E-DRONE DELIVERING SYSTEM

B. Tech
In
Computer Science and Communication Engineering
By

Abhisek Omkar Prasad
Pradumn Mishra
Urja Jain

ABSTRACT

Systems can broadly be divided into two types; open-loop systems and closed-loop systems. An open-loop system is also called an uncontrolled system. The output of such a system is not controlled as the system has no feedback. A closed-loop system is also called a controlled system. The output of such a system is controlled as it receives feedback. PID is an example of a close loop system. Proportional-integral-derivative (PID) controllers are widely used in industrial systems despite the significant developments of recent years in control theory and technology. They perform well for a wide class of processes. Also, they give a robust performance for a wide range of operating conditions, and they are easy to implement using analog or digital hardware.

TABLE OF CONTENTS

:

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Drones or Unmanned Aerial Vehicles (UAVs) come in two variants – fixed-wing and rotary drones. Rotary drones or multirotor drones consist of various configurations, and some common ones are the helicopter, four-rotor quadcopter, and six-rotor hex copter. Each rotor type has a specific usage and is useful for specific applications. The commonly used type of multirotor is the quadcopter (often shortened to quad). This is because it is a very mechanically, simple system. In quads, each motor spins in the opposite direction of the adjacent motor, as shown in Figure 1. This allows it to achieve vertical lift. Disadvantages include low bandwidth, less gain, and low polarization purity.

In order to maintain its 'pose' in-flight and stay stable, a quadcopter relies heavily on sensors that constantly monitor the quad's 'attitude.' These sensors provide feedback to the quad using which the flight controller makes corrections in the motor spin speed and thus adjusting and shifting the quad in flight so that it remains stable.

Sensors:

- **Inertial Measurement Unit (IMU)**: This sensor is a combination of an accelerometer, which measures linear acceleration (i.e., in X, Y & Z axis relative to the sensor) and a gyroscope which measures angular velocity in three axes, roll, pitch, and yaw. Typically a magnetometer is also present, which by measuring the earth's magnetic field, serves to give a reference direction.
- **Barometer**: Barometers are used to measure the air pressure and hence help flight controllers to predict the height of the quad from the ground. Barometers generally come in handy when the quad is at large enough heights.
- **Ultrasonic Sensor**: Ultrasonic sensors help give a precise height of the quad from the ground. Ultrasonic sensors are extremely useful when flying a quad within 50cm from the ground. Beyond that height, it is recommended to rely on the barometer for estimating the quad height with respect to the ground.
- **Time of Flight Sensor**: It is a distance sensor that uses a laser to estimate the distance.
- **GPS Reciever**: Incorporating this on the UAV enables it to locate itself using the Global Positioning System and other satellite positioning systems.

Quadcopter Motion:

A quadcopter's thrust, roll, pitch, and yaw is changed by manipulating the angular velocity of the motors. Following details about how to move the quad by manipulating the angular velocity of the motors:

- **Thrust/Throttle -** In order to change the quad's height, we can decrease or increase the velocity of all four motors. Reducing the velocity of all the motors lowers the quad height while increasing the velocity increases the height of the quad with respect to the ground.
- **Pitch -** By changing the motor speed of the front and back motors, we can move the quad forward and backward. Increasing the speed of the forward motors moves the quad back while increasing the speed of the back motors moves the quad forward.
- **Roll -** To move the quad left or right, we simply manipulate the left and right motors. By increasing the speed of the two left motors, the quad bends to the right. By increasing the speed of the two right motors, the quad bends to the left.
- **Yaw -** By changing the speed of the alternate motors, the quad yaws to the left or right, respectively.

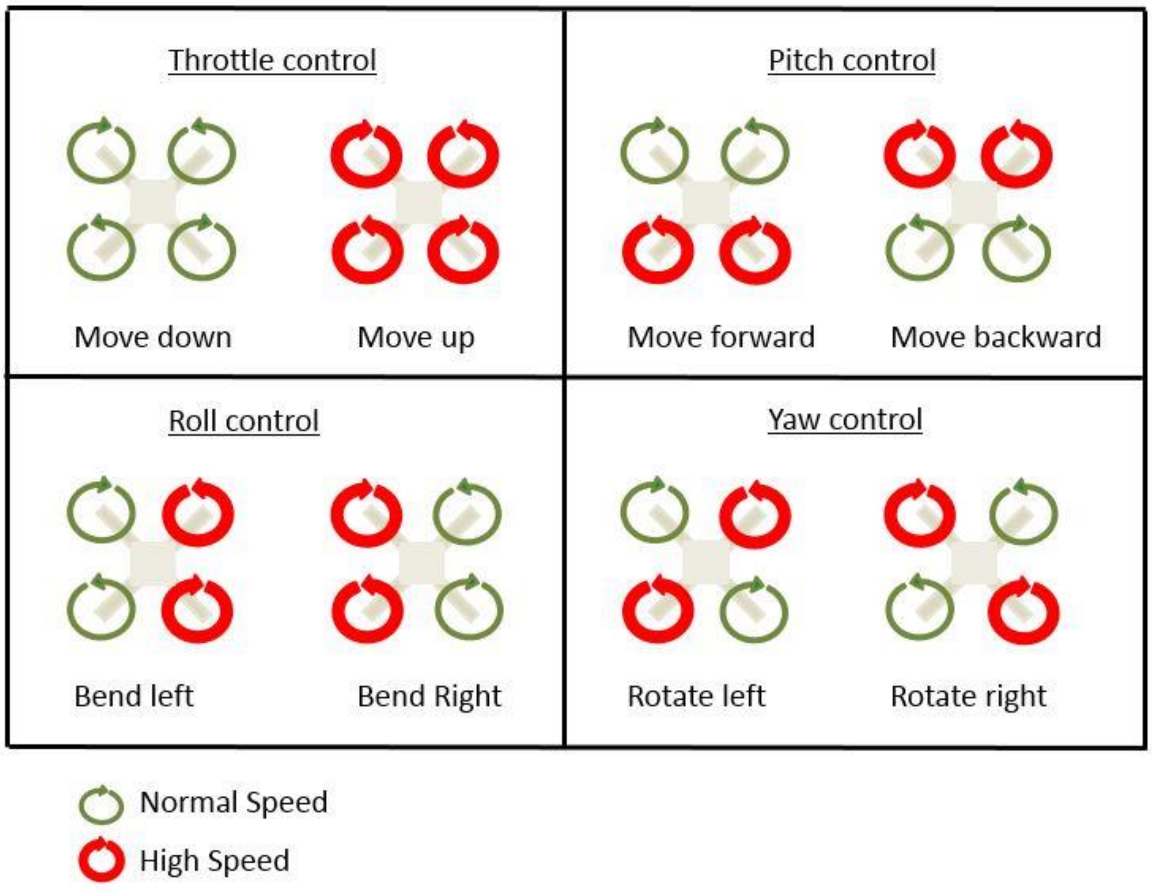


Figure 1: Motion in 3D space

Flight Controller:

A flight controller is responsible for issuing commands to the motors as per the required motion. The flight controller relies on sensor data to generate accurate motion commands so that the UAV maintains its pose as it moves from Point A to Point B. Commonly used flight controllers in quadcopters are PixHawk, KK, and MultiWii.

PID

The PID or Proportional Integral Derivative algorithm is a widely used control loop feedback mechanism in control theory. At the crux of the algorithm, an error value is calculated `e(t)` based on the difference between a set point or the desired point and the measured current point. A correction value is calculated based on Proportional, Derivative, and Integral terms and is then applied to the system to reduce the error value. PID can be applied to any system oscillating like a pendulum and desires to maintain a center reference.

Common examples in robotics that use the PID algorithm are line following robot, object tracking using image processing, and maintaining UAV pose during flight.

CHAPTER 2
CONTROL SYSTEM

Abstract

Systems can broadly be divided into two types; open-loop systems and closed-loop systems. An open-loop system is also called an uncontrolled system. The output of such a system is not controlled as the system has no feedback. A closed-loop system is also called a controlled system. The output of such a system is controlled as it receives feedback. PID is an example of a close loop system. Proportional-integral-derivative (PID) controllers are widely used in industrial systems despite the significant developments of recent years in control theory and technology. They perform well for a wide class of processes. Also, they give a robust performance for a wide range of operating conditions, and they are easy to implement using analog or digital hardware.

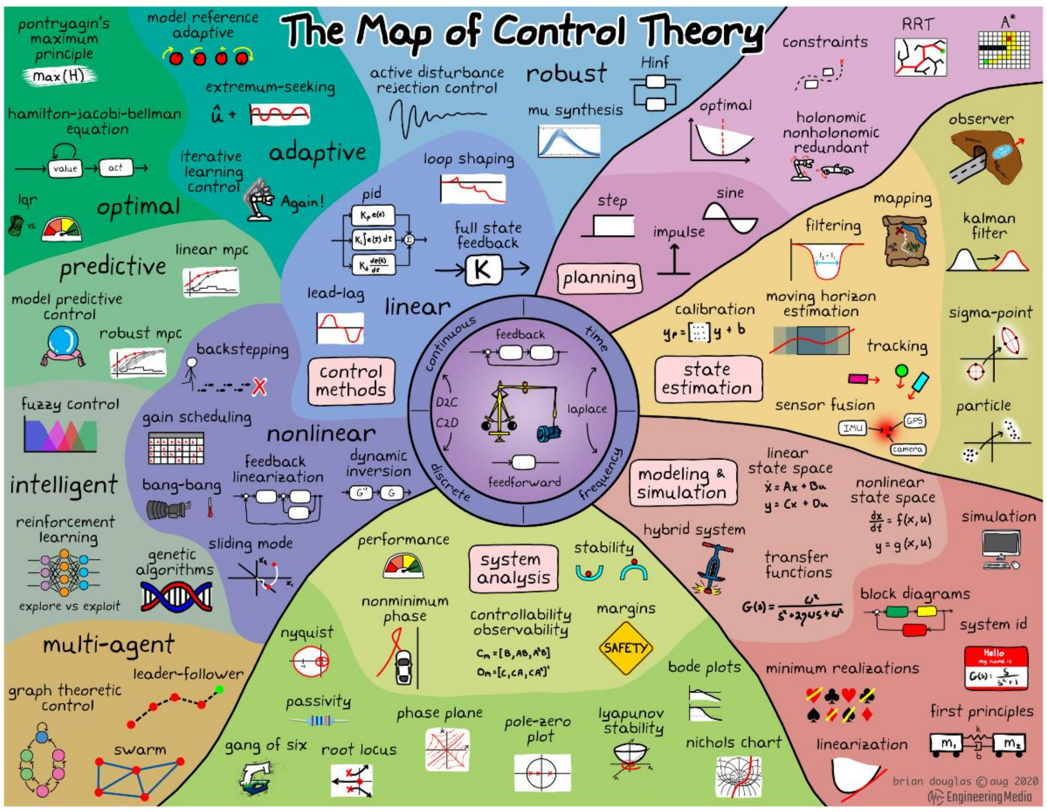


Figure 2: The Map of Control Theory

Types of Controllers

Proportional Controller (P Controller)

P controller is mostly used in first-order processes with single energy storage to stabilize the unstable process. The main usage of the P controller is to decrease the steady-state error of the system. As the proportional gain factor K increases, the steady-state error of the system decreases. However, despite the reduction, P control can never manage to eliminate the steady-state error of the system. As we increase the proportional gain, it provides a smaller amplitude and phase margin, faster dynamics satisfying a wider frequency band, and larger sensitivity to the noise. We can use this controller only when our system is tolerable to a constant steady-state error. In addition, it can be easily concluded that applying the P controller decreases the rise time, and after a certain value of reduction on the steady-state error, increasing K only leads to an overshoot of the system response. P control also causes oscillation if sufficiently aggressive in the presence of lags and/or dead time. The more lags (higher-order), the more problem it leads. Plus, it directly amplifies process noise.

Proportional Derivative Controller (PD Controller)

The aim of using a P-D controller is to increase the stability of the system by improving control since it has the ability to predict the future error of the system response. In order to avoid the effects of the sudden change in the value of the error signal, the derivative is taken from the output response of the system variable instead of the error signal. Therefore, D mode is designed to be proportional to the change of the output variable to prevent the sudden changes occurring in the control output resulting from sudden changes in the error signal. In addition, D directly amplifies process noise; therefore, D-only control is not used.

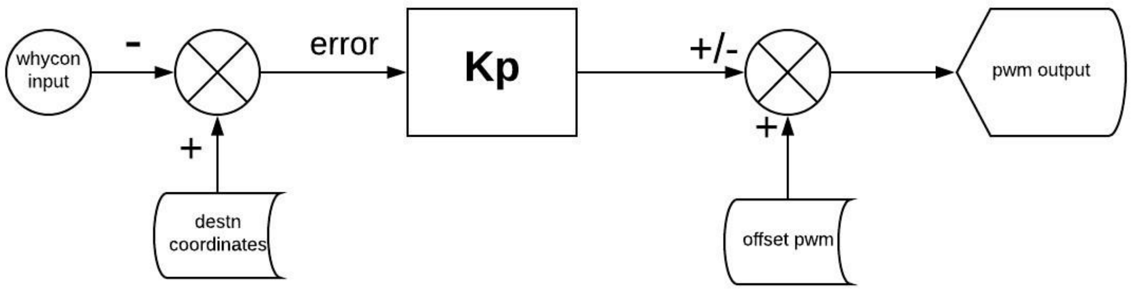
Proportional Integral Derivative (PID Controller)

P-I-D controller has the optimum control dynamics, including zero steady-state error, fast response (short rise time), no oscillations, and higher stability. The necessity of using a derivative gain component in addition to the PI controller is to eliminate the overshoot and the oscillations occurring in the output response of the system.

P Controller

A P controller consists of only a linear gain Kp. The output of such a controller can be simply given as

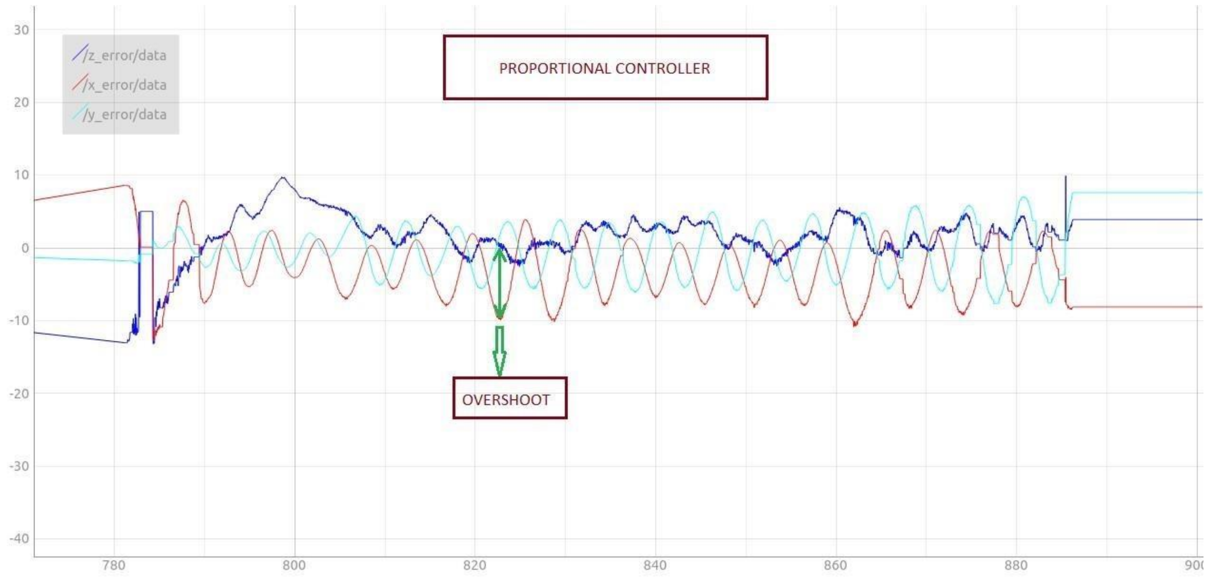
```
output = Kp * error
```



Block diagram

Observations

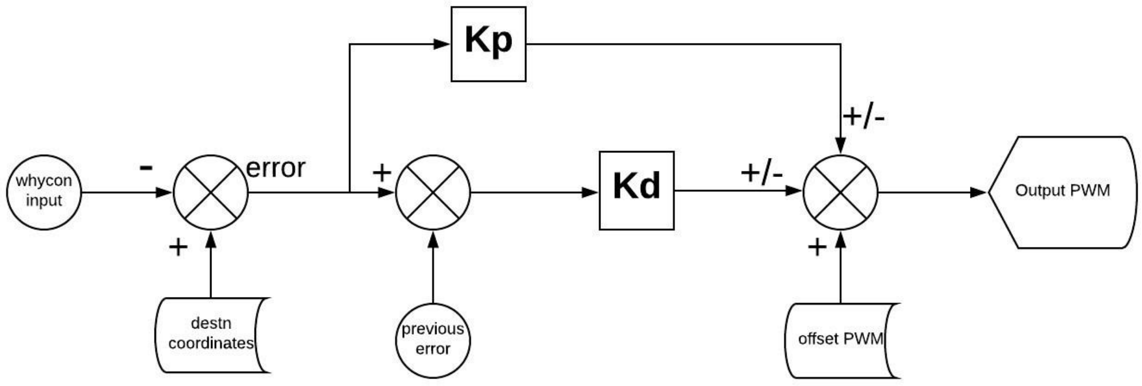
It was observed that the drone never settled at its destination. Instead, it oscillated about its destination. The higher the value of Kp, the more the amplitude of the oscillation, and the drone would be out of the flying zone. Below is a plot between errors in x, y, and z coordinates and time. Clearly, it is visible that the drone was not able to stabilize itself at the destination.



Hence, it was concluded that a P-controller by itself wasn't able to stabilize the system at a required point.

PD Controller

In the previous section, we saw how the P-Controller wasn't successful in stabilizing the system at a given point. It was observed that there were oscillations instead. These oscillations can be damped by using a differential gain along with the P-Controller. The system as a whole is said to be a PD Controller



Block diagram

Here we keep track of the error over time, i.e., sum up the errors over a specified sampling time.

```
Iterm = (Iterm + error) * Ki
```

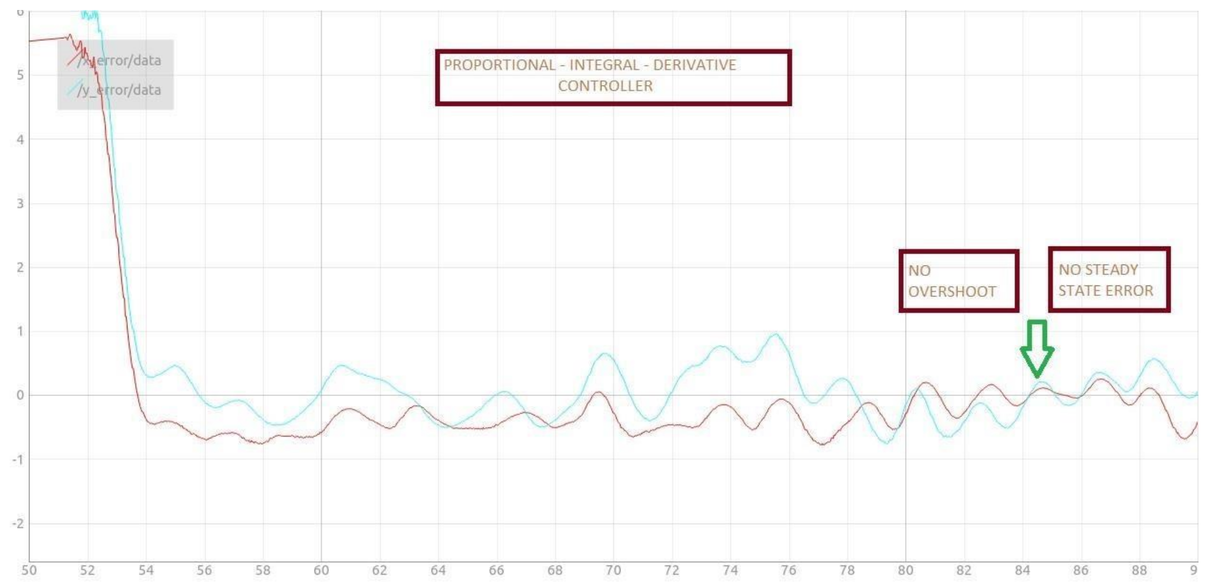
Further explanation regarding the implementation of this Item is given in the code. Note that Ki is calculated keeping the sampling time in consideration. This output will be further added or subtracted to the offset pwm as the need be to give the final result.

```
output = Kp*error + Iterm + Kd*(error - previous error)
```

Observations

The PID controller was successful in hovering the drone above the destination point with minimal error.

It overcomes the steady-state error, which was noticeable in previous controllers. Here is a plot of error and time for a PID controller implemented on the Pluto drone.



CHAPTER 3

QR CODE DETECTION

In today's date, you might have seen QR codes are used in various places to maintain some info. Basically, the QR code is used to get data instantly rather than typing. It can store data on two axes. We will be using this QR in our theme to get the delivery location of any parcel.3.1 Design of microstrip antenna 1 for 2.45 GHz.

Figure 3: QR code sample

Structure Of QR

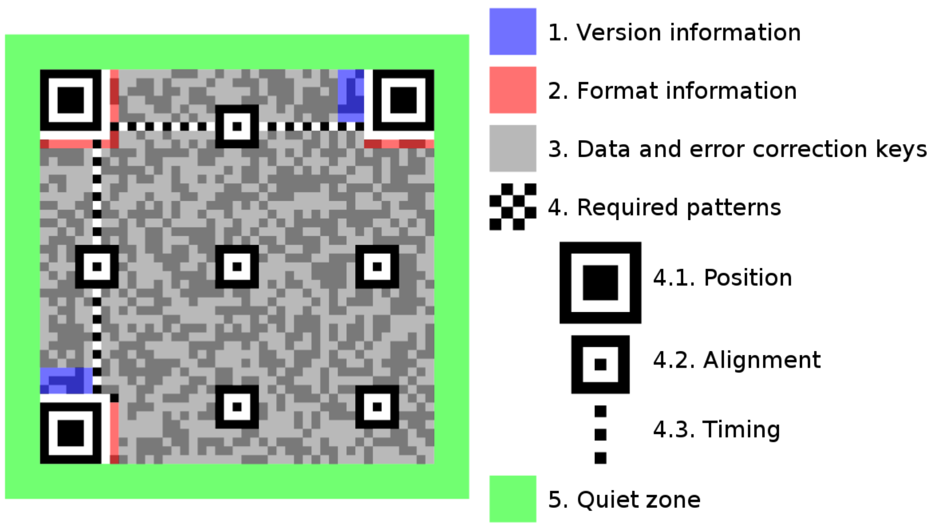


Figure 3.1: Structure of the QR code

The three big blocks show the orientation of QR; hence you can use it in any direction it will detect in which direction it has to start reading. The black and white strip (Timing) shows the size of QR. The marked red area shows the formatting style of QR, such as error correction, format correction, and which type of masking has been used in QR.

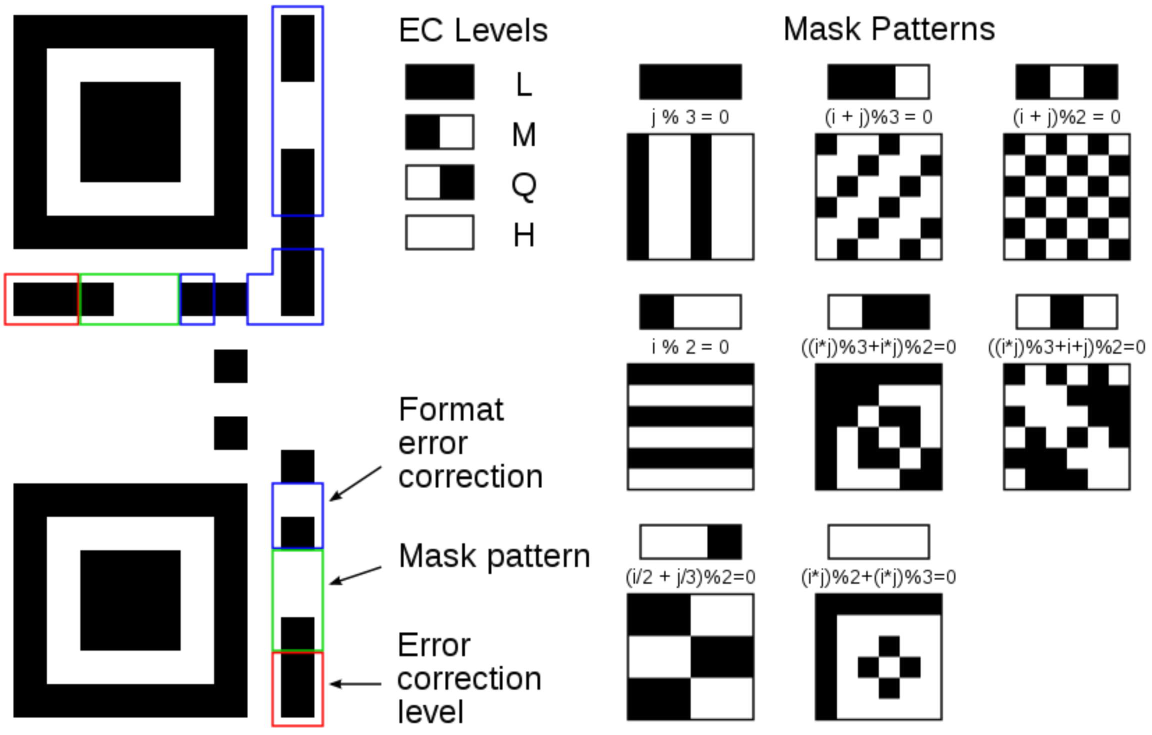


Figure 3.2: QR code masking

Here when it detects the mask pattern, then it applies this masking over the QR data hence making black boxes white and white boxes black. Here is how it looks when we unmask a QR.

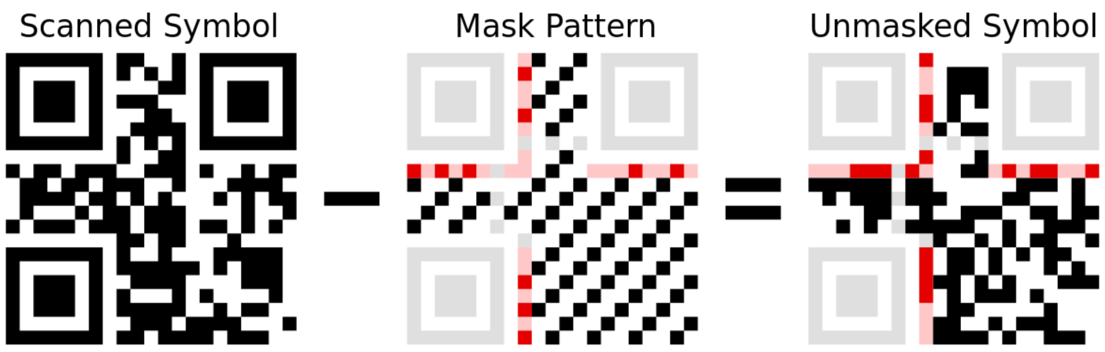


Figure 3.3: QR code unmasking

After unmasking, we are good to go for reading QR.

After unmasking, we are good to go for reading QR.

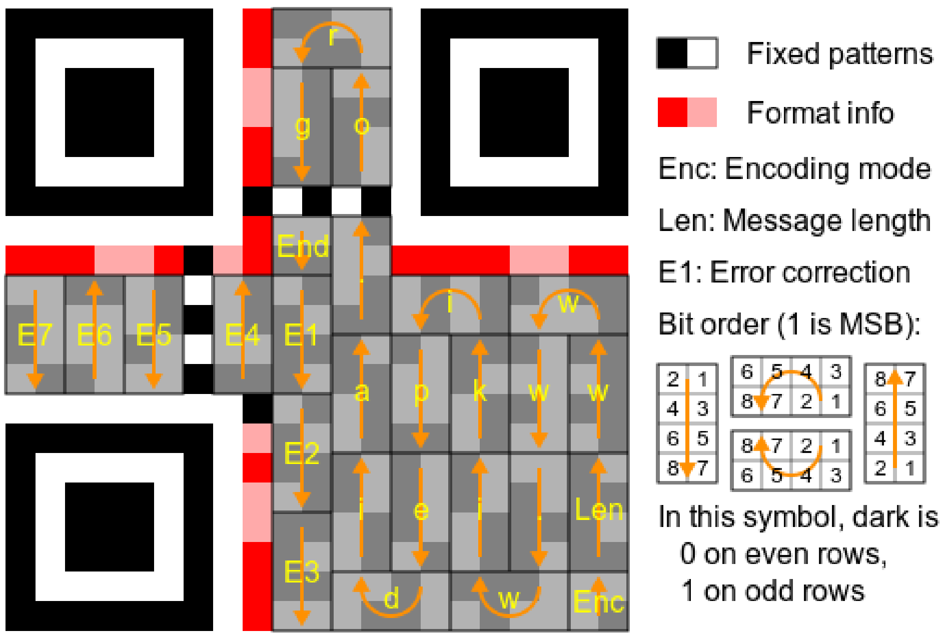
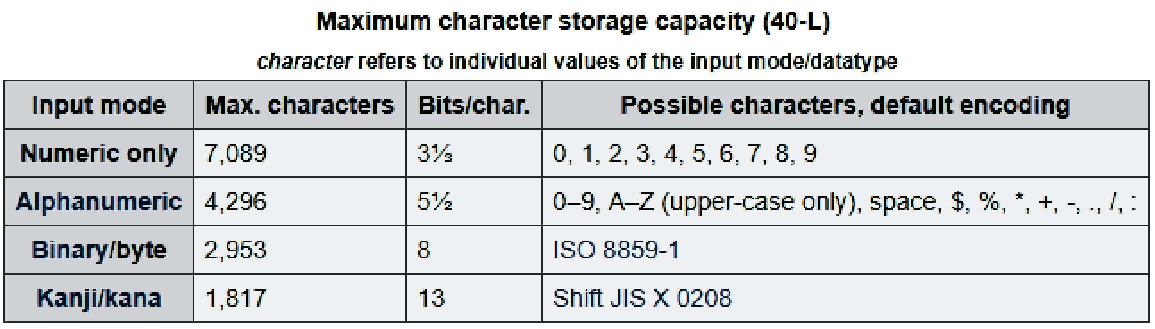


Figure 3.4: Detection of QR code

Here reading starts from the right bottom block. Which is a four-bit block which denotes the type of encoding used in QR code. Then the next eight-block shows the length of the characters. Then from the next block, our data starts. In this, the sum of numbers of the whole block-wise is calculated. for, e.g., when you sum the next block after length block:-

it gives 8 + 7 + 5 + 3 + 1 + 4 = 28

and now this 28 is converted to the respective encoding, which has been applied. Each number represents a specific symbol. Hence that block represents what 28 represents in that encoding data list.



Maximum character storage capacity (40-L)

character refers to individual values of the input mode/datatype

Input mode	Max. characters	Bits/char.	Possible characters, default encoding
Numeric only	7,089	3⅓	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alphanumeric	4,296	5½	0–9, A–Z (upper-case only), space, $, %, *, +, -, ., /, :
Binary/byte	2,953	8	ISO 8859-1
Kanji/kana	1,817	13	Shift JIS X 0208

Table 1: QR code encoding

Here you can find the types of encoding, its store data capacity, and type of data.

QR detection using python

This was the overall working behind the scene. Well, you don't need to do this whole calculation; there are various libraries by which you can scan a QR with a single line of code (just passing an image to these libraries). Example *pyzbar* is such a library that is used to detect QR codes. It is not necessary to stay with this; you can search and find better libraries than this. Detecting QR code using the library is easy, but you must know about how it works and the calculation behind the reading.

CHAPTER 4

TRAINING AND USING CLASSIFIERS

"Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images."

Training Cascades:

Training Haar Cascades requires a lot of sample images. The sample images are categorized into two types:

1. Positive Samples: Images containing the pattern you want to detect, e.g., a photo of a brand logo
2. Negative Samples: Images not containing the pattern you want to detect; these can be any photos not containing the brand logo.

Samples are required in the thousands in order to get robust detections!

Step 1: Creating positive samples

Now, where will we find thousands of images of the landing pad you will have to detect? The answer is we don't; we create them! We essentially take the image we want to detect and overlay it on the negative sample images in varying positions, orientations, and magnifications, and thus create positive samples.

In order to do this, we need to have a folder containing all of the negative images, and we create a .txt file containing all of the filenames in that folder. We will use this file to list all of the background images when creating positives. It looks like follows:

We can create this by using simple Python commands; I won't get into that here; just read and dump filename to a text file. Here is the file for the negative image folder shared with you all.

Back to creating the positives, we do this automatically, with the command:

```
opencv_createsamples -img landing_marker.jpg -bg bg.txt -info info/info.lst -
maxxangle 0.1 -maxyangle 0.1 -maxzangle 1.6 -num 34320
```

Where the parameters: `-img` specifies the image we want to overlay `-bg` takes the background file list `-info` creates a .lst file which lists the positions where the image is overlaid in each negative `-num` is the number of negatives `-maxxangle -maxyangle -maxzangle` is the amount of rotation which we want the positive images to vary in angles (in radians) along the x, y, and z (roll, pitch, and yaw) axes respectively. The camera on our quadcopter does not have many distortions. Hence x and y angles are kept low since we are unlikely to encounter such scenarios.

Since there are 30k+ images, this operation might take a few minutes. These are some of the output images:

Now we have both the required positive and negative images!

We will create a positive. Vec file that will keep a record of our positives. We do that using the following command:

```
opencv_createsamples -info info/info.lst -num 34320 -w 24 -h 24 -vec
positives.vec
```

where the parameters: `-w` & `-h` are the width and height of the samples respectively, 24 pixels is an optimal number, going above yields little benefits `-vec` specifies the file to be created

Step 2: Training

- Training involves a lot of parameters; we are going to keep most of the parameters to their default values. Refer to this link if you want more information.
- We can train our classifiers to either detect Haar-like feature types or Local Binary Patterns (LBP). Haar features undergo floating-point computation, while the LBP features just require integer computations.

- Thus while training, LBP is much much faster than Haar. There is no noticeable difference in the outcomes in our use case. Hence we will be using LBP features.
- This process runs only on a single core of your CPU (requires OpenCV to be differently compiled to run on multiple cores), yet training using LBP in our case requires less than 10 minutes, compared to potentially hours if not days with the Haar Cascades.

Training involves feeding a certain number of positive and negative samples into each stage.

```
opencv_traincascade -data data -vec positives.vec -bg bg.txt -numPos 2287 -numNeg
2287 -numStages
```

You will notice that while running this, it will not complete the full 15 stages we have given; this is because the false positive error rate will be reached at Stage 10 itself. The resulting output is perfectly usable and works well. You might add more negative images to refine the algorithm according to your decisions and thinking.

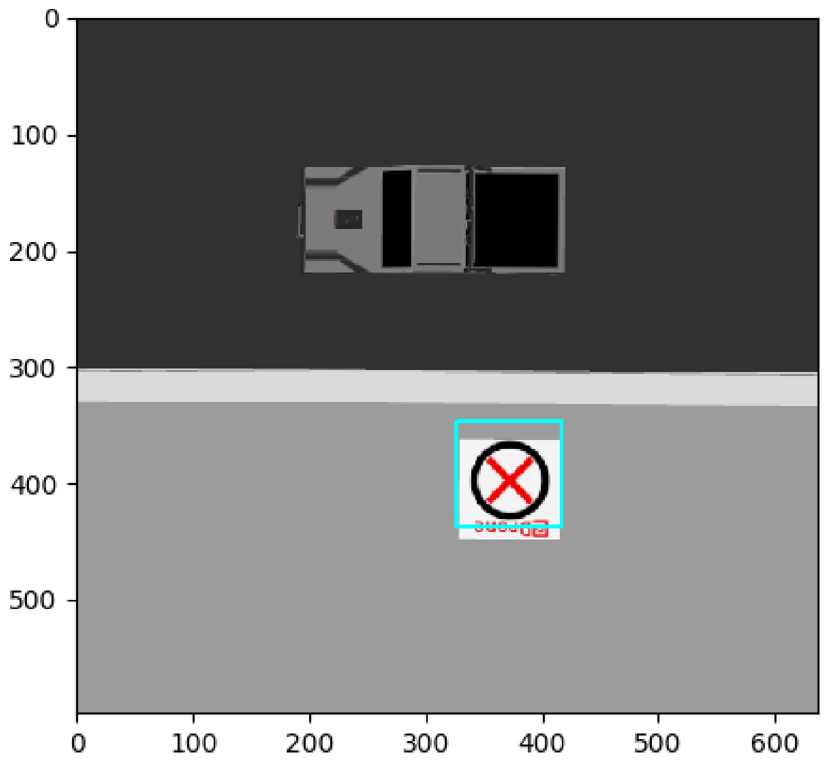
After the training is complete, you will notice that the cascades have been generated in the data folder. We will use these for our detection. These files can be used anywhere. Their small size is one of the advantages of this technique.

Step 3: Detection

- The weblink shared above contains a sample script. Refer to it.
- To learn and experiment with the various parameters, the most important ones are the `scaleFactor` and the `main neighbor's` threshold. Adjust them accordingly. Here is a helpful post for the latter (check this one out too), but in our use case, if there are a lot of repeated/false detections, you may need to consider training again.
- Here is a short script to detect the landing marker and display it

```
import cv2
from matplotlib import pyplot as plt

logo_cascade = cv2.CascadeClassifier('data/cascade.xml')

img = cv2.imread('test_1.png')  # Source image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

logo = logo_cascade.detectMultiScale(gray, scaleFactor=1.05)

for (x, y, w, h) in logo:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 0), 2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

You can see the output below. Notice the symmetric nature of the marker allows us to get away with a smaller angle range while creating the positives.



Conclusion

Drones are set to become the future of logistics with their reduced cost, higher convenience, and delivery times of less than 30 minutes. Both online retailers and brick and mortar stores will adopt it to smoothen their last-mile delivery process. Stores like Walmart can experience increased efficiency and convenience with their local presence, while online retailers like Amazon can offer quick deliveries at reduced costs. The early adopters will be the winners as they will be able to provide their services at cheaper and quicker rates leading to brand promotion.