KhuatDangSon_20002159_Lab2_1

September 26, 2023

1 Drawing with opencv

1.1 Activity 1: Import Library

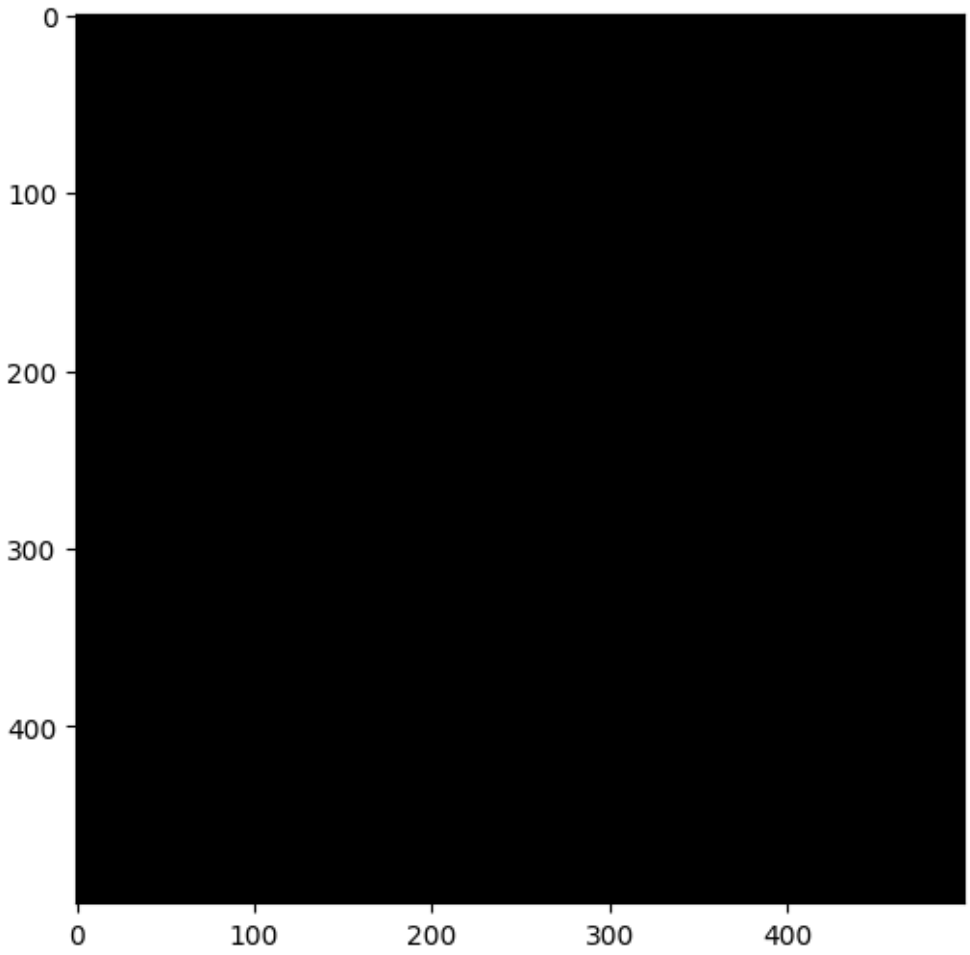
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

1.2 Activity 2: Shape Drawing

Initial Image

[2]:
```
# Initial Image
img = np.zeros([500, 500, 3])
plt.figure(figsize=(6, 6))
plt.imshow(img)
```

[2]:
```
<matplotlib.image.AxesImage at 0x7d50acc0b400>
```

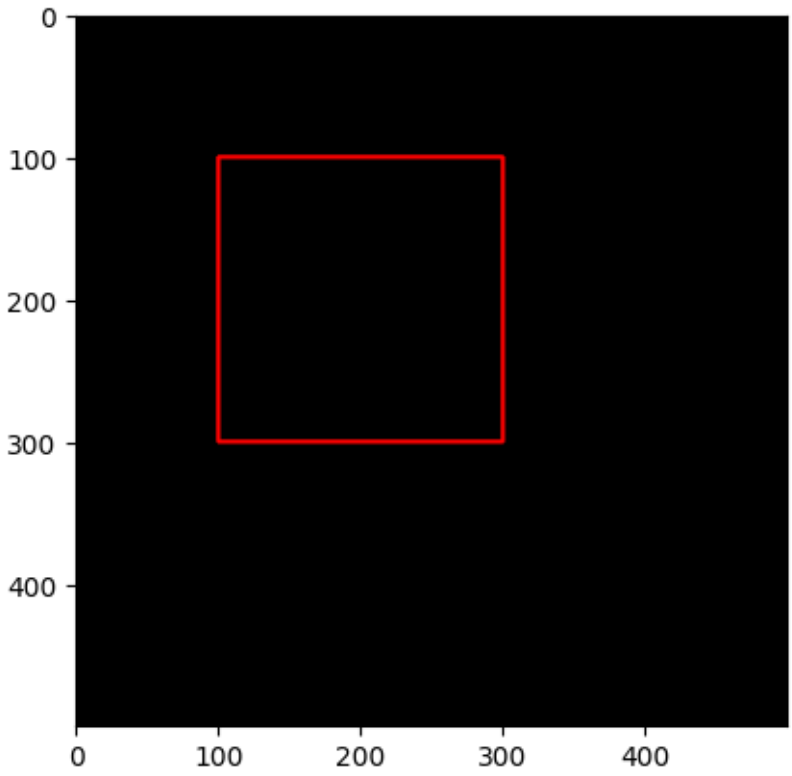


Draw rectangle

[3]:
```
# Draw a rectangle
cp = img.copy()
rectang = cv2.rectangle(cp, (100, 100), (300, 300), (255, 0, 0), 2)
plt.imshow(rectang)
```

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

[3]:
```
<matplotlib.image.AxesImage at 0x7d50ac90b490>
```

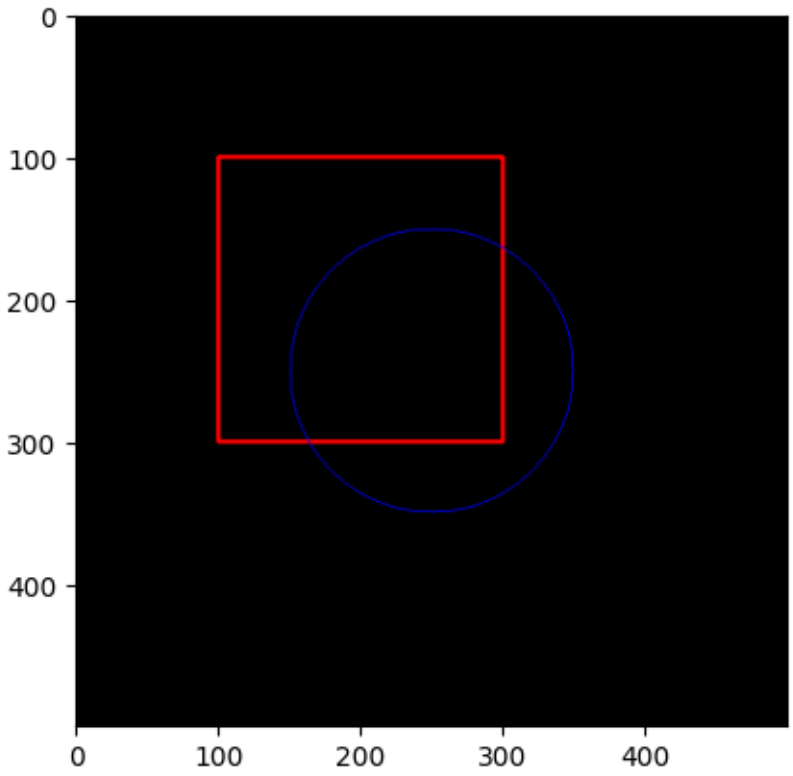


Draw Circle

[4]:
```
cir = cv2.circle(cp, (250, 250), 100, (0, 0, 255), 1)
plt.imshow(cir)
```

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

[4]: <matplotlib.image.AxesImage at 0x7d50ac71d3c0>

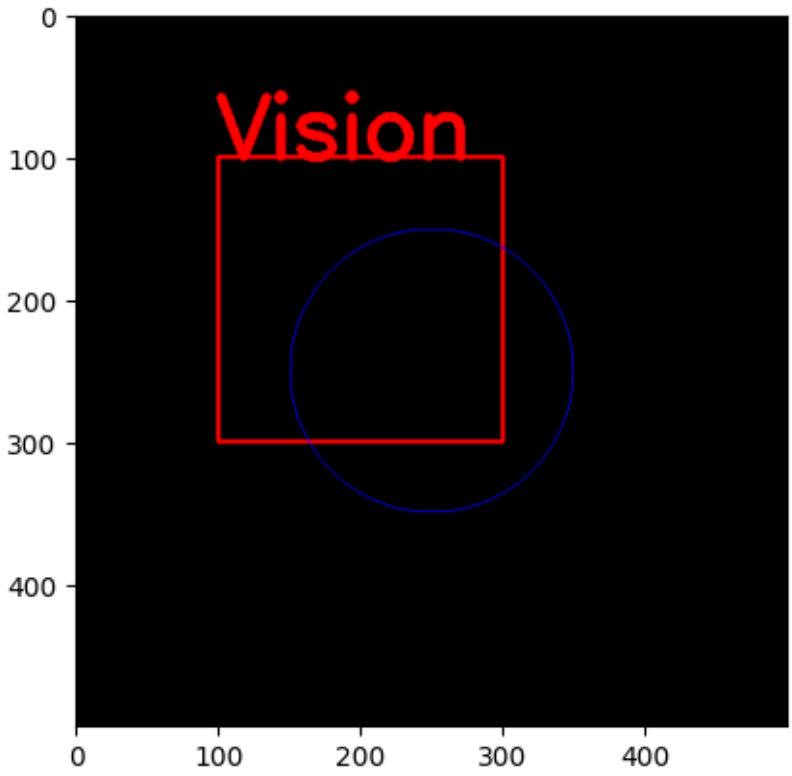


Insert Text

[5]:
```
font = cv2.FONT_HERSHEY_SIMPLEX
new_img = cv2.putText(cp, 'Vision', (100, 100), font, 2, (255, 0, 0), 5, cv2.
↪LINE_AA)
plt.imshow(new_img)
```

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

[5]:
```
<matplotlib.image.AxesImage at 0x7d50ac7dffd0>
```

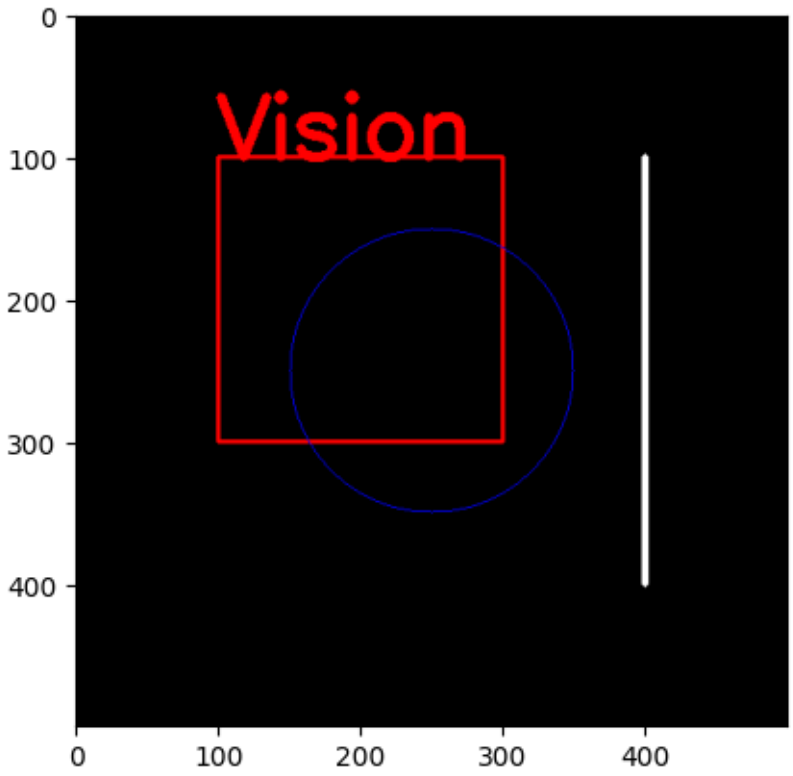


Draw Line

[6]:
```
line = cv2.line(cp, (400, 100), (400, 400), (255, 255, 255), 3)
plt.imshow(line)
```

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

[6]:
```
<matplotlib.image.AxesImage at 0x7d50ac4b2d10>
```



Draw Polygon

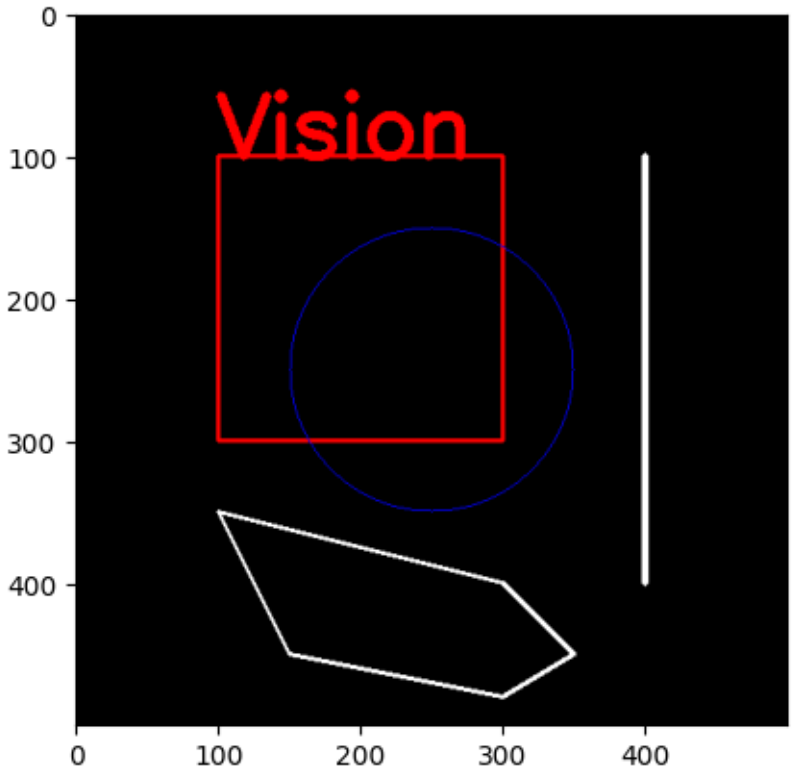
[7]:
```
pts = np.array([[100, 350], [300, 400],
                [350, 450], [300, 480],
                [150, 450], [100, 350]],
               np.int32)

pts = pts.reshape((-1, 1, 2))
print(pts)

a = cv2.polylines(cp, [pts], True, (125, 125, 125), 2)

plt.imshow(a)
```

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

```
[[[100 350]]

 [[300 400]]

 [[350 450]]
```

```
[[300 480]]

[[150 450]]

[[100 350]]]
```

[7]: <matplotlib.image.AxesImage at 0x7d50ac8c3f70>



1.3 Activity 3: Image Interactive

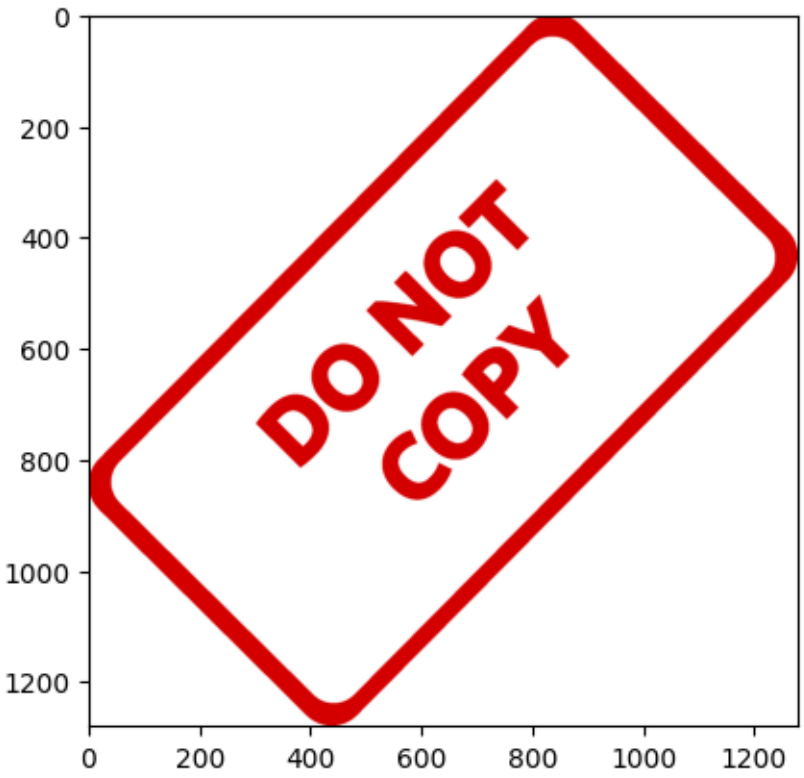
Add Image

[8]:
```
img1 = cv2.imread('car.jpg')
# Convert image to rgb
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
# Show image
plt.imshow(img1)
```

[8]: <matplotlib.image.AxesImage at 0x7d50ac6b7bb0>

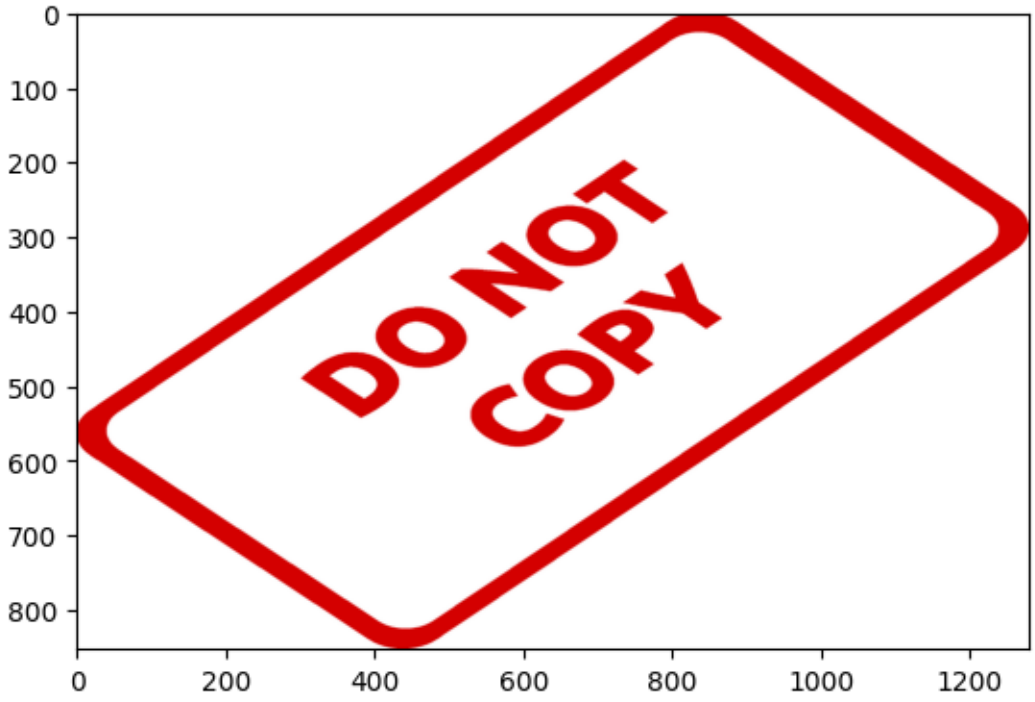
[9]:
```
img2 = cv2.imread('watermark_no_copy.png')
# Convert image to rgb
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
# Show image
plt.imshow(img2)
```

[9]: <matplotlib.image.AxesImage at 0x7d50ac269a50>



[10]:
```
img2_resized = cv2.resize(img2, (1280, 853))
plt.imshow(img2_resized)
```

[10]: <matplotlib.image.AxesImage at 0x7d50ac2aab30>



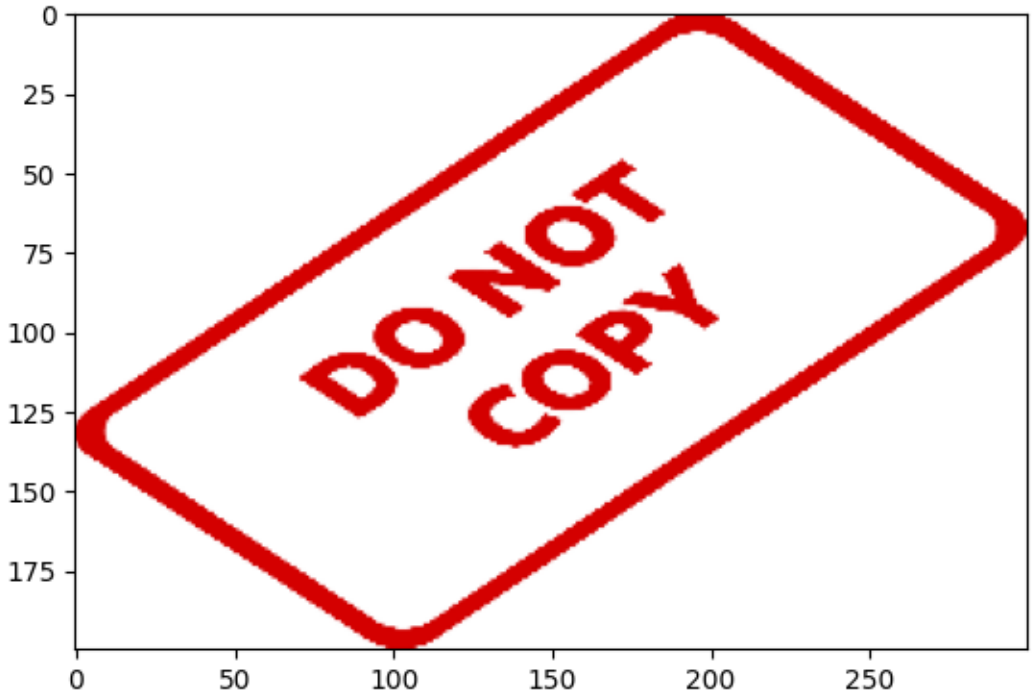
[11]:
```
dst = cv2.addWeighted(img1, 0.5, img2_resized, 0.5, 0.0)
plt.imshow(dst)
```

[11]: <matplotlib.image.AxesImage at 0x7d509e114640>

Paste Image

[13]:
```
small_img = cv2.resize(img2, (300, 200))
plt.imshow(small_img)
```

[13]:
```
<matplotlib.image.AxesImage at 0x7d50a33deb60>
```



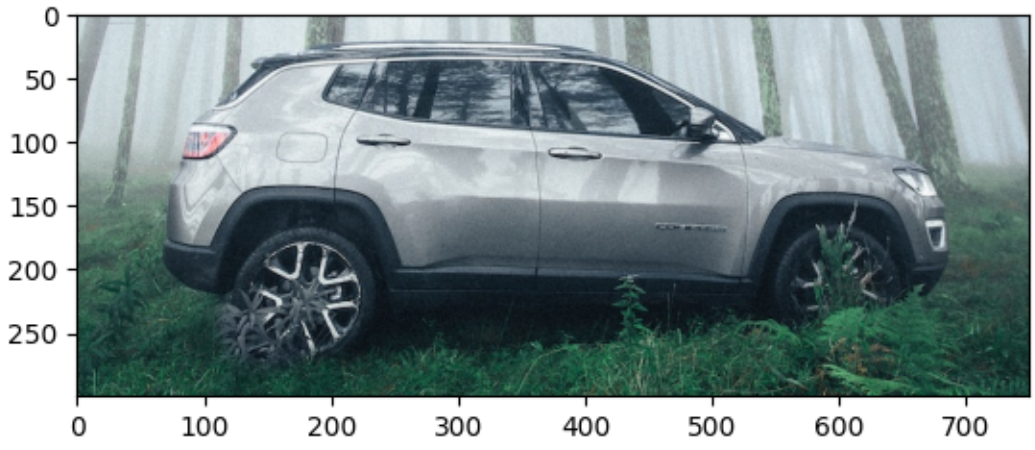
[14]:
```
img1_cp = img1.copy()
img1_cp[0:small_img.shape[0], 0:small_img.shape[1]] = small_img
plt.imshow(img1_cp)
```

[14]: <matplotlib.image.AxesImage at 0x7d508ed9fa00>

Image Clip

[16]:
```
roi = img1[400:700, 250:1000]
plt.imshow(roi)
```

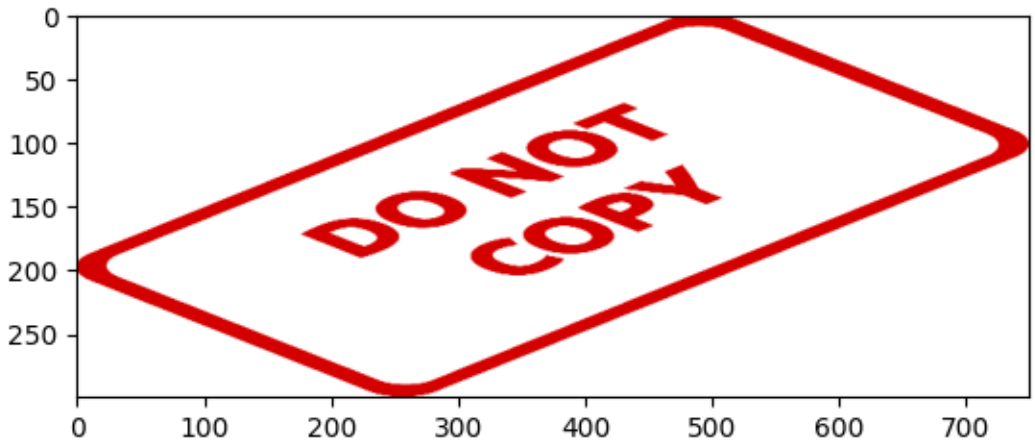
[16]:
```
<matplotlib.image.AxesImage at 0x7d508ec002e0>
```



Mix Image

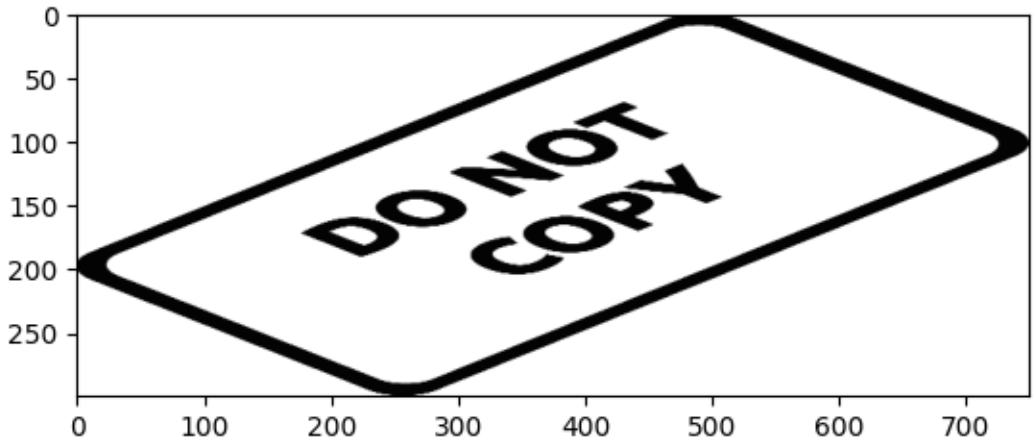
[17]:
```
small_img = cv2.resize(img2, (750, 300))
plt.imshow(small_img)
```

[17]: <matplotlib.image.AxesImage at 0x7d508ec62500>



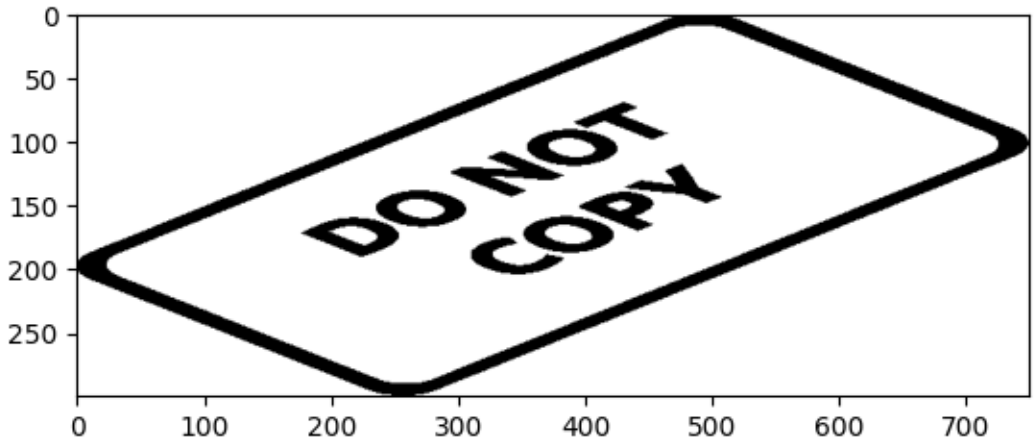
[18]:
```
# Convert to gray
small_img_gray = cv2.cvtColor(small_img, cv2.COLOR_RGB2GRAY)
plt.imshow(small_img_gray, cmap='gray')
```

[18]: <matplotlib.image.AxesImage at 0x7d508eb232b0>



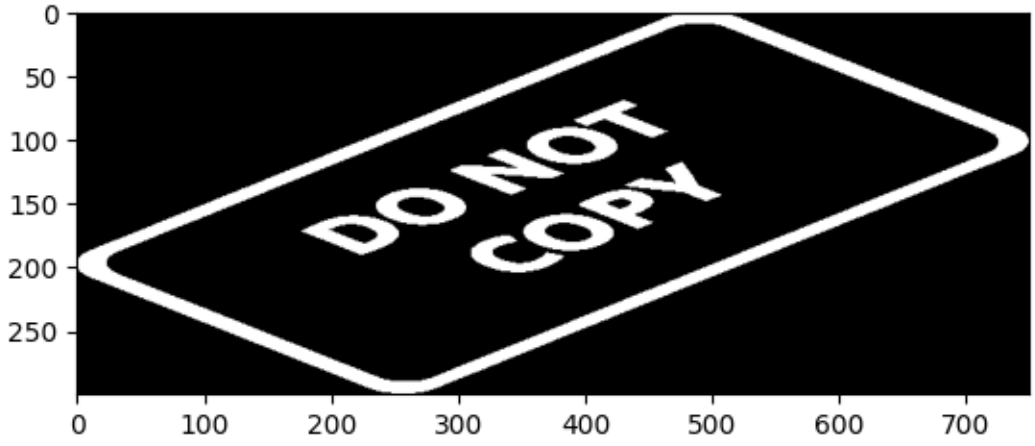
[19]:
```
# Take mask
_, mask = cv2.threshold(small_img_gray, 127, 255, cv2.THRESH_BINARY)
plt.imshow(mask, cmap='gray')
```

[19]: <matplotlib.image.AxesImage at 0x7d508eaebd30>



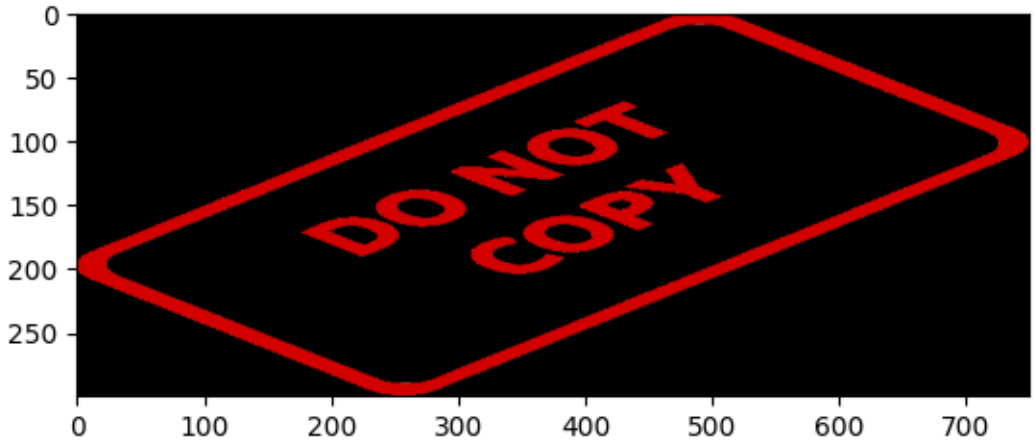
[20]:
```
mask_inv = cv2.bitwise_not(mask)
plt.imshow(mask_inv, cmap='gray')
```

[20]: <matplotlib.image.AxesImage at 0x7d508ea06e60>



[21]:
```
foreground = cv2.bitwise_and(small_img, small_img, mask=mask_inv)
plt.imshow(foreground)
```

[21]: <matplotlib.image.AxesImage at 0x7d508ea59750>



[22]:
```
background = cv2.bitwise_or(roi, roi, mask=mask)
plt.imshow(background)
```

[22]: <matplotlib.image.AxesImage at 0x7d508e7b52d0>

[23]:
```
final_roi = cv2.add(background, foreground)
plt.imshow(final_roi)
```

[23]: <matplotlib.image.AxesImage at 0x7d508e6d7700>

[24]:
```
img1[400:700, 250:1000] = final_roi
plt.imshow(img1)
```

[24]: <matplotlib.image.AxesImage at 0x7d508e749bd0>

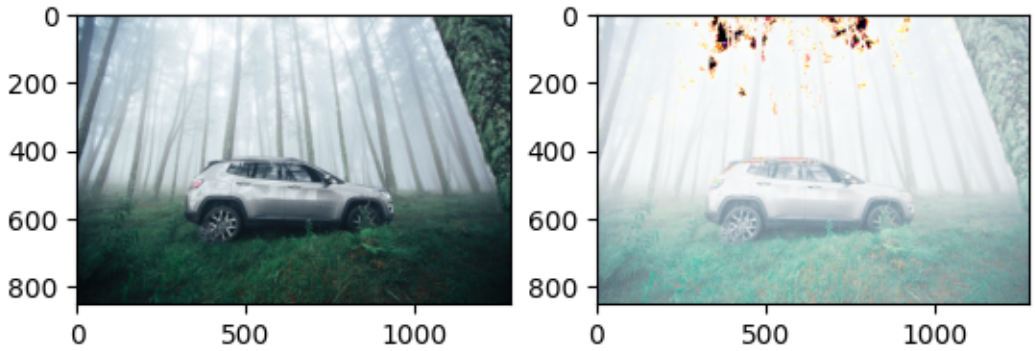
2 Intensity Transformations

2.1 Log Transformations

[25]:
```
# Open image
img = cv2.imread('car.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# Apply log transform
c = 255 / (np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)
# Specify the data type
log_transformed = np.array(log_transformed, dtype=np.uint8)
plt.subplot(1, 2, 1)
plt.imshow(img)
plt.subplot(1, 2, 2)
plt.imshow(log_transformed)
```

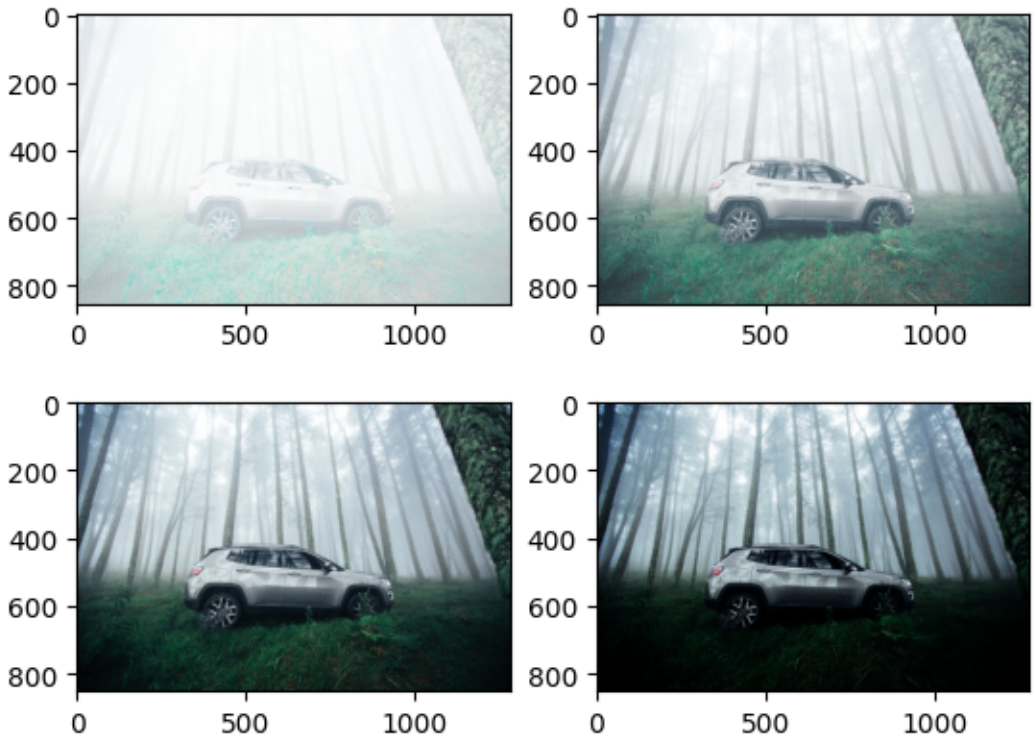
```
<ipython-input-25-8bceccfaef49>:6: RuntimeWarning: divide by zero encountered in
log
  log_transformed = c * np.log(1 + img)
```

[25]:
```
<matplotlib.image.AxesImage at 0x7d508e5af700>
```



2.2 Gamma Transformations

[26]:
```
i = 1
for gamma in [0.1, 0.5, 1.2, 2.2]:
  # Apply gamma correction
  gamma_corrected = np.array(255*(img / 255) ** gamma, dtype='uint8')
  plt.subplot(2, 2, i)
  plt.imshow(gamma_corrected)
  i += 1
```



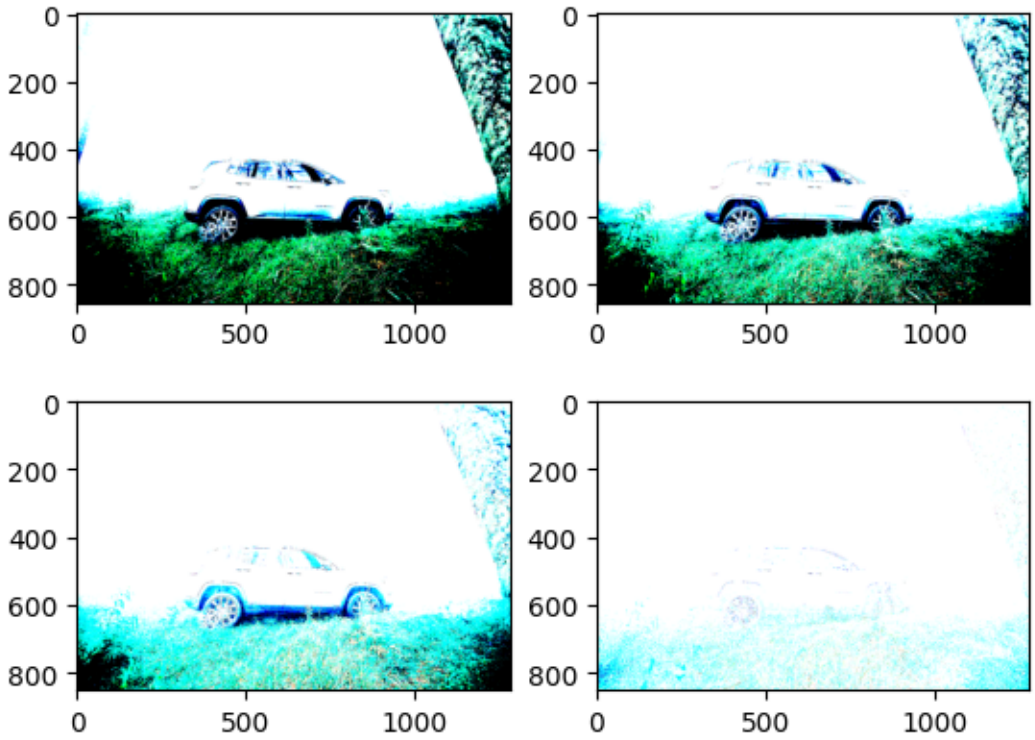
2.3 Piecewise - Linear Transformation

[27]:
```
def pixelVal(pix, r1, s1, r2, s2):
  if (0 <= pix and pix <= r1):
    return (s1 / r1) * pix

  elif (r1 < pix and pix <= r2):
    return ((s2-s1)/(r2-r1)) * (pix-r1) + s1

  else:
    return ((255-s2)/(255-r2)) * (pix-r2) + s2
```

[36]:
```
# Define params
r1 = [70, 50, 30, 10]
s1 = [0, 0, 0, 0]
r2 = [140, 160, 180, 200]
s2 = [255, 255, 255, 255]

# Vectorize the function to apply it to each value in the numpy array
pixelVal_vec = np.vectorize(pixelVal)

for i in range(4):
```

```
  # Apply contrast stretching
  contrast_stretched = pixelVal_vec(img, r1[i], s1[i], r2[i], s2[i])
  plt.subplot(2, 2, i+1)
  plt.imshow(contrast_stretched)
```

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



2.4 Equalize Histogram

[34]:
```
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
hist_eq_gray = cv2.equalizeHist(gray)
hist = cv2.calcHist([gray], channels = [0], histSize=[256], ranges = (0, 255),␣
 ↪mask = None)
```

```
hist2 = cv2.calcHist([hist_eq_gray], channels = [0], histSize=[256], ranges =␣
 ↪(0, 255), mask = None)

plt.subplot(2, 2, 1)
plt.plot(hist)

plt.subplot(2, 2, 2)
plt.plot(hist2)

plt.subplot(2, 2, 3)
plt.imshow(gray, cmap='gray')
plt.subplot(2, 2, 4)
plt.imshow(hist_eq_gray, cmap='gray')
```

[34]: <matplotlib.image.AxesImage at 0x7d5088336fe0>