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

contour\_val = get\_fruit(image,variance)

window = get\_window(contour\_val)

**preprocessing(image):**

maxsize=max(image.shape)

scale=700/maxsize

image=cv2.resize(image,None,fx=scale,fy=scale)

image\_blur=cv2.GaussianBlur(image,(7,7),0)

image\_blur\_hsv=cv2.cvtColor(image\_blur,cv2.COLOR\_BGR2HSV)

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

min\_color=np.array([0,100,80])

max\_color=np.array([10,256,256])

min\_color2=np.array([170,100,80])

max\_color2=np.array([180,256,256])

mask1=cv2.inRange(image,min\_color,max\_color)

mask2=cv2.inRange(image,min\_color2,max\_color2)

mask=mask1+mask2

**apply\_morphology(mask):**

kernel = np.ones((3,3),np.uint8)

dilation = cv2.dilate(mask,kernel,iterations = 2)

erosion = cv2.erode(dilation,kernel,iterations = 3)

dilation = cv2.dilate(erosion,kernel,iterations = 15)

mask\_closed=cv2.morphologyEx(dilation,cv2.MORPH\_CLOSE,kernel)

mask\_cleaned=cv2.morphologyEx(mask\_closed,cv2.MORPH\_OPEN,kernel)

**get\_contours(image):**

image, contours, hierarchy = cv2.findContours(image,cv2.RETR\_LIST,cv2.CHAIN\_APPROX\_SIMPLE)

hull = cv2.convexHull(contour)

approx=cv2.approxPolyDP(hull,0.1\*cv2.arcLength(hull,True),True)

area = cv2.contourArea(hull)

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

im,im\_blur,im\_blur\_hsv = preprocessing(image)

color\_mask = get\_color\_mask(image\_blur\_hsv)

color\_morphed\_mask = apply\_morphology(color\_mask)

color\_filed\_mask=apply\_flood\_filling(color\_morp\_mask)

colour\_masked\_image = cv2.bitwise\_and(image,image,mask = color\_filled\_mask)

contour\_val = get\_contours(colour\_masked\_image)

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

rect = cv2.minAreaRect(tracked\_contour)

box = cv2.boxPoints(rect)

box = np.int0(box)

window\_width = ( get\_distance(box[0],box[1]) + get\_distance(box[2],box[3]) )/2.0

window\_height = ( get\_distance(box[0],box[3]) + get\_distance(box[1],box[2]) )/2.0

mid\_x = (box[0][0]+box[1][0]+box[2][0]+box[3][0])/4

mid\_y = (box[0][1]+box[1][1]+box[2][1]+box[3][1])/4

window = [mid\_x,mid\_y,window\_width,window\_height]

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

r,t,w= cv2.CamShift(dst, track\_window, term\_crit)

pts = cv2.boxPoints(ret)

pts = np.int0(pts)

img2 = cv2.polylines(frame,[pts],True, 255,2)

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

fruit\_width = get\_width()

fruit\_window\_width = fruit\_window[2]

fruit\_window\_width = get\_adjusted\_window\_width(fruit\_window\_width,approx,area,no\_of\_contours)

distance = get\_distance\_to\_camera(fruit\_width,fruit\_window\_width)

pivot\_y,pivot\_x = int(fruit\_image.shape[0]/2),int(fruit\_image.shape[1]/2)

if(not(consistency\_check(distance,(pd\_1,pd\_2,pd\_3)))):

direction = get\_direction(fruit\_window,distance,(pivot\_y,pivot\_x))

if(distance>160 or no\_of\_contours>10):

move\_drone(fruit\_image, direction[:-1],"Far")

elif(distance<=100):

if(len(fruit\_contour)>=5):

tracked\_contour\_image = draw\_ellipse(fruit\_contour,fruit\_image)

move\_drone(fruit\_image, direction[:-1],"Close")

else:

move\_drone(fruit\_image,direction[:-1],"Medium")

**get\_width():**

apple = 7.5

return apple

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

focalLength = 1200

distance = (knownWidth \* focalLength) / perWidth

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

int(math.sqrt( (p1[0]-p2[0])\*\*2 + (p1[1]-p2[1])\*\*2))

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

x,y = window[:2]

pivot\_x,pivot\_y = pivot

Distance = distance-30

Tilt = int((pivot\_x-x)\*(pivot\_x/180))

Height = pivot\_y-y

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

cv2.putText(img, "Distance:%.2fcm Tilt:%.2funits Height:%.2funits %s"%(direction[0],direction[1],direction[2],msg),(0, img.shape[0] - 20), cv2.FONT\_HERSHEY\_SIMPLEX,0.5, (0, 255, 0), 1)

display(img,"Final Image")

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

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