**Non Functional Requirements**

**Accessibility**: Can be downloaded over the internet and ported on the drone. The end user need not know how to run the code. Just need to know how to save the executable on the drone and keep it running.

**Availability**: When there are unexpected bugs, the drone may run haywire. It is necessary to force stop the drone and resolve the error.

**Confidentiality**: Since the system is aimed to be completely autonomous, there shall be no breach of any data.

**Efficiency**: The system handles a continuous stream of recorded or live video feed. The response time is less than 20ms per frame.

**Reliability**: Since it is a computer vision based system, if the payload is not differed, the results will remain very accurate. But if there is an anonymous environment, then the accuracies cannot be predicted right away. Also this is highly subjectible to the whether conditions. If it is rainy, the image input shall be hazy, thus providing a low reliability.

**Safety**: Since the system involves using a drone for movement and some physical cutting mechanism to extract the fruit, it is important to maintain safety to avoid harm to the drone, trees or humans in the vicinity. But this is out of this project’s scope and is highly dependent on the drone calibration. It is also advised to have a method of manual override the drone in case of emergency.

**Dependencies**: A drone that can be controlled using an executable python code and can send a readable video stream from the camera attached to it to the computer board on which the python code runs.

**Assumptions**: There is availability of a programmable drone and an optimal camera for video feed. It is also assumed that the drone can be calibrated and manoeuvred accurately. The input also needs to be in focus and should have to other obstructions so that the entire fruit can be accessed.

**Functional Requirements**

EXTRACT\_FRUIT

**preprocessing(image)**

i/p: Image

o/p: Image

Functionality: Rescales, Gaussian blurs and converts from RGB to HSV

**get\_color\_mask(image)**

i/p: Image

o/p: Masked Binary Image

Functionality: Identifies colour on the image and creates a binary mask of the same

**apply\_morphology(mask)**

i/p: Binary Mask

o/p: Binary Mask

Functionality: Applies methods on the mask to separate joined masks and joining split blobs.

**apply\_flood\_filling(mask)**

i/p: Binary Mask

o/p: Binary Mask

Functionality: Removes noises from the Binary masks

**get\_contours(image)**

i/p: Masked RGB image

o/p: Contour image, approximate shape of largest contour, area of contour

Functionality: Find the contours on the masked image and use it to obtain various features and shape specific parameters.

**extract\_fruit(image,variance)**

i/p: Image and laplacian variance of the image

o/p: Window co-ordinates

Functionality: Apply various techniques to recognize fruits in the image and extract a bounding rectangle around the fruit.

**get\_window(contour\_val)**

i/p: Contour Values

o/p: Window co-ordinates

Functionality: Use boxPoints to estimate the bounding rectangle around the contour

**get\_fruit(image,variance)**

i/p: Image and laplacian variance of the image

o/p: Contour values

Functionality: Find the contour of the fruit and it’s respective parameters

TRACK\_FRUIT

**camshift(fruit\_image,fruit\_window,cap)**

i/p: Image and window co-ordinates

o/p: None

Functionality: It continuously tracks the same fruit contour and manoeuvrers the drone accordingly.

**get\_adjusted\_window\_width(fruit\_window\_width,poly\_approx\_len,area,no\_of\_contours)**

i/p: window width, contour parameters

o/p: adjusted window width

Functionality: This uses camera or contour parameters to calibrate and readjust window width so that triangular similarity property can used on it for better distance calculations.

**track\_fruit(fruit\_contour,fruit\_window,cap,pd)**

i/p: Fruit contour and window co-ordinates

o/p: None

Functionality: It takes input the contour and fruit window to estimate distance to the fruit in the image, keep tracking the window if needed and send appropriate directions to the drone.

DISTANCE

**get\_width()**

i/p: None

o/p: Real life width of the fruit to calculate

Functionality: Keep track of the real width of fruits and use it to calculate real-time distance using triangular similarity property.

**get\_distance\_to\_camera(knownWidth, perWidth)**

i/p: Width of the fruit and Width of the recognized fruit contour

o/p: Distance (in cm)

Functionality: Use the triangular similarity property to calculate distance to the fruit in the image

**get\_distance(p1,p2)**

i/p: Two co-ordinate points

o/p: Distance in pixel units

Functionality: Calculate the Euclidean distance between two given co-ordinate points.

**consistency\_check(distance,pd)**

i/p: Present distance and previous 3 calculated distances

o/p: True or False

Functionality: Checks if present distance is matching the previous distances to make sure the same contour is detected to avoid noises and return True if consistency is proved.

DRONE

**get\_direction(window,distance,pivot)**

i/p: Window co-ordinates and camera scope centre pivot point

o/p: Direction (Distance, Tilt and Height rise)

Functionality: The difference between centre of the camera image scope and the image contours centroid is calculated. This is used to instruct the drone how much to move, tilt or raise/fall.

**move\_drone(img,direction,msg)**

i/p: Direction

o/p: None

Functionality: Used to instruct the drone how and where to move.

DRAW

**draw\_ellipse(tracked\_contour,tracked\_contour\_image)**

i/p: Contour and image

o/p: Image, ellipse co-ordinates

Functionality: We first estimate an ellipse around the contour, draw it on the image and obtain the ellipse parameters

**find\_stalk(tracked\_contour,tracked\_contour\_image)**

i/p: Contour and image.

o/p: Stalk co-ordinates

Functionality: Given the elliptical parameters ,get the major and minor axis. Then use the axis with higher slope and obtain the topmost point as the co-ordinate of the stalk.

**display(image,msg)**

i/p: Image

o/p: None

Functionality: Given a image, rescale it and display it