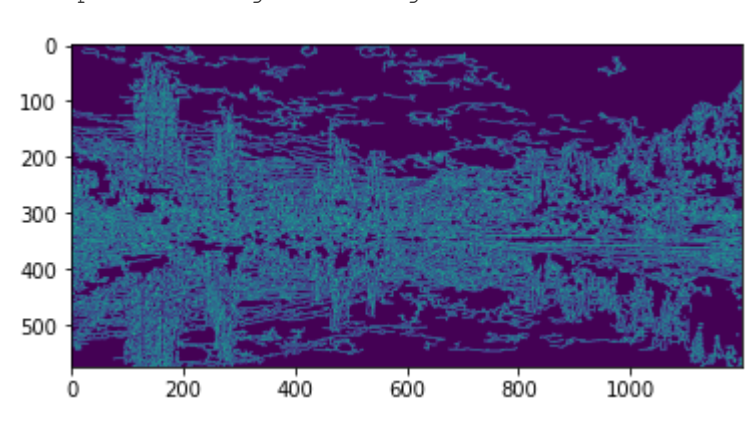
In [1]:
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

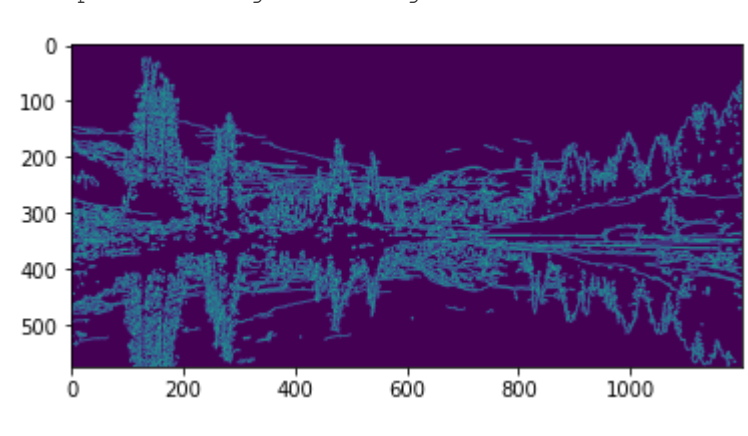
In [2]:
```python
# 1(A)
img = cv2.imread('Resources\\high_resolution.jpg')
img = cv2.Canny(img, 0, 50)
plt.imshow(img)
```

Out[2]: <matplotlib.image.AxesImage at 0x20228ffc488>



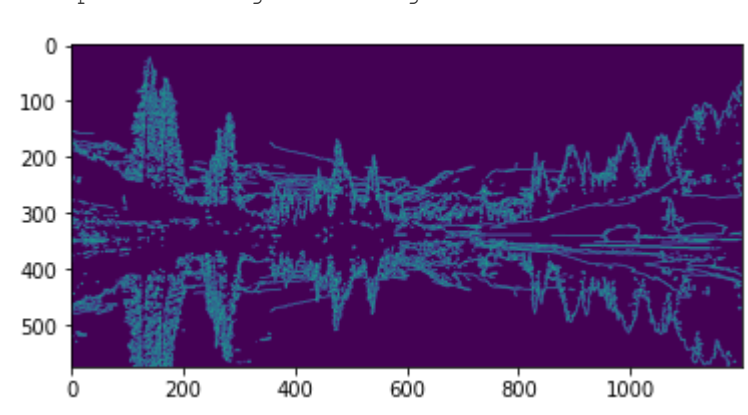
In [3]:
```python
# 1(B)
img = cv2.imread('Resources\\high_resolution.jpg')
img = cv2.Canny(img, 50, 100)
plt.imshow(img)
```

Out[3]: <matplotlib.image.AxesImage at 0x2022910f7c8>



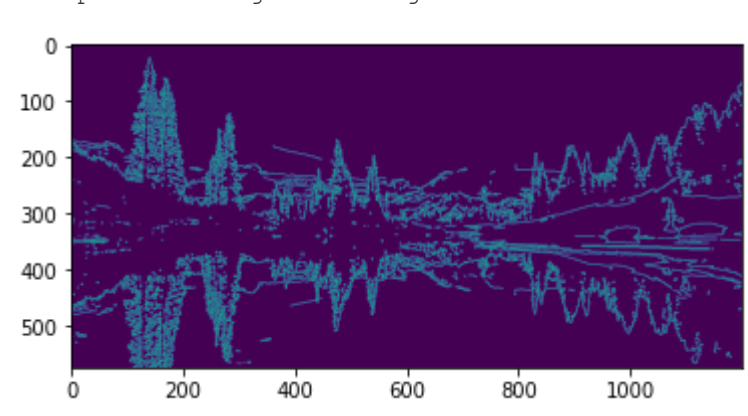
In [4]:
```python
# 1(C)
img = cv2.imread('Resources\\high_resolution.jpg')
img = cv2.Canny(img, 100, 150)
plt.imshow(img)
```

Out[4]: <matplotlib.image.AxesImage at 0x20229182948>



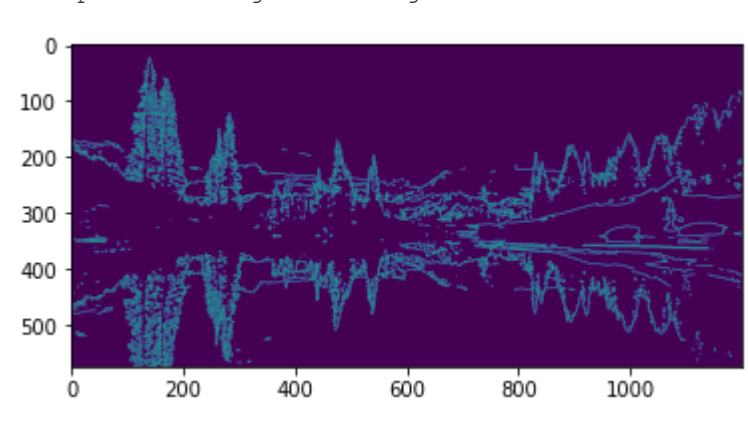
In [5]:
```python
# 1(D)
img = cv2.imread('Resources\\high_resolution.jpg')
img = cv2.Canny(img, 150, 200)
plt.imshow(img)
```

Out[5]: <matplotlib.image.AxesImage at 0x202291e9948>



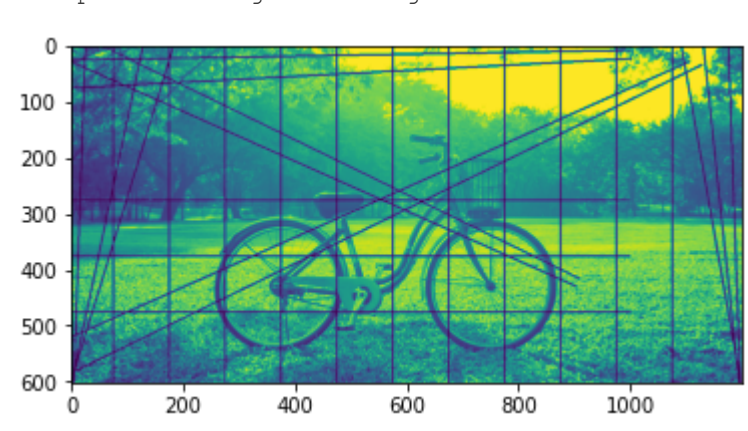
In [6]:
```python
# 1(E)
img = cv2.imread('Resources\\high_resolution.jpg')
img = cv2.Canny(img, 200, 250)
plt.imshow(img)
```

Out[6]: <matplotlib.image.AxesImage at 0x20229252808>



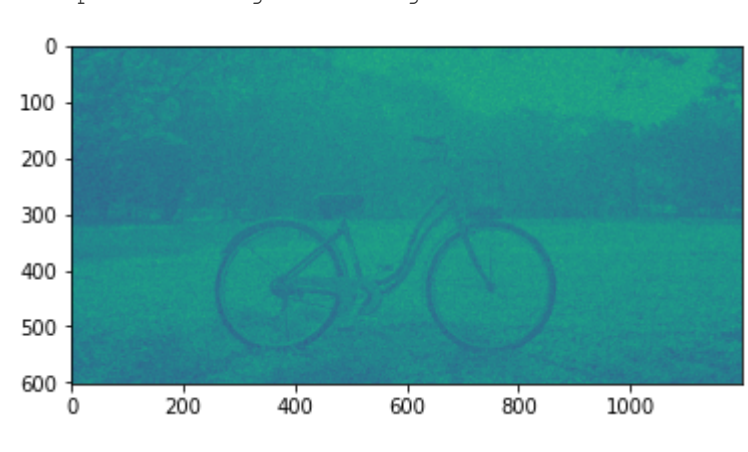
In [58]:
```python
# 2
img = cv2.imread('Resources\\bicycle.jpg', 0)
lines = cv2.HoughLines(img,50, np.pi/180, 200)
for line in lines:
    for rho,theta in line:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))

        cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

plt.imshow(img)
```

Out[58]: <matplotlib.image.AxesImage at 0x2022ea84e88>



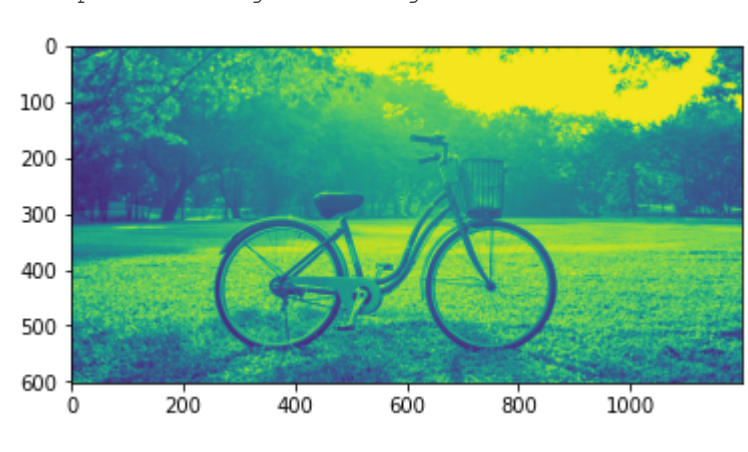
In [9]:
```python
# 4
img = cv2.imread('Resources\\bicycle.jpg', 0)

f = np.fft.rfft2(img)
f2 =  np.fft.rfft2(np.random.normal(0, 100, img.shape))
new_im = np.fft.irfft2(f + f2)
plt.imshow(new_im)
```

Out[9]: <matplotlib.image.AxesImage at 0x20229330ac8>



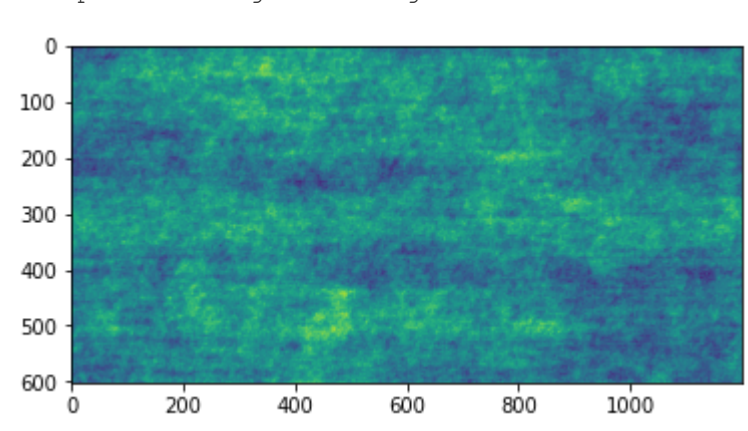
In [10]:
```python
img = cv2.imread('Resources\\bicycle.jpg', 0)

f = np.fft.rfft2(img)
f2 =  np.fft.rfft2(np.random.normal(0, 1, img.shape))
new_im = np.fft.irfft2(f + f2)
plt.imshow(new_im)
```

Out[10]: <matplotlib.image.AxesImage at 0x2022939d548>



In [11]:
```python
img = cv2.imread('Resources\\bicycle.jpg', 0)

f = np.fft.rfft2(img)
f2 =  np.fft.rfft2(np.random.normal(0, 1, img.shape))
new_im = np.fft.irfft2(f * f2)
plt.imshow(new_im)
```

Out[11]: <matplotlib.image.AxesImage at 0x2022d4c5f08>



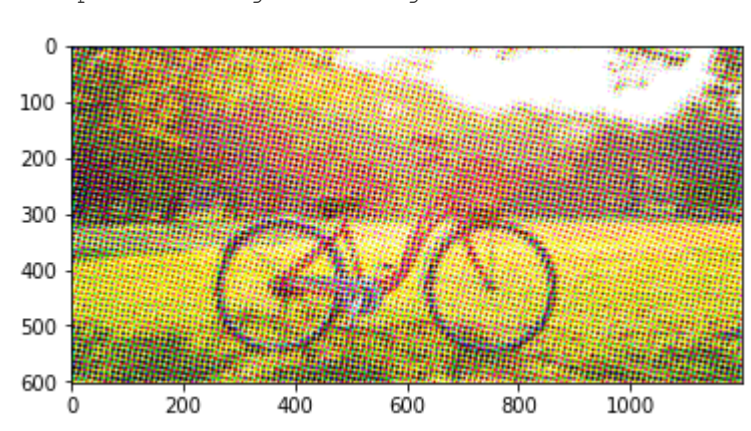
In [15]:
```python
# 4
from PIL import Image, ImageDraw, ImageStat
```

In [20]:
```python
def halftone(im, cmyk, sample, scale):
    '''Returns list of half-tone images for cmyk image. sample (pixels),
       determines the sample box size from the original image. The maximum
       output dot diameter is given by sample * scale (which is also the number
       of possible dot sizes). So sample=1 will presevere the original image
       resolution, but scale must be >1 to allow variation in dot size.'''
    cmyk = cmyk.split()
    dots = []
    angle = 0
    for channel in cmyk:
        channel = channel.rotate(angle, expand=1)
        size = channel.size[0]*scale, channel.size[1]*scale
        half_tone = Image.new('L', size)
        draw = ImageDraw.Draw(half_tone)
        for x in range(0, channel.size[0], sample):
            for y in range(0, channel.size[1], sample):
                box = channel.crop((x, y, x + sample, y + sample))
                stat = ImageStat.Stat(box)
                diameter = (stat.mean[0] / 255)**0.5
                edge = 0.5*(1-diameter)
                x_pos, y_pos = (x+edge)*scale, (y+edge)*scale
                box_edge = sample*diameter*scale
                draw.ellipse((x_pos, y_pos, x_pos + box_edge, y_pos + box_edge), fill=255)
        half_tone = half_tone.rotate(-angle, expand=1)
        width_half, height_half = half_tone.size
        xx=(width_half-im.size[0]*scale) / 2
        yy=(height_half-im.size[1]*scale) / 2
        half_tone = half_tone.crop((xx, yy, xx + im.size[0]*scale, yy + im.size[1]*scale))
        dots.append(half_tone)
        angle += 15
    return dots
```

In [38]:
```python
def gcr(im, percentage):
    '''basic "Gray Component Replacement" function. Returns a CMYK image with
       percentage gray component removed from the CMY channels and put in the
       K channel, ie. for percentage=100, (41, 100, 255, 0) >> (0, 59, 214, 41)'''
    cmyk_im = im.convert('CMYK')
    if not percentage:
        return cmyk_im
    cmyk_im = cmyk_im.split()
    cmyk = []
    for i in xrange(4):
        cmyk.append(cmyk_im[i].load())
    for x in xrange(im.size[0]):
        for y in range(im.size[1]):
            gray = min(cmyk[0][x,y], cmyk[1][x,y], cmyk[2][x,y]) * percentage / 100
            for i in xrange(3):
                cmyk[i][x,y] = cmyk[i][x,y] - gray
            cmyk[3][x,y] = gray
    return Image.merge('CMYK', cmyk_im)
```

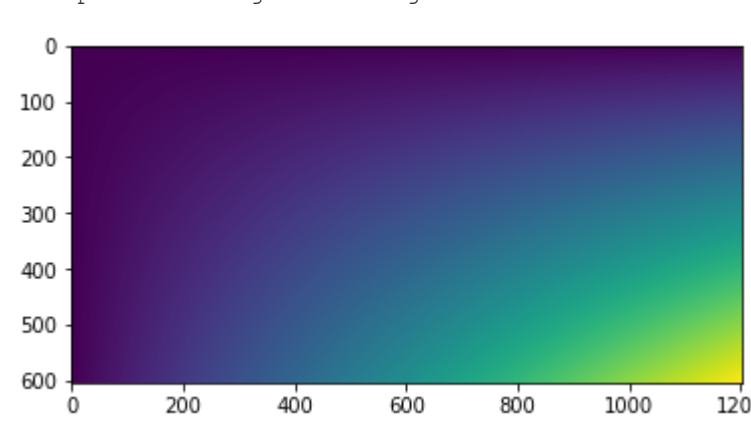
In [39]:
```python
im = Image.open('Resources\\bicycle.jpg')

cmyk = gcr(im, 0)
dots = halftone(im, cmyk, 10, 1)

new = Image.merge('CMYK', dots)
plt.imshow(new) # get halftone image
```

Out[39]: <matplotlib.image.AxesImage at 0x2022ea4ca48>



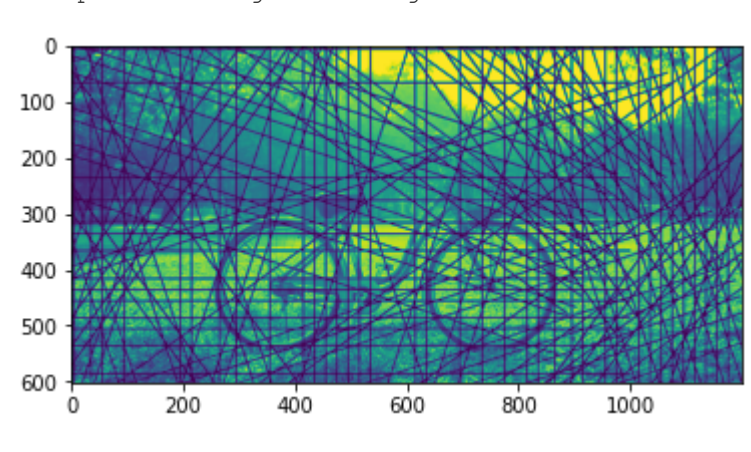
In [73]:
```python
new = np.array(new) # convert to numpy array
img = cv2.integral(img.astype(np.uint8))
plt.imshow(img) # show integral img
```

Out[73]: <matplotlib.image.AxesImage at 0x2022e92fd48>



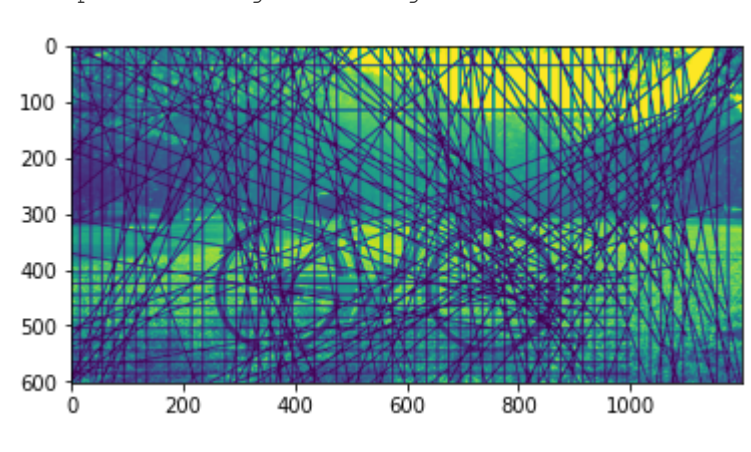
In [81]:
```python
edges = cv2.Canny(img.astype(np.uint8), 50, 150, apertureSize = 3)
img = cv2.imread('Resources\\bicycle.jpg', 0)
lines = cv2.HoughLines(edges, 10, np.pi/10, 100)
for line in lines:
    for rho,theta in line:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))

        cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

plt.imshow(img)
```

Out[81]: <matplotlib.image.AxesImage at 0x202302a7548>



In [84]:
```python
edges = cv2.Canny(img.astype(np.uint8), 50, 150, apertureSize = 3)
img = cv2.imread('Resources\\bicycle.jpg', 0)
lines = cv2.HoughLines(edges, 10, np.pi/18, 100)
for line in lines:
    for rho,theta in line:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))

        cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

plt.imshow(img)
```

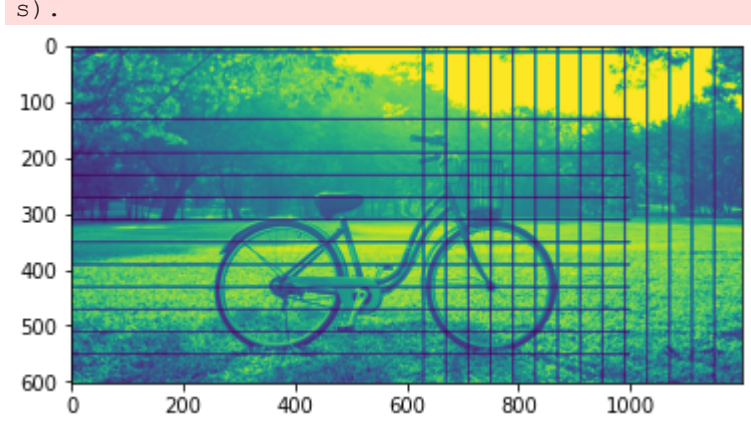
Out[84]: <matplotlib.image.AxesImage at 0x202303e72c8>



In [111...
```python
def write_edges(egde_degree, img_path, times):
    im = Image.open(img_path)

    cmyk = gcr(im, 0)
    dots = halftone(im, cmyk, 10, 1)

    new = Image.merge('CMYK', dots)
    plt.imshow(new) # get halftone image
    new = np.array(new) # convert to numpy array
    img = cv2.integral(np.array(im).astype(np.uint8))
    plt.imshow(img) # show integral img
    edges = cv2.Canny(img.astype(np.uint8), 50, 150, apertureSize = 3)
    img = cv2.imread(img_path, 0)
    lines = cv2.HoughLines(edges, times, egde_degree, 255)
    for line in lines:
        for rho,theta in line:
            a = np.cos(theta)
            b = np.sin(theta)
            x0 = a*rho
            y0 = b*rho
            x1 = int(x0 + 1000*(-b))
            y1 = int(y0 + 1000*(a))
            x2 = int(x0 - 1000*(-b))
            y2 = int(y0 - 1000*(a))

            cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

    plt.imshow(img)
```

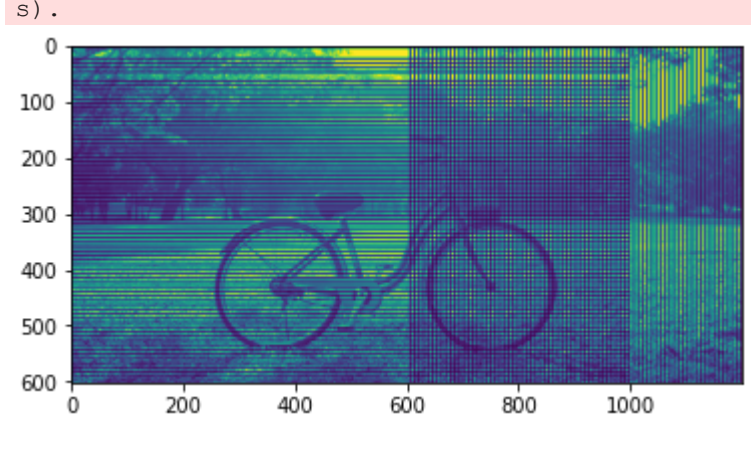
In [114...
```python
# 5
write_edges(np.pi/2, 'Resources\\bicycle.jpg', 20)
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



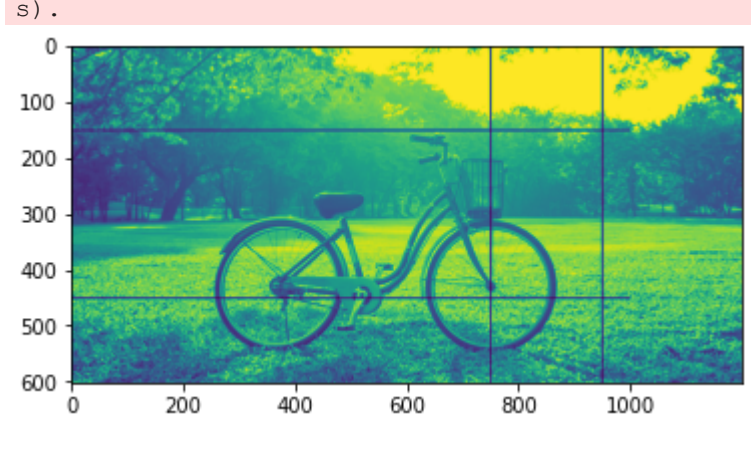
In [115...
```python
write_edges(np.pi/2, 'Resources\\bicycle.jpg', 4)
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



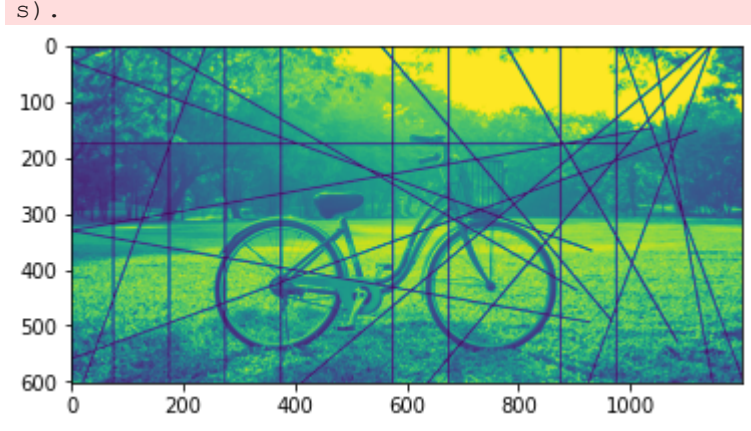
In [118...
```python
write_edges(np.pi/2, 'Resources\\bicycle.jpg', 100)
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



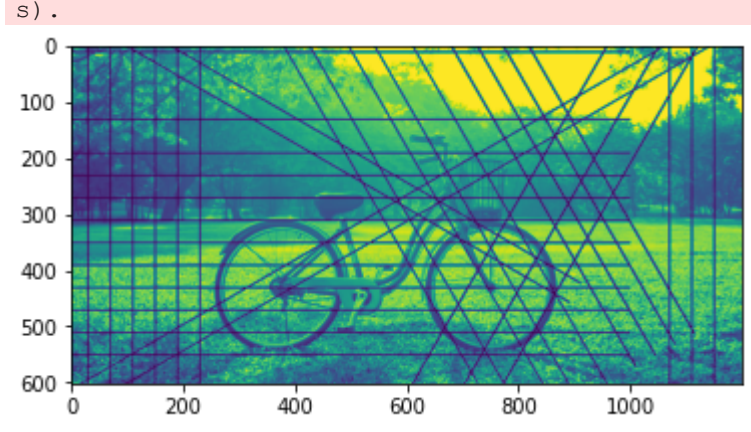
In [119...
```python
write_edges(np.pi/18, 'Resources\\bicycle.jpg', 50)
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



In [120...
```python
write_edges(np.pi/6, 'Resources\\bicycle.jpg', 20)
```

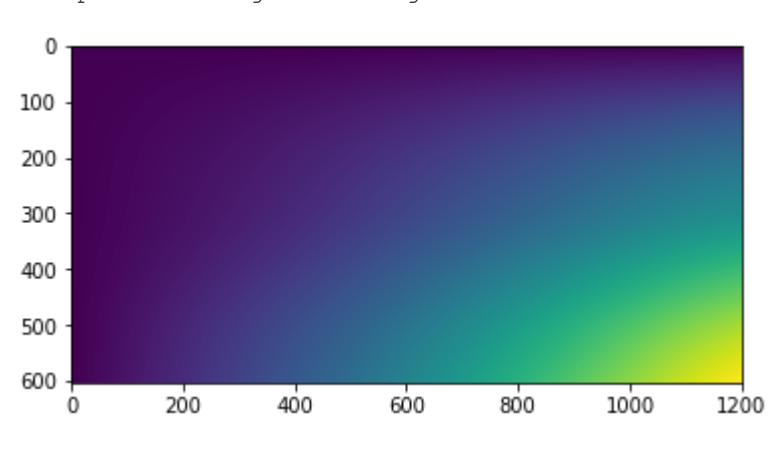
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



In [124...
```python
# 6
def rotate_image(image, angle):
    image = cv2.imread(image, 0)
    image_center = tuple(np.array(image.shape[1::-1]) / 2)
    rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    result = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)
    return cv2.integral(result.astype(np.uint8))
```

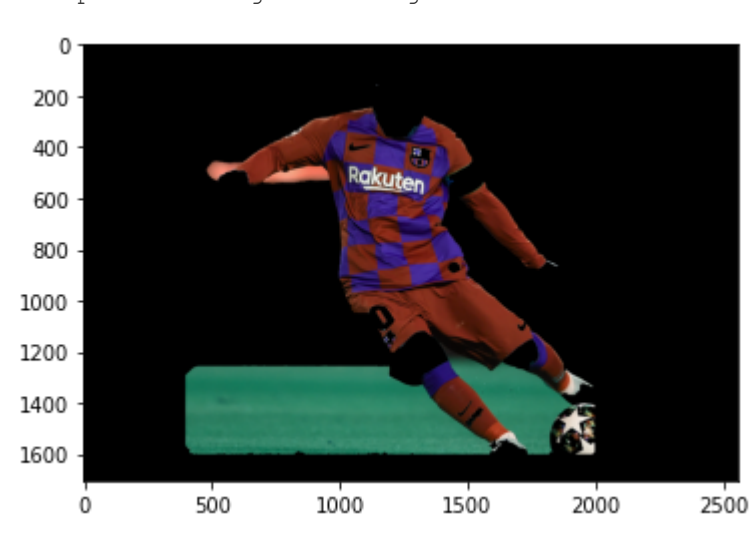
In [128...
```python
plt.imshow(rotate_image('Resources\\bicycle.jpg', np.pi/4))
```

Out[128...: <matplotlib.image.AxesImage at 0x20237c1f8c8>



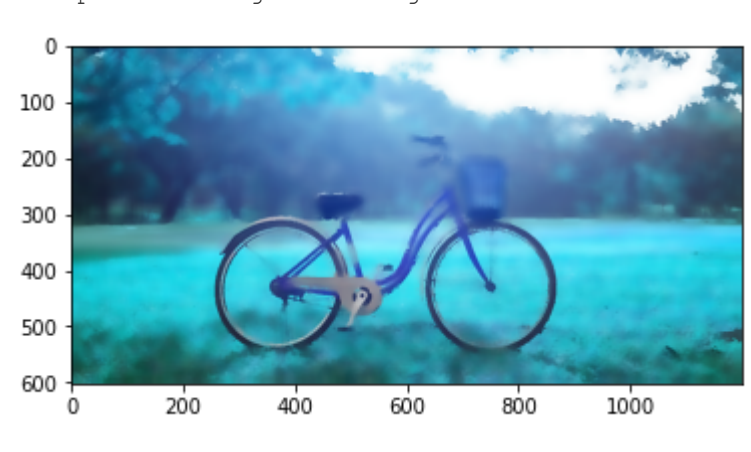
In [148...
```python
# 8
img = cv2.imread('Resources\\messi.jpg')
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (400, 0,1600,1600)
cv2.grabCut(img,mask,rect,bgdModel,fgdModel,2,cv2.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis] # === imapint INPAINT_NS
plt.imshow(img)
```

Out[148...: <matplotlib.image.AxesImage at 0x20234daa388>



In [159...
```python
# 10
img = cv2.imread('Resources\\bicycle.jpg')
dst = cv2.pyrMeanShiftFiltering(img, 10, 100)
plt.imshow(dst)
```

Out[159...: <matplotlib.image.AxesImage at 0x202365eb9c8>



In [160...
```python
gray = cv2.cvtColor(dst, cv2.COLOR_BGR2GRAY)

thresh = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY_INV,3,1)

contours, hierarchy = cv2.findContours(thresh,cv2.RETR_CCOMP,cv2.CHAIN_APPROX_NONE)

mask = np.zeros(img.shape[:-1],np.uint8)

cv2.drawContours(mask,contours,-1,(255,255,255),-1)

height, width = img.shape[:-1]

mask1 = np.zeros((height+2, width+2), np.uint8)     # line 26
cv2.floodFill(mask,mask1,(0,0),255)     # line 27
mask_inv=cv2.bitwise_not(mask)
plt.imshow(mask_inv)
```

Out[160...: <matplotlib.image.AxesImage at 0x20234e84d48>

In [ ]: