Implementation By Simon Kalouche

Running on the local Pi Camera Module

1. Every XX minutes capture a picture
2. Name the picture with a timestamp in the name
3. Upload the named image to dropbox folder using Pi's Wifi
4. Delete the image from the Pi local directory to not fill up memory

Running on a non-local Server

1. Initialize iPhone API and connection using PyiCloudService
2. Initialize DropBox Connection
3. Input images to inventory tracking program
    a. Download the two most recent images from DropBox that were taken from the raspberry pi module and uploaded via wifi (their names are time stamped)
    b. Manually select 2 images and enter them as arguments in terminal
4. Implemented auto-alignment algorithm that accounts for slight misalignments in the two images (alignments due to vibration or small changes in camera viewpoint) so that the two images can be compared
    a. Using OpenCV find ECC transform of image 1 with respect to image 2
    b. Run ECC (Enhanced Correlation Coefficient Maximization) algorithm to warp perspective and affine of the second image to match that of the first image.
5. Pre-process the image
    a. Resize and rescale the input images to be 1000 pixels wide while maintaining aspect ratio
    b. Convert images to grayscale
    c. Run Canny edge detector on both gray-scale images
    d. Increase contrast on gray images
6. Find the difference between the two images (pixel difference)
    a. Subtract pixel values of second image from first in gray-scale and edge images
7. Manually threshold the gray-scale image to filter out noise
    a. Convert low confidence pixels (pixels with gray-scale value above a certain threshold) in the difference image between the two gray-scale images to 255 (black).
8. Find the contours in the gray difference image then calculate the area of each found contour. Contours in the difference image represent areas of where the second image differs from the first.
9. Filter contours by area size and only keep contours above a certain threshold defined to be the image area (number of square pixels) that make up a small item on a retail shelf
10. Check for overlapping contours in the image
    a. If a contour is overlapping and fully enclosed within a larger contour, delete the smaller, enclosed contour

11. Draw a box around all missing items (i.e. Enscribe the contours that have area greater than a certain threshold with a rectangle)
12. Crop that rectangle out of the original colored image 1. This cropped image is the item that was removed from the shelf between the first and second image captures.
13. Determine what the removed product is
    a. Convert cropped image to grayscale
    b. Use SURF to extract features from the removed item
    c. Find keypoints and descriptors directly using SURF
    d. Match descriptors using Brute Force of Flann Matchers
        i. Compare each image in a database of product images to the cropped image (image of the removed item) running SURF on all the images in the database and comparing features between each image in the database with the SURF features found in the cropped image (removed item image).
        ii. Choose the product with the highest correlation and confidence from the feature matching algorithm
14. Determine if the product is out of stock or if there are some left
    a. Look at the pixels on the second input image in the location of the cropped portion of the difference image (i.e. look at the shelf in the spot where an item was removed after it was removed).
    b. Crop this rectangle from the second image, convert it to grayscale and detect SURF features on this image.
    c. Run the brute force matcher to check if there are any feature matches between the image of the product take from the shelf (predicted in step 13) and the image of what was directly behind the product taken off the shelf.
    d. Count of the number of matches there were between the item taken and the item behind the item taken (if there is one). A low stock confidence number means there weren't many feature matches and the product is most likely now out of stock
15. Compile stock status and product(s) removed from shelf information and create a text file log of which items were taken off the shelf with a timestamp for each.
    a. Upload the log file to dropbox
16. Send out-of-stock predictions directly to iPhone as alert for the store manager to replace the out-of-stock item.
17. Repeat every time a new image is detected to be uploaded to DropBox using always the 2 most recent images (i.e. the first image in the instance i+1 of running the algorithm will be the same as the second image from instance i)