Computer Vision Assignment 3

Name: Mrinal Misra

Roll: 121CS0132

January 31, 2024

1) By use OpenCV to load an image, implement a mouse click event, and retrieve the coordinate along with the color values of the clicked position on the image

[11]:
```
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
```

[2]:
```
img=cv2.imread("R.png")
```

[3]:
```
cv2.imshow("hi",img)
cv2.waitKey()
cv2.destroyAllWindows()
```

[4]:
```
def mouse_callback(event,x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        pix = img[x, y]
        print(f"Pixel value at ({x},{y}): {pix}")
```

[5]:
```
cv2.namedWindow('Image')
cv2.setMouseCallback('Image', mouse_callback)

cv2.imshow("Image", img)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
Pixel value at (257,185): [61 15 81]
Pixel value at (214,210): [ 90 37 111]
```

2) Read an image with OpenCV and perform drawing operations by using coordinate values, including lines, rectangles, triangle, circle and adding the text “Write your name” in a single operation

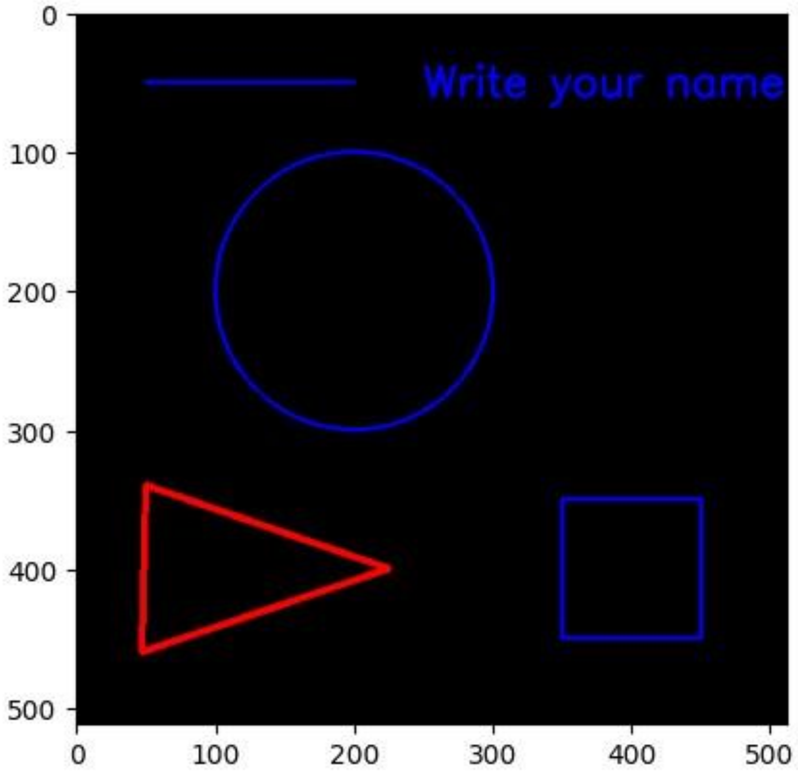
[12]:
```
 img = np.zeros((512,512,3), np.uint8)
cv2.line(img, (50,50), (200,50), (0,0,255), 2)
```

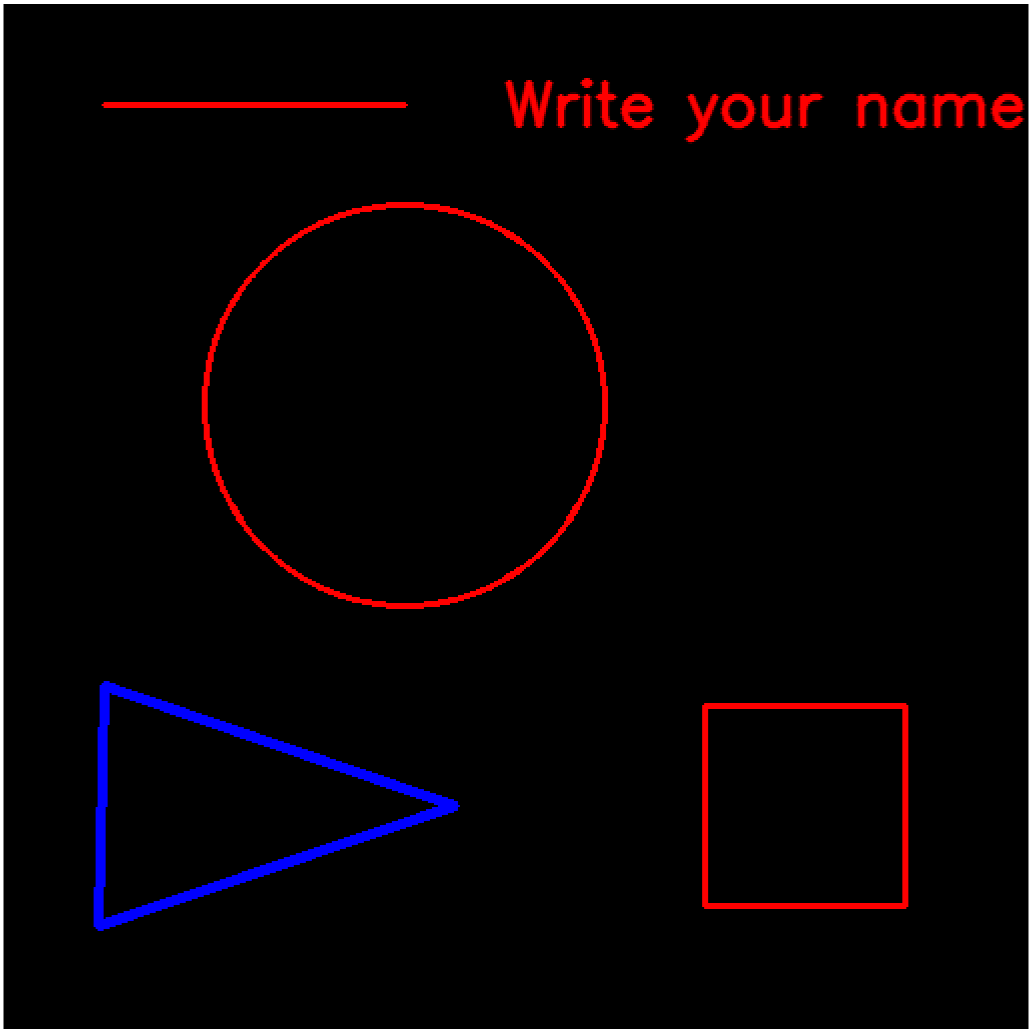
```
cv2.rectangle(img, (350,350), (450,450),
(0,0,255), 2)
```

```
cv2.circle(img, (200,200), 100,  0,0,255), 2)

p1 = (47, 460)
p2 = (50, 340)
p3 = (225, 400)

cv2.line(img, p1, p2, (255, 0, 0), 3)
cv2.line(img, p2, p3, (255, 0, 0), 3)
cv2.line(img, p1, p3, (255, 0, 0), 3)

image = cv2.putText(img, 'Write your name',  250,60), cv2.FONT_HERSHEY_SIMPLEX␣
 ↪, 1,  0,0,255), 2, cv2.LINE_AA)
cv2.imshow("Image", img)
cv2.waitKey()
cv2.destroyAllWindows()
```

[13]:

```
plt.imshow(img)
plt.show()
```





3) By utilize OpenCV to perform various geometric transformations such as

a)Image scaling (use different interpolation like Cubic, Linear, Nearest-neighbor, Area and sinusodial) b) Rotation

[14]:

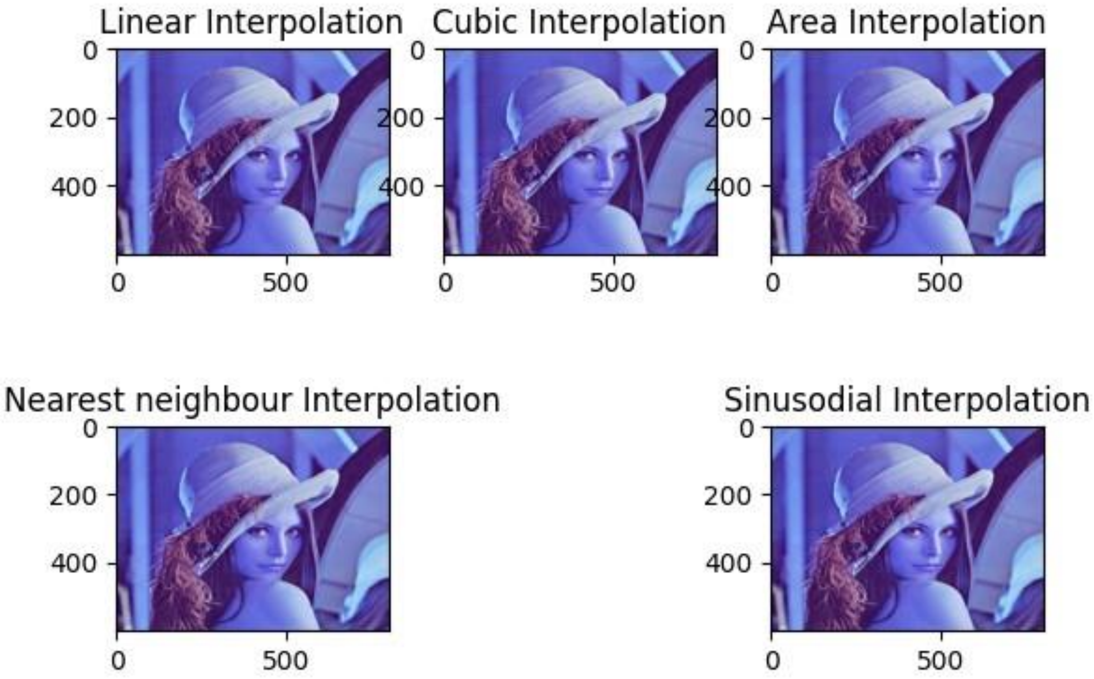
```
img=cv2.imread("R.png")
```

[15]:

```
linear = cv2.resize(img, (800, 600), interpolation = cv2.INTER_LINEAR)
cubic = cv2.resize(img, (800, 600), interpolation = cv2.INTER_CUBIC)
area = cv2.resize(img, (800, 600), interpolation = cv2.INTER_AREA)
nearest = cv2.resize(img, (800, 600), interpolation = cv2.INTER_NEAREST_EXACT)
sinusodial = cv2.resize(img, (800, 600), interpolation = cv2.INTER_LANCZOS4)
cv2.imshow("Linear", linear)
cv2.imshow("cubic", cubic)
cv2.imshow("area", area)
cv2.imshow("nearest", nearest)
cv2.imshow("sinusodial", sinusodial)
cv2.waitKey()
cv2.destroyAllWindows()
```

[19]:

```
plt.subplot(231)
plt.imshow(linear)
plt.title('Linear Interpolation')

plt.subplot(232)
plt.imshow(cubic)
plt.title('Cubic Interpolation')

plt.subplot(233)
plt.imshow(area)
plt.title('Area Interpolation')

plt.subplot(234)
plt.imshow(nearest)
plt.title('Nearest neighbour Interpolation')

plt.subplot(236)
plt.imshow(sinusodial)
plt.title('Sinusodial Interpolation')

plt.show()
```



Linear Interpolation

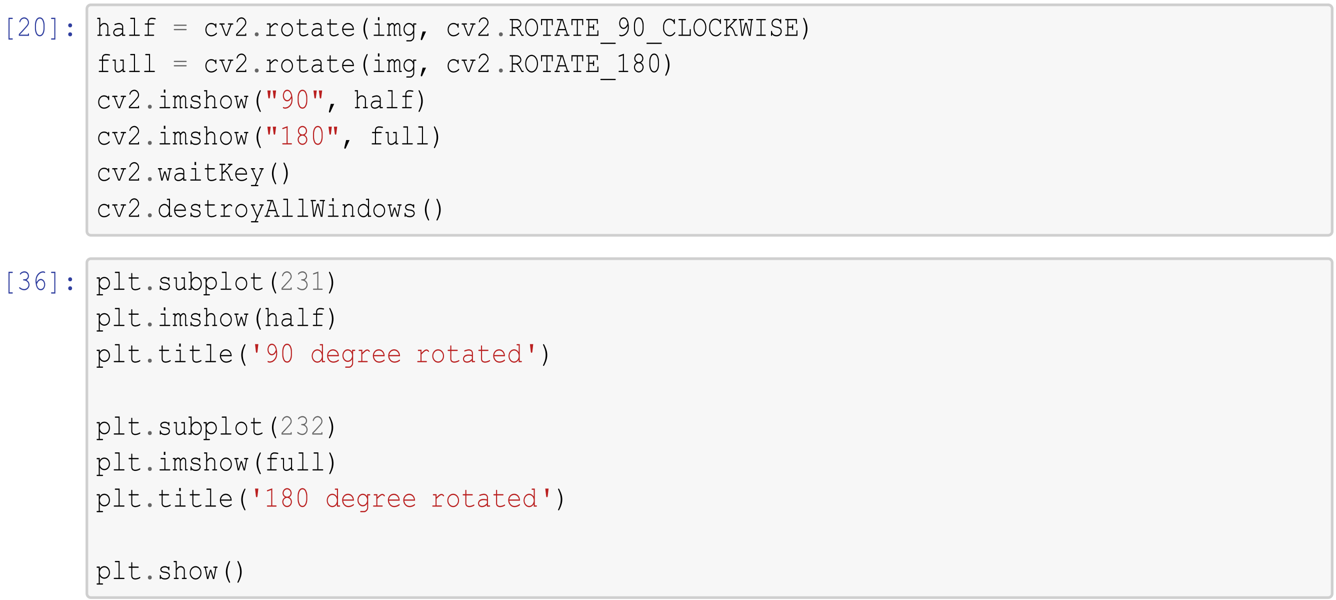
Area Interpolation

Cubic Interpolation

Nearest Interpolation

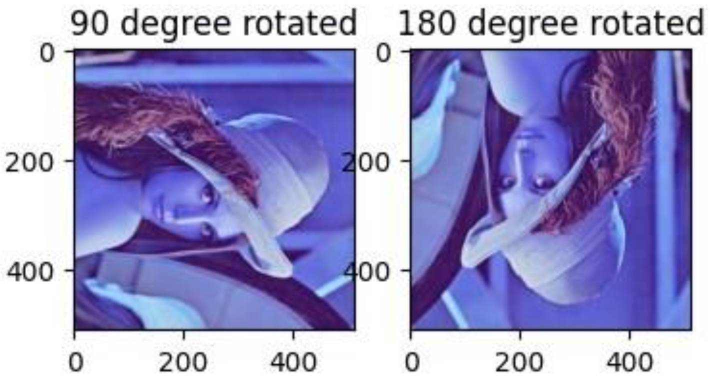
Sinusodial Interpolation

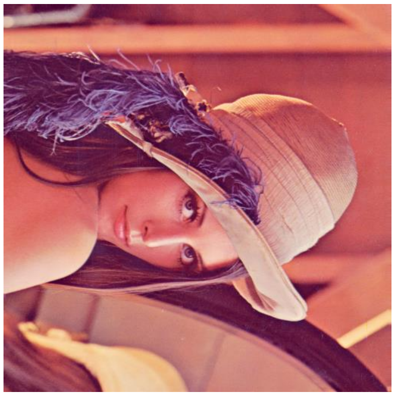
Usung In-built function to rotate



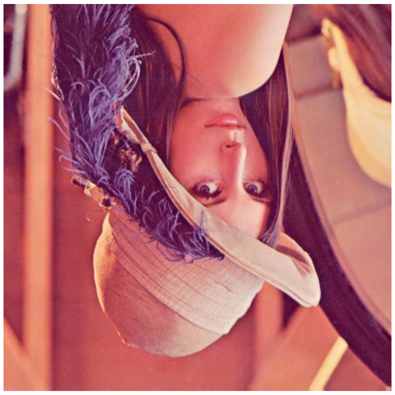
```
[20]: half = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
      full = cv2.rotate(img, cv2.ROTATE_180)
      cv2.imshow("90", half)
      cv2.imshow("180", full)
      cv2.waitKey()
      cv2.destroyAllWindows()
```

```
[36]: plt.subplot(231)
      plt.imshow(half)
      plt.title('90 degree rotated')

      plt.subplot(232)
      plt.imshow(full)
      plt.title('180 degree rotated')

      plt.show()
```





90 degrees rotated



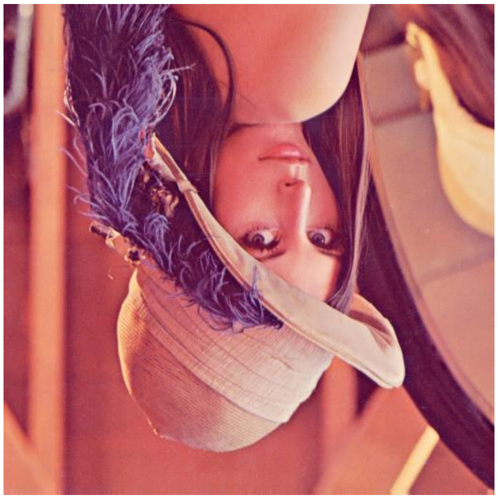
180 degrees rotated

```
[37]: img=cv2.imread("R.png")
```

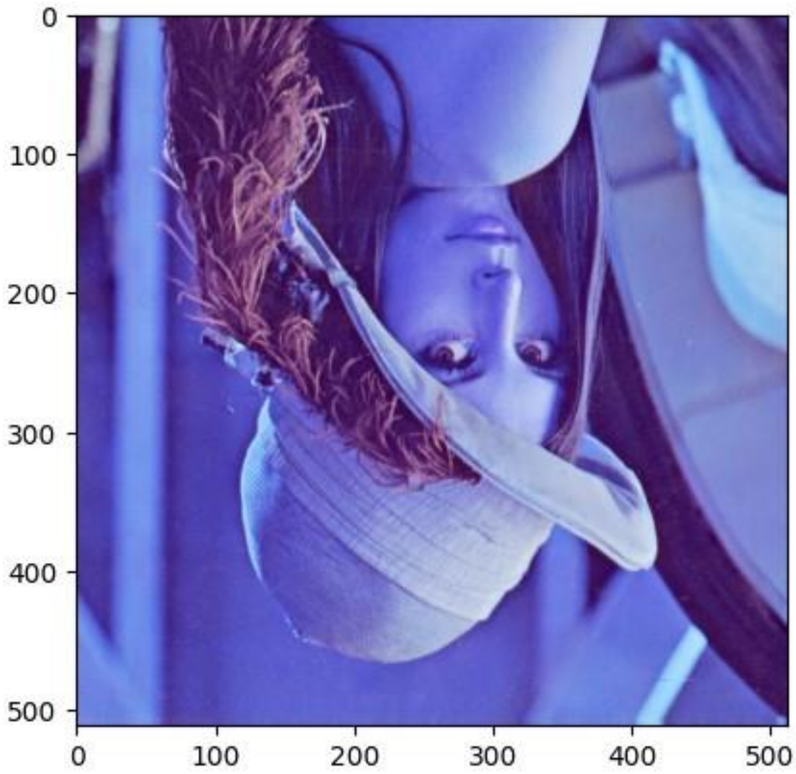
Creating own logic for rotating image without using inbuilt function

```
[40]: l,r,h = img.shape
      flip=np.empty_like(img)
      for i in range (l):
         flip[i,:,:]=img[l-1-i,:,:]

      cv2.imshow("rotated",flip)
      cv2.waitKey()
      cv2.destroyAllWindows()
```

```
[41]: plt.imshow(flip)
      plt.show()
```



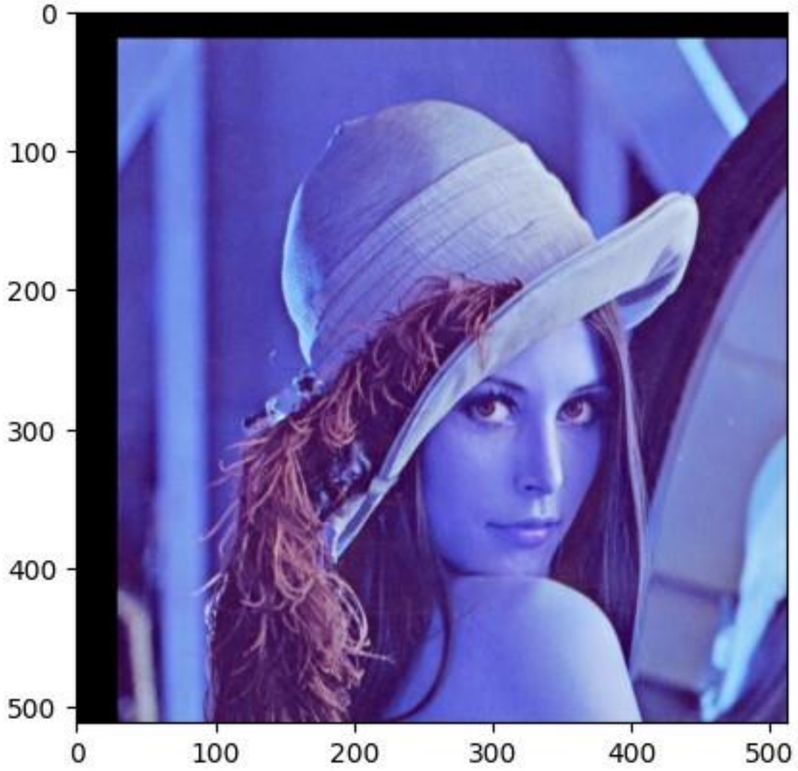
180 degrees rotated



4) Write code using OpenCV to read an image and apply an affine transformation with a translation of 20 pixels in the x-axis and 30 pixels in the y-axis. Display both the original and transformed images.

[42]:
```
l,r,h = img.shape
flip_new = np.zeros((l,r,3), np.uint8)
for i in range (0,l):
    for j in range (0,r):
        if(i+20<l and j+30<r):
            flip_new[i+20][j+30]=img[i][j]
cv2.imshow("translated",flip_new)
cv2.imshow("original",img)
cv2.waitKey()
cv2.destroyAllWindows()
```

[44]:
```
plt.imshow(flip_new)
plt.show()
```



Original Image

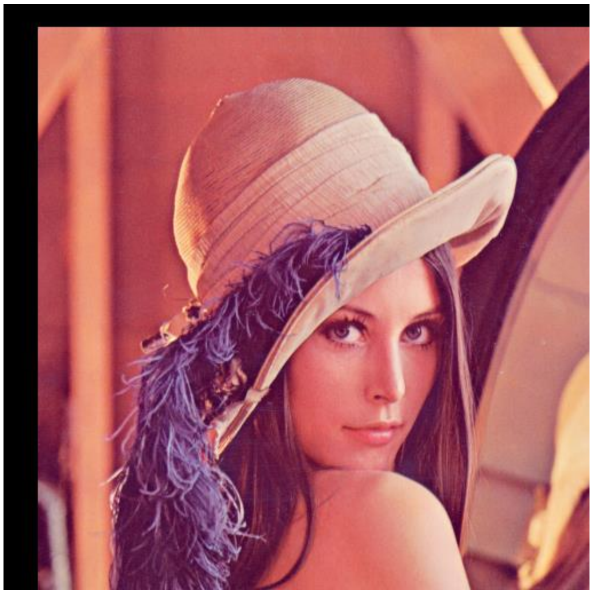


Image after translation(affine transformation)

5) Create a program that reads an image and applies a Motion blur to it using the filter shown in the image below. Display both the original image and the blurred image.

[45]:

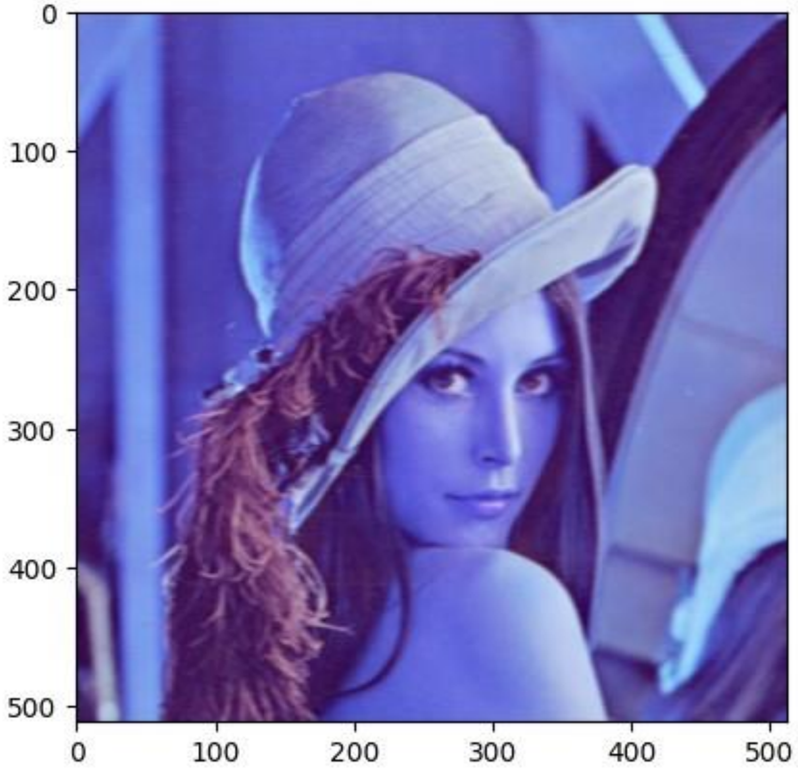
```
img = cv2.imread('R.png')
blur_mat = np.array([[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0],
[1, 1, 1, 1, 1],
[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0]])
blur_mat = blur_mat/5
blur_img = cv2.filter2D(img, -1, blur_mat)
cv2.imshow("original",img)
cv2.imshow("Blurred",blur_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite("blurred.png",blur_img)
```

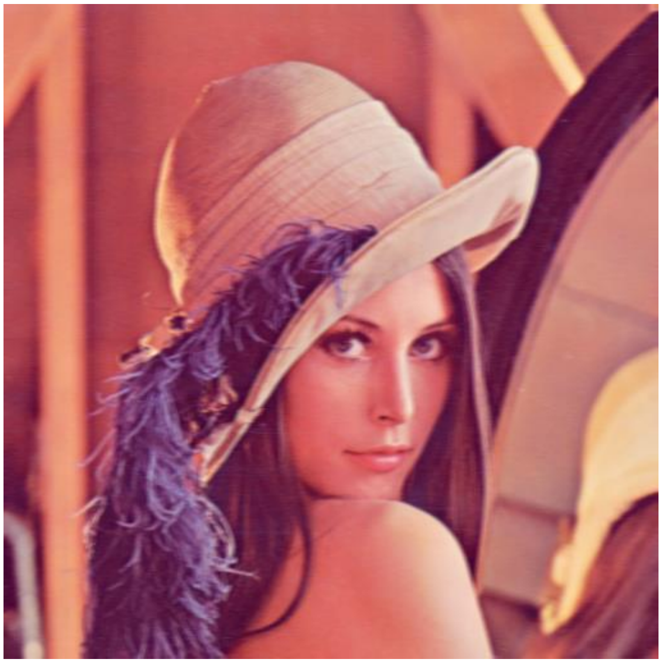
[45]:

```
True
```

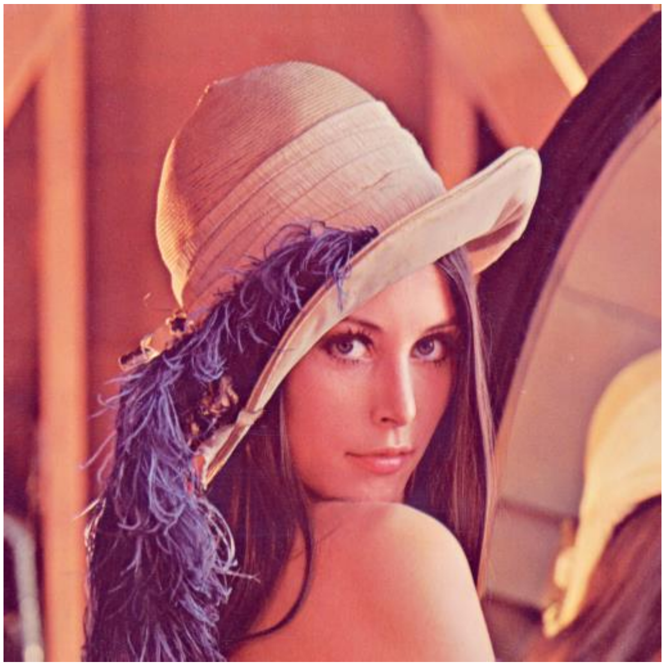
[46]:

```
plt.imshow(blur_img)
plt.show()
```





Blurred Image



Actual Image

Thank You!