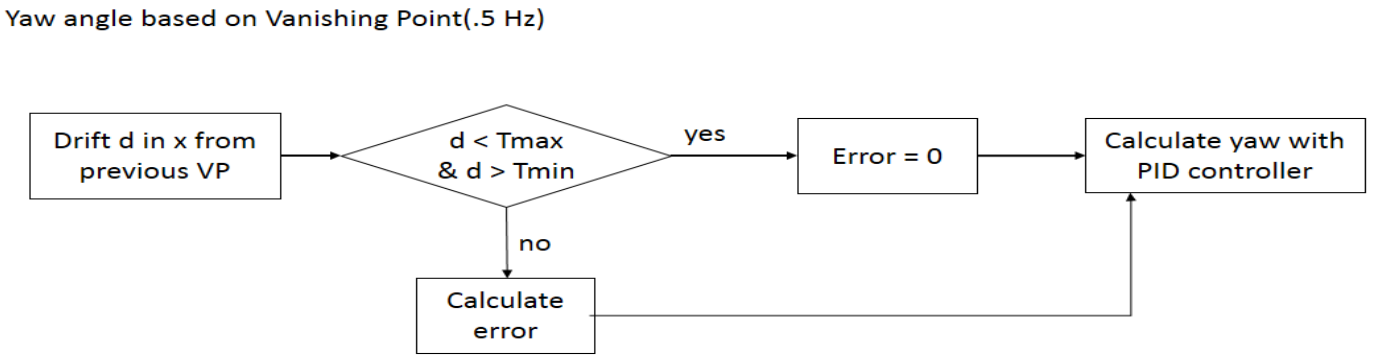
$$(x_{vp}, y_{vp}) = (current_{vp}.x/2 + previous_{vp}.x/2, current_{vp}.y/2 + previous_{vp}.y/2)$$

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.

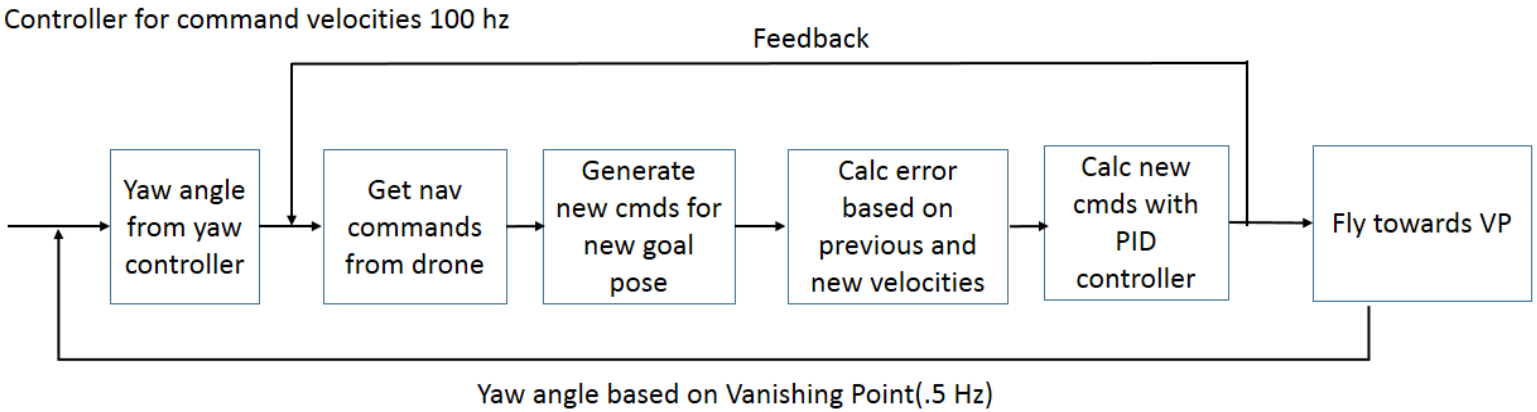
Controller

- Controller receives estimated pose from image processing module and sends flight commands to the AR.Drone via ardrone autonomy.
- Aim to maintain at zero, the horizontal distance between the vanishing point and the center of the image.
- PID controller is used in our approach to directly control the quadrocopter.
  - **P**: helps to reduce the error
  - **I**: helps to maintain the stable state(hold state)
  - **D**: damps occurring oscillations
- Separate controller for yaw angle and command velocities

Controller



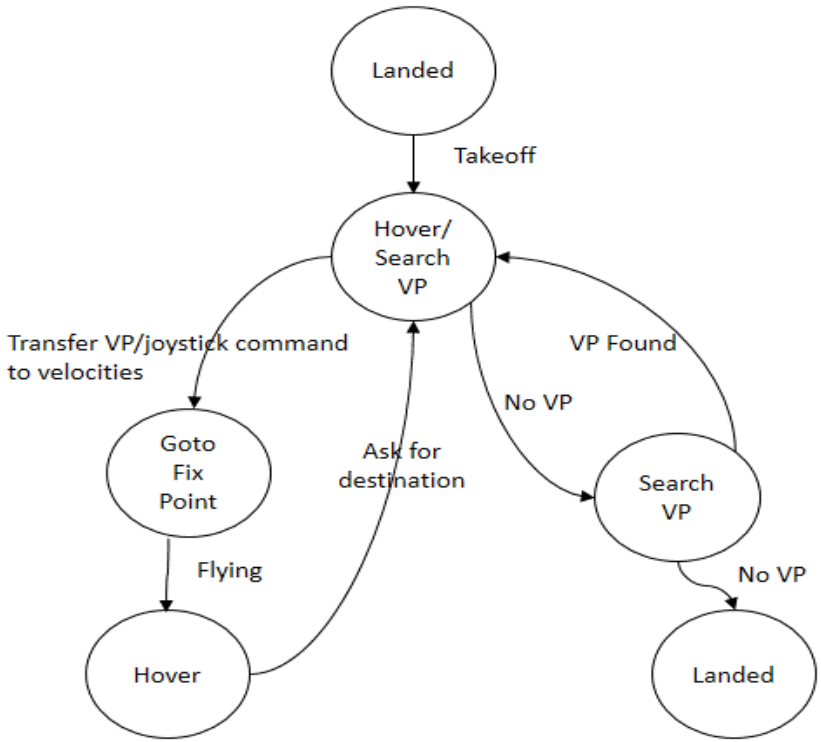
Controller for yaw angle



Controller for navigation commands

- Vanishing point auto mode can be suppressed by joystick mode.

State diagram of the drone



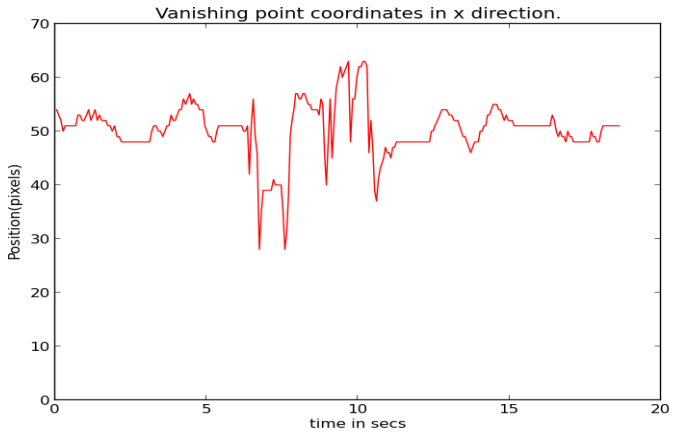
State diagram of drone

Experimental results

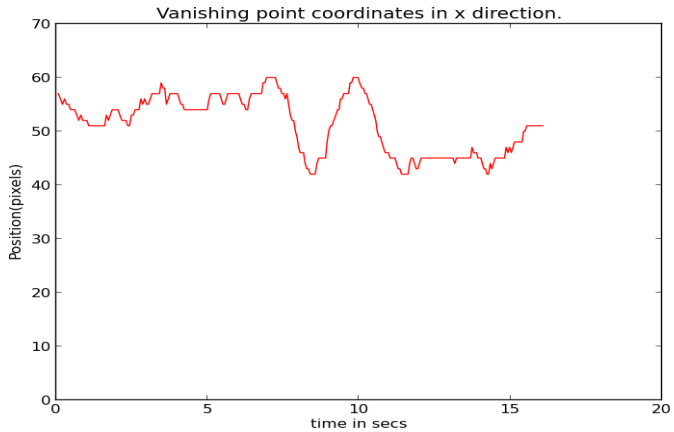
Tested our approach in several corridors such as narrow, broad, corridors with obstacles and dark corridors.

- Accuracy of vanishing point detection approach

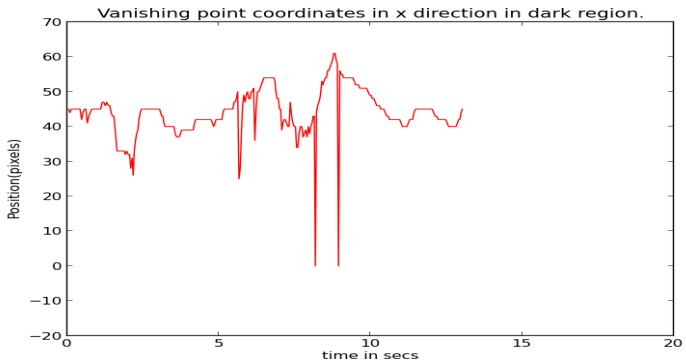
with sharp maneuvers



without sharp maneuvers



Dark Corridor(bad illumination)

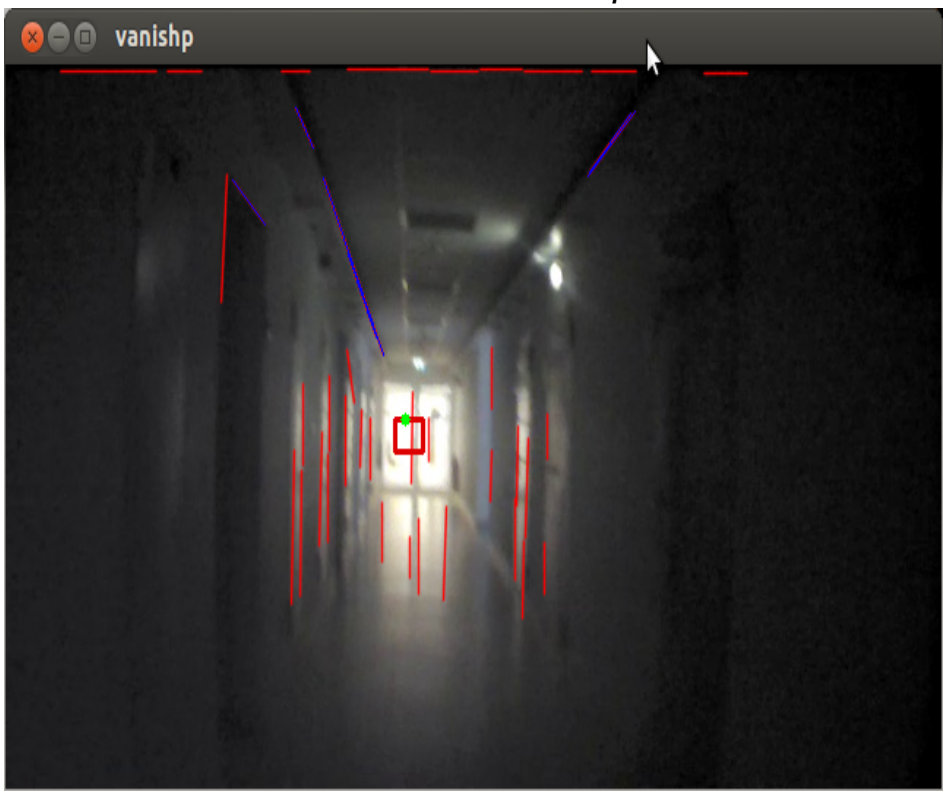


VP Evaluation

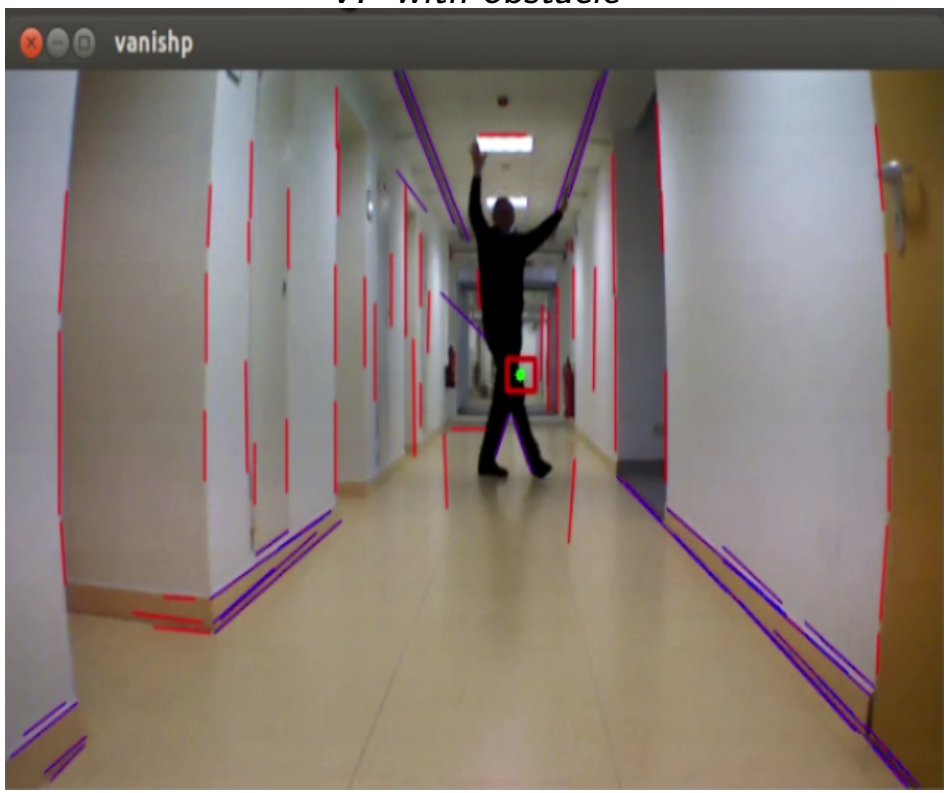
- Different corridors for VP testing

Current system does not consider obstacles if there are enough lines to calculate the vanishing point

Dark corridor example



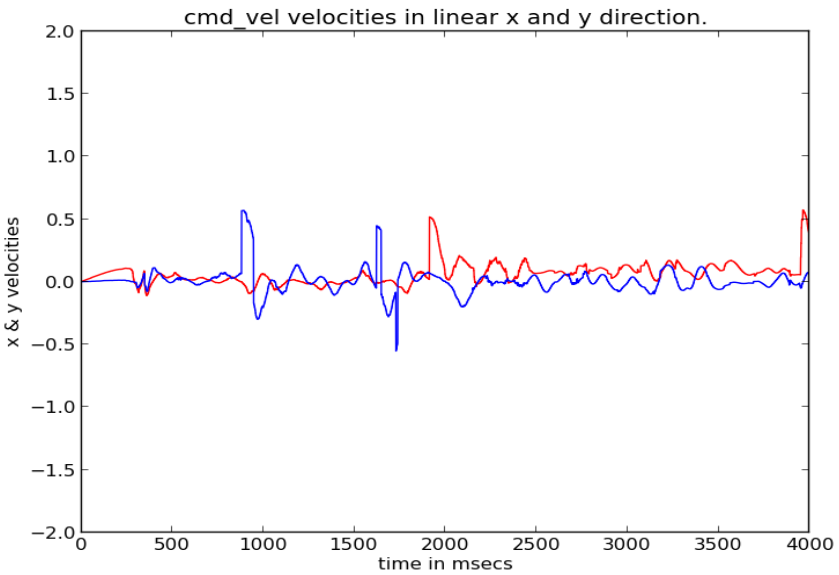
VP with obstacle



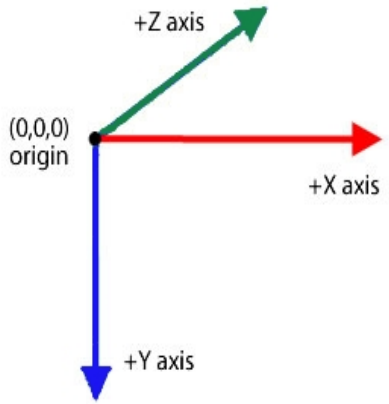
Controller Evaluation

We evaluated controller based on the variation in Linear x (horizontal) ,Linear y (vertical) and angular z (yaw) velocities

Linear roll and pitch velocities on real flight

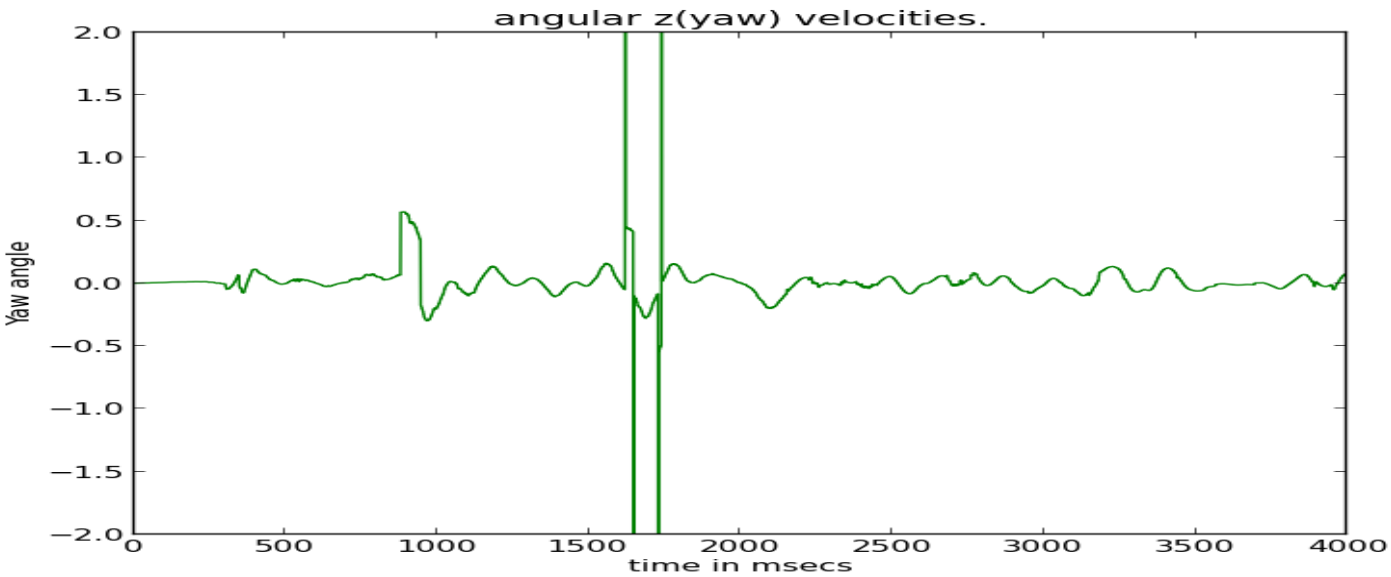


Linear XYZ axis



Controller Evaluation

Variation in yaw angle shows the robustness of the controller



Angular velocities of the drone during flight

Future work

- **Density based VP detection** is promising and less computationally expensive. Can be improved for useful VP based navigation
- **Edge based VP detection:** 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.
- **Onboard Computation:** Currently algorithm works on host machine, still need to be implemented on Asctec Pelican.

VIDEOS

References


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.


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.


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.


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


Haiyang Chao; Yu Gu; Napolitano, M., "A survey of optical flow techniques for UAV navigation applications," Unmanned Aircraft Systems (ICUAS), 2013 International Conference on , vol., no., pp.710,716, 28-31 May 2013


G. Conte and P. Doherty "Vision-based unmanned aerial vehicle navigation using georeferenced information", EURASIP J. Adv. Signal Process., vol. 2009, pp.10 2009

THANK Y😄U

Devvrat Arya
devvrat.arya@smail.inf.h-brs.de

Edge extracting using Canny operator

Canny Edge detector - optimal Edge detector

- Gaussian Filter De-noising

- Gradient Operator Sobel
  Finds the intensity gradient of the image in vertical and horizontal direction using Sobel Operators.

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

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

$$\|\bar{G}(x,y)\| = \sqrt{\bar{G}_x^2 + \bar{G}_y^2}, \qquad \theta = \texttt{arctan}\left(\frac{\bar{G}_y}{\bar{G}_x}\right)$$

- Control of Gradient Value
  Suppress any pixel value (i.e. set it equal to zero) that is not considered to be an edge.

- Hysteresis
  Track along the remaining pixels that have not been suppressed.

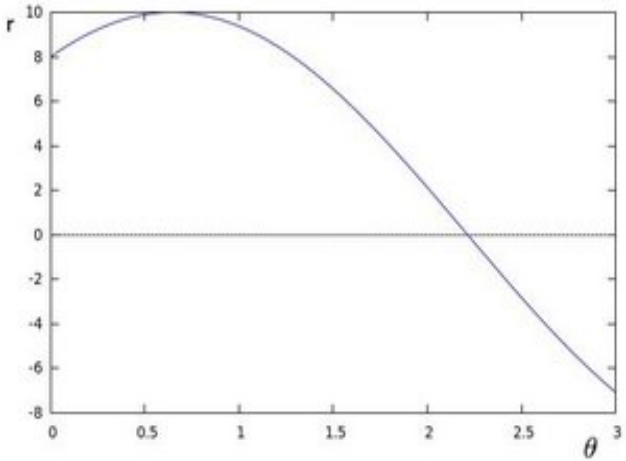
Extracting lines by PHT(Probabilistic Hough Transform)

- The algorithm is based on the parametric representation of a line:

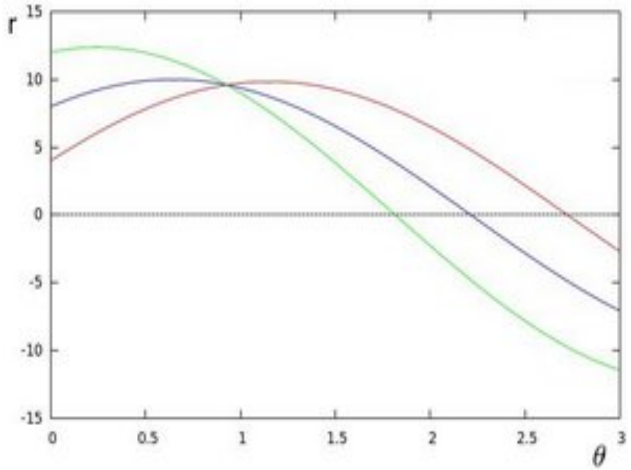
$$\rho = x \cos\theta + y \sin\theta$$

where ρ is perpendicular distance from the origin to the line and θ is angle between horizontal axis and this perpendicular.

- Family of lines that goes through a point (x, y), gives a sinusoid.



(ρ, θ) plot for points (x, y)



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

- *Same operation is done for all the points in an image. If curves of two different points intersect in the plane (θ - ρ), that means both points belong to a same line.*