HW 1 - Abhijay (11632196)

```
A = reshape( 1:25, 5, 5)'
```

```
A = 5×5
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
    16    17    18    19    20
    21    22    23    24    25
```

```
% 1a.
A = reshape( 1:25, 5, 5)';
A(end,:)=[] % Del last row
```

```
A = 4×5
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
    16    17    18    19    20
```

```
A(:,end)=[] % Del last col
```

```
A = 4×4
     1     2     3     4
     6     7     8     9
    11    12    13    14
    16    17    18    19
```

```
% 1b.
A = reshape( 1:25, 5, 5)';
A = A(1:end-1,1:end-1) % Extract first 3x3 matrix
```

```
A = 4×4
     1     2     3     4
     6     7     8     9
    11    12    13    14
    16    17    18    19
```

```
% 1c.
A = reshape( 1:25, 5, 5)';
A(4,:) = zeros(5,1) % Replace fourth row by zeros
```

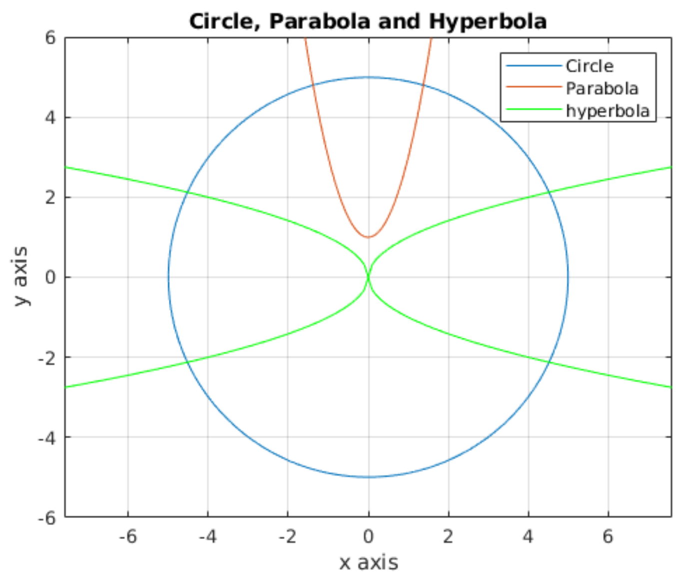
```
A = 5×5
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
     0     0     0     0     0
    21    22    23    24    25
```

```
% 1d.
A = reshape( 1:25, 5, 5)';
A(2,2:3) = ones(2,1)' % Replace the second and third values of the second row by ones
```

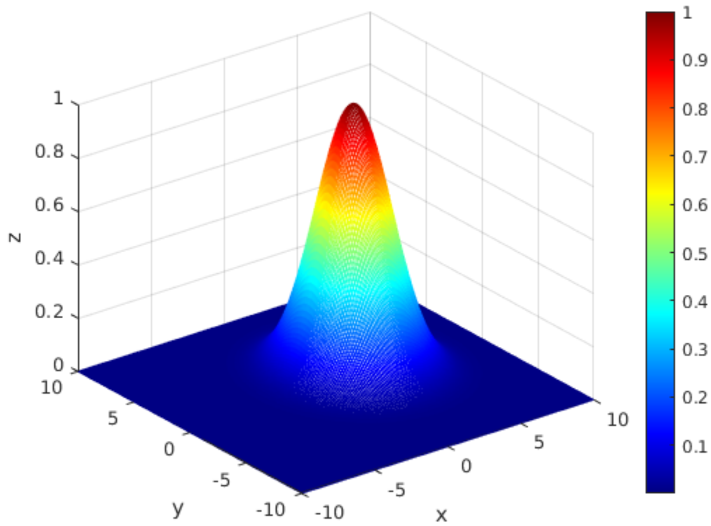
```
A = 5×5
     1     2     3     4     5
     6     1     1     9    10
    11    12    13    14    15
    16    17    18    19    20
    21    22    23    24    25
```

```
% 2

% Circle
rad=5;
t=0:0.001:2*pi;
x=rad*cos(t);
y=rad*sin(t);
plot(x,y);
%   axis square;
xlim([-6 6]);
ylim([-6 6]);
grid on;
axis equal;
title('Circle, Parabola and Hyperbola');
```

```
xlabel('x axis');
ylabel('y axis');
hold on;

% Parabola

% a < 0: The cup faces down.
% a > 0: The cup faces up.
% a = 0: It’s not a parabola but a straight line
a = 2;

% b = 0: The vertex of the parabola is on the y-axis in the point (0,c)
% b ≠ 0: The vertex of the parabola is not on the y-axis.
b = 0;

% c is the point of intersection between the parabola and the y-axis
%  (because y=c when x=0 in the quadratic function).
c = 1;

x = -5:0.01:5;
y = a*(x.^2) + b*x + c;
plot(x,y);
legend('parabola');

% hyperbola
x = 0:0.1:10;
plot(x, sqrt(x), 'g', x, -sqrt(x), 'g', -x, sqrt(x), 'g', -x, -sqrt(x), 'g');
legend('Circle', 'Parabola','hyperbola');
hold off;
```




```
% 3.
% Plot a 3D surface of a 2D Gaussian function
[X,Y] = meshgrid(-10:0.1:10, -10:0.1:10);
mu_x = 2;
mu_y = 1;
sig_x = 4;
sig_y = 5;
A = 1;
twoDgauss = A*exp( -( ((X - mu_x).^2 / (2*sig_x)) + ((Y - mu_y).^2 / (2*sig_y)) ) );
mesh(X, Y, twoDgauss);
xlabel('x');
ylabel('y');
zlabel('z');
colormap(jet(256));
colorbar;
```




```
% 4
[img,map] = imread('Apples.JPG');
imgCell = { class(img), size(img,1), size(img,2), img(:,:,1), img(:,:,2), img(:,:,3)}
```

```
imgCell = 1×6 cell array
    {'uint8'}    {[525]}    {[700]}    {525×700 uint8}    {525×700 uint8}    {525×700 uint8}
```

```
% 5
imgStruct.dataType = class(img);
imgStruct.ncols = size(img,2);
imgStruct.nrows = size(img,1);
imgStruct.red = img(:,:,1);
imgStruct.green = img(:,:,2);
imgStruct.blue = img(:,:,3)
```

```
imgStruct = struct with fields:
    dataType: 'uint8'
       ncols: 700
       nrows: 525
         red: [525×700 uint8]
       green: [525×700 uint8]
        blue: [525×700 uint8]
```

```
% 6
imgStruct_ = imgData(img) % See function in imgData.m file
```

```
imgStruct_ = struct with fields:
    dataType: 'uint8'
       ncols: 700
       nrows: 525
         red: [525×700 uint8]
       green: [525×700 uint8]
        blue: [525×700 uint8]
```

Part 2: Binary Image Processing and Morphological Operators

In [1]:

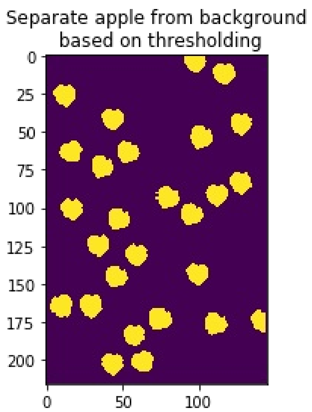
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import NoNorm
from skimage import measure

def process_and_count_apples(img):

    # get hsv values for thresholding; with rgb it was not coming to be be less accurate
    img_hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Threshold based on both light and dark red color ranges

    # lower mask (0-10)
    lower_red = np.array([0,50,50])
    upper_red = np.array([10,255,255])
    mask0 = cv2.inRange(img_hsv, lower_red, upper_red)

    # upper mask (170-180)
    lower_red = np.array([170,50,50])
    upper_red = np.array([180,255,255])
    mask1 = cv2.inRange(img_hsv, lower_red, upper_red)

    # join my masks
    mask = mask0+mask1

    # set my output img to zero everywhere except my mask
    output_img = img.copy()
    output_img[np.where(mask==0)] = 0
    gray = cv2.cvtColor( output_img, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Separate apple from background\n based on thresholding")
    plt.show()

    # Select a kernel for applying morphological operation
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(4,5))

    # Applying Erosion to find round (based on kernel) objects or apples
    erosion = cv2.erode( output_img, kernel, iterations = 1)
    gray = cv2.cvtColor( erosion, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Erosion")
    plt.show()

    # Applying Opening to remove noise
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(2,4))
    opening = cv2.morphologyEx(erosion, cv2.MORPH_OPEN, kernel)
    gray = cv2.cvtColor( opening, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Opening")
    plt.show()

    # Applying Closing to close apples which appear slashed
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(12,12))
    closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
    gray = cv2.cvtColor( closing, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Closing")
    plt.show()

    # Applying Erosion
    kernel = np.ones((4,4),np.uint8)
    erosion = cv2.morphologyEx(closing, cv2.MORPH_ERODE, kernel)
    gray = cv2.cvtColor( erosion, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Erosion")
    plt.show()
```

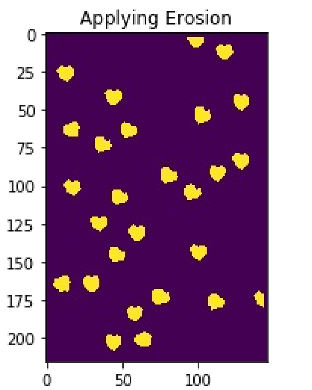
```
    # Applying Opening to remove noise
    kernel = np.ones((3,3),np.uint8)
    opening = cv2.morphologyEx(erosion, cv2.MORPH_OPEN, kernel)
    gray = cv2.cvtColor( opening, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Opening")
    plt.show()

    # Applying Dilation to increase to original apple size
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,8))
    dilation = cv2.dilate(opening,kernel,iterations = 1)
    gray = cv2.cvtColor( dilation, cv2.COLOR_BGR2GRAY)
    gray[gray>0] = 255
    imgplot = plt.imshow( gray)
    plt.title("Applying Dilation")
    plt.show()

    # Now let's count the apples
    f = plt.figure(figsize=(15,13))
    ax= f.subplots(1,2)
    img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    ax[0].imshow(img)
    ax[1].imshow(gray)
    plt.title("Extracted apples")
    plt.show()

    labels = measure.label(gray)
    return labels.max()
```

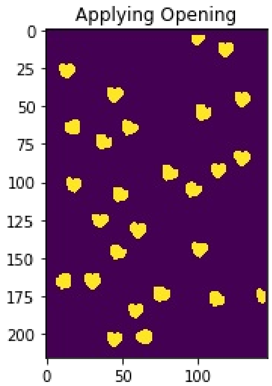
a. FakeApples.bmp

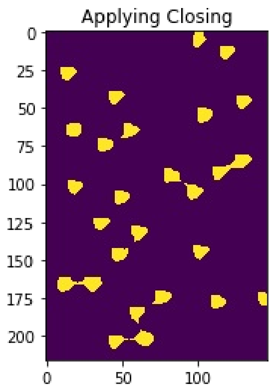
In [2]:

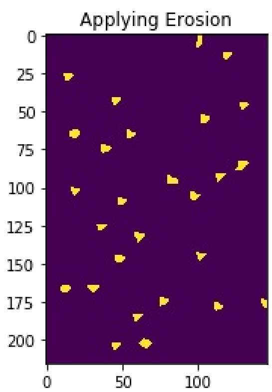
```
img=cv2.imread("FakeApples.bmp")
num_apples = process_and_count_apples(img)
```

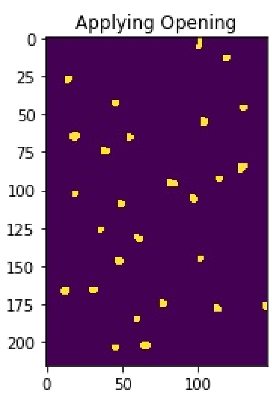


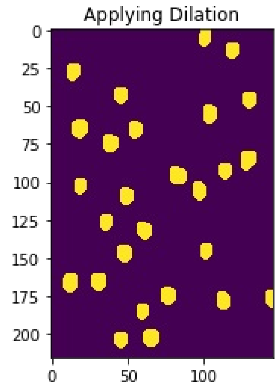


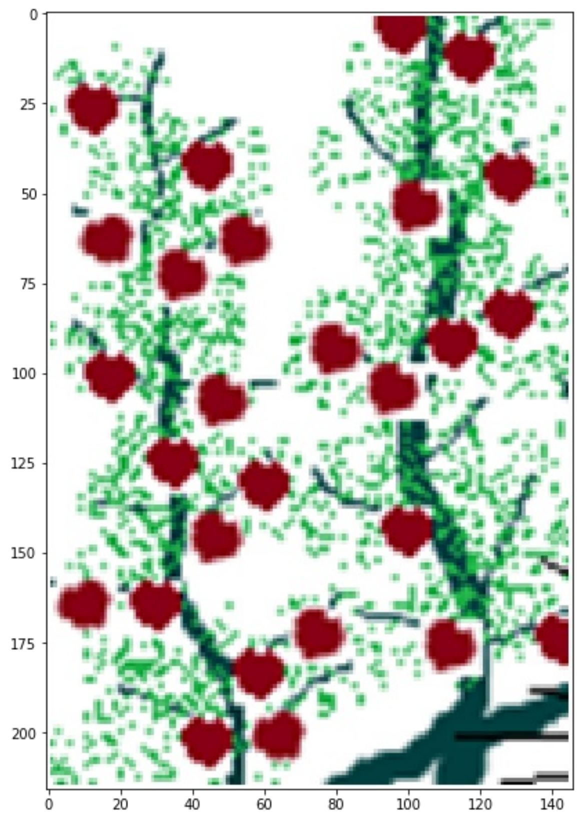

Applying Opening
0
25
50
75
100
125
150
175
200
0
50
100

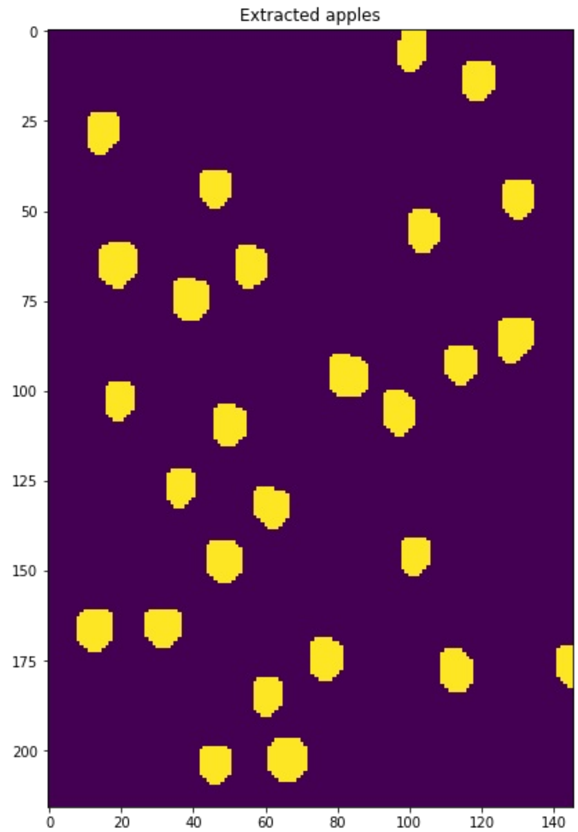

Applying Closing
0
25
50
75
100
125
150
175
200
0
50
100


Applying Erosion
0
25
50
75
100
125
150
175
200
0
50
100


Applying Opening
0
25
50
75
100
125
150
175
200
0
50
100







Number of apples detected

In [3]:

```
num_apples
```

Out[3]:

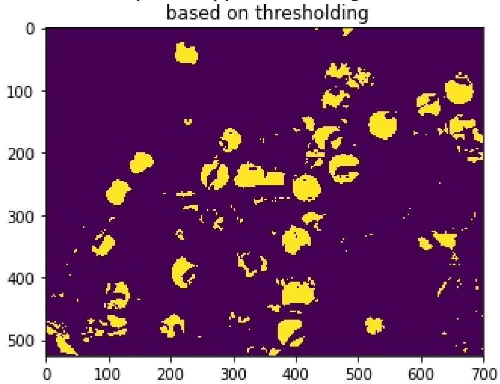
27

b. Apples.JPG

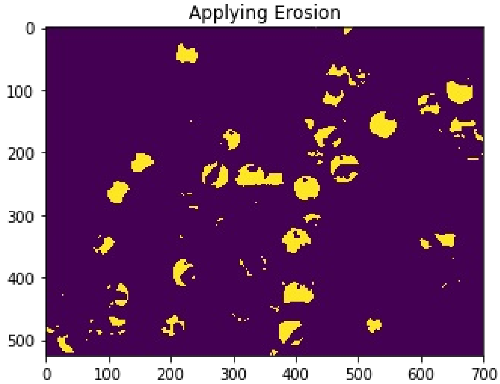
In [4]:

```
img=cv2.imread("Apples.JPG")
num_apples=process_and_count_apples(img)
```

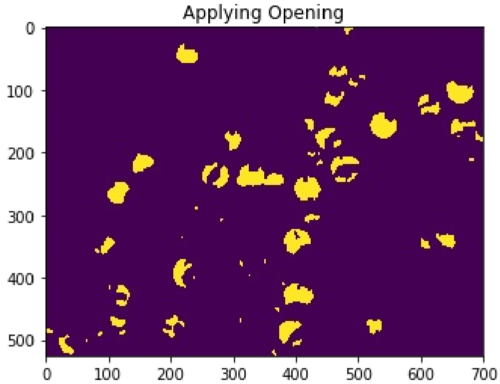
Separate apple from background
based on thresholding



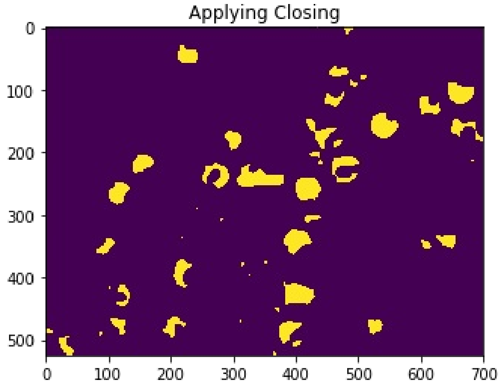
Applying Erosion

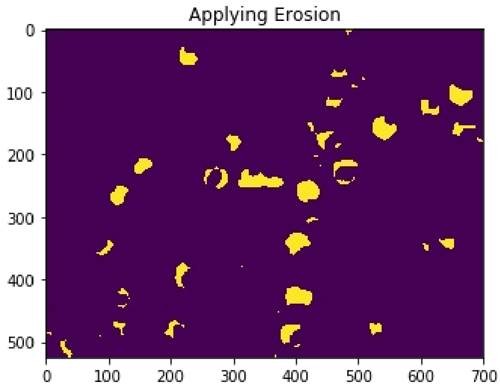


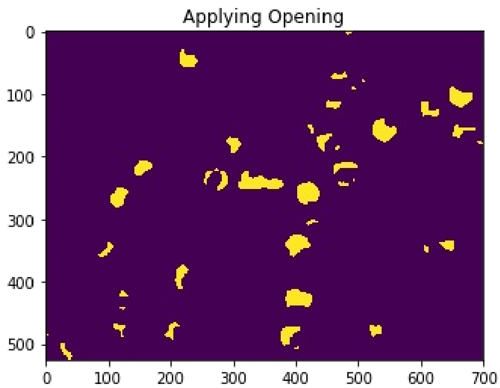
Applying Opening

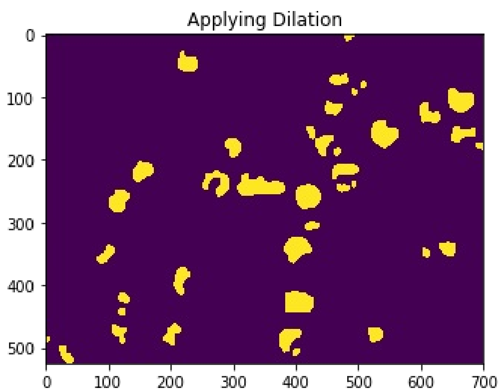


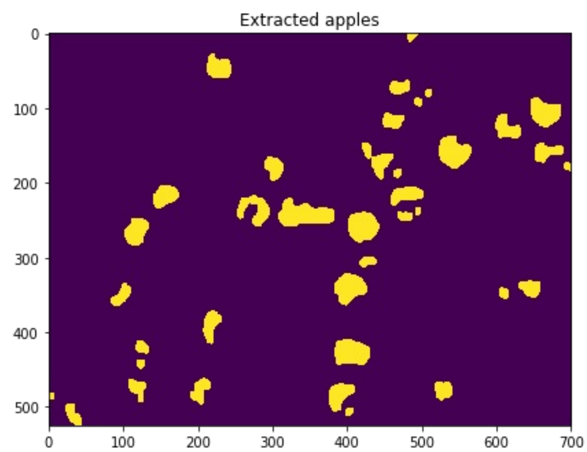
Applying Closing











Number of apples detected

In [5]:

```
num_apples
```

Out[5]:

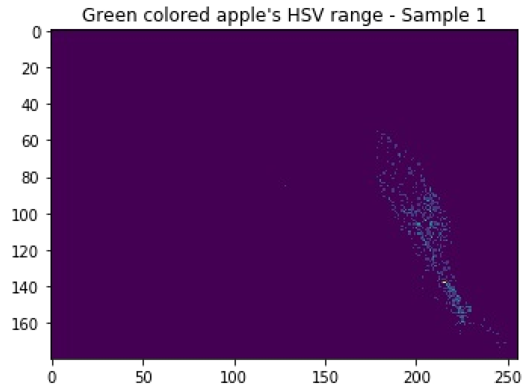
38

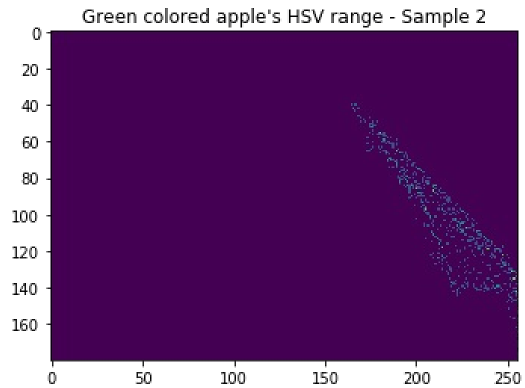
c. AuroraWall.JPG

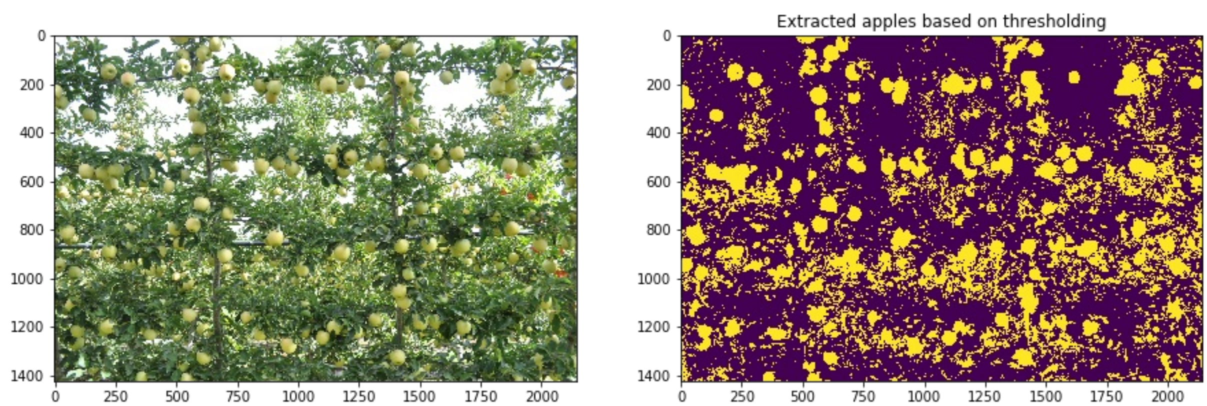
In [6]:

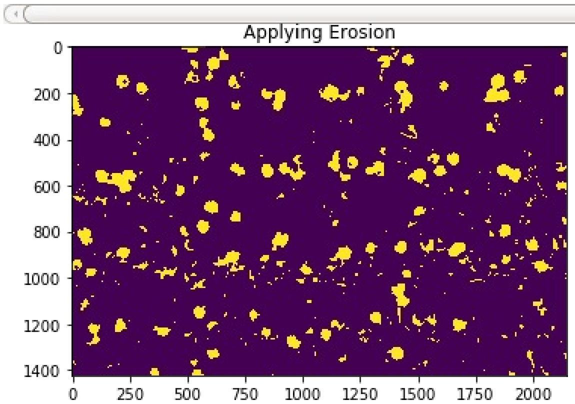
```
# Histogram of green colored apple's HSV
img=cv2.imread("1.png")
img_hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hist = cv2.calcHist( [img], [0, 1], None, [180, 256], [0, 180, 0, 256])
plt.imshow(hist,interpolation = 'nearest')
plt.title('Green colored apple\'s HSV range - Sample 1')
plt.show()

img=cv2.imread("2.png")
img_hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hist = cv2.calcHist( [img], [0, 1], None, [180, 256], [0, 180, 0, 256])
plt.imshow(hist,interpolation = 'nearest')
plt.title('Green colored apple\'s HSV range - Sample 2')
plt.show()

img=cv2.imread("AuroraWall.JPG")

# get hsv values for thresholding; with rgb it was not coming to be be less accurate
img_hsv=cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Threshold based on both light green color ranges

lower_green = np.array([10,20,60])
upper_green = np.array([41,255,255])
mask = cv2.inRange(img_hsv, lower_green, upper_green)

# set my output img to zero everywhere except my mask
output_img = img.copy()
output_img[np.where(mask==0)] = 0

gray = cv2.cvtColor( output_img, cv2.COLOR_BGR2GRAY)
gray[gray>0] = 255
img_=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
f = plt.figure(figsize=(15,12))
ax= f.subplots(1,2)

ax[0].imshow(img_)
ax[1].imshow(gray)
plt.title("Extracted apples based on thresholding")
plt.show()

# Select a kernel for applying morphological operation
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(11,12))

# Applying Erosion to find round (based on kernel) objects or apples
erosion = cv2.erode( output_img, kernel, iterations = 1)
gray = cv2.cvtColor( erosion, cv2.COLOR_BGR2GRAY)
gray[gray>0] = 255
imgplot = plt.imshow( gray)
plt.title("Applying Erosion")
plt.show()

# Applying Opening to remove noise
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(11,12))
opening = cv2.morphologyEx(erosion, cv2.MORPH_OPEN, kernel)
gray = cv2.cvtColor( opening, cv2.COLOR_BGR2GRAY)
gray[gray>0] = 255
imgplot = plt.imshow( gray)
plt.title("Applying Opening")
plt.show()

# Applying Dilation to increase to original apple size
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(8,9))
dilation = cv2.dilate(opening,kernel,iterations = 1)
gray = cv2.cvtColor( dilation, cv2.COLOR_BGR2GRAY)
gray[gray>0] = 255
imgplot = plt.imshow( gray)
plt.title("Applying Dilation")
plt.show()

# Now let's count the apples
f = plt.figure(figsize=(15,13))
ax= f.subplots(1,2)
img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
ax[0].imshow(img)
```

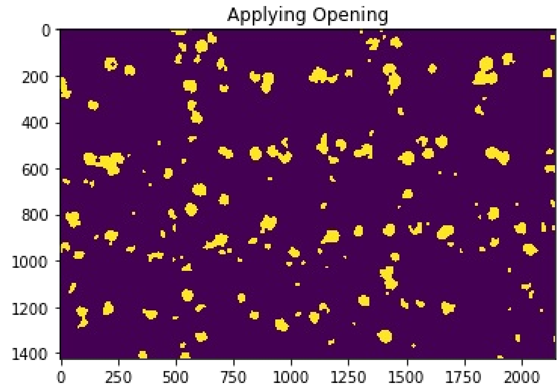
```
ax[0].imshow(img)
ax[1].imshow(gray)
plt.title("Extracted apples")
plt.show()

labels = measure.label(gray)
num_apples=labels.max()
```

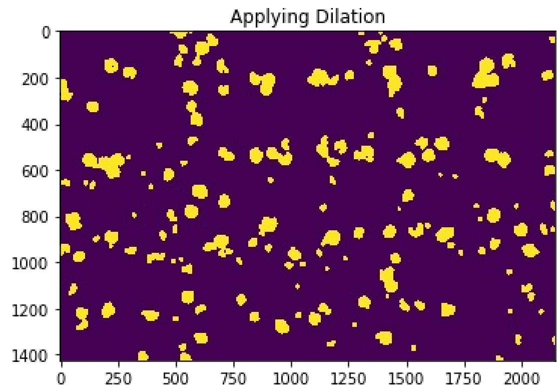


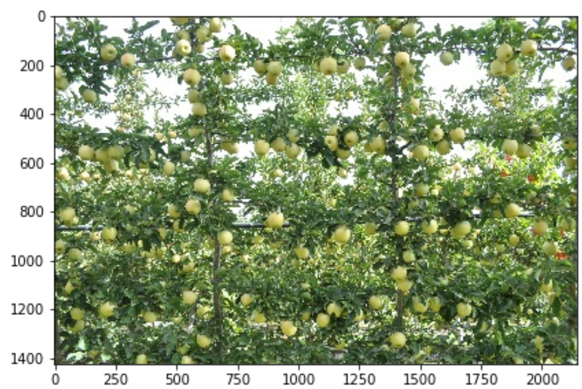


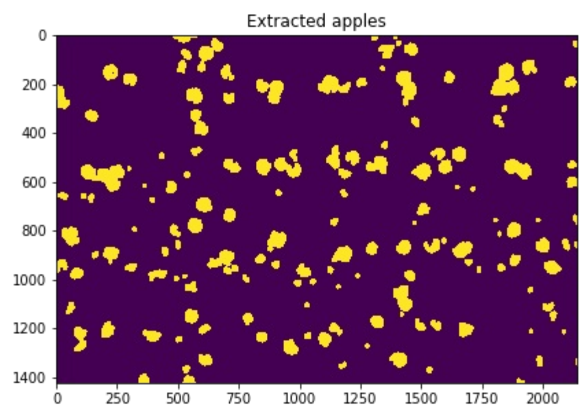












Number of apples detected

In [7]:

```
num_apples
```

Out[7]:

148