# Pill removal from Blister Pack

**Introduction**

The project incorporates techniques which can be implemented so as to automate the task of removing pills from Blister Packs without touching them. Further, it is extended to keep a count of the remaining tablets in the pack so that the user can be intimidated well in advance to buy a new pack.

This report basically emphasises on Image Processing which has been used to detect the position and later, the number of pills in the blister pack. Image processing is a technique which is becoming more and more popular in everyday application. The image captured is processed as per our requirement so as to extract the desired features from it. As a consequence, only the parts of image conveying relevant information is kept while the rest is discarded. This helps to eliminate the noise which can cause interference while applying algorithms to perform what we want.

The project describes two approaches taken to correctly identify the pills in the blister.

**Approach 1:**

1. The user is given the option of either using a downloaded image (for calibration or demo purpose) or to capture a new image from the webcam or an external camera.
2. The image acquired is checked whether it is monochrome or coloured index image.
3. The captured image is an RGB image. So, it is separated into its R, G and B components.
4. Here, the program is written for identifying red colour pills. But, it can altered so as to identify other colour too, just by modifying the threshold values of the mask.
5. Our aim is to create a Red colour mask which can be applied to all the three R, G & B images so as to intensify the red characteristic of the image.
6. We need to set the threshold values for each image so that relevant features can be extracted. For this to happen judiciously for each blister image, we use Otsu's Method which is incorporated in the 'graythresh' function available in the Image processing toolbox of Matlab. It sets the threshold to a value beyond which the pixels of the R, G or B 2D image starts becoming prominent. The returned value is scaled to a value between 0 and 1. Due to this, it needs to be multiplied with 255.
7. Since we want red color to be prominent here, we set the low and high threshold as shown in the program. The blue and green thresholds are taken complimentary to the red one.
8. These threshold values are applied to the original R, Y and B images.
9. We form our mask by combining the above obtained images. Therefore it can be thought of as a red mask. To enhance this mask, we use functions bwareaopen( ) and imfill( ) to fill any holes or areas which are lesser than the desired area.
10. Now that our mask is ready, we apply it to the R, Y and B images and concatenate them to form the final processed RGB image which contains prominently Red part only.
11. This makes identification of the pills easy. To actually identify the pills, we use the function imfindcircles( ) which detect circular shapes in the processed image as per the given radius. To visually see the results, we use viscircle( ) to draw actual circles over the identified areas, i.e, tablets.
12. To count the number of pills, we actually count the number of circles identified by taking the length of the radii[ ] array returned by imfindcicles( ), which contains the radius of the identified circles.

13. Coordinates of the tablets are also returned by imfindcircles( ) function which can be used to compute the Euclidean distance of each tablet from the top left corner of the screen in pixels. So, we can identify the next target pill to be popped out. So, the coordinates of that pill can be mapped to the

actual blister pack and respective command can be given to a stepper motor through Arduino to hit that particular spot so that the pill pops out of the Blister Pack.

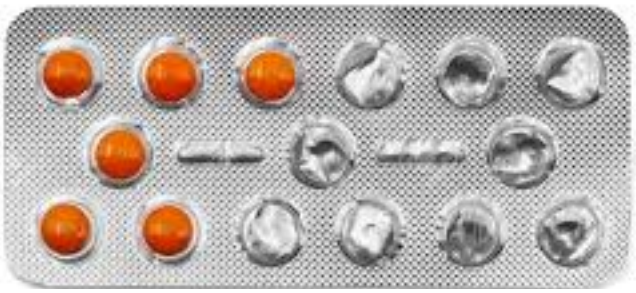

**Fig.: Original Blister Image**

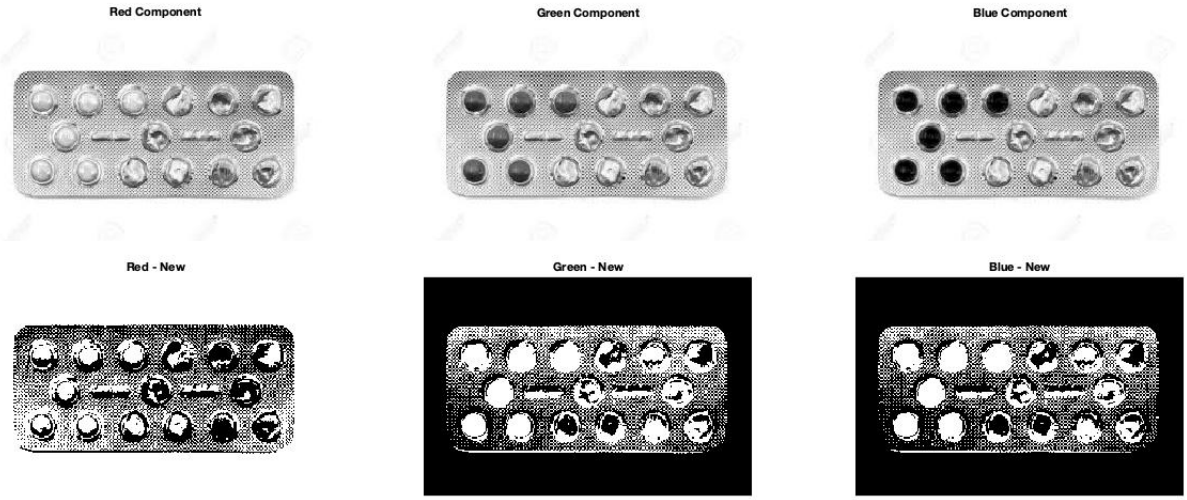


**Fig.: R, G and B components separated from original image**

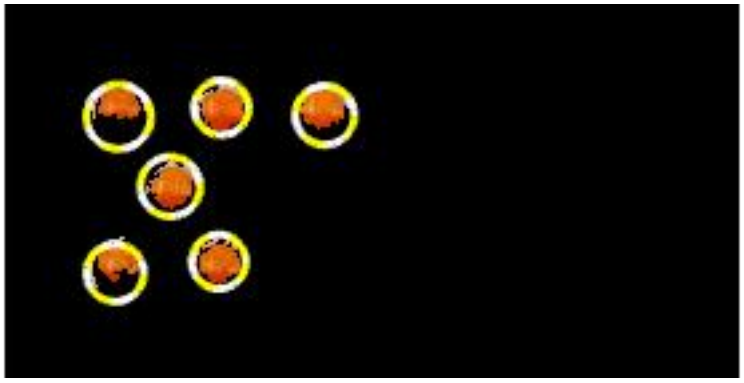

**Fig.: Tablets identified and circles drawn on them**

```
Number of Tablets left:
     6

Nearest Tablet is at:
   43.1636   68.6650
```

**Fig.: Output for the above blister pack**

**Matlab Program for Approach 1:**

```
close all;
clear all;

message = sprintf('Use stored image or take a fresh picture?');
reply = questdlg(message, 'Please choose one', 'Use Downloaded', 'Snap now',
'Snap now');
if strcmpi(reply, 'Use Downloaded')
    [I, colorMap] = imread('try1.jpeg');
else
% capturing image of blister from webcam
    cam = webcam;
    pause(2);
    I = snapshot(cam);
    clear cam;
end

% checking whether it is an 8-bit image
if (strcmpi(class(I), 'uint8'))
    bit8 = true;
end
% extracting color bands from the original image
redImg = I(:,:,1);
greenImg = I(:,:,2);
blueImg = I(:,:,3);

figure(1)
% displaying all three color components seperately
subplot(2,3,1)
imshow(redImg);
title('Red Component');
subplot(2,3,2)
imshow(greenImg);
title('Green Component');
subplot(2,3,3)
imshow(blueImg);
title('Blue Component');

%Setting a rough Threshold Value for grayscale image - Otsu's Method

redThreshL = graythresh(redImg);        %graythresh returns the value beyond
which red's pixel come in majority
redThreshH = 255;
greenThreshL = 0;
greenThreshH = graythresh(greenImg);
blueThreshL = 0;
blueThreshH = graythresh(blueImg);
if bit8
```

```
    redThreshL = uint8(redThreshL*255);
    greenThreshH = uint8(greenThreshH*255);
    blueThreshH = uint8(blueThreshH*255);
end
% Applying the above thresholds to the R, G, B Images
redNewImg = (redImg >= redThreshL) & (redImg <= redThreshH);
greenNewImg = (greenImg >= greenThreshL) & (greenImg <= greenThreshH);
blueNewImg = (blueImg >= blueThreshL) & (blueImg <= blueThreshH);

%Displaying the three new images - seperately as R, Y and B
subplot(2,3,4)
imshow(redNewImg);
title('Red - New');
subplot(2,3,5)
imshow(greenNewImg);
title('Green - New');
subplot(2,3,6)
imshow(blueNewImg);
title('Blue - New');

figure(2)
% Combining new images to form a red mask
redEnhancedMask = uint8(redNewImg & greenNewImg & blueNewImg); %since images
are arrays only - concatenating them

% Removing small holes in images
minUsefulArea = 75;
redEnhancedMask = uint8(bwareaopen(redEnhancedMask, minUsefulArea));
redEnhancedMask = uint8(imfill(redEnhancedMask, 'holes'));
imshow(redEnhancedMask, []);
title('Masked Image');

% Converting data type of redEnhancedMask from logical array to that of redImg
redEnhancedMask = cast(redEnhancedMask, class(redImg));

% Using this mask to bring forward red parts of the original Image
redAtTop = redEnhancedMask .* redImg;
greenAtTop = redEnhancedMask .* greenImg;    % Green Mask is actually decreasing
intensity of Green in image
blueAtTop = redEnhancedMask .* blueImg;      % -- same with Blue --

% Concatenating the three images to bring out the red in all three components -
R,G,B
enhancedRGB = cat(3, redAtTop, greenAtTop, blueAtTop);
figure(3)
imshow(enhancedRGB, []);
title('Final Enhanced Image');          %Final Processed Image
[centresTablet, radiiTablet] = imfindcircles(enhancedRGB,[10 18],
'sensitivity', 0.965);
viscircles(centresTablet, radiiTablet,'LineStyle','--', 'edgecolor', 'y');
```

```
numberOfTablet = length(radiiTablet);
disp('Number of Tablets left: ');
disp(numberOfTablet);

min = 1000;
min_coordinate = [0 0];
for i = 1:numberOfTablet
    d = sqrt(centresTablet(i, 1)^2 + centresTablet(i, 2)^2);
%calculating distance of each pill from top left of the screen
    if (d<min)
        min = d;                                                    %finding
nearest pill
        min_coordinate = [centresTablet(i,1) centresTablet(i,2)];   %storing
coordinates of nearest pill, i.e, the next target pill
    end
end
disp ('Nearest Tablet is at:'), disp(min_coordinate);
```

**Approach 2:**

1. Initial steps of choosing the image is same as that of the first approach.
2. Necessary pre-processing of image is done such as:
   a. Conversion to grayscale
   b. Threshold adjustment
   c. Binarizing the image
   d. Applying averaging filter to the image
   e. Negating (or complimenting) the image
   f. Removing the negated image from the filtered image of (d) so as to remove background effects
3. Next, we try to locate the circles in the image using imfindcircles( ). The parameters of this function play a key role in correctly detecting the circles in the image. Sensitivity adjustment must be done for different design of blister so that they can be identified correctly.
4. To visualize the identified circles, we use viscircles( ) which draws the identified circles on the image itself.
5. Similar to the first approach, coordinates of the tablets are returned using which Euclidean distance of each of them can be calculated from the top left corner of the screen thereby helping us to identify the next target tablet.
6. The coordinates of each tablet can then be mapped to the actual blister pack so that relevant command can be given to stepper motors through Arduino which can pop out the tablets from the Blister Pack.

**Fig.: Original Image**

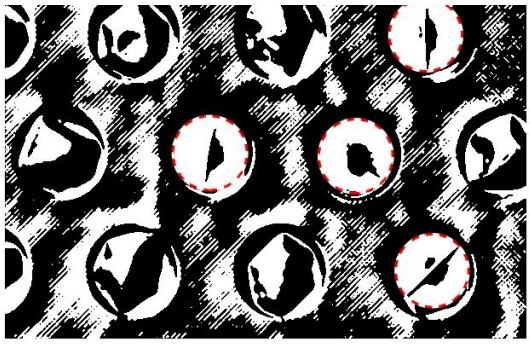

**Fig.: 4 tablets identified by red dotted circles**

```
The number of tablets left in the blister is:
     4

Nearest Tablet is at:
   63.3899  118.6216
```

**Fig.: Output for the above Blister Pack**

**Matlab Program for Approach 1:**

```
clear all;
close all;

message = sprintf('Use stored image or take a fresh picture?');
reply = questdlg(message, 'Please choose one', 'Use Downloaded', 'Snap now',
'Snap now');
if strcmpi(reply, 'Use Downloaded')
    I = imread('pills_used_2.jpeg');
else

% capturing image of blister from webcam
    cam = webcam;
    pause(2);
```

```
    I = snapshot(cam);
    clear cam;
end

I = imread("pills_used_2.jpeg");
gs = im2gray(I);
gsAdj = imadjust(gs);

BW = imbinarize(gsAdj, "adaptive");

H = fspecial("average");                % creating an averaging filter
BW1 = imfilter(BW, H);                  % filtering the B&W image with
averaging filter
BW2 = ~imfilter(BW1, H, "replicate");   % negating and again filtering the
image with the same filter while eliminating edge lines
BW3 = BW1 - BW2;                        % removing negated components from
filtered image
imshow(BW3);

[centresTablet, radiiTablet] = imfindcircles(BW3,[30 50],'ObjectPolarity',
'bright', 'sensitivity', 0.95);
viscircles(centresTablet, radiiTablet,'LineStyle','--', 'edgecolor', 'r');
n = length(centresTablet);
disp('The number of tablets left in the blister is:')
disp(n)
min = 1000;
min_coordinate = [0 0];
for i = 1:n
    d = sqrt(centresTablet(i, 1)^2 + centresTablet(i, 2)^2);
%calculating distance of each pill from top left of the screen
    if (d<min)
        min = d;                    %finding nearest pill
%storing coordinates of nearest pill, i.e, the next target pill
        min_coordinate = [centresTablet(i,1) centresTablet(i,2)];
     end
end
disp ('Nearest Tablet is at:'), disp(min_coordinate);
```