LINCODE HACKATHON

Cookie Challenge

25.09.2020

TEAM : The ChainCoders

DARSHAN DALAL

HARDIK GANDHI

VAIBHAV POPAT

Overview

Created a vision system which takes an image of a cookie as input and gives the vertical and horizontal diameters as output. Counts the number of chocolate chips in the cookie. It also detects if the cookie has any damages. Finally, it displays the hex codes of the top 3 colors in the cookie.

Features

1. Image Acquisition

To capture the image and save it in our disk, *VideoCapture* function from *OpenCV* library is used. [1]

2. Finding diameter of the Cookie

Read the cookie image from the *image* folder.

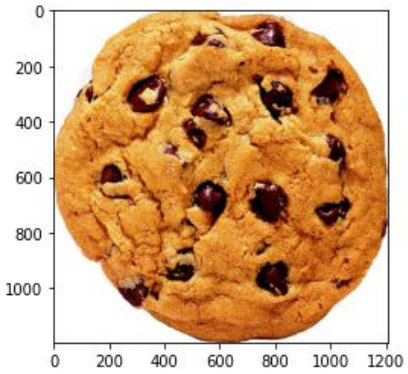
Processed and thresholded the image for precise bounding box detection.

The bounding box is drawn over the image - sides of which give the vertical and horizontal diameters.

Key functions used -

1. GaussianBlur
2. Cv2.threshold

Next, the image is cropped to capture only the portion in the bounding box - thus eliminating other irrelevant information.



Drawing bounding box over the cookie

[1] Reference - https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html

3. Finding Dominant Colors in the Cookie

The image is reshaped to be a list of pixels.

K-means clustering algorithm is used over the pixel intensities.[2]

Built a histogram of clusters and then created a figure representing the number of pixels labeled to each color.[3]

Normalized the histogram.

Calculated and plotted the relative percentage of each cluster on the bar chart.

The length of the color in the chart is proportional to its number of pixels in the image.

Finally, the RGB values are converted to display the top 3 Hex Codes in the image.

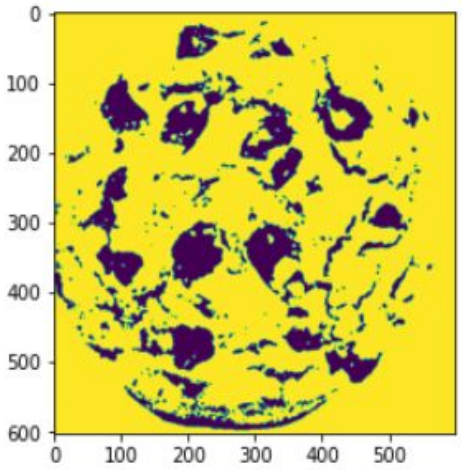
Color Palette showing the hex codes of the top 3 colors

4. Checking the number of chocoChips to be greater than a minimum

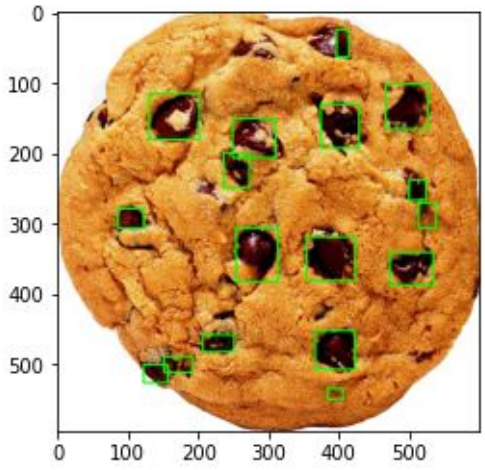
1. MedianBlur -Image Smoothing to remove noise
2. Binary Thresholding of the image- Image Thresholding
3. Resized image maintaining the aspect ratio with width as 600 px
4. Cv2.dilate- Morphological Operation
5. cv2.findContours - Detected contours in the image i.e. the group of pixels having the same color/intensity values.

[2] Reference - https://docs.opencv.org/master/d1/d5c/tutorial_py_kmeans_opencv.html
[3] Reference - https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html

6. boundingRectangle- to draw bounding rectangle around the contour
7. Non Maximal Suppression - For taking the union of overlapping choco chips
8. Computed the area of the bounding boxes and sorted them by the bottom-right y-coordinate of the bounding box
9. Looped over the picked bounding boxes and drew them over the image.
10. Compared the chocochips count to the minimum required threshold



Global thresholding of the image



Detecting exact location of choco chips

5. Detecting Broken or Damaged Cookies

1. cv2.adaptiveThreshold - Thresholded the image based pixels based on its surrounding pixels.
2. cv2.Canny - Detected significant edges in the image.
3. cv2.morphologyEx - Performed morphological operations (smoothening and removal of noise)
4. cv2.findContours - Detected contours in the image i.e. the group of pixels having the same color/intensity values.

5. cv2.contourArea - Thresholded major contours by limiting the area to values greater than 1,50,000 px (observed value after experiments with a variety of images).
6. cv2.arcLength - Thresholded major contours by limiting the arc length to values less than 3,000 px (observed value after experiments with a variety of images) -

    a. If arc length is less than 3000 px, the cookie is a complete circle meaning a quality product.

    b. If arc length is greater than 3000 px, it was found that all of them were

Fig. drawing arc length over normal cookie

Fig. drawing arc length over damaged cookie