Computer Vision

PROBLEM SET 1

Image Cartoonifier

March 14, 2019

abstract

Applying Image Processing Filters For Image Cartoonifying

Team

Dahlia Chehata 27

Fares Mehanna 52

TABLE OF CONTENTS

Overview

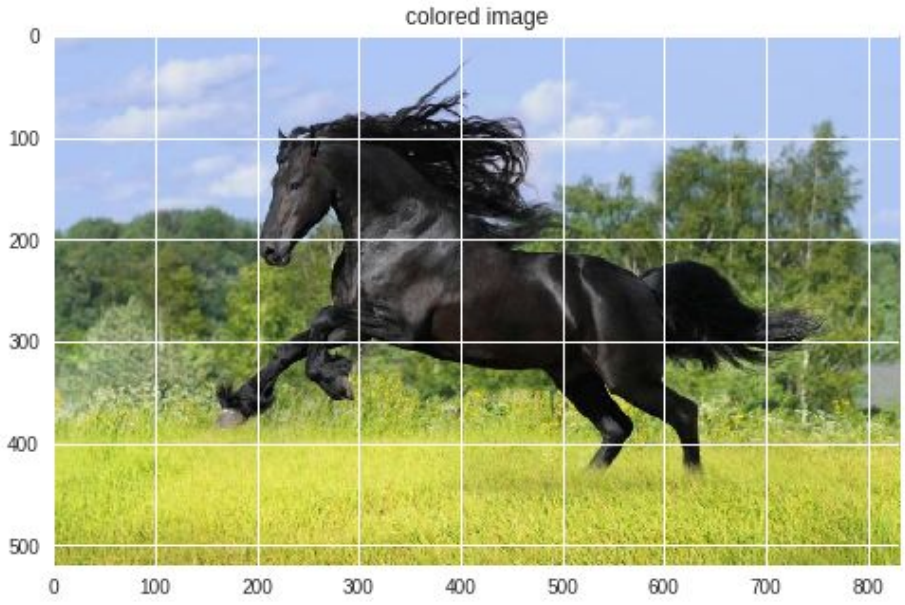
The problem set covers

- Applying image processing filters for image smoothing and edge detection.
- Converting a real-life image to a sketch drawing.
- Converting a real-life image to a painting and overlaying the sketch to produce a cartoon.

Generating a black-and-white sketch/ colored painting

1. **for sketch (black and white drawing) :** edge detection filter such as Sobel, Scharr, Laplacian filters, or Canny-edge detector . We use Laplacian filter
2. **for colored painting :** edge-preserving filter (bilateral filter) to further smooth the flat regions while keeping the edges intact
3. **cartoonized image :** By overlaying the sketch drawing on top of the color painting, we obtain a cartoon effect

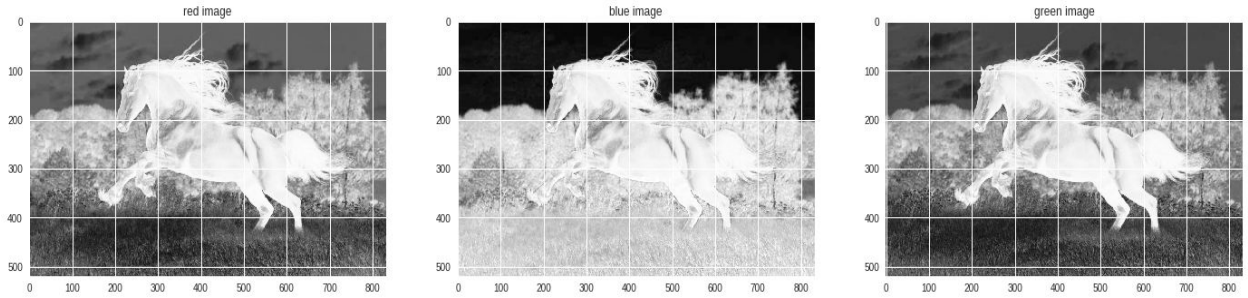
```
To generate the colored version and the grey scaled version of the
image, We use:

RGB_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
gray_image = cv2.cvtColor(RGB_image, cv2.COLOR_BGR2GRAY)
```


colored image
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800


Grey scaled
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

RGB components

```
# get RGB components

comp= [None]*3
comp[1],comp[2],comp[0] = cv2.split(image)       # get b,g,r
titles = ['red image','blue image','green image']
plt.rcParams['figure.figsize'] = (24,16)
gs = gridspec.GridSpec(1, 3)
plt.figure()
for i in range (3):
  ax = plt.subplot(gs[0, i])
  plt.title (titles[i])
  plt.imshow(comp[i])
```



Noise Reduction Using Median Filter

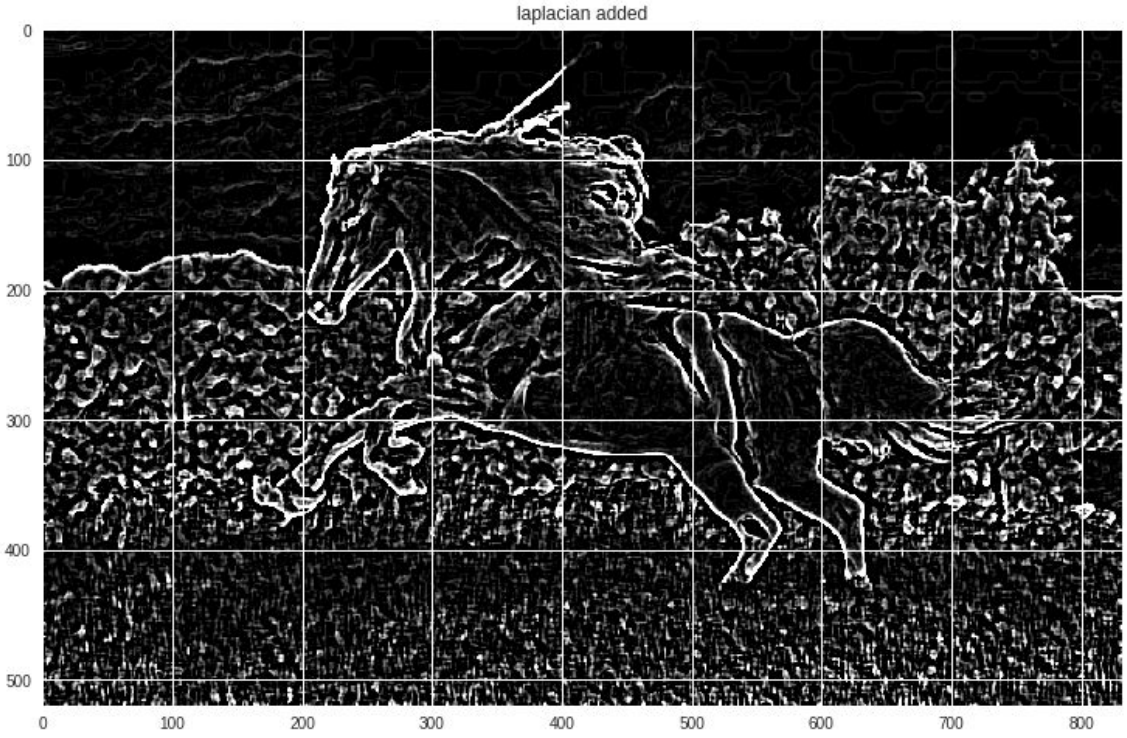
Before applying Laplacian filter for edge detection, we smooth first with median filter

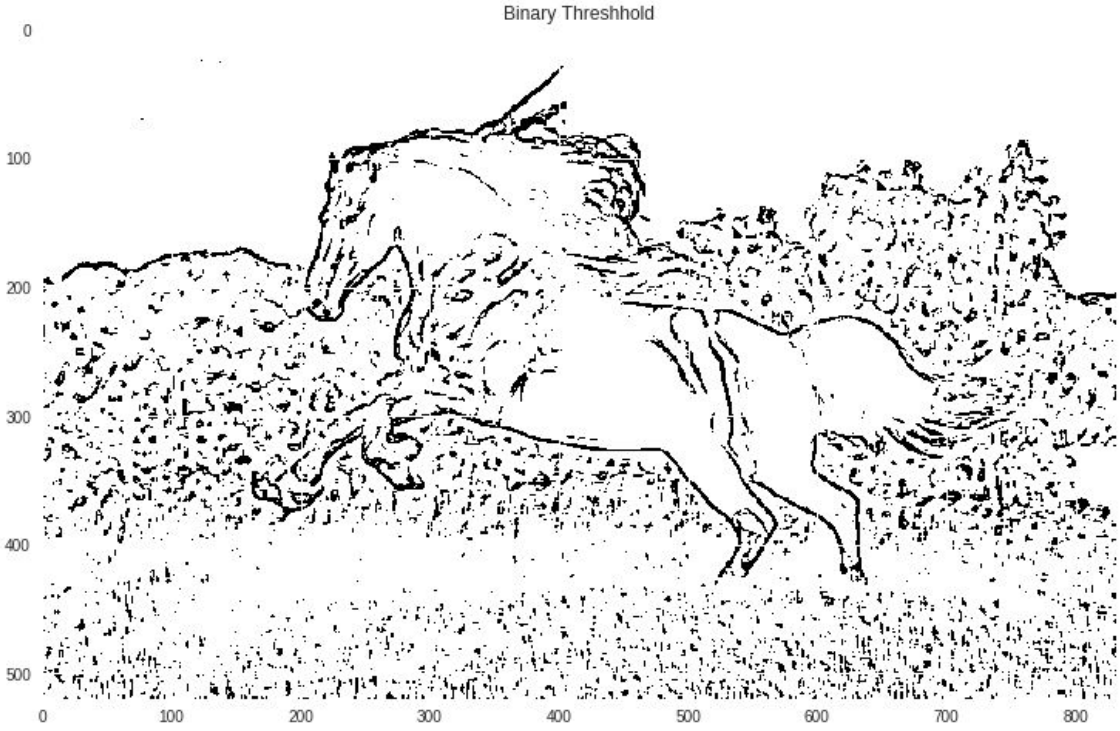
```
median = cv2.medianBlur(gray_image,7)
plt.figure()
plt.title('smoothed with median filter')
plt.imshow(median , cmap='gray')
```

Edge Detection Using Laplacian Filter

We apply Laplacian filter with aperture size = 5 for edge detection, then we apply binary threshold to make the edges either white or black with threshold value = 125

```
laplacian = cv2.Laplacian(median,ddepth=-1, ksize = 5)
ret,thresh1 = cv2.threshold(laplacian,125,255,cv2.THRESH_BINARY)
plt.figure()
plt.title('laplacian added')
plt.imshow(laplacian, cmap='gray')
plt.figure()
plt.title('Binary Threshhold')
plt.imshow(thresh1 )
plt.show()
```


laplacian added
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

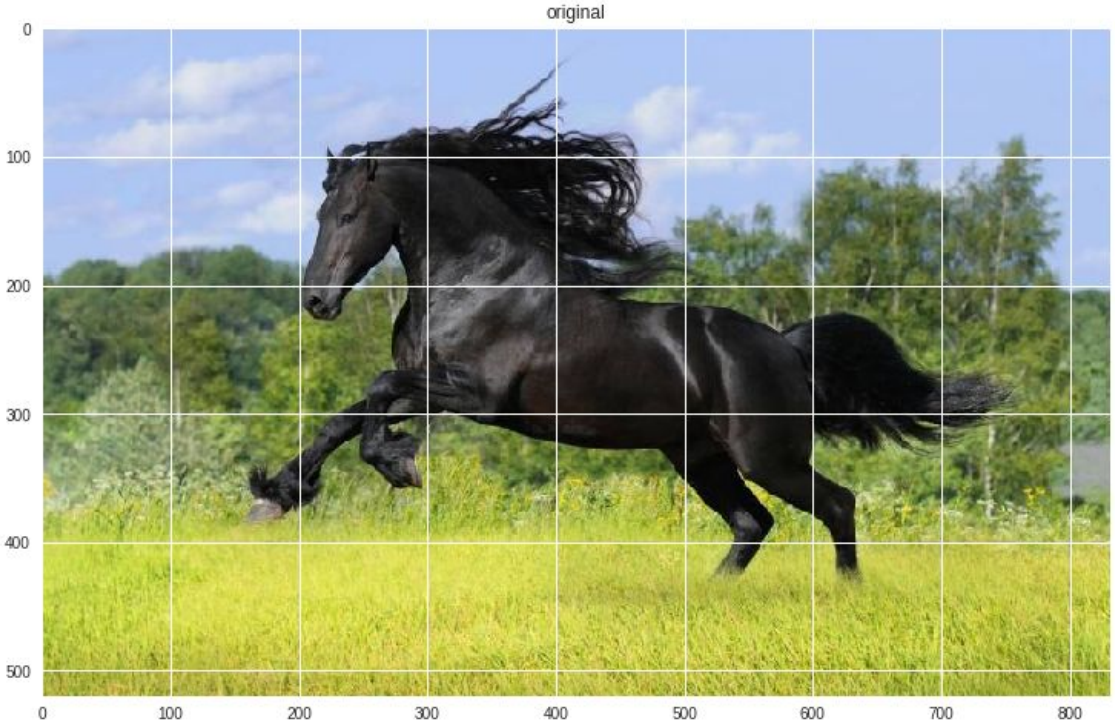

Binary Threshhold
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

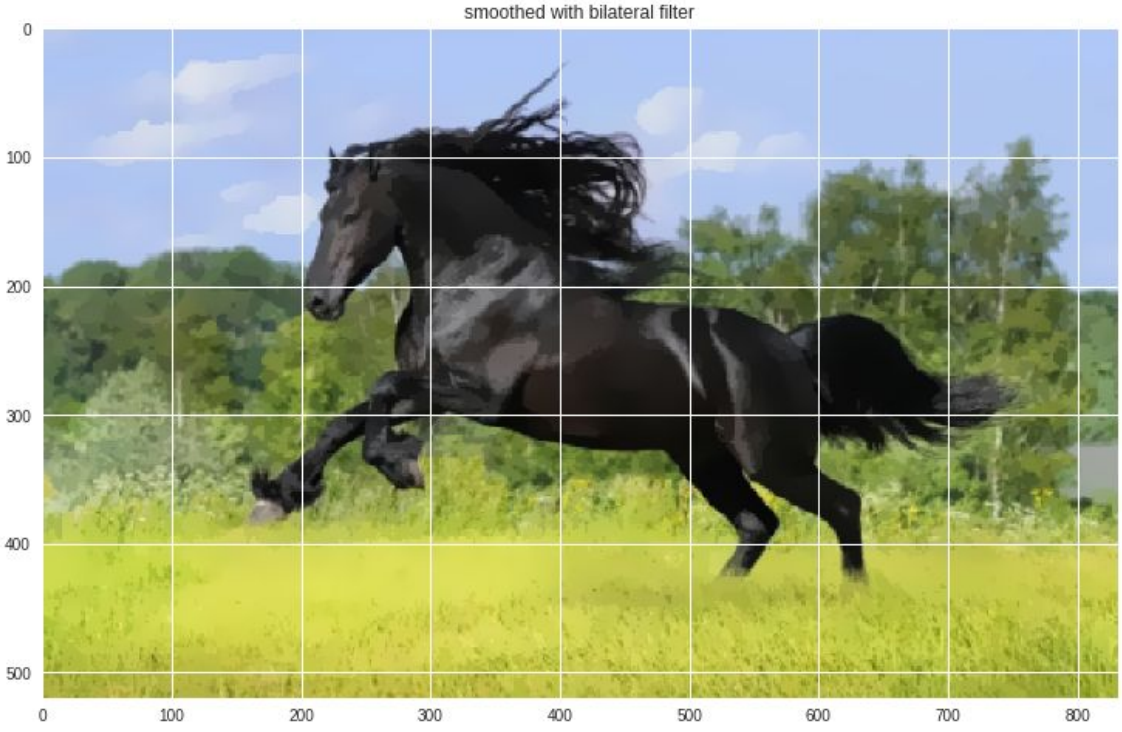
Generating a color painting and a cartoon

Since bilateral filter is extremely slow , we resize the image first to 60% of its original size

Resizing

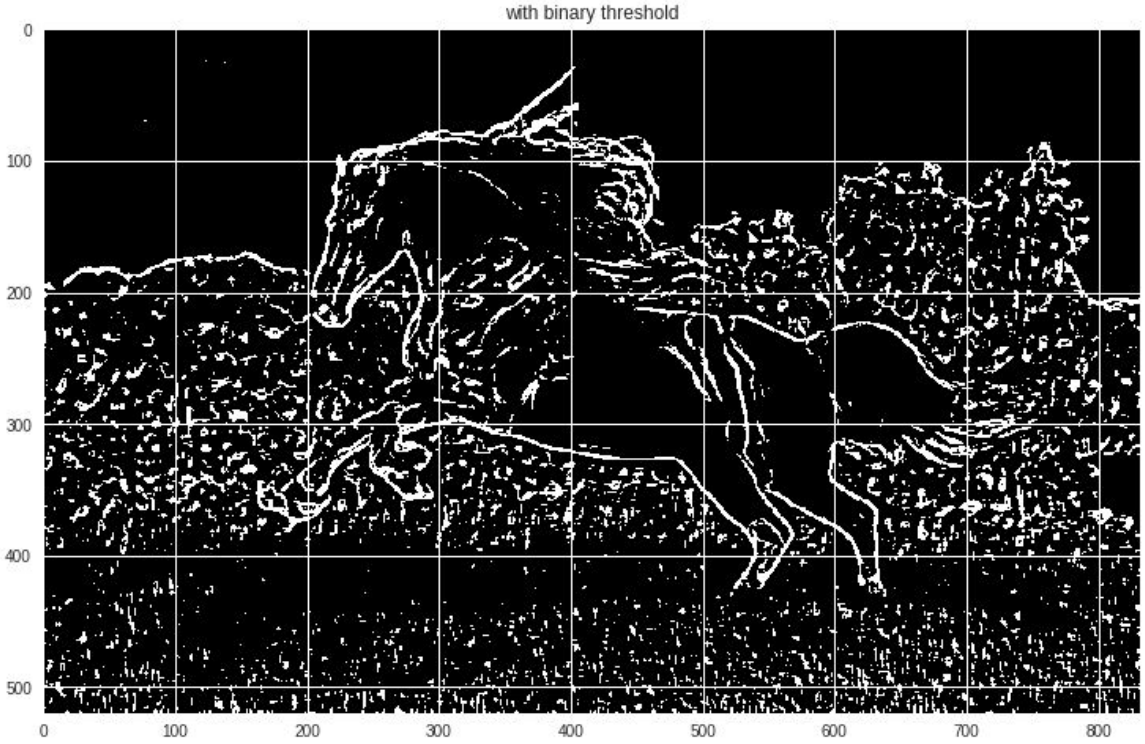
```
numBilateralFilters = 7

# 60 % dimensions
scale_percent = 60 # percent of original size
width = int(RGB_image.shape[1] * scale_percent / 100)
height = int(RGB_image.shape[0] * scale_percent / 100)
dim = (width, height)
resized = cv2.resize(RGB_image, dim, interpolation =
cv2.INTER_AREA)
colored_img = resized


# repeatedly apply small bilateral filter instead of applying one
large filter

for _ in range(numBilateralFilters):
    colored_img = cv2.bilateralFilter(img_color,
d=9,sigmaColor=9,sigmaSpace=7)

# resize again to original size
dim = (RGB_image.shape[1], RGB_image.shape[0])
resized = cv2.resize(colored_img, dim, interpolation =
cv2.INTER_AREA)

plt.title('original')
plt.imshow(RGB_image)
plt.figure()

plt.title('smoothed with bilateral filter')
plt.imshow(resized)
plt.show()
```

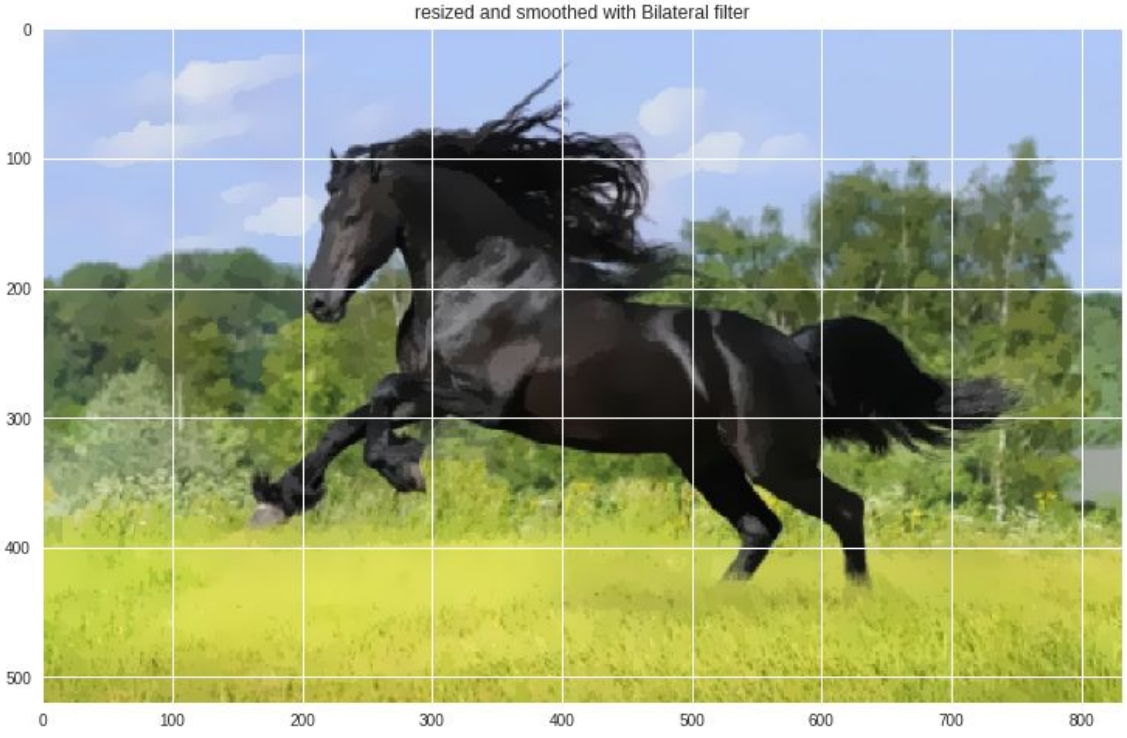

original
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800


smoothed with bilateral filter
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

Cartoonization

```
# convert the grayscaled image smoothed with laplacian after applying
binary threshold to RGB format
img_edge = cv2.cvtColor(thresh1,cv2.COLOR_GRAY2RGB)
cartoonized= cv2.subtract(resized,img_edge)

print(thresh1.shape)
plt.figure()
plt.title ('with binary threshold')
plt.imshow(thresh1, cmap='gray' )

print(resized.shape)
plt.figure()
plt.title ('resized and smoothed with Bilateral filter')
plt.imshow(resized, cmap='gray' )

plt.figure()
plt.title('cartoonized image')
plt.imshow(cartoonized, cmap='gray' )
cv2.imwrite('/content/drive/My
Drive/Photos/cartoonized.png',cv2.cvtColor(cartoonized,cv2.COLOR_RGB2BGR))
```

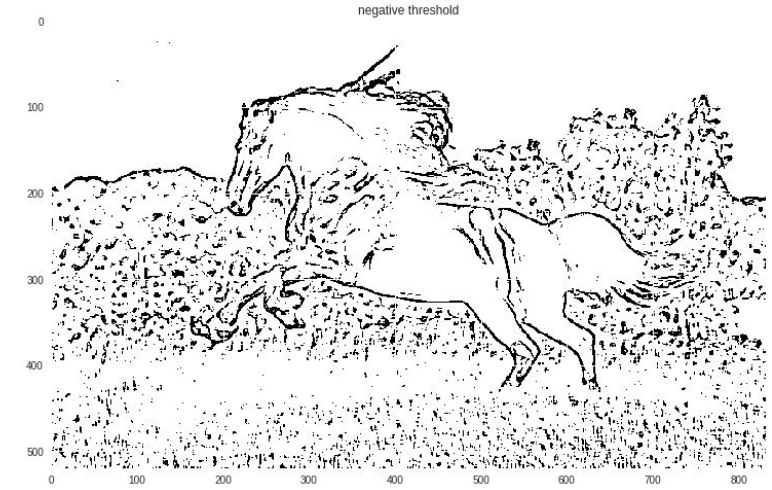

with binary threshold
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800


resized and smoothed with Bilateral filter
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

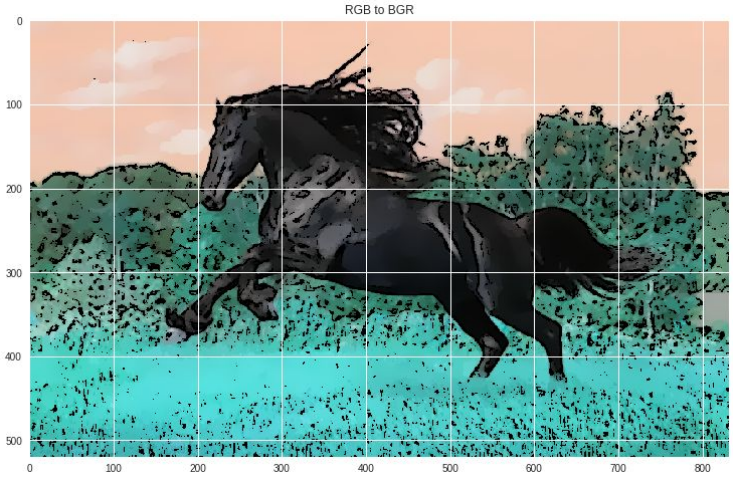
Applying the cartoonized effect on negative thresholding

```
thresh2 = cv2.bitwise_not(thresh1)
img_edge = cv2.cvtColor(thresh2,cv2.COLOR_GRAY2RGB)
plt.figure()
plt.title ('negative threshold')
plt.imshow(img_edge)
plt.show()

cartoon = cv2.bitwise_and(resized, img_edge)
plt.figure()
plt.title('cartoonized image')
plt.imshow(cartoon)
plt.show()

cartoon_converted = cv2.cvtColor(img_cartoon,cv2.COLOR_RGB2BGR) # RGB to BGR
plt.figure()
plt.title ('RGB to BGR')
plt.imshow(cartoon_converted)
```

```
plt.show()
```


negative threshold
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800


cartoonized image
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800


RGB to BGR
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800