Introduction

Farming has always been the backbone of Indian economy, and fruit harvesting has been a pioneer in this field. With the rising demand for the same, the agricultural world has turned to technological advancements for better produce. In this paper, we aim to try to solve one demand and make a contribution. The main users for this project would essentially be farmers and industries pertaining to fruit farming sectors.

Problem Statement

The main purpose here to harvest/extract fruits from it’s tree while having a video feed for the same. Provided a video stream and a controllable vehicle (with a camera docked on it and an extracting mechanical mechanism) calibrate to detect and recognize the fruit in vision of the camera and cut the fruit.

And since we plan to cut the fruit in this scope, the point of cutting would be the stalk of the fruit as this is the most fragile point on the fruit.

Literature Survey

Gantt Chart

Activity Diagram

Requirements

## Module / Scenario 1

|  |  |
| --- | --- |
| **Reqmt 1** | **Requirement** |
| Drone | A programmable drone which is compatible with drone kit library and has necessary communication models. (Along with a docking kit). Includes appropriate FCU. |

## Module / Scenario 2

|  |  |
| --- | --- |
| **Reqmt 2** | **Requirement** |
| Stereo Camera | A good resolution light weighted stereo camera with wireless interfaces. |

## Module / Scenario 3

|  |  |
| --- | --- |
| **Reqmt 3** | **Requirement** |
| Computing System | A high-power computation unit and a storage unit to perform necessary computer vision task. |

## Software Requirements

This is run as a OSX to raspberry PI integration.

A basic SQL database to store the 3D coordinates.

Python integrated with drone kit to control the drone.

Tensor Flow, opencv for recognition algorithms.

# Performance Requirements

(Tentative to change, vision-based parameters and metrics will be implemented)

* 1 drone caters to 1 user at a time only. (1:1)
* The analytical data may be used on the cloud by multiple users. (1:N)
* The primary database will consist of configuration files and a DB of minimum records to store path and waypoints to the 3D coordinates of the fruit.
* The program can be reused on multiple drones at different locations or eventually useful to coordinate drones together.

Review of Our methods

This paper shall explain in detail about how to detect, recognize the fruit, approximate the distance in the image and recognize the stalk using two methods

1. Computer Vision Methods

2. Deep Learning Methods

Both these methods shall take in input a basic video frame and provide the co-ordinates of the bounding window of the recognized fruit.

These window co-ordinates shall be used for further computer vision methods to send respective directions to the controllable vehicle for further movement.

We shall now venture deep into both the techniques used.

**COMPUTER VISION METHODS**

**Introduction**

Computer vision has been used since the early 40s, but has evolved exponentially since then. We shall use OpenCV, a python based library to use the concepts of computer vision and solve our problem.

Installations needed: python, opencv, imutils, numpy.

The entire process involves the following steps:

1. Preprocessing

2. Colour Masking

3. Morphological methods

4. Contour Extraction

5. Window Extraction

1. PRE-PROCESSING

We need to refine the image to remove the noises and make it suitable to be shape-recognizable.

The first step id to rescaling it to a smaller size. This is done so that the redundant noise is removed from the image and only the essential features are retained.

The next process would be to apply Gaussian Blur. This is done so that when we want to obtain a colour mask, the excessive colour noise is removed.

Finally we perform a RGB to HSV conversion. The RGB channel is light sensitive, thus making is unreliable to use on a generic scale and time. Thus we try to extract out the features of HSV (Hue, Saturation, Value) so that further techniques remain light invariant.

Image Laplacian variance for Blur detection. The variance of the Laplacian histogram of an image can be used as a relative threshold to determine if an image is blur or crisp (on a relative scale only).

2. COLOUR – MASKING

Once the image has been processed, it is now time to decipher what’s in the image. The first and easiest aspect of determining something in the image is using colour. Since the image has been converted to HSV, we can now apply certain thresholds to choose what range of colour needs to be displayed and what can be masked out.

Apple: ([0,100,80] – [10,256,256]) + ([170,100,80] – [180,256,256])

Banana: ([20,50,50] – [30,256,256]) + ([60,50,50] – [70,256,256])

3. MORPHOLOGICAL METHODS

Once the image has been colour masked, it is now present as multiple blobs due to similar colour noise. Thus we resort to morphological methods to alter these methods and apply

Set an appropriate kernel size and shape ( (3,3) square or ellipse )

i) Erode to remove small blobs and split masks spanning over multiple fruits.

ii) Dilate to combine multiple blobs that are pertaining to the same fruit.

iii) Morphological Open and Close ... ???

Note: The ratio of cycles of Erode and Dilate in whatsoever order plays an important role in obtaining the right mask.

4. CONTOUR EXTRACTION

Once the final mask has been obtained, it is now required to obtain the border of each blob in the mask. (Contour extraction can be obtained directly on the original image without the above steps, and also canny edge can be used for contour approximation, but in this use case the above method proved to be better).

On obtaining the contour, we need to smooth en the obtained contour shape, thus we apply convex hull to obtain a trench filled shape of the contour. (Hough circle transforms can also be used, but are restricted to circular shapes and might cause over-fitting).

On the further estimated contour, we now obtain various parameters to determine the shape and properties of the contour. We obtain the area of the contour. We apply approxPolyDP and check the number of results obtained. If the length of the results are over a certain threshold, we can classify it as a circular shape, else other polygon shapes.

5. WINDOW EXTRACTION

Once we recognize the contour of our necessary shape and area, It is now necessary to obtain a bounding window around it. Thus we use minAreaRect and boxPoints to estimate the co-ordinates of the window and convert it to the necessary format we desire.

We also obtain the centroid of this window that is used as a pivot reference for vehicular movement later.

The number of contours obtained are further used to estimate if the vehicle is closer or further from the object in the image. (more contours means farther distance)

**DEEP LEARNING TECHNIQUES**

**DRONE CONTROL and FRUIT TRACKING**

Once the window of the fruit has been obtained (using either techniques) it is now needed to manoeuvrer the vehicle (drone) closer to the fruit and at the right angle. Since the scope of the camera is constant, we assume that the mid pivoted view of the drone camera is also constant relative to the complete image shape(size).

**DISTANCE CALCULATIONn**

But before any analysis can be performed, we need to calculate the distance to the object from the camera. This could be done using sensors or using a stereo camera for accurate results. But here we would use a single camera and no sensors to approximate the distance to a known object.

We use the “Triangular similarity property”.

We first assume that the size of a fruit, apple is 7.5cm by calculating manually with a scale. Then we place it a known distance and record an image with it. Now we use triangular similarity to estimate the focal length of the camera used.

distance = (knownWidth \* focalLength) / perWidth

distance – known distance. knownWidth- 7.5cm. PerWidth-width of the contour of the fruit in the image

Once we calculate the focalLength in the above formula, we can use the same later to estimate new “distance” to the fruit in the image that is recognized by the bounding window of it’s contour around it.

**FRUIT TRACKING**

We now have an estimate distance to the fruit. We want to cut the fruit only after we reach a certain threshold closeness distance to it. Thus we classify the distance into three aspects so as to perform different and appropriate actions.

1. Far: >200 cm. At this distance, not much can be derived, thus the drone is instructed to just move closer to the recognized fruit.

2. Medium: >120 cm & <200 cm. At this distance we want to keep the focus on a single fruit only. Thus we now use Cam-shift algorithm to keep tracking the fruit window as we move closer to the fruit.

3. Close: <120 cm. When the drone is extremely close to the fruit, it should now identify the stalk and cut the fruit at the right point.

The drone is made to move only after checking consistency with the distance to the contour to ensure that the same contour is being tracked.

The directions to the drone for it’s movement are defined by:

1. Distance to the object from the camera

2.Tilt: The horizontal tilt the drone should endure to align the centre of fruit to the camera’s scope

3. Height: The vertical rise the drone should endure to align the centre of fruit to the camera’s scope

These parameters are calculated as a relative difference between the pivot co-ordinate point of the camera scope and the centroid/mid-point of the contour of the recognized fruit.

*#Here 30cm is the distance the drone should be from the fruit to cut it*

*Distance = distance-30*

*#Tilt degree is dependent on camera scope angle, assuming scope angle is 180*

*degree = pivot\_x/180*

*Tilt = int((pivot\_x-x)\*degree)*

*#Height is the rise and fall amount and depends on the image height dimensions*

*Height = pivot\_y-y*

**ELLIPSE AND STALK SHAPE DETECTION**

When the drone is at a close range to the fruit, we now estimate an Ellipse with the contour and obtain their major and minor axes. The slope of each of these axis are calculated and the top most co-ordinate of the axis with higher slope on average with the topmost extreme point is assumed to be the point of extraction on the stalk

*ellipse = cv2.fitEllipse(tracked\_contour)*

*topmost = tuple(tracked\_contour[tracked\_contour[:,:,1].argmin()][0])*