**ARIES : LINE FOLLOWING DRONE WITH LANDING**

**Overview**

-> This project aims to design a Line-tracking algorithm through Vision-based control with image processing techniques.

-> A significant advantage that characterizes this technique is the auto-code generation, which allows us to automatically translate the blocks of the model built through Simulink into a C-code executable by the hardware, instead of writing it manually.

-> This project aims to make a drone follow a line of a specific color(here red) and land at the end of it on a circle while remaining stable and following path as precisely as possible.

-> The environment used to design and develop the control system is MATLAB, with Simulink and their add-on toolboxes like Aerospace blockset, image processing, computer vision, and Hardware support package for Parrot mini drone.

-> Firstly, the preliminary goal is the accomplishment of the stabilization of flight maneuvers through a suitable control system architecture and PID controllers tuning.

Then, the Flight Control System design proceeds with Image processing and Path planning subsystems design.

-> The line-tracking algorithm implementations is based on the analysis of the pixels of the image acquired from the downward-facing camera and elaboration through image processing techniques like color thresholding and edge detection.

-> The path planning logic was implemented through Stateflow, which is an add-on tool of Simulink, useful for Statae machines design.

**Introduction:**

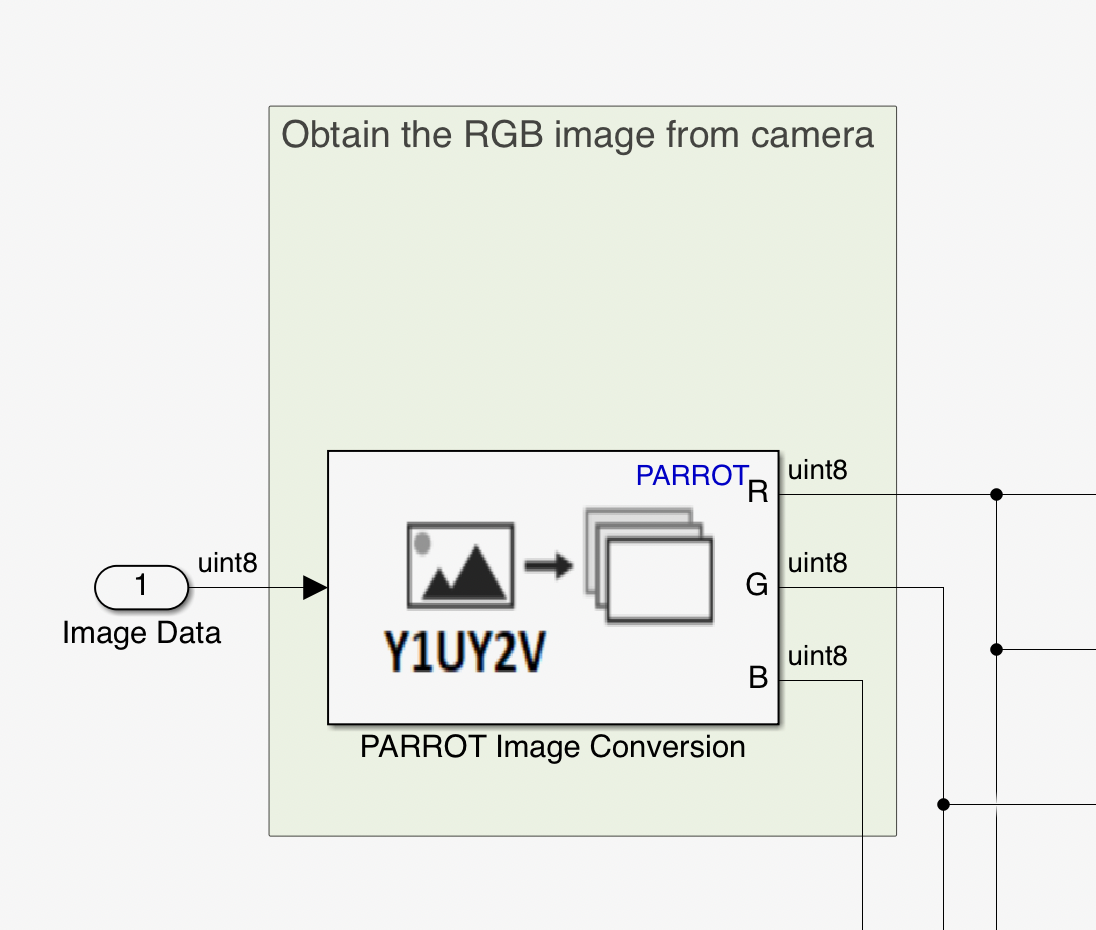
Line/path following is a relevant application problem within the Unmanned Aerial Vehicle (UAV) field. For instance, in precision agriculture scenarios having good path following algorithms is a fundamental requirement to preserve high productivity rates and plant growth. In civilian applications, monitoring of power lines can be encoded as a path following problem between several target regions that need to be inspected. Whatever the field of application is, drones have to follow a path to accomplish the mission specifications safely and successfully.

Main building block in this simulation are IPS and CS.

Let’s start with IPS

**Parrot Image Conversion Block:**

The PARROT Image Conversion block converts an image that is encoded in Y1UY2V format, as obtained from the Parrot minidrone's downward-facing camera, to RGB format. The block separates the color components and provides them as the output.



**Image Binarization System:**

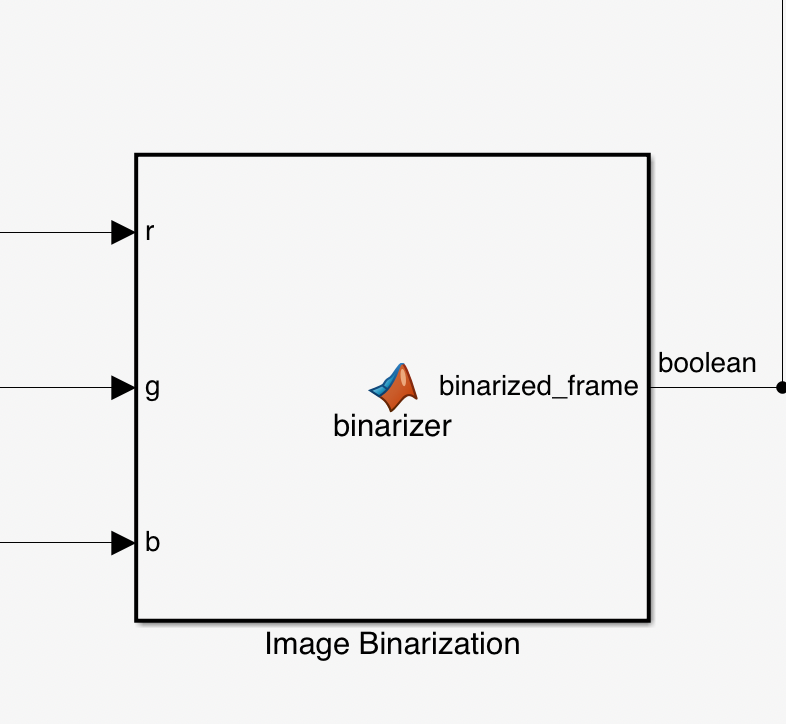
Image Binarization is the conversion of a document image into bi-level document image. Image pixels are separated into a dual collection of pixels, i.e., black, and white. The main goal of image binarization is the segmentation of documents into foreground text and background.

**Image Binarizer MATLAB Function(ss):**

The MATLAB function computes a binarized frame starting from a frame expressed in the RGB file format. Inputs are red, green, and blue channel frame and output is the binarized frame. Parameters are:

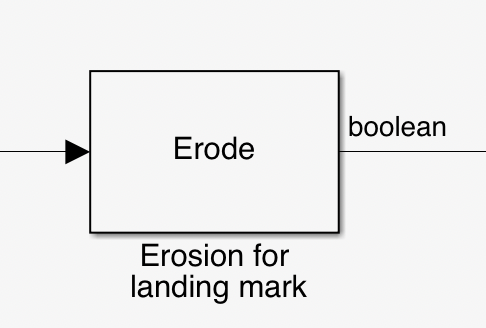
(i) G\_B\_GAIN: used to reduce the amount of green and blue colors inside the frame

(ii) BINARIZER\_THRESHOLD: threshold to detect only the track inside the frame



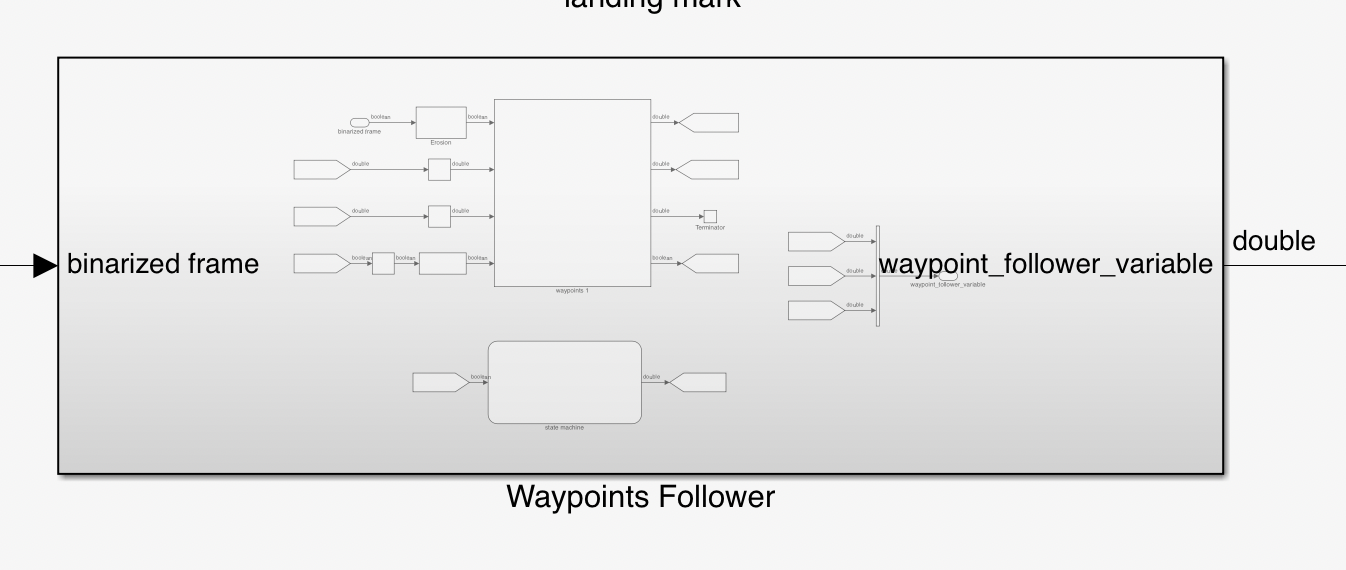
**Erosion:**

The Erosion block slides the neighborhood or structuring element over an image, finds the local minima, and creates the output matrix from these minimum values. If the neighborhood or structuring element has a center element, the block places the minima there.



**Waypoint Follower:**

In this vision based path following algorithm, it starts selecting a target position ahead of the drone which has to be reached typically on the path.



**Error-pixel generator MATLAB function(ss):**

The MATLAB function computes the error along the x and y-axis between the drone COG(Center of Gravity) and the point of the track. This error is later used by the path planner for the lane tracking.

Inputs:

-> x\_ref\_prev: previous value of the tracking point along the x-axis of

the track line. It is used as a reference for tracking the path.

-> y\_ref\_prev: previous value of the tracking point along the y-axis of

the track line. It is used as a reference for tracking the path. -> flag\_track\_prev: previous value assumed by the flag variable, i.e., the previous state of the drone (over the track / not over the track).

Outputs:

-> error\_y\_track: error between the CoG of the drone and a point of the

lane in the forward direction along the y-axis. It is expressed in

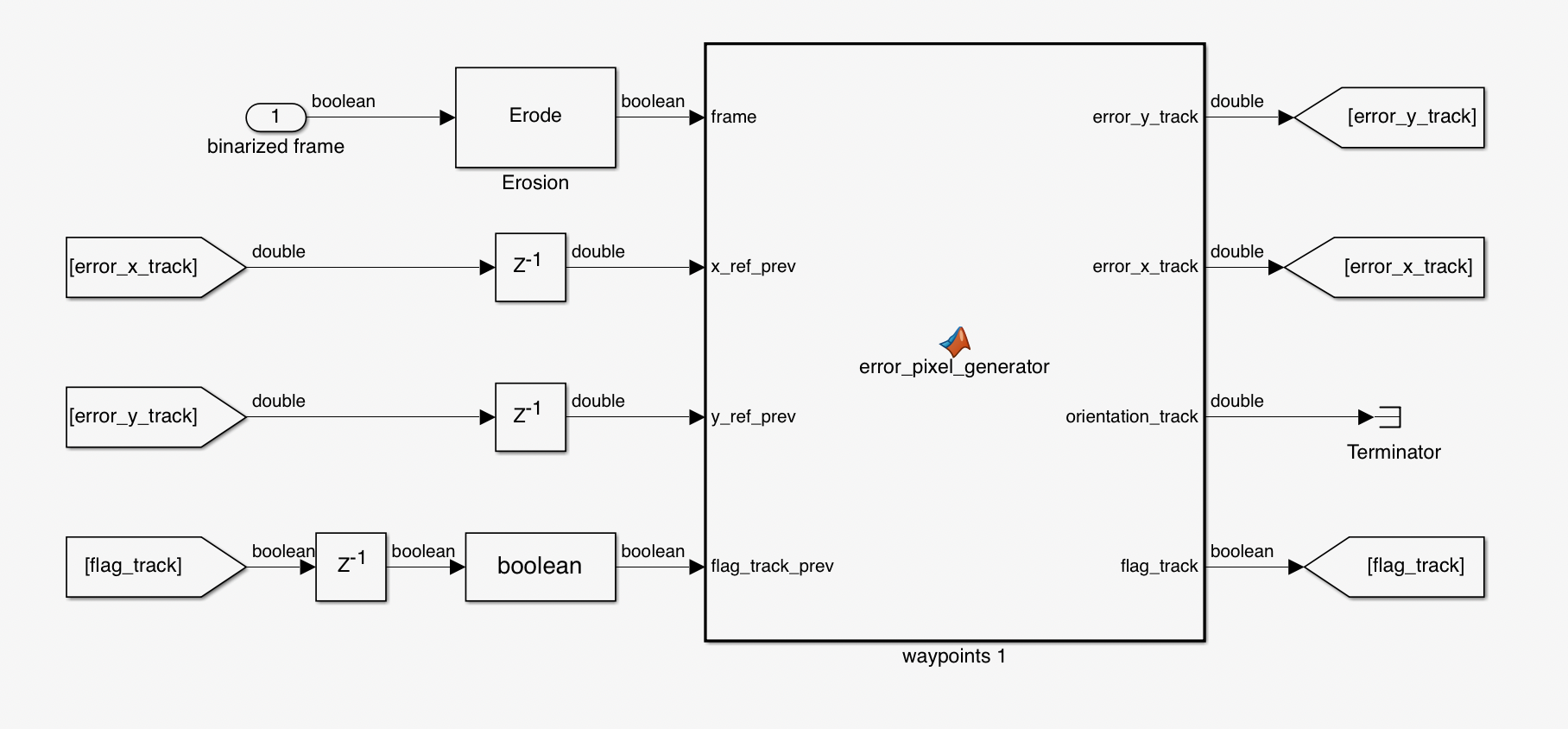
pixels.

-> error\_x\_track: error between the CoG of the drone and a point of the

lane in the forward direction along the x-axis. It is expressed in pixels.

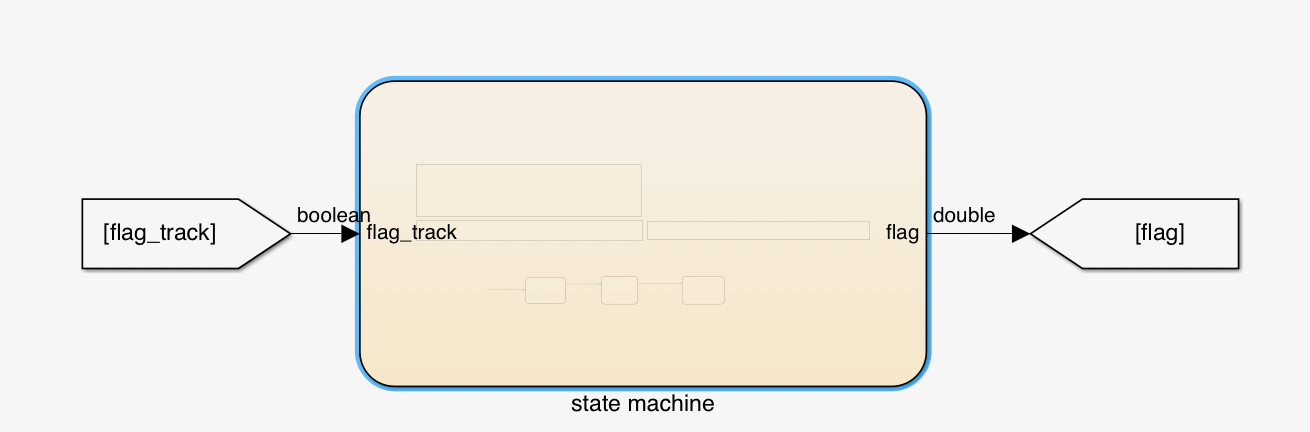
-> flag\_track: it indicates if the drone is currently over the track (TRUE).

yaw: the heading angle of the track in the image plane.



**State Machine:**

State machine gives whether the drone is over the lane/track by making the FLAG variable either 0 or 1. The input flag-track raises when the track is detected from the image processing system.



**Land Marker Detector MATLAB Function (ss):**

The MATLAB function computes the errors along the x- and y-axis between the drone COG and the middle of the lane. The end marker is detected by using a disk kernel. This function is active only when the drone has arrived at the end of the path.

Inputs:

binary: binarized frame obtained as the convolution with a disk kernel

x\_ref\_prev: previous value of the tracking point along the x-axis of he track line. It is used as a reference for tracking the path.

y\_ref\_prev: previous value of the tracking point along the y-axis of the track line. It is used as a reference for tracking the path.

flag\_track\_prev: previous value assumed by the flag variable, i.e., the previous state of the drone (over the track / not over the track).

Outputs:

error\_y\_end\_marker: error between the CoG of the drone and a point of the lane in the forward direction along the y-axis. It is expressed in pixels.

error\_x\_end\_maker: error between the CoG of the drone and a point of the lane in the forward direction along the x-axis. It is expressed in pixels.

flag\_end\_marker: it is TRUE if the Image Processing System detects the end marker.

yaw: heading angle of the lane in the image plane.

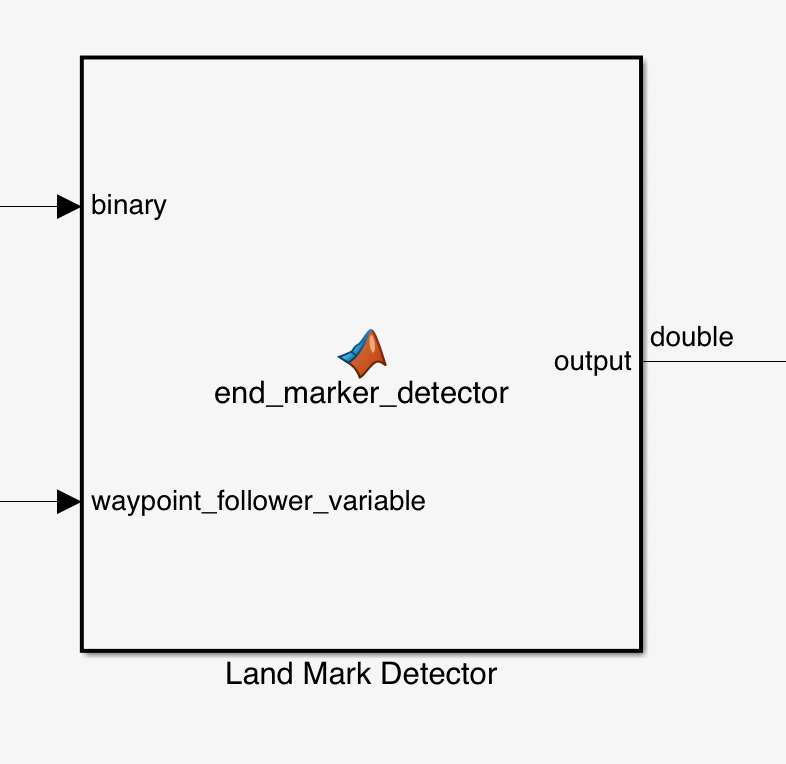
Parameters:

-COG\_X: position of the drone CoG along x-axis of the reference frame.

- COG\_Y: position of the drone CoG along y-axis of the reference frame.

- FRAME\_SIZE\_WIDTH: width of the frame.

- FRAME\_SIZE\_HEIGHT: height of the frame.



Different Sensors used in drone

-> Accelerometer

Accelerometers are used to determine the orientation of the drone in flight and it measures the proper acceleration. It’s values help determine the roll and pitch movements of the drone. It measures the change in the acceleration due to gravity along the X axis (forward and backward) or along the Y axis(sideward).

-> Barometer

Barometer measures air pressure. This air pressure is used to measure the height of the drone above the MSL (mean sea level). As we move to higher altitude, the air pressure decreases. This data is used with Kalman Filters to provide Altitude Hold (Althold) for the drone.

-> Gyro Sensor

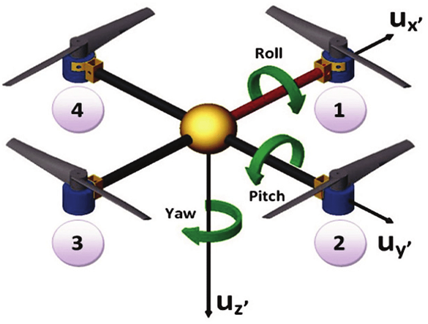
Gyro sensors, also known as angular rate sensors or angular velocity sensors, are devices that sense angular velocity . Gyro sensors work according to the Coriolis effect that is a force that acts on objects that are in motion relative to a rotating reference frame

-> Magnetometer

A magnetometer is an instrument that measures magnetism—either magnetization of magnetic material like a Ferro magnet, or the strength and, the direction of magnetic field at a point in space. It measure the earth’s magnetic field by using the Hall Effect or the magneto resistive effect. It is used as a digital compass that helps identify the magnetic north. This helps in obtaining more precise readings for headings.

Different Motion in drone:

Any type of motion can be described as combination of these three following motions



1. Roll

The axis along the length (front - back direction) of the aircraft, usually passing through its center of gravity is longitudinal axis. The rotation of the aircraft along the longitudinal axis is called as ‘Roll’.

2. Pitch

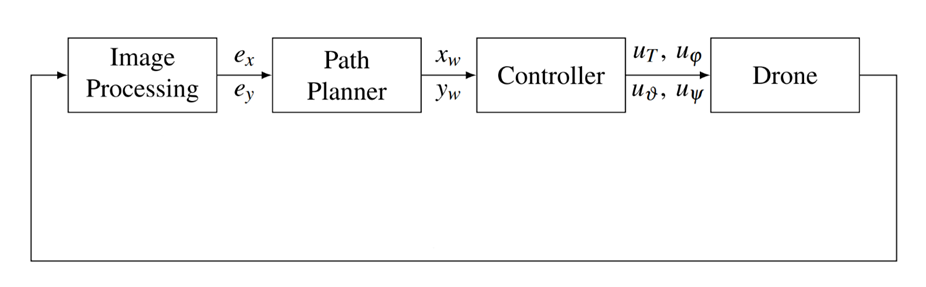
The axis that runs below the wing, from one wingtip to another (left - right), passing through the airplane's center of gravity is lateral/transverse axis. The rotation of the aircraft along the lateral axis is called as ‘Pitch’.

3. Yaw

The axis perpendicular to the wings and body of the aircraft (up - down), passing through the airplane's center of gravity is the perpendicular axis. The rotation of the aircraft along the perpendicular axis is called as ‘Yaw'.

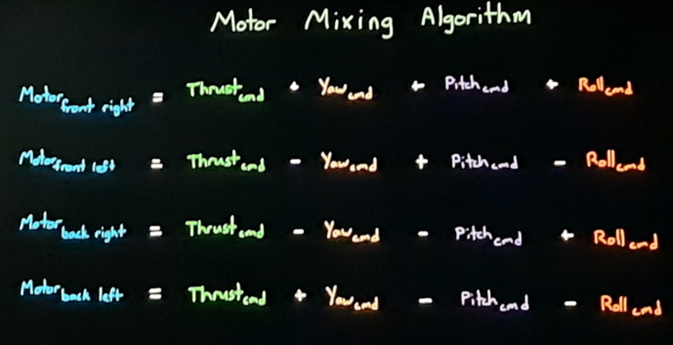
Control System :

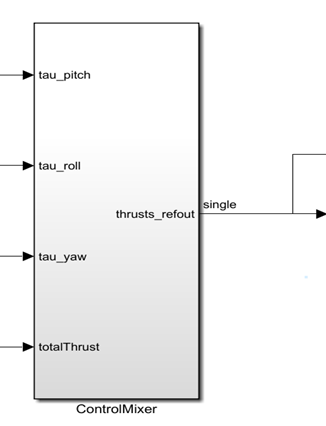
The basic idea of a control system is to figure out how to generate the appropriate actuated signal, the input, so that our system will produce the desired controlled variable, the output.



In this we will discuss about motor control algorithm after taking the input through Image Processing System like VTP(Virtual tracking point) and input from various sensors about current position(x,y,z) and state variables like roll, pitch and yaw.

As we have current position reference and estimated yaw, roll , pitch and thrust(from IPS). So all these inputs are feeded to PID controller to signal required motor speeds of all respective 4 motors according to below motor mixing algorithm.



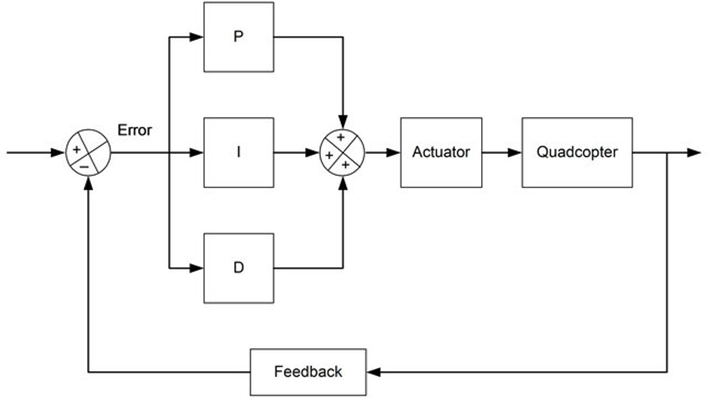


PID Controller

**PID** stands for **Proportional, Integral, Derivative**, it’s part of a [flight controller software](https://oscarliang.com/mini-quad-fc-firmware/) that reads the data from sensors and calculates how fast the motors should spin in order to retain the desired rotation speed of the aircraft.

The goal of the PID controller is to correct the “**error**“, the difference between a **measured value** (gyro sensor measurement), and a **desired set-point** (the desired rotation speed). The “error” can be minimized by adjusting the control inputs in every loop, which is the speed of the motors.

Those PID and PD controllers have as outputs the force and torque commands that are subsequently sent to the MMA. It produces the necessary motor thrusts, the commands are turned into motors’ speeds.



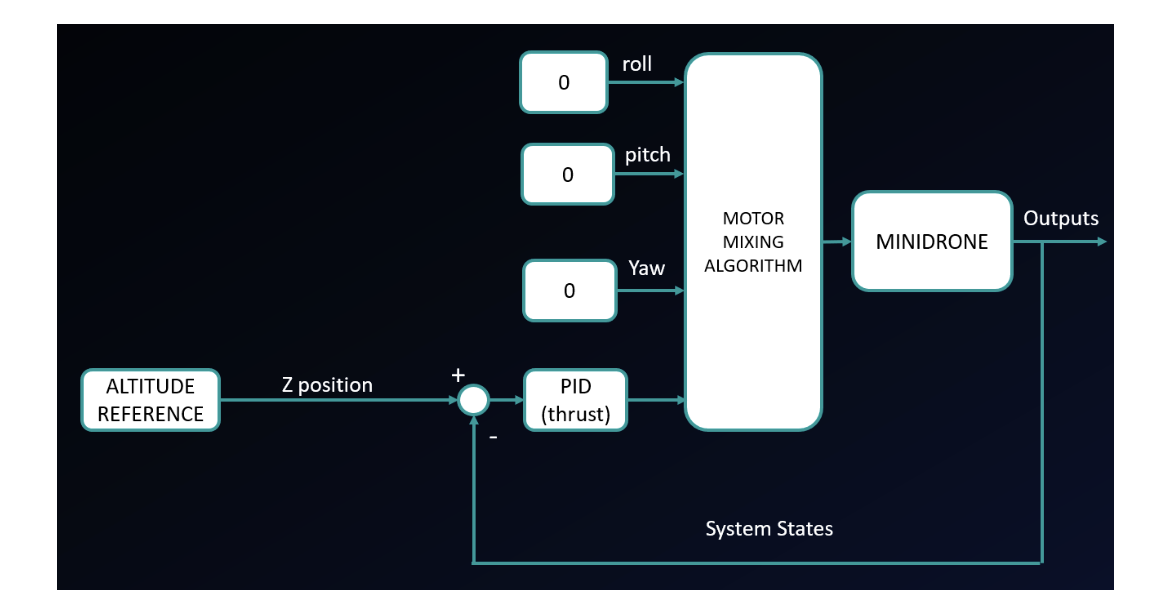
Need for PID controller can be explained in this way:

First we want to hover our drone at some altitude. At the time of starting error will be desired altitude. Let’s assume we need N rpm to hover then error multiplied by gain will give the propellor speed. While decreasing error by increasing gain, error will get smaller but it won’t go away. Error is called steady state error.

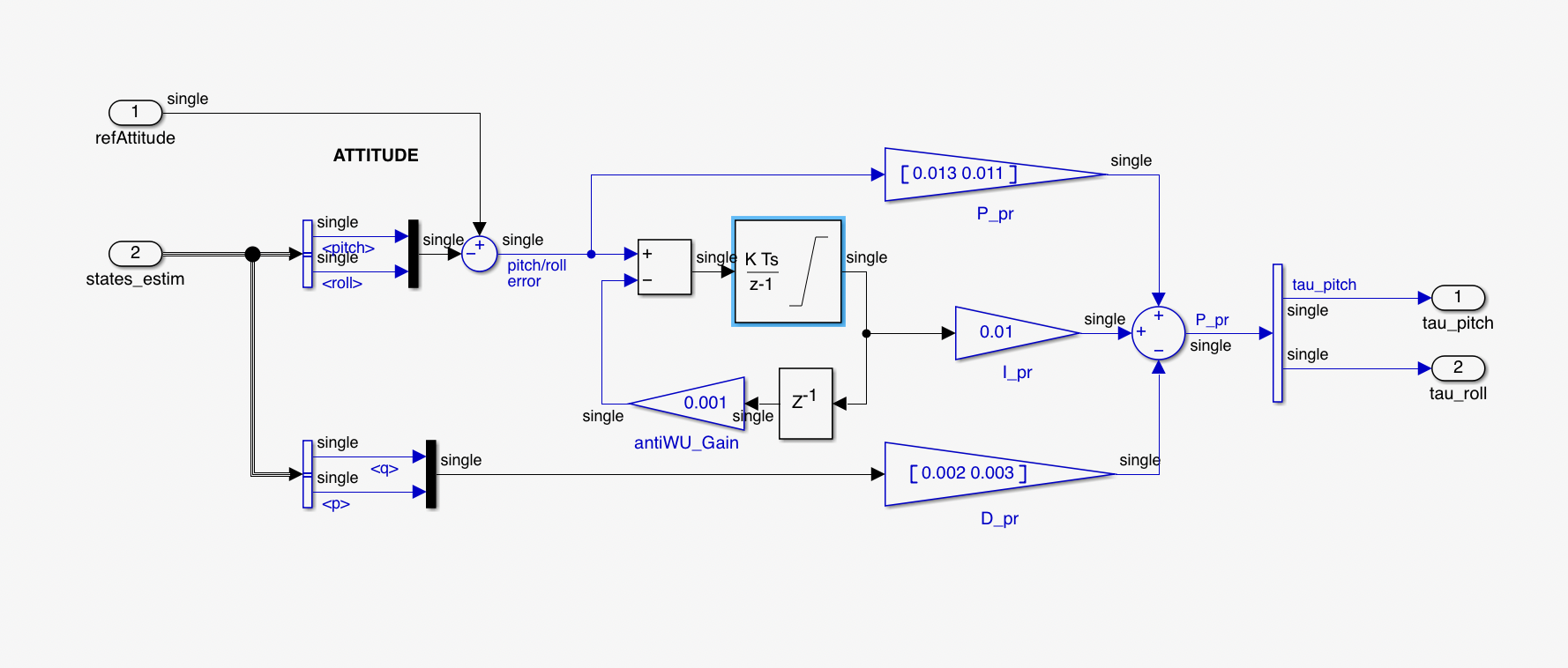
To get rid of it we can use past information or specifically adding an integrator path. Integrator sums up the input signal over time keeping the running total. It has memory of what has happened before. If the drone gets to steady state below the desired altitude, the error term is non-zero and when a non-zero value is integrated, the output will increase. Proportional and the integral path work with each other to drive the error down to zero. Sometimes it overshoots the desired goal. It can be handled by adding a path to our controller that can predict the future i.e., the derivative.

Example:

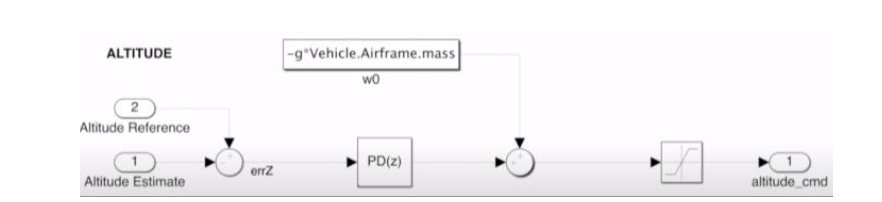
Altitude Controller Tuning



**Control System simplification for altitude controller tuning**



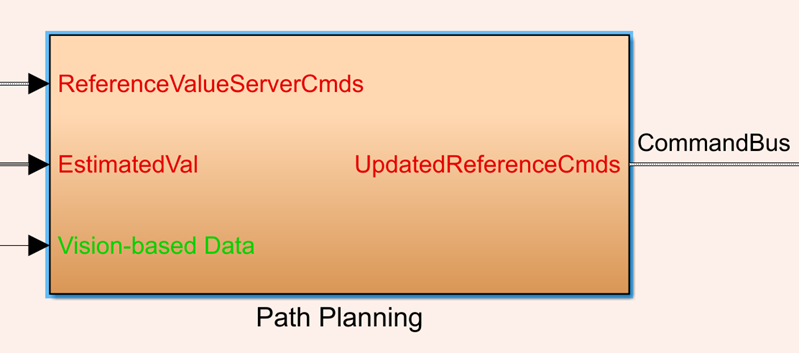
**Linearized control System used for altitude controller tuning**

****

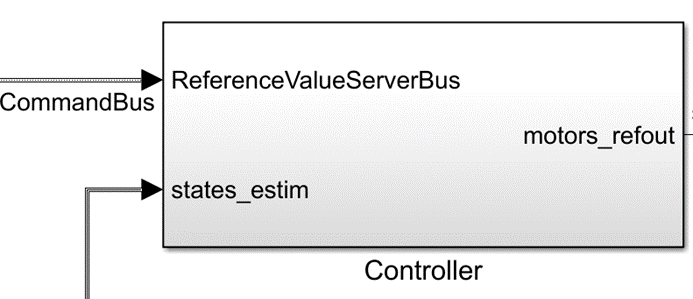
**Simplified altitude controller used for tuning**

Important Blocks used in Control System part:-

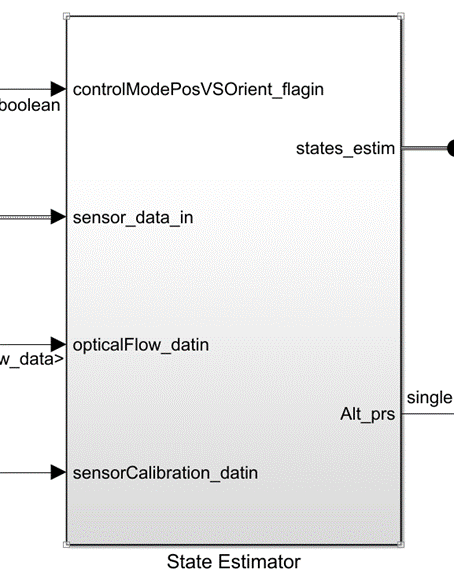
1. Path planning: where the logic for the line-tracking algorithm will be designed.



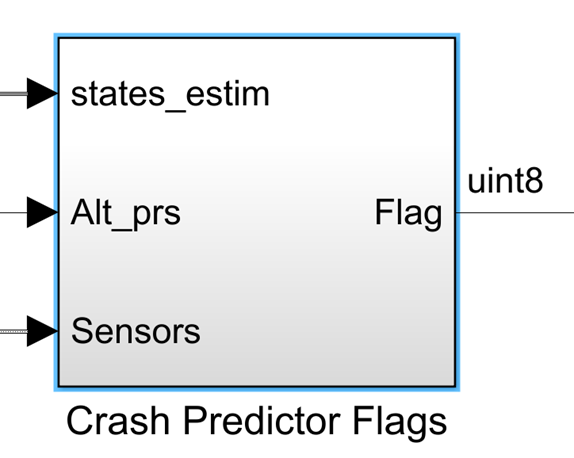
2. Controller: where all the PIDs of the flight controller reside.



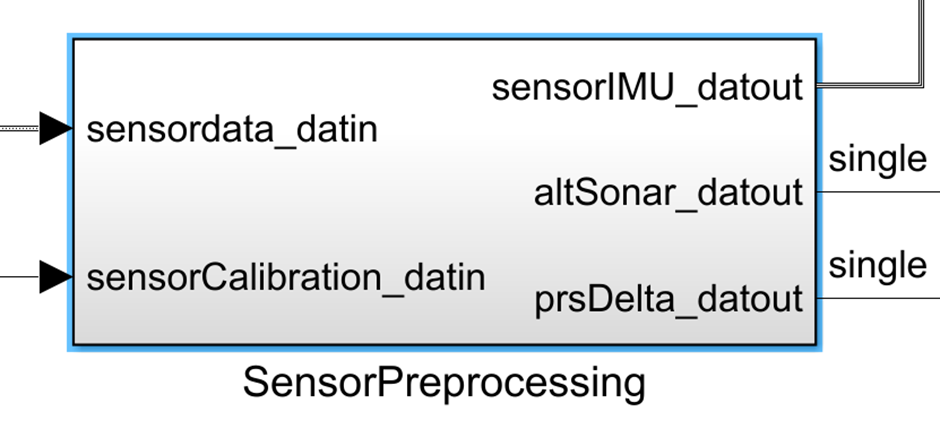
3. State estimator: contains the state observer.



4. Crash Predictor Flags: this contains the logic to turn off the drone in case of anomalies in flight.



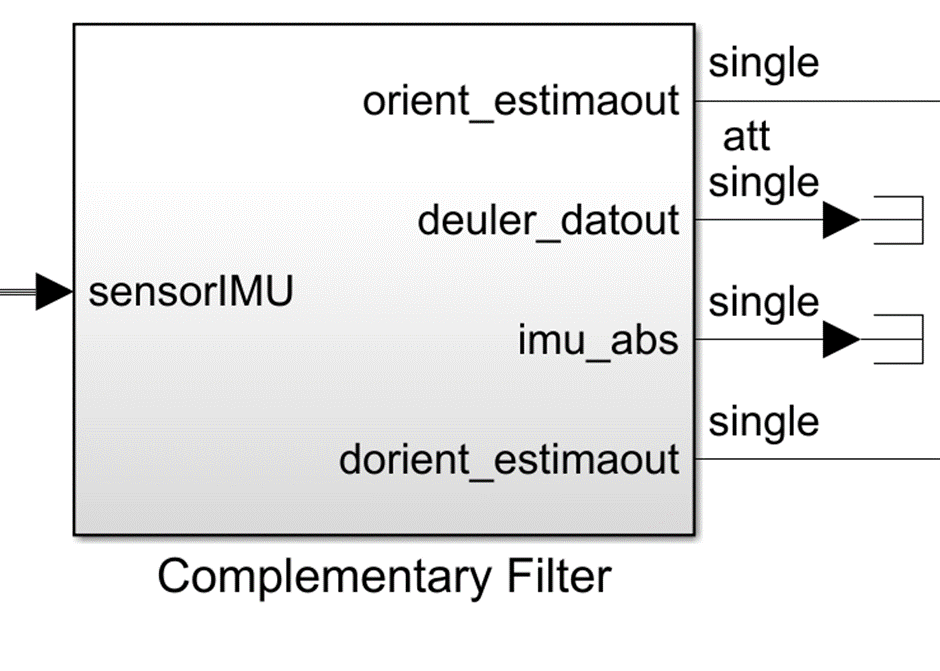
5. Sensor processing block- It consists of bias removal, coordinate transformation and filtering.



6. Rate transition block- The two subsystems work at different rates: The Image Processing System at almost 20 milliseconds (60 Hz as the camera frame rate), while the Control System at 5 milliseconds (200 Hz). This block allows the data transfer between systems with a different rate.

8. Commonly used filters-

Complementary Filter- It is used for estimation of altitude, roll, pitch and yaw angles



**Commonly used filters :**

Complementary Filter- It is used for estimation of altitude, roll,

pitch and yaw angles.

Kalman Filters- It is used for estimation of position and velocity.

**Learnings:**

PID Controllers and its tuning

Image processing: Hough transform, vision based path following concepts, image Binarization

State Machine Diagram with the help of stateflow, Different sensors like accelerometer, barometer, gyrosensor, Magnetometer, IMU

**Further scope of improvements:**

-> There are some noise at the turnings of the path which can be eliminated i.e., it can be made more smooth.

-> It can be modelled to fly in an environment having obstacles in the form of buildings.

**References:**

-> Mathworks

-> Model-based Design of a Line-tracking Algorithm for a Low-cost Mini Drone through Vision-based Control by **PAOLO CEPPI**

-> A Vision-Based Algorithm for a Path Following Problem by **Terlizzi, Mario, and Silano, Giuseppe and Russo, Luigi and Muhammad, Aatif and Basiri, Amin and Mariani, Valerio and Iannelli, Luigi and Glielmo, Luigi**, IEEE

-> ATL\_Drone\_Module - Get, Set, Fly! - **Dr. Ayesha Chaudhary**, Atal Innovation Mission, NITI Aayog.

-> Wikipedia, Quora, Researchgate