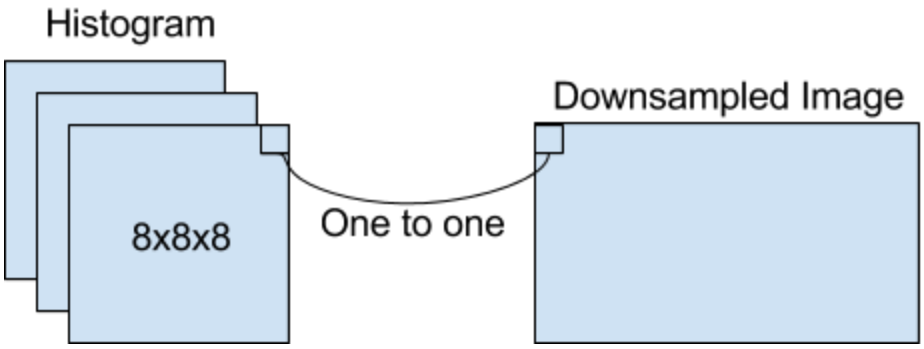
CMPT 412 Assignment 3

Kai Zhang
301087916

Part 1

While working with the Swain Ballard, a lot of difficulty arises from constructing the histogram out of the image. After downsampling the image from 8 bits to 3 bits, it is unclear how the downsampled image maps to histogram, let alone agreeing on the shape of the histogram. In the end, I used 8x8x8 matrix to represent a color histogram. For every pixel in the downsample image, there exists a direct one to one relation to a bin in the histogram.




```
I = zeros(8,8,8);
for i = 1:size(collage,1)
    for j = 1:size(collage,2)
        px = collage(i,j,:);
        I(px(1), px(2), px(3)) = I(px(1), px(2), px(3)) + 1;
    end
end
```

Figure . For every pixel in the image, it is one to one mapped one bin in the histogram
https://github.com/jeholmes/MATLAB-Backprojection/blob/master/tracker.m#L289

Now that I can get a histogram for each image, I can generate 2 histograms [M, I] where M is for mask image and I is for the collage image. Now I get R which is same size histogram as I and M.

$$R_i = \min(\frac{M_i}{I_i}, 1)$$

```
rhisto = min(M ./ I, 1);
```

Figure. R can computed in one line through elementwise divide
https://www.mathworks.com/help/fixedpoint/ref/rdivide.html

From the histogram R back to the image, we can take advantage of one to one relationship and back project R onto the collage image. For every pixel in the back projected image, the R has a corresponding bin count that tell us the match in that image. Higher count corresponds to greater match.

```
for i = 1:size(collage,1)
    for j = 1:size(collage,2)
        px = collage(i,j,:);
        resultimg(i,j,:) = rhisto(px(1),px(2),px(3));
    end
end
```

Figure . Simple back projection from R to result image
https://github.com/jeholmes/MATLAB-Backprojection/blob/master/tracker.m#L289

Now we can filter some noises using median filter and locate the bright spots. Below are some sample results:

Finding Crunchberries

Finding Garan

Part 2

Part 2 heavily relies on work from part 1. The black cup video clip contains 40 frames and we want to track black cup for every frame. A big issue is when motions blur occurs on the cup; now the histogram of the cup is no longer accurate and we can find the cup.

Figure. As the black cup becomes more motion blurred, tracking becomes harder

However out of total 40 frames, only 6 frames out of track so performance is somewhat appropriate. Implementation is similar to part 1 that is applied to every frame of the video. Back projection is nevertheless fine tuned using sobel edge detector to capture the rim of the cup. Unfortunately it also capture other edges with the same color histogram.

```
resultimg1 = edge(resultimg,'Sobel',[],'horizontal','nothinning');
resultimg2 = bwareaopen(resultimg1, 305);
imshow(aframe)
blk = regionprops(resultimg2, 'BoundingBox');
blkregions = [blk(1).BoundingBox(1), blk(1).BoundingBox(2),blk(1).Boundin
rectangle('Position', blkregions, 'EdgeColor','r', 'LineWidth',3);
```

Figure. Result image is post processed with sobel edge detector and noise removal to improve tracking

The main issue is that I have no way specifying which part of the frame is the most important. Even the edges are similar in color, they are not what I am looking for. I am not taking advantage of the information from the previous frame. That is exactly what mean shift sets out to do.

Part 3

It was unclear from the slides whether mean shift should calculate straight moving means from raw intensity values of the image or from resulting image back projected from color histogram. So referenced from jholmes, color histogram simplifies the mean shift calculation process due it only being 512 bins. We can calculate the mean at each frame as follows:

```
for i = 1:size(resultimg,1)
    for j = 1:size(resultimg,2)
        if sqrt((i-mean(2))^2 + (j-mean(1))^2) < 2*r
            all = all + resultimg(i,j);
            xcnt = xcnt + j*resultimg(i,j);
            ycnt = ycnt + i*resultimg(i,j);
        end
    end
end
mean(1) = xcnt/all;
mean(2) = ycnt/all;
```

Figure . Mean calculated at each frame from back projection image

For every pixel on the back projected image, if the pixel is within the radius of the previous mean, we want to include it in the total count and as well as row x and column y count. Once total count is updated, mean in both x, y directions can be shifted in the right direction.

```
imshow(aframe);
blkregions = [mean(1), mean(2), 50, 50];
rectangle('Position', blkregions, 'EdgeColor','r', 'LineWidth',3);
```

Figure . Mean Shift plotting requires no post processing

```
resultimg1 = edge(resultimg,'Sobel',[],'horizontal','nothinning');
resultimg2 = bwareaopen(resultimg1, 305);
imshow(aframe)
blk = regionprops(resultimg2, 'BoundingBox');
blkregions = [blk(1).BoundingBox(1), blk(1).BoundingBox(2), 50, 50];
rectangle('Position', blkregions, 'EdgeColor','r', 'LineWidth',3);
```

Figure . Part 2 Color Tracking has noise issues so post processing is required

Mean shift significantly outperforms basic color tracking frame by frame. The major advantage being that is the range is limited to radius of the previous mean. Similar color edges can now be easily ignored because it is not within radius. Nevertheless, the results cannot be generalized to both cups so more investigation is need.

(Please see the tracking directory for different tracking results)

REFERENCES

1. Jeholmes
   https://github.com/jeholmes/MATLAB-Backprojection/blob/master/tracker.m#L28

2. Swain and Ballard
   https://canvas.sfu.ca/courses/35529/files/folder/Notes%20and%20Reprints?preview=6678254

3. Mean shift
   https://canvas.sfu.ca/courses/35529/files/folder/Notes%20and%20Reprints?preview=6683953