CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)

HOMEWORK – 10

HW10_Lastname_Firstname_part_a:
Create a model to classifying pixels as raspberries (based on the color) or not-raspberries, based only on the color of the pixel.
We did this for a banana or an orange doll in class. Use the input image with a picture of a raspberry bush in it.
Your program may change the color space of the image to any other color space it wants to use.
It then uses a pre-built classifier to determine which pixels are raspberry pixels, and returns 1 if the pixel is a raspberry
and it returns a 0 if the pixel is not.
The returned image type is a uint8 image.
Your output image should be called HW10_Lastname_Firstname_part_a_output.jpg.
Ans: The important thing to observe here is that I have to classify the raspberries using the **color**.

I have referred the code which was explained by professor in class to classify raspberries.
Steps to classify raspberries (based on color):

1) Read the input image
   Code:
   ```
   im_rgb = imread( filename );
   ```

2) Manually select foreground and background from the input image. We are using ginput() for selecting the foreground and background.
   Code:
   ```
   [x_foreground, y_foreground] = ginput();
   [x_background, y_background] = ginput();
   ```

3) I am using LAB color space as it segments the image more clearly.

4) Recollecting all the foreground and background pixels from the a* and b* channel of image. I am using sub2ind. It returns the linear index equivalent of given vector. Then we prepared the attributes that we will use further to calculate the Mahalanobis distances.
   Code:
   ```
   % out = sub2ind( matrix_size,rowSub, colSub)
   % sub2ind returns the linear index equivalent to row subscript(rowSub)
   % and col subscript(colSub)
   % Using this we are extracting forground and background indices
   foreground_indices  = sub2ind( size(im_a), round(y_foreground), round(x_foreground) );
   background_indices  = sub2ind( size(im_a), round(y_background), round(x_background) );

   % Recollecting the pixels corresponding to background and forground
   % indices from a and b channel.
   foreground_a       = im_a( foreground_indices );
   foreground_b       = im_b( foreground_indices );
   background_a       = im_a( background_indices );
   background_b       = im_b( background_indices );

   % Creating attributes of background, foreground and entire image
   foreground_ab      = [ foreground_a, foreground_b ];
   background_ab      = [ background_a, background_b ];
   im_ab              = [ im_a(:) im_b(:) ];
   ```

5) Compute the Mahalanobis distance for each pixel in image with the selected background and foreground.
   Code:
   ```
   % Compute mahalanobis distance of each pixel in image from the
   % foreground selected.
   mahal_foreground    = mahal( im_ab, foreground_ab );

   % Compute mahalanobis distance of each pixel in image from the
   % background selected.
   ```

```
mahal_background    = mahal( im_ab, background_ab );
```

6) Classify to prepare a class of all the pixels whose foreground Mahalanobis distance is lesser then the background Mahalanobis distance. By doing this we are classifying foreground raspberries from the background.
Code:

```
% Classify a Class using the condition if distance to FG is < distance to BG
b_is_target_object_class    = ( mahal_foreground.^(1/2)) < ( mahal_background.^(1/2));
```

7) From the class created in the above step, we will compute the mean and standard deviation and using that we will prepare the matrix of inliers.
Code:

```
% Prepare a model of the foreground Mahalanobis distance
% NOTE:  We must take the Square Root
foreground_dists      = mahal_foreground.^(1/2);
fg_dists_class0   = foreground_dists(b_is_target_object_class);
dist_mean       = mean( fg_dists_class0 );
dist_std_01     = std(  fg_dists_class0 );

% Toss everything outside of one std, and re-adjust the mean value:
b_inliers     = ( fg_dists_class0 <= (dist_mean + dist_std_01) ) & ( fg_dists_class0 >=
      (dist_mean - dist_std_01));
the_inliers    = fg_dists_class0( b_inliers );
dist_mean       = mean( the_inliers );
```

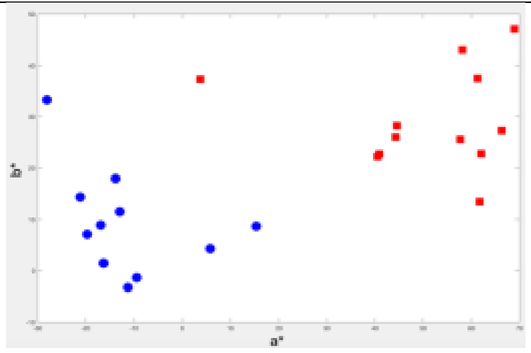
8) Then we will compute the mean of the inliers and use it as threshold. That is all the points below threshold are considered as foreground and selected.
Code:

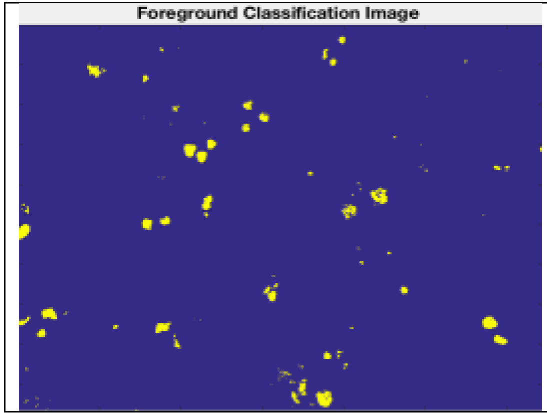
```
% Toss everything outside of one std, and re-adjust the mean value:
b_inliers     = ( fg_dists_class0 <= (dist_mean + dist_std_01) ) & ( fg_dists_class0 >=
      (dist_mean - dist_std_01));
the_inliers    = fg_dists_class0( b_inliers );
dist_mean       = mean( the_inliers );

% Use a distance to target variable as rules for inclusion
threshold          = dist_mean;
guess_class0         = foreground_dists <= threshold;
out_image          = reshape( guess_class0, size(im_a, 1), size(im_a, 2) );
```

9) Then we will save the output image in the form of output JPEG file.
Code:

```
final_output  = im2uint8(out_image);
imwrite(final_output,'HW10_Jain_Yash_part_a_output.jpg');
```

(Original image)	 (Selected background and foreground points)



HW10_Lastname_Firstname_part_b:
Create a model to classifying pixels as golf ball or not, based only on the local texture. Use your favorite texture metric.
Use the golf-ball image provided to you.
Build a pre-built classifier to determine which pixels are golf-ball pixels. Then run the classifier on the input image and have it
write out an output image, called HW10_Lastname_Firstname_part_b_output.jpg. In this image, a pixel is a 1 if the pixel is a
golf-ball, and a 0 if it is not.
Ans: The important thing to observe here is that I have to classify the golf balls using the **texture**. So, we have to select background and foreground based on texture. *Once the background and foreground is selected rest of the steps will remain the same as 'part a'*.

Why can't we use color classification in this image?
We are classifying balls by texture but not color because we have multiple colors of balls available. It was not the case in raspberries. All raspberries are red in color but we have 3 different balls colors in this image. So, we have to use texture to classify this image.

As we have to classify using texture, I need to classify the background and foreground (balls) based on the texture. So, I am using the concept of superposition and template matching taught in class. In part a, we are manually selecting the pixels based on color, but here we cannot do that as have to identify by texture. So, I hardcoded values to select the patch of grass and then prepare the local average filter of same length as that of patch. I played around with values between 0 to 100 and found 20-70 works the best.

Before applying the local average filter on the patch, we find the mean of that patch. Then we subtract that patch by this mean value to convert it to edge detector. By doing this we can detect the correlation. For this part of homework, I patch is have selected is 'grass' which is actually background while the non-selected part will be foreground 'balls'. So, I am trying to use the template matching fundamentals to select the background. One such example was taken by professor in class for camouflage image.

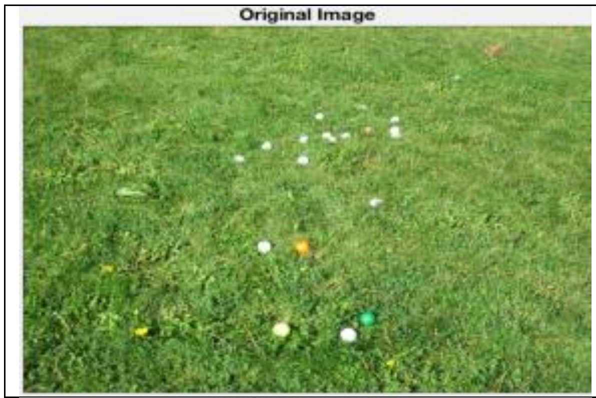
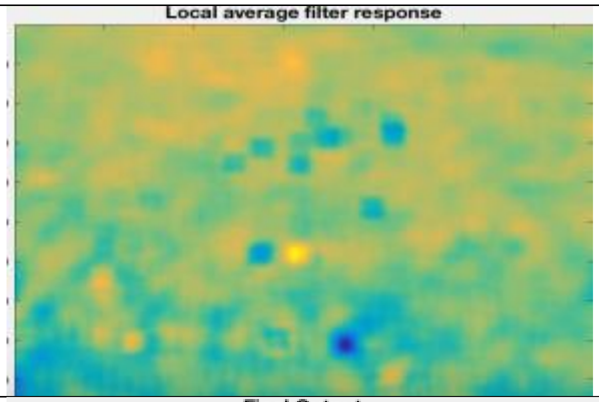
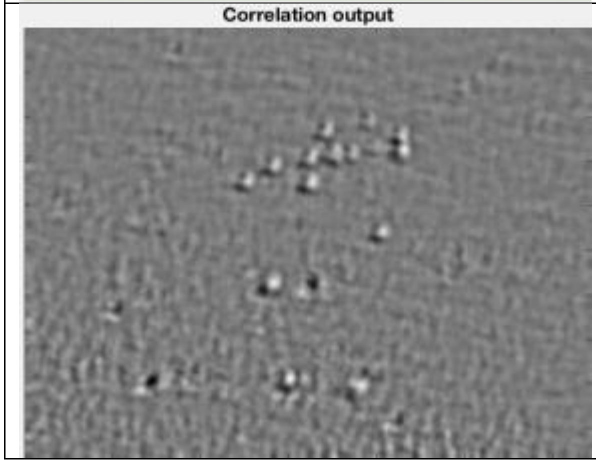
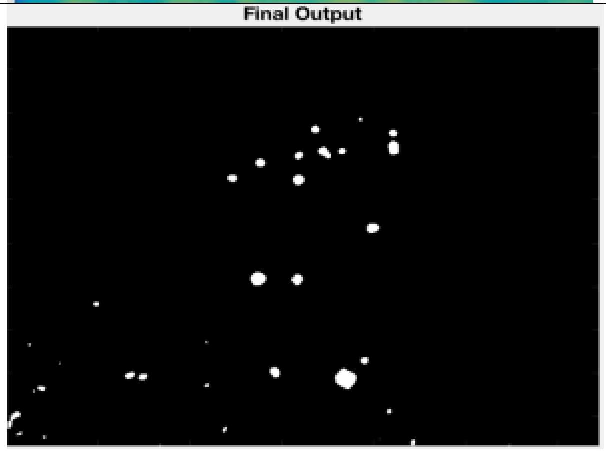
Once we are done selecting foreground and background, further steps will remain the same as part a.

I tried taking h and s channel of hsv and tried combining them but it gives lots of noise in output, though balls are segmented easily. So, I commented this code and tried with only b channel of lab and it gives much better output.

Using this approach, it is selecting the yellow leaves on the ground. We can ignore them by computing the area of the region selected. But there is an issue with that, since the leaves are larger in size, if we try to remove them area wise we may end up removing small sized balls.

Saving final output in image file using imwrite:

```
imwrite(final_output,'HW10_Jain_Yash_part_b_output.jpg');
```

General algorithm:

Flowchart of image chaining is below (next page):

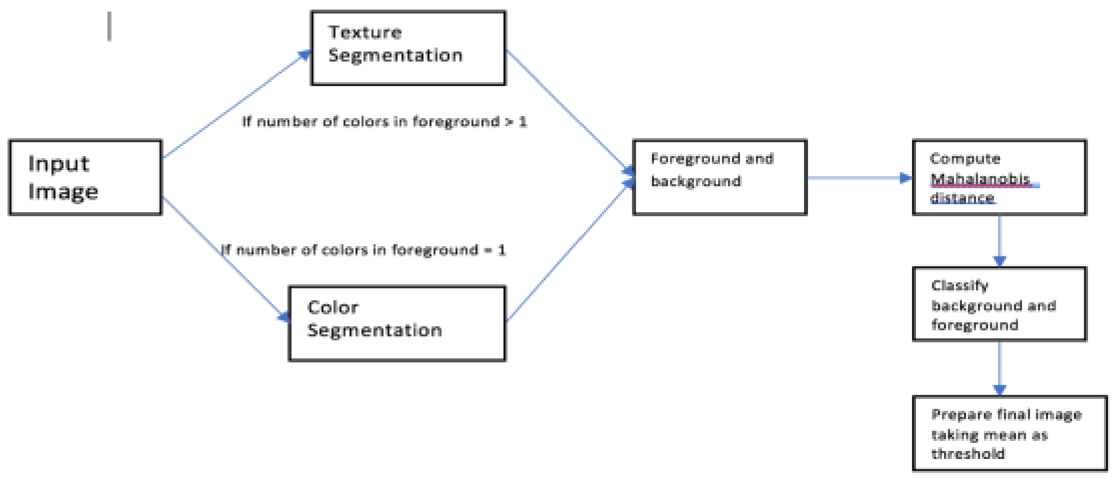

Input Image
Texture Segmentation
If number of colors in foreground > 1
If number of colors in foreground = 1
Color Segmentation
Foreground and background
Compute Mahalanobis distance
Classify background and foreground
Prepare final image taking mean as threshold

Figure: Flowchart showing steps of the generalized algorithm to follow