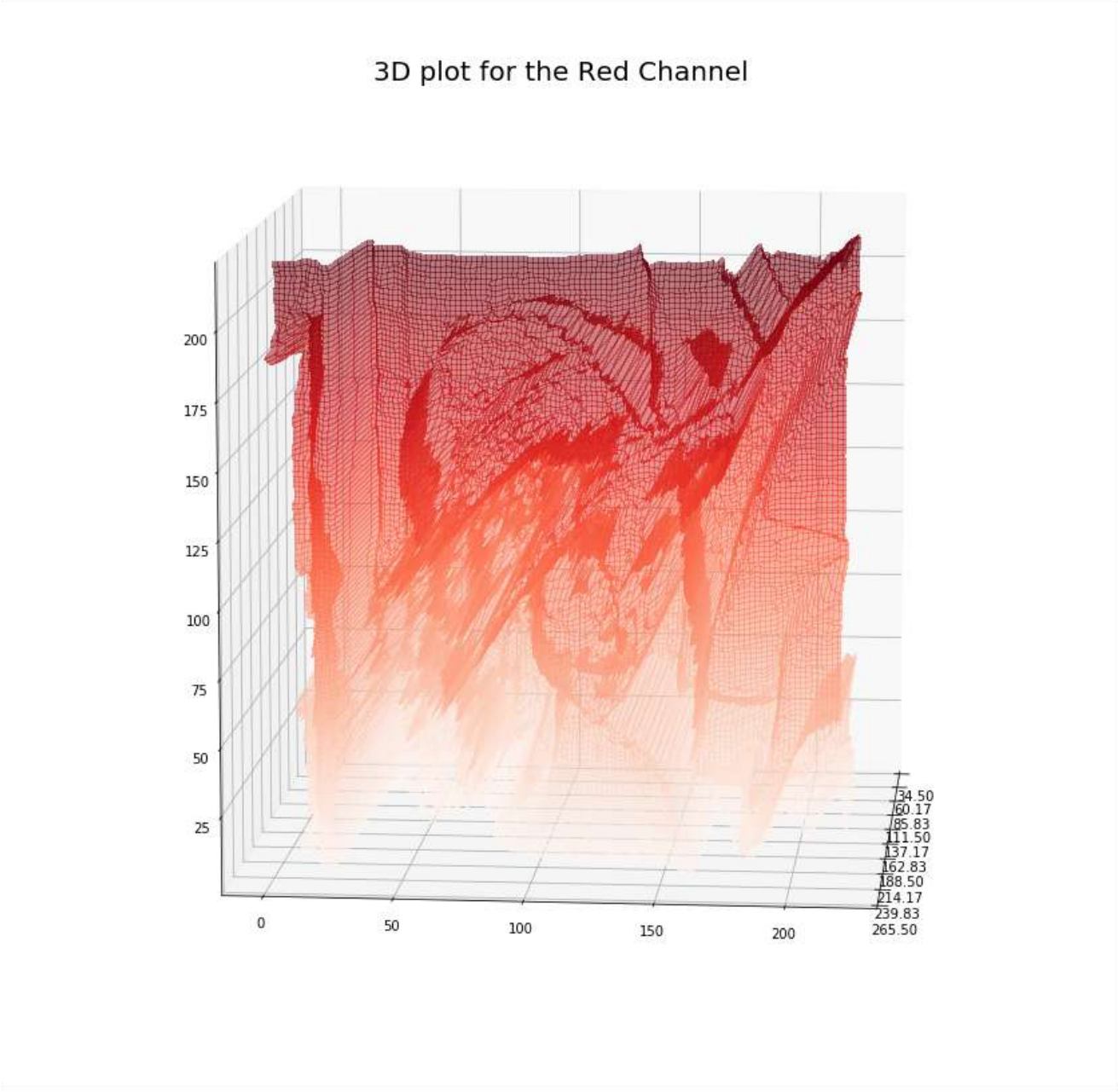
Chapter 1: Basic Image and Video Processing

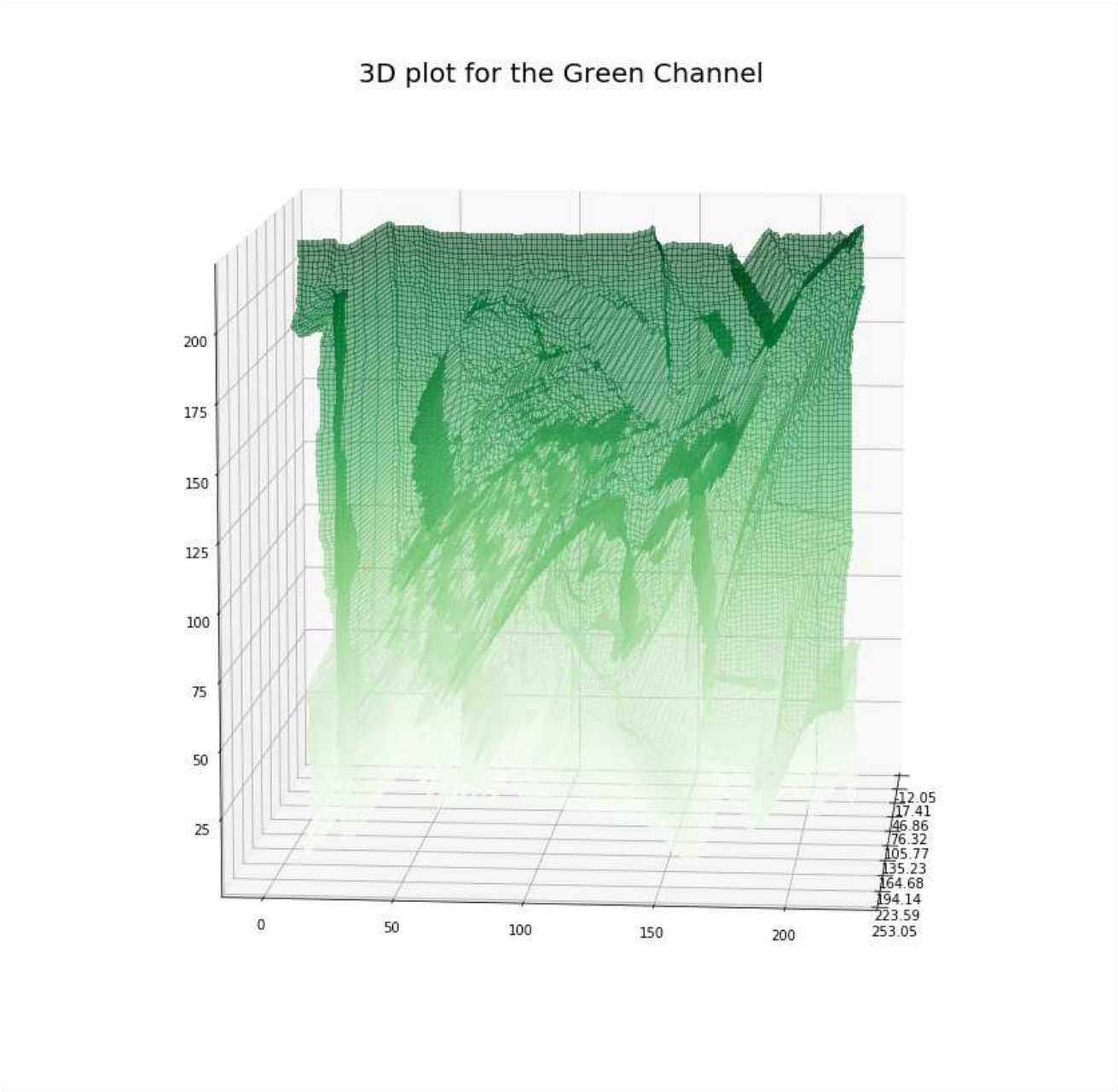
Problems

1. Display RGB image color channels in 3D

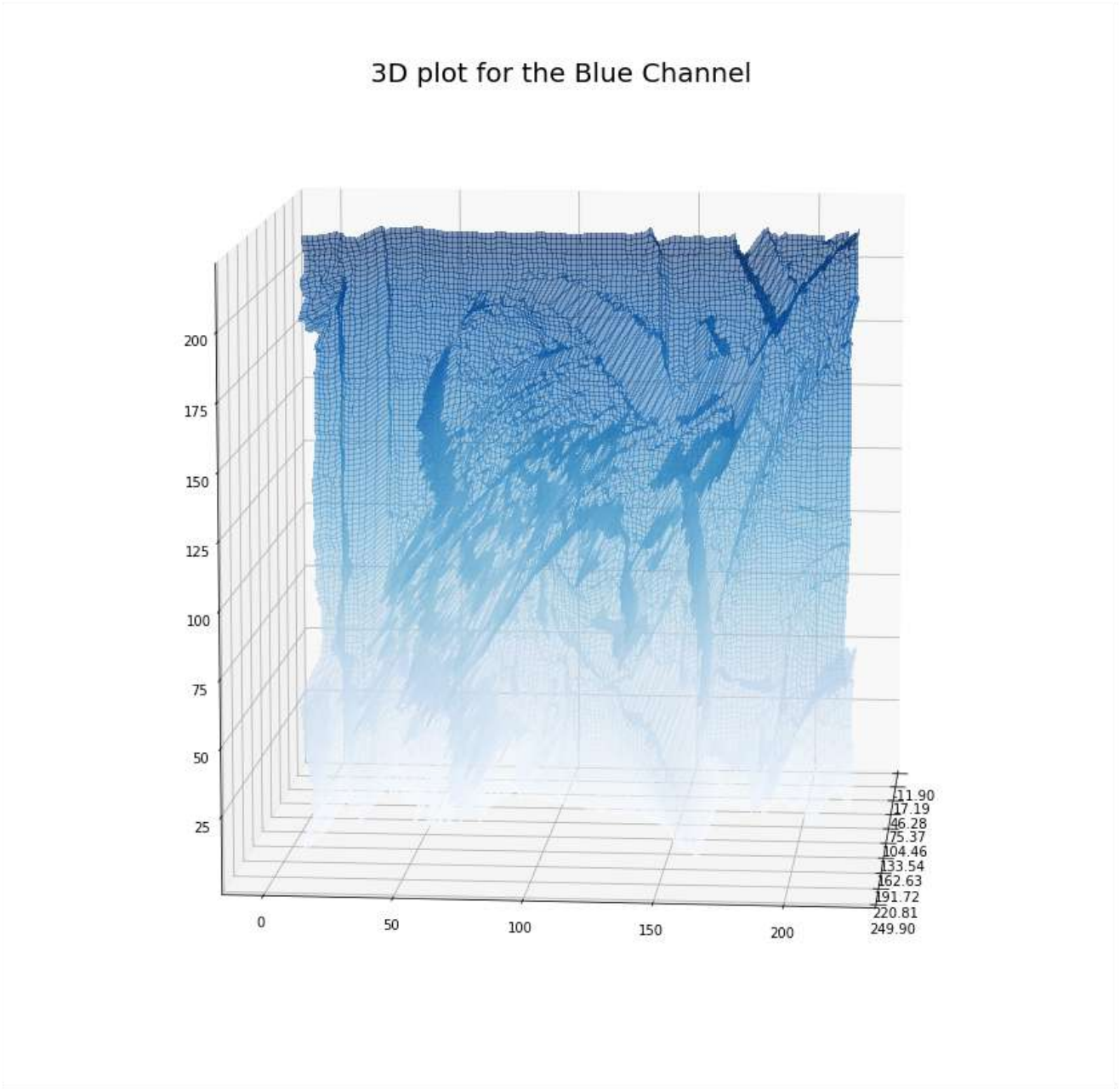
In [22]:
```
# plot 3D visualizations of the R, G, B channels of the image respectively
plot_3d(Z1, X, im.shape[1]-Y, cmap='Reds', title='3D plot for the Red Channel')
```



In [23]:
```
plot_3d(Z2, X, im.shape[1]-Y, cmap='Greens', title='3D plot for the Green Channel')
```



In [24]:
```
plot_3d(Z3, X, im.shape[1]-Y, cmap='Blues', title='3D plot for the Blue Channel')
```

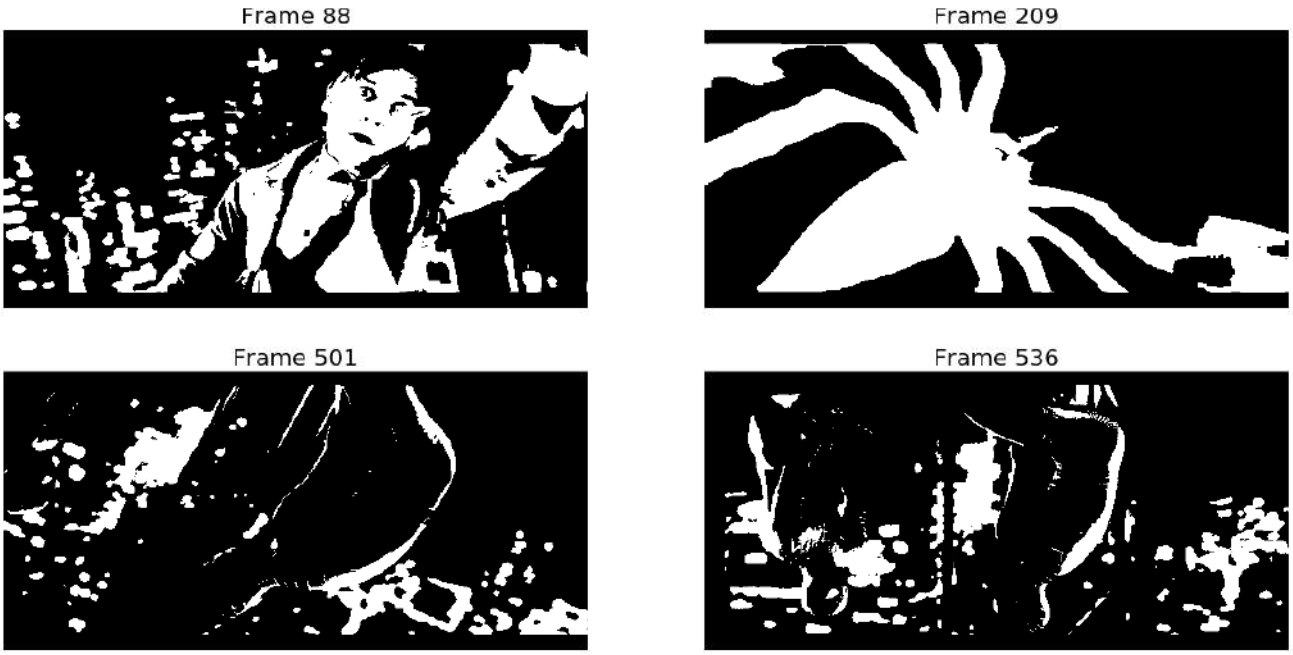


2. Video I/O

2.1 Read/Write Video Files with scikit-video

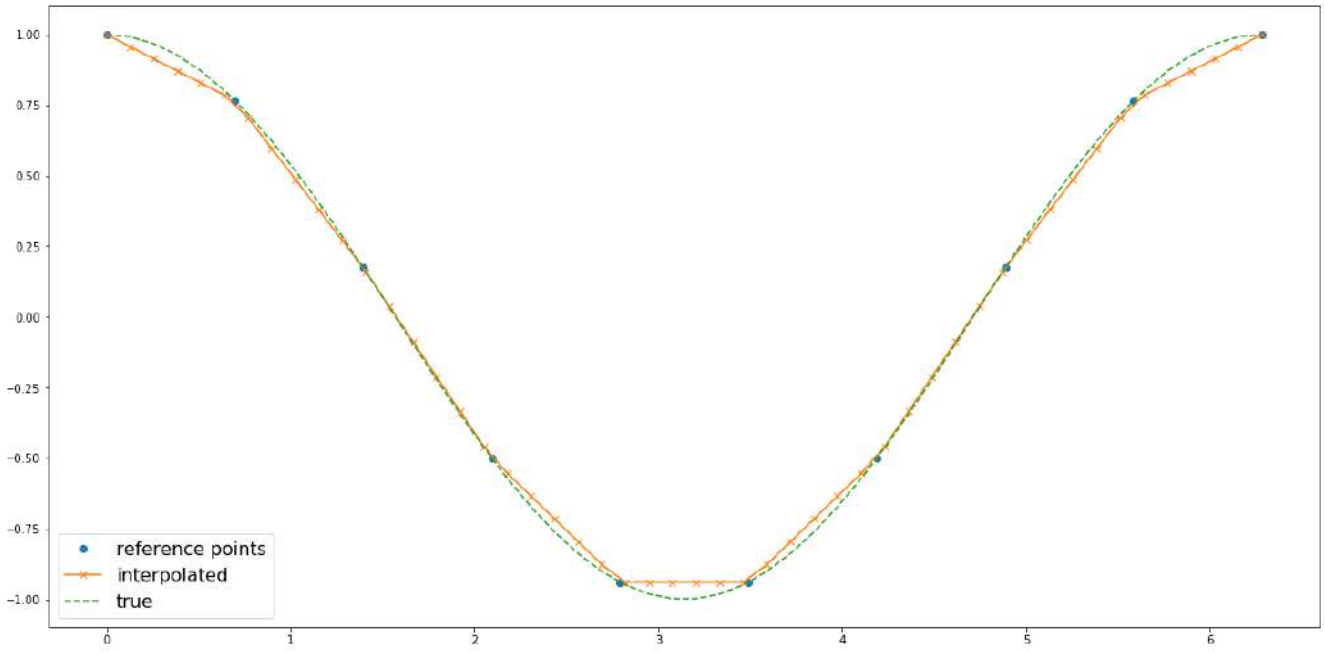
In [36]:
```
plt.figure(figsize=(20,10))
# iterate through the frames and display a few frames
frame_list = np.random.choice(num_frames, 4)
i, j = 0, 1
for frame in reader.nextFrame():
    if i in frame_list:
        plt.subplot(2,2,j)
        plt.imshow(frame)
        plt.title("Frame {}".format(i), size=20)
        plt.axis('off')
        j += 1
    i += 1
plt.show()
```

In [19]:
```
plt.figure(figsize=(20,10))
# iterate through the frames and display a few frames
reader = skvideo.io.FFmpegReader("images/spiderman_binary.mp4")
num_frames, height, width, num_channels = reader.getShape()
frame_list = np.random.choice(num_frames, 4)
i, j = 0, 1
for frame in reader.nextFrame():
    if i in frame_list:
        plt.subplot(2,2,j)
        plt.imshow(frame)
        plt.title("Frame {}".format(i), size=20)
        plt.axis('off')
        j += 1
    i += 1
plt.show()
```

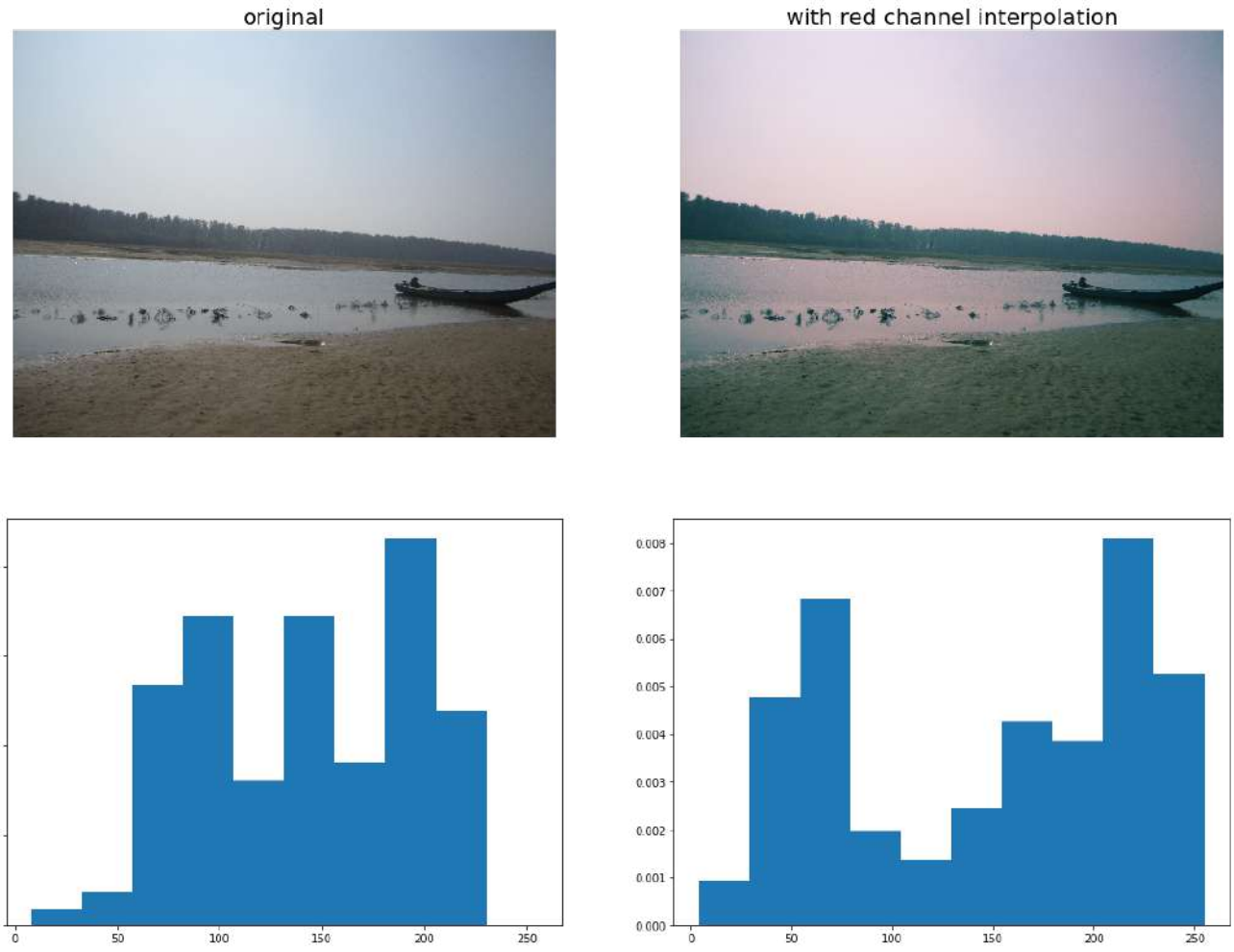


3. Implement Instagram-like Gotham Filter

In [5]:
```
# reference points
x_p = np.linspace(0, 2*np.pi, 10) # generate sequence of 10 points (numbers) evenly spac
# true values at reference points
y_p = np.cos(x_p)
# test points
x = np.linspace(0, 2*np.pi, 50) # generate sequence of 50 test points (numbers) evenly s
# true values at all points
y = np.cos(x)
# interpolated values at all test points
y_interp = np.interp(x, x_p, y_p)
# now plot
plt.figure(figsize=(20,10))
plt.plot(x_p, y_p, 'o', label='reference points')
plt.plot(x, y_interp, '-x', label='interpolated')
plt.plot(x, y, '--', label='true')
plt.legend(prop={'size': 16})
plt.show()
```



In [12]:
```
# plot with 2x2 subplots
plt.figure(figsize=(20,15))
plt.subplot(221)
plt.imshow(im)
plt.title('original', size=20)
plt.axis('off')
plt.subplot(222)
im1 = Image.merge('RGB', (r1, g, b))
plt.imshow(im1)
plt.axis('off')
plt.title('with red channel interpolation', size=20)
plt.subplot(223)
plt.hist(np.array(r).ravel(), normed=True)
plt.subplot(224)
plt.hist(np.array(r1).ravel(), normed=True)
plt.show()
```

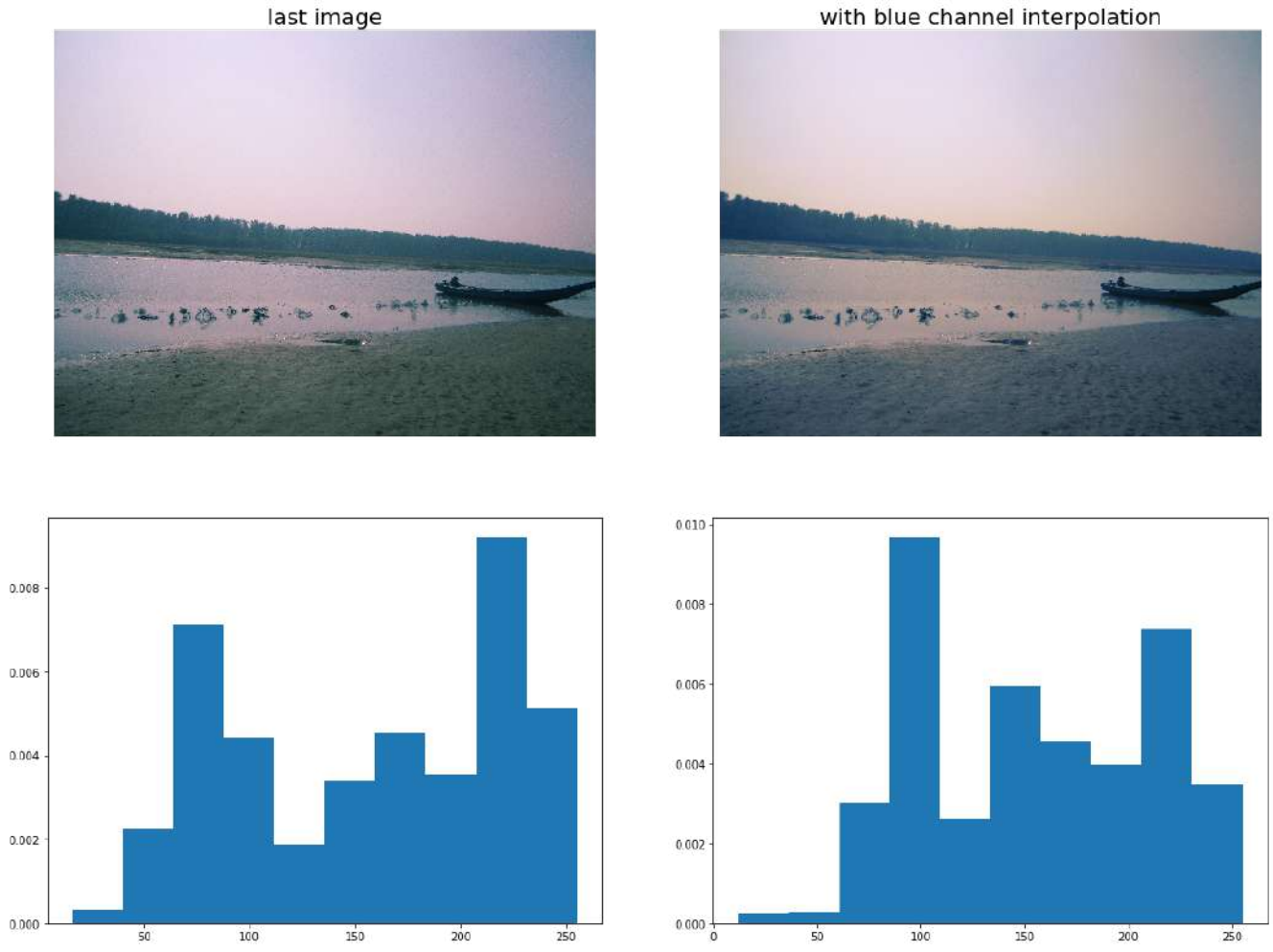


In [5]:
```
plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(im1)
plt.title('last image', size=20)
plt.axis('off')
b1 = Image.fromarray(np.clip(np.array(b) + 7.65, 0, 255).astype(np.uint8))
im1 = Image.merge('RGB', (r1, g, b1))
plt.subplot(122)
plt.imshow(im1)
plt.axis('off')
plt.title('with transformation', size=20)
plt.tight_layout()
plt.show()
```

In [6]:
```
from PIL.ImageEnhance import Sharpness

plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(im1)
plt.title('last image', size=20)
plt.axis('off')
im2 = Sharpness(im1).enhance(3.0)
plt.subplot(122)
plt.imshow(im2)
plt.axis('off')
plt.title('with transformation', size=20)
plt.tight_layout()
plt.show()
```

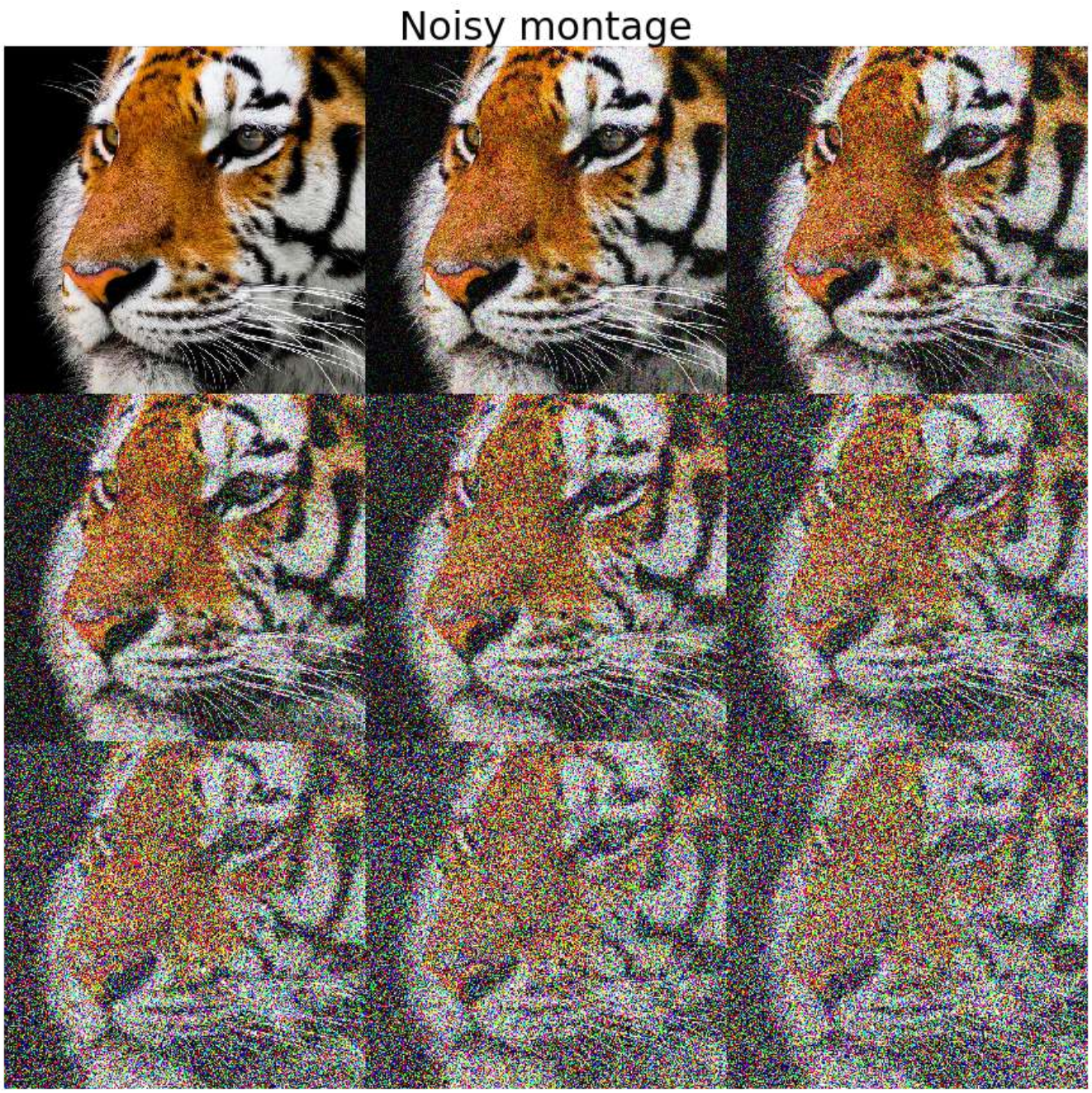
In [7]:
```
plt.figure(figsize=(20,15))
plt.subplot(221)
plt.imshow(im2)
plt.title('last image', size=20)
plt.axis('off')
plt.subplot(222)
im3 = Image.merge('RGB', (r1, g, b2))
plt.imshow(im3)
plt.axis('off')
plt.title('with blue channel interpolation', size=20)
plt.subplot(223)
plt.hist(np.array(b1).ravel(), normed=True)
plt.subplot(224)
plt.hist(np.array(b2).ravel(), normed=True)
plt.show()
```



4. Explore image manipulations with different python libraries

4.1 Plot image montage with scikit-image

In [1]:
```
plt.figure(figsize=(15,15))
plt.imshow(noisy_images_montage)
plt.title('Noisy montage', size=30)
plt.axis('off')
plt.show()
```



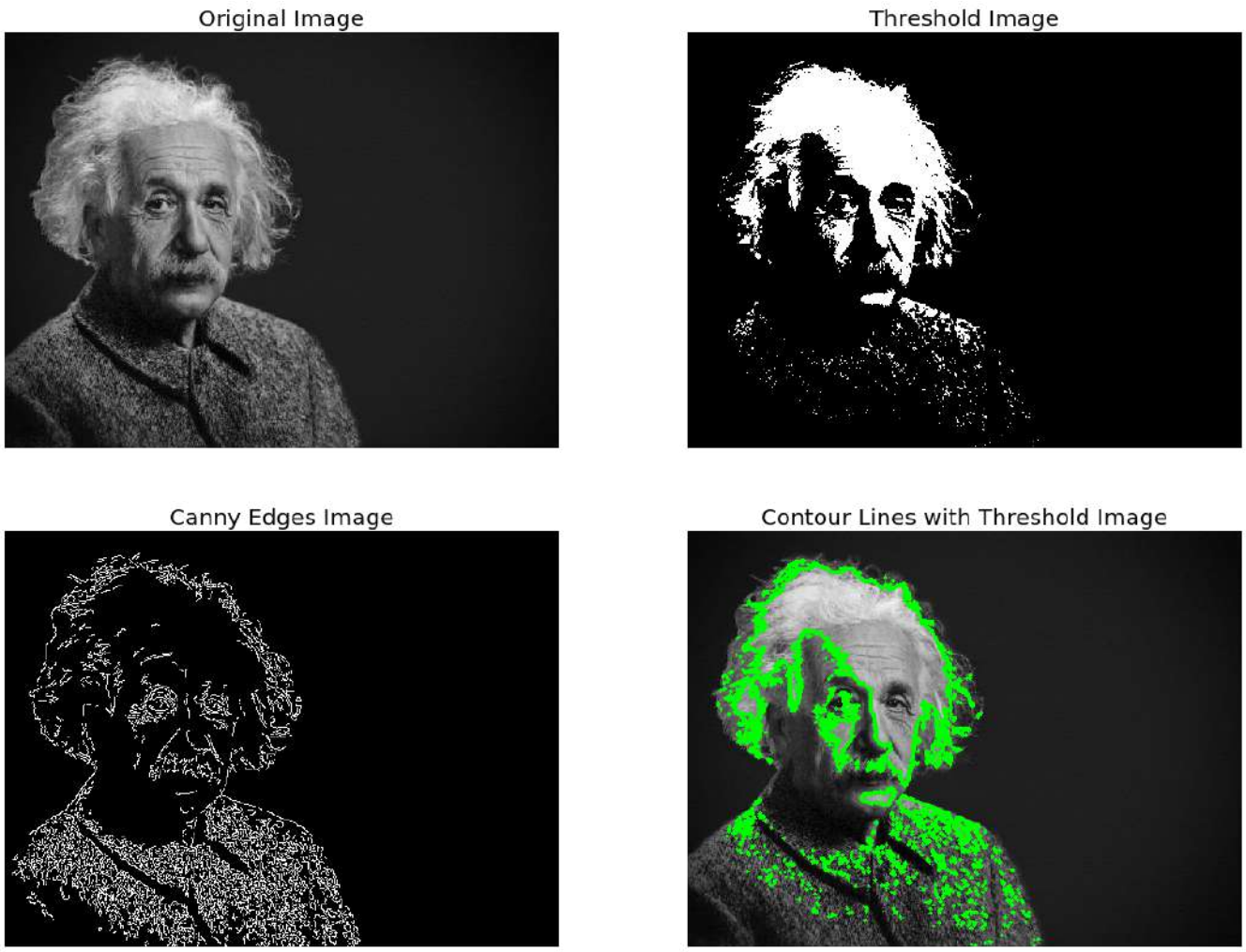
4.2 Crop / Resize images with scipy ndimage module

In [8]:
```
plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(im)
plt.title('Original Image', size=25)
plt.subplot(122)
plt.imshow(zoomed_im[125:325,375:550,:]) # crop the enlarged face
plt.title('Zoomed and Cropped Image', size=25)
plt.show()
```

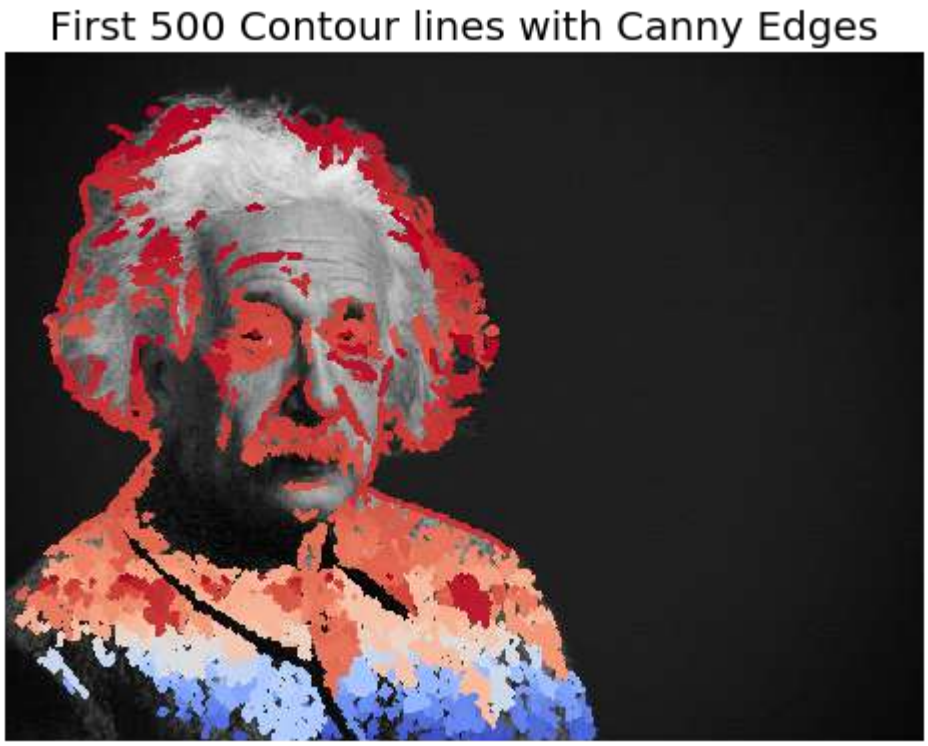
4.3 Draw contours with opencv-python

In [5]:
```
plt.figure(figsize=(20,15))
plt.subplot(221), plt.imshow(image), plt.title('Original Image', size=20), plt.axis('of
plt.subplot(222), plt.imshow(thresh, cmap='gray'), plt.title('Threshold Image', size=20
plt.subplot(223), plt.imshow(edged, cmap='gray'), plt.title('Canny Edges Image', size=2
plt.subplot(224), plt.imshow(cv2.drawContours(np.copy(image), contours_thresh, -1, (0,25
plt.title('Contour Lines with Threshold Image', size=20), plt.axis('off')
```

Out[5]:
```
(Text(0.5, 1.0, 'Contour Lines with Threshold Image'),
 (-0.5, 639.5, 479.5, -0.5))
```

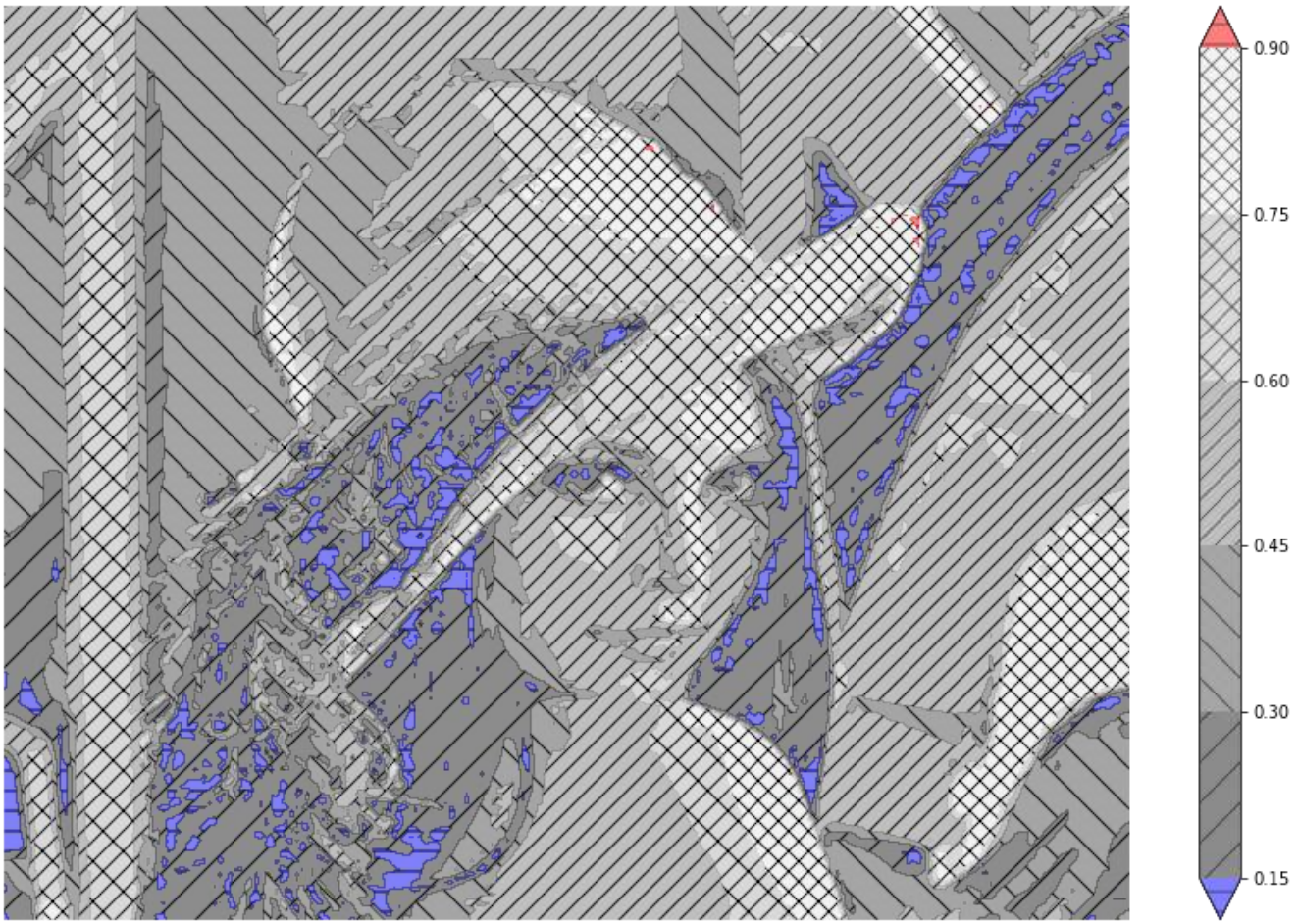


In [12]:
```
n = 500
plt.figure(figsize=(7,7))
colors = plt.cm.coolwarm(np.linspace(0, 1, n))
for i in range(n):
    image = cv2.drawContours(image, contours_edged, i, 255*colors[i], 3)
plt.imshow(image)
plt.title('First ' + str(n) + ' Contour lines with Canny Edges', size=20), plt.axis('of
plt.tight_layout()
plt.show()
```



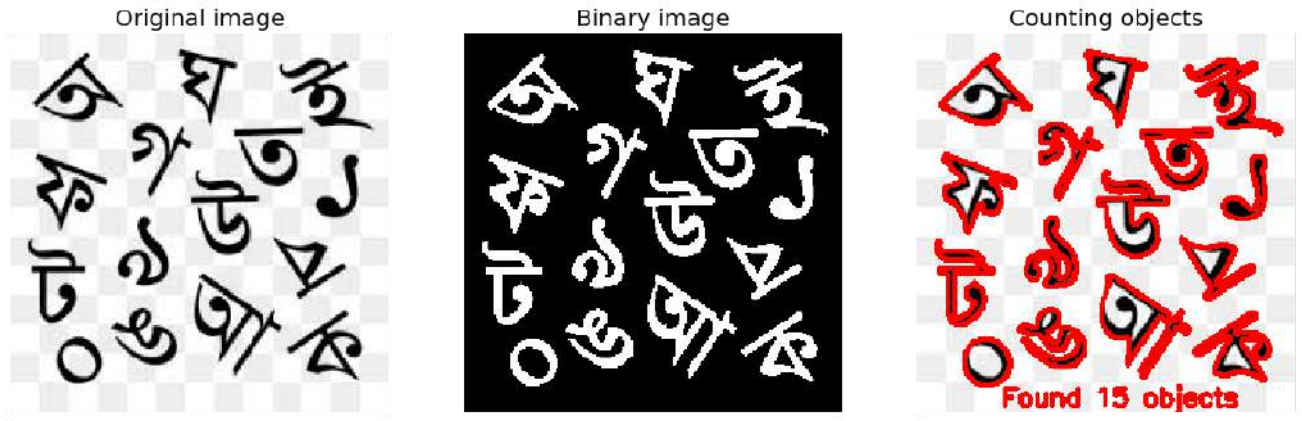
4.4 Creating different Hatched Contour Patterns for different levels with matplotlib

In [7]:
```
plt.figure(figsize=(15,10))
cs = plt.contourf(x, img.shape[0]-y, z, hatches=['-', '/', '\\', '//', '//\\', '//\\\\'
cs.cmap.set_over('red')
cs.cmap.set_under('blue')
cs.changed()
plt.colorbar()
plt.axis('off')
plt.show()
```



4.5 Counting Objects in an image

In [14]:
```
text = "Found {} objects".format(len(cnts))
cv2.putText(output, text, (50, 220),  cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
plt.figure(figsize=(20,7))
plt.subplot(131), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)), plt.axis('off'),
plt.subplot(132), plt.imshow(thresh, cmap='gray'), plt.axis('off'), plt.title('Binary im
plt.subplot(133), plt.imshow(cv2.cvtColor(output, cv2.COLOR_BGR2RGB)), plt.axis('off'),
plt.show()
```



4.6 Convert a png image with palette to gray scale with PIL

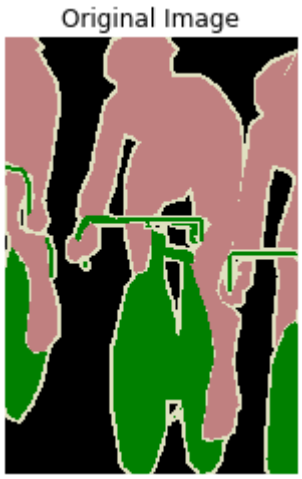
In [40]:
```
#The easy way

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open('images/Img_01_07.png')
print(img.mode)

plt.imshow(img)
plt.axis('off')
plt.title('Original Image')
plt.show()
```

P



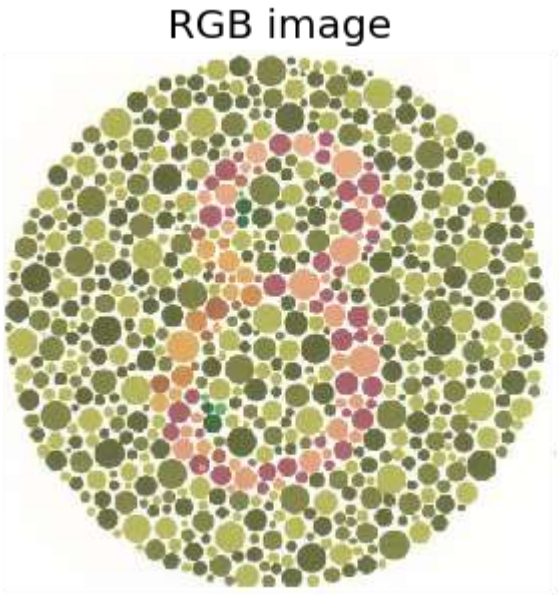
In [41]:
```
img = img.convert('RGB').convert('L')
print(img.mode)
plt.imshow(img, cmap='gray')
plt.axis('off')
plt.title('Grayscale Image')
plt.show()
```

```
L
```

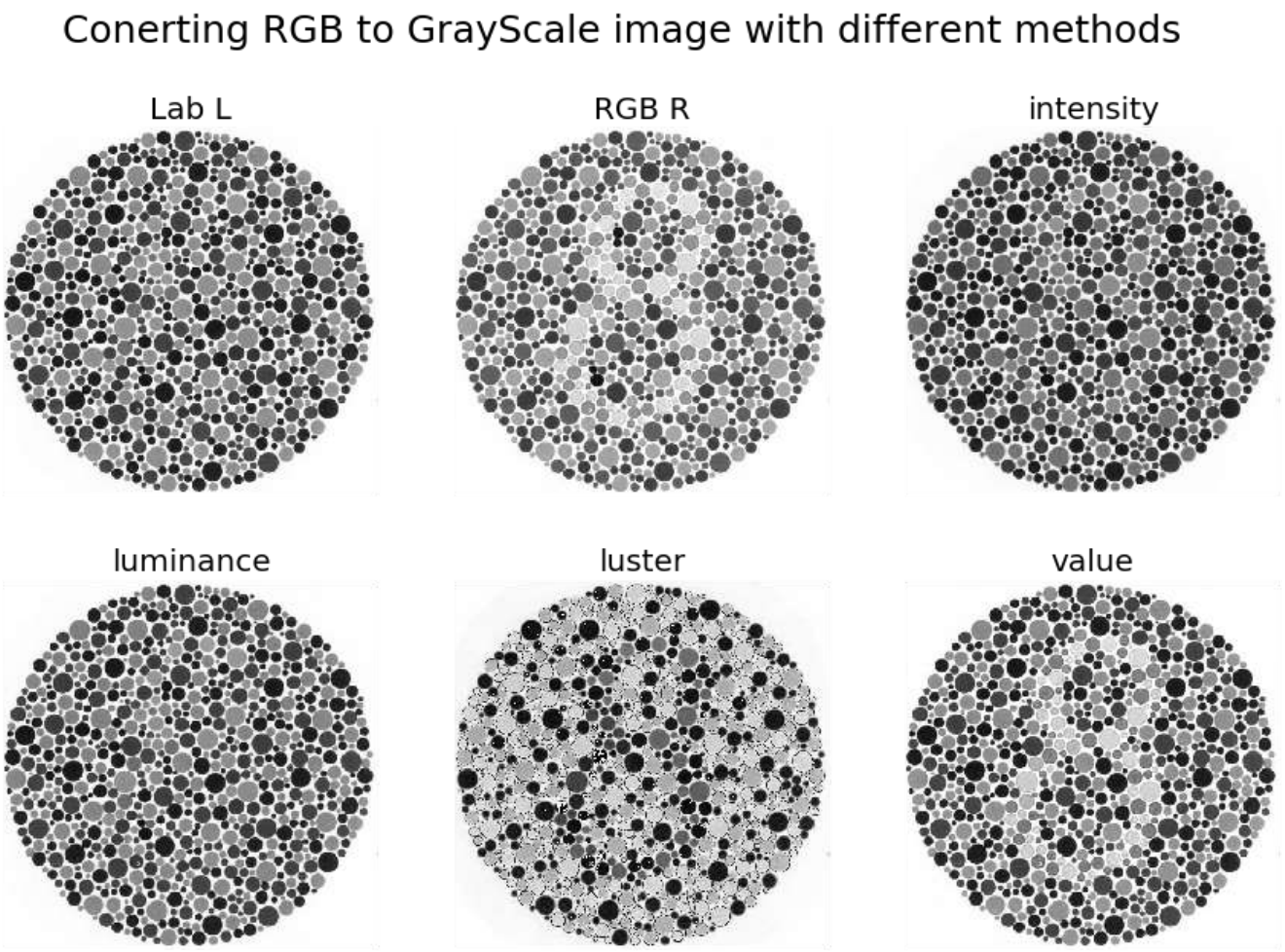
In [215]:
```
plt.imshow(arr, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')
plt.show()
```

4.7 Different ways to convert an RGB image to GrayScale

In [44]:
```
image = imread('images/Img_01_17.png')
plt.figure(figsize=(5,5))
plt.imshow(image), plt.axis('off'), plt.title('RGB image', size=20)
plt.show()
```



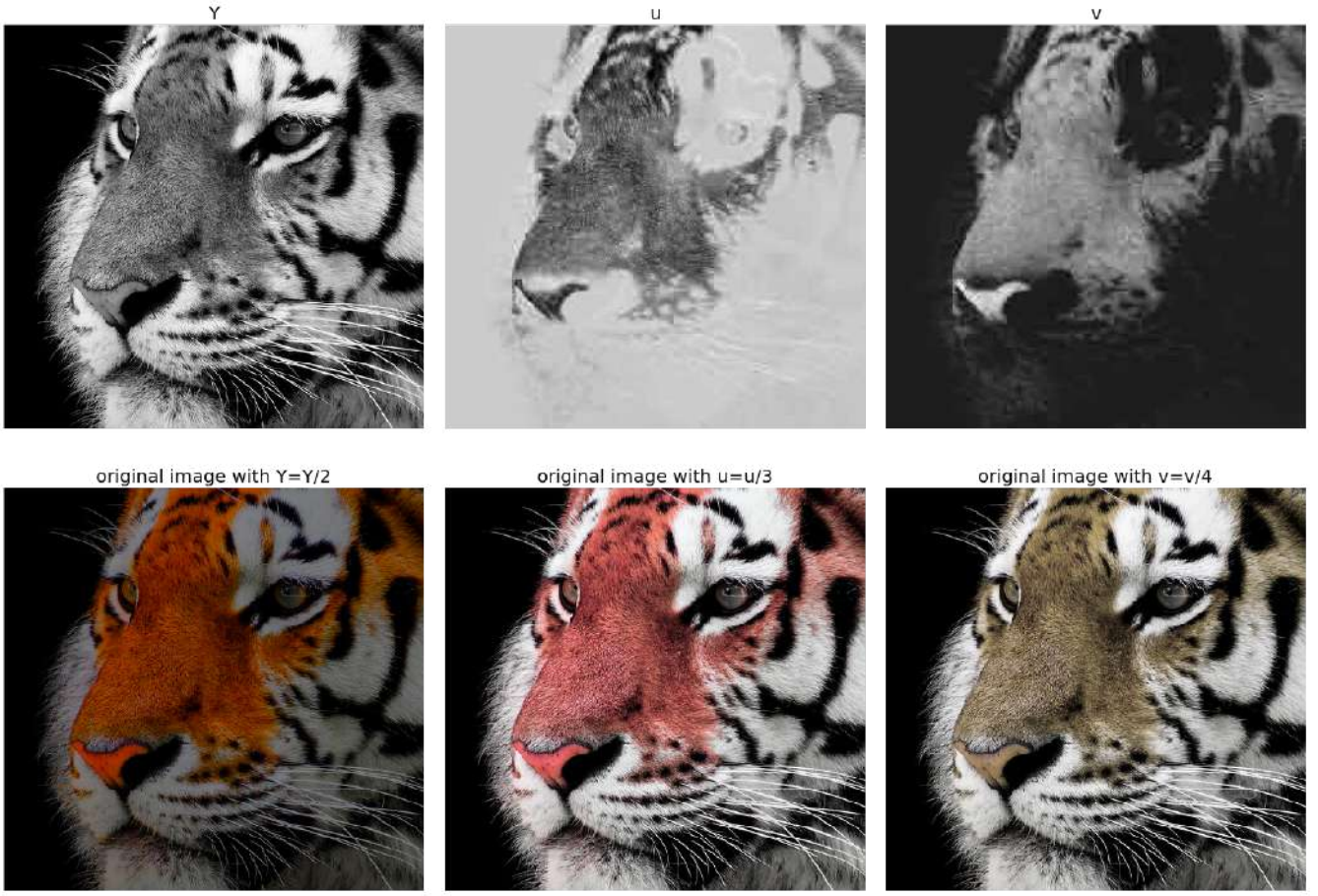
In [45]:
```
gray_images = rgb2gray(image)
i = 1
plt.figure(figsize=(15,10))
plt.gray()
for gray_type in sorted(gray_images):
    plt.subplot(2,3,i), plt.imshow(gray_images[gray_type]), plt.axis('off'), plt.title(g
    i += 1
plt.suptitle('Conerting RGB to GrayScale image with different methods', size=25)
plt.show()
```



4.9 RGB to hsv and Yuv color spaces with scikit-image

In [16]:
```
plt.figure(figsize=(20,12))
plt.subplots_adjust(0,0,1,0.925,0.05,0.05)
plt.gray()
plt.subplot(231), plt.imshow(im_hsv[...,0]), plt.title('h', size=20), plt.axis('off')
plt.subplot(232), plt.imshow(im_hsv[...,1]), plt.title('s', size=20), plt.axis('off')
plt.subplot(233), plt.imshow(im_hsv[...,2]), plt.title('v', size=20), plt.axis('off')
im_hsv_copy = np.copy(im_hsv)
im_hsv[...,0] /= 4
plt.subplot(234), plt.imshow(np.clip(hsv2rgb(im_hsv), 0, 1)), plt.title('original image
im_hsv = im_hsv_copy
im_hsv[...,1] /= 3
plt.subplot(235), plt.imshow(np.clip(hsv2rgb(im_hsv), 0, 1)), plt.title('original image
im_hsv = im_hsv_copy
im_hsv[...,2] /= 5
plt.subplot(236), plt.imshow(np.clip(hsv2rgb(im_hsv), 0, 1)), plt.title('original image
plt.show()
```

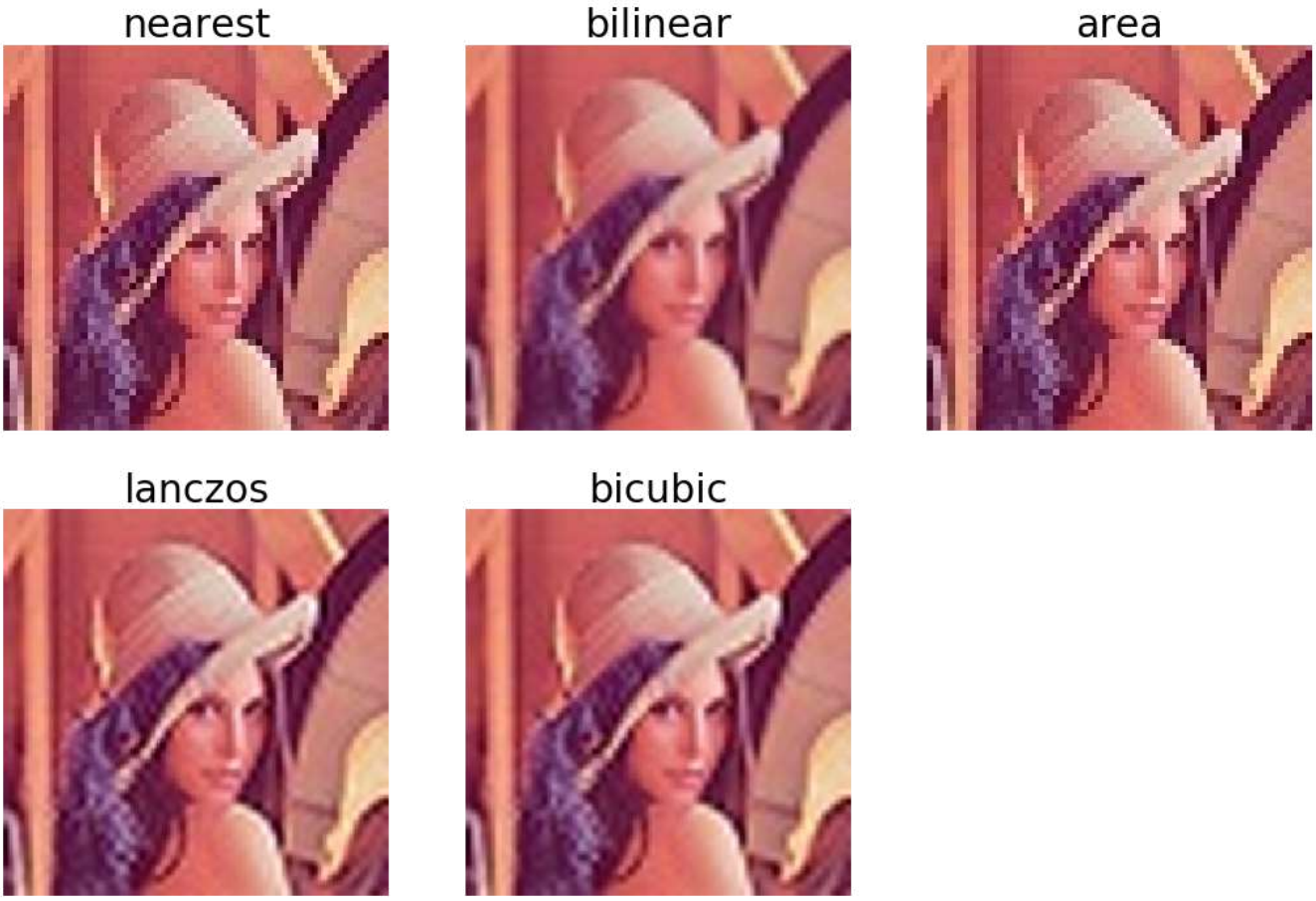
In [25]:
```
plt.figure(figsize=(20,15))
plt.subplots_adjust(0,0,1,0.925,0.05,0.05)
plt.gray()
plt.subplot(231), plt.imshow(im_Yuv[...,0]), plt.title('Y', size=20), plt.axis('off')
plt.subplot(232), plt.imshow(im_Yuv[...,1]), plt.title('u', size=20), plt.axis('off')
plt.subplot(233), plt.imshow(im_Yuv[...,2]), plt.title('v', size=20), plt.axis('off')
im_Yuv_copy = np.copy(im_Yuv)
im_Yuv[...,0] /= 2
plt.subplot(234), plt.imshow(np.clip(yuv2rgb(im_Yuv),0,1)), plt.title('original image wi
im_Yuv = im_Yuv_copy
im_Yuv[...,1] /= 3
plt.subplot(235), plt.imshow(np.clip(yuv2rgb(im_Yuv),0,1)), plt.title('original image wi
im_Yuv = im_Yuv_copy
im_Yuv[...,2] /= 4
plt.subplot(236), plt.imshow(np.clip(yuv2rgb(im_Yuv),0,1)), plt.title('original image wi
plt.show()
```



4.10 Resizing an image with opencv-python

In [2]:
```
i = 1
plt.figure(figsize=(18,12))
for interp in [cv2.INTER_NEAREST, cv2.INTER_LINEAR, cv2.INTER_AREA, cv2.INTER_LANCZOS4,
 im1 = cv2.resize(im, None, fx=4., fy=4., interpolation = interp) # 4 times
 plt.subplot(2,3,i)
 plt.imshow(cv2.cvtColor(im1, cv2.COLOR_BGR2RGB))
 plt.axis('off')
 plt.title(interps[i-1], size=30)
 i += 1
print(im.shape, im1.shape)
plt.show()
```

```
(55, 55, 3) (220, 220, 3)
```



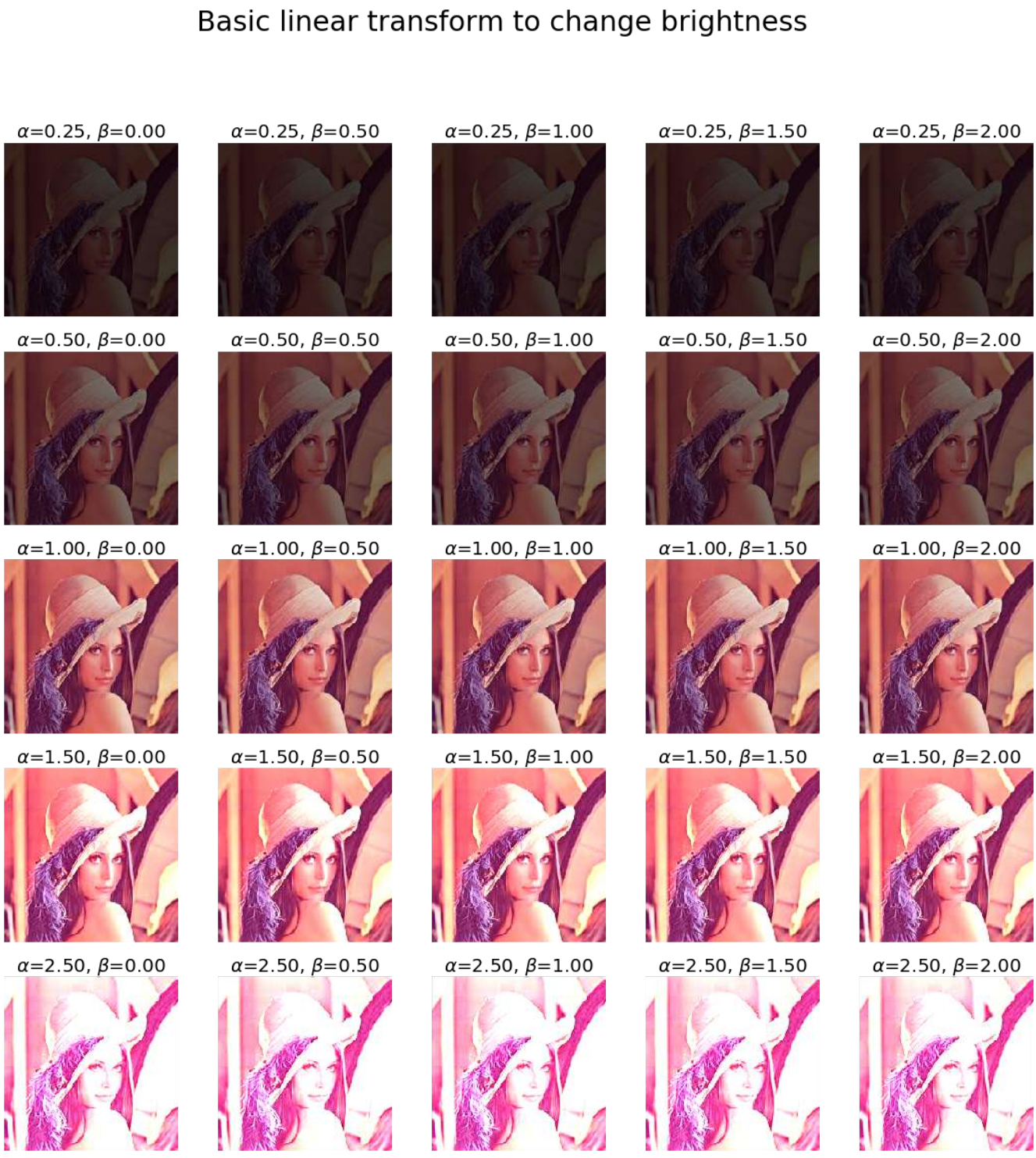
4.11 Add a logo to an image with scikit-image

In [68]:
```
# Put logo in ROI and modify the main image
dst = img1_bg + img2_fg
img1[0:rows, 0:cols ] = dst
plt.figure(figsize=(20,20))
plt.imshow(img1)
plt.axis('off')
plt.show()
```

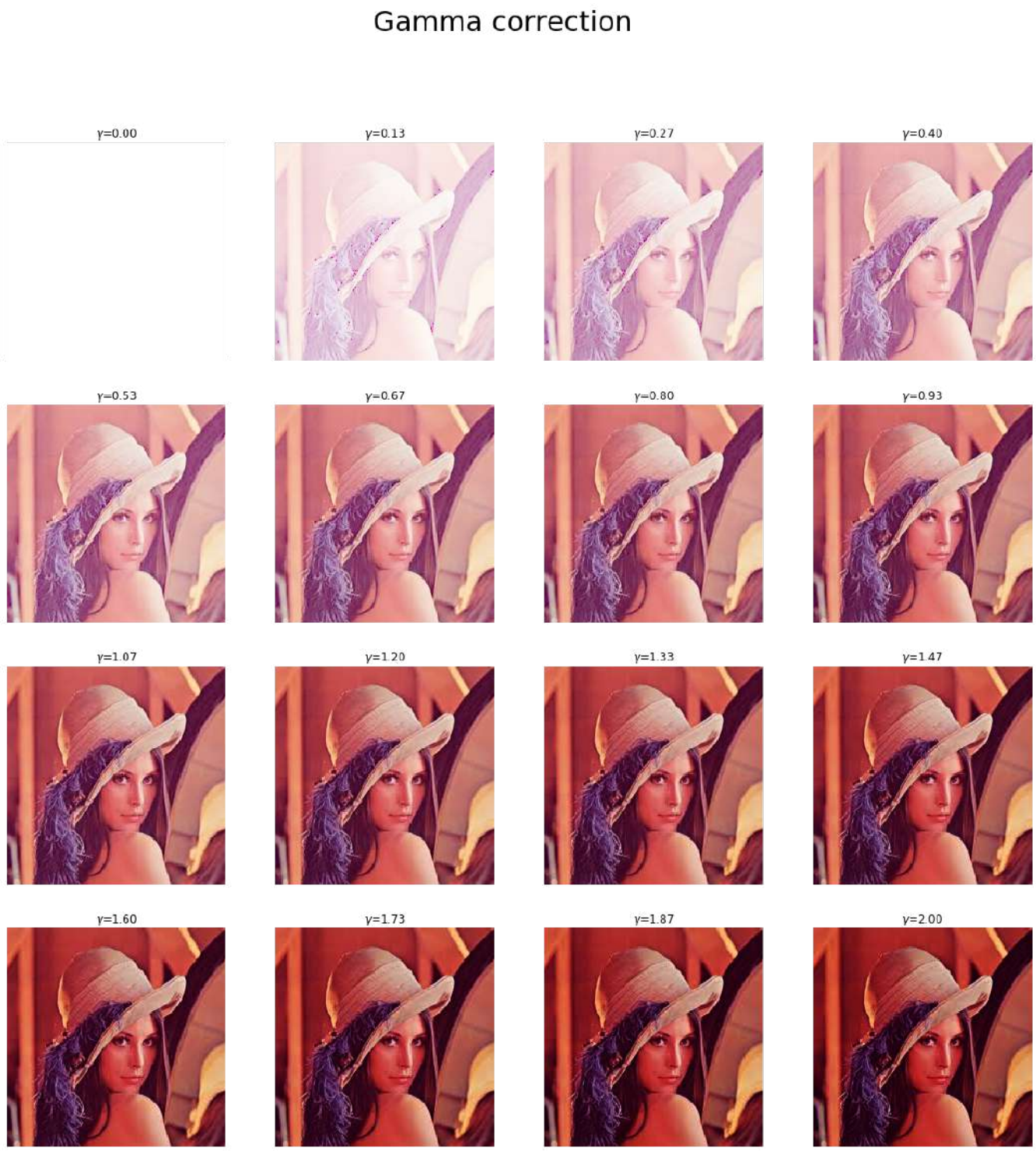
4.12 Change brightness / contrast of an image with linear transform and gamma correction with opencv-python

In [4]:

```
plt.figure(figsize=(20,20))
i = 1
for alpha in [0.25, 0.5, 1, 1.5, 2.5]:
    for beta in [0, 0.5, 1, 1.5, 2]:
        image_corrected = basic_linear_transform(image, alpha, beta)
        plt.subplot(5,5,i), plt.imshow(cv2.cvtColor(image_corrected, cv2.COLOR_BGR2RGB))
        plt.title(r'$\alpha$={:.2f}, $\beta$={:.2f}'.format(alpha, beta), size=20)
        i += 1
plt.suptitle('Basic linear transform to change brightness', size=30)
plt.show()
```



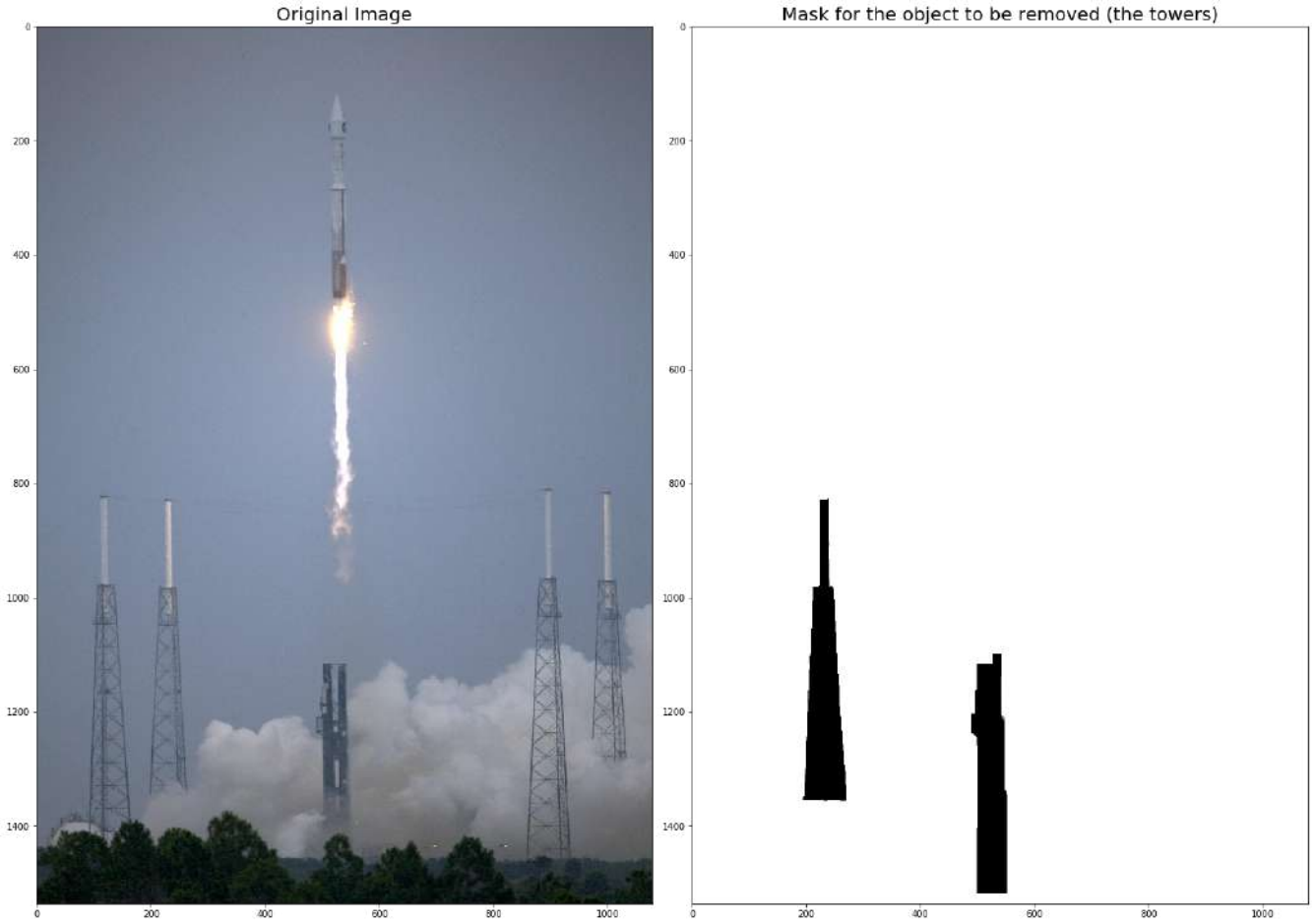
In [5]:
```
plt.figure(figsize=(20,20))
i = 1
for gamma in np.linspace(0, 2, 16):
    image_gamma_corrected = gamma_correction(image, gamma)
    plt.subplot(4,4,i), plt.imshow(cv2.cvtColor(image_gamma_corrected, cv2.COLOR_BGR2RGB
    plt.title(r'$\gamma$={:.2f}'.format(gamma))
    i += 1
plt.suptitle('Gamma correction', size=30)
plt.show()
```



5. Object Removal with Seam Carving

In [9]:
```
image = imread('images/Img_01_27.png')
mask_image = rgb2gray(imread('images/Img_01_28.png'))
print(image.shape)
plt.figure(figsize=(20,20))
plt.subplot(121), plt.imshow(image), plt.title('Original Image', size=20)
plt.subplot(122), plt.imshow(mask_image, cmap='gray'), plt.title('Mask for the object to
plt.tight_layout()
plt.show()
```

```
(1536, 1079, 3)
```



In [6]:
```
plt.figure(figsize=(10,15))
plt.title('Objects Removed', size=15)
out = seam_carve(image, mask_image, 'verical', 120)
plt.imshow(out)
plt.show()
```

In [36]:
```
im = Image.open("images/Img_01_29.png")
mask = Image.open("images/Img_01_30.png")
out = create_fake_miniature(im, mask)
plt.figure(figsize=(20,10))
plt.imshow(im), plt.axis('off'), plt.title('Original image', size=20)
plt.show()
```

In [37]:
```
plt.figure(figsize=(10,10))
plt.imshow(mask), plt.axis('off'), plt.title('(Bell Whistles) Mask image', size=20)
plt.show()
```

In [38]:
```
plt.figure(figsize=(20,10))
plt.imshow(out), plt.axis('off'), plt.title('Fake Miniature image', size=20)
plt.show()
```