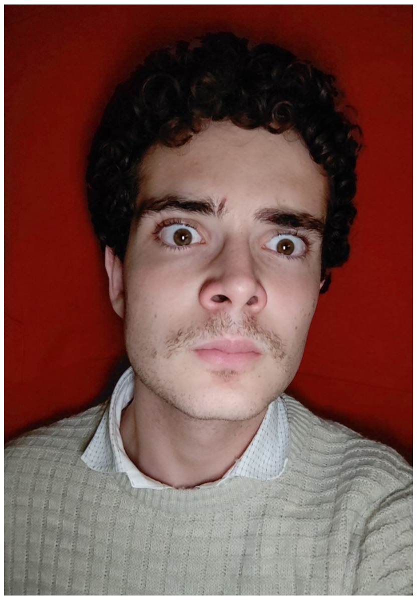
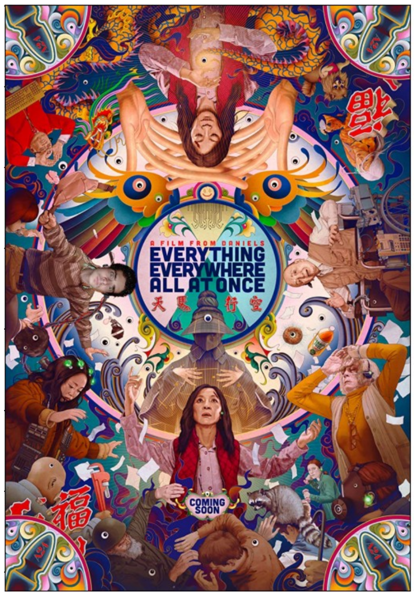
HOMEWORK SET 5 – Víctor Mira Ramírez

Exercise 1

To solve this exercise, I took a picture of myself ensuring that the image size was the same as the poster you gave us, although I could have just trimmed it on the program. The picture was taken wiith the best chroma that I could get, a red jacket hanged from the bathroom curtain bar. It did more or less the job and let me key out the background. My clothes were more of an issue. Then I translated and rotated my keyed image into position, got a negative of the mask and pass the poster through it. Finally added both of them et voilà.



A FILM FROM DANIELS
EVERYTHING EVERYWHERE ALL AT ONCE
COMING SOON



The step that could be improved the most is the masking. Although more work could be done in the software side, probably the easiest solution would be to take a better picture with a better chroma in a better lit scenario that is not a bathroom with dim lights.

Code:

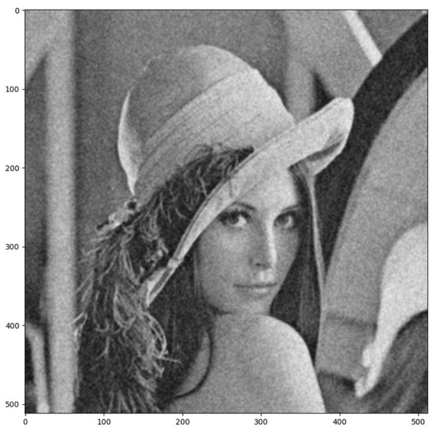
```
path=r'/home/victor/fisicaua/tercero/SIUE/robotic_vision/entregas/hw5/victor.jpeg'

path2 = r'/home/victor/fisicaua/tercero/SIUE/robotic_vision/entregas/hw5/poster.jpg'

# READ victor
img_victor = cv2.imread(path, cv2.IMREAD_COLOR)
victor = cv2.cvtColor(img_victor, cv2.COLOR_BGR2RGB)

# READ poster
img_poster = cv2.imread(path2, cv2.IMREAD_COLOR)
poster = cv2.cvtColor(img_poster, cv2.COLOR_BGR2RGB)

# img size
h, w = img_victor.shape[:2]

# HSV MASKING victor
HSV_victor = cv2.cvtColor(img_victor, cv2.COLOR_BGR2HSV)
mask5 = cv2.inRange(HSV_victor, (0,0,10), (3,200,255))
mask0 = cv2.inRange(HSV_victor, (4,0,50), (4,220,255))
mask1 = cv2.inRange(HSV_victor, (5,0,10), (10,200,255))
mask2 = cv2.inRange(HSV_victor, (11,21,0), (110,200,255))
mask3 = cv2.inRange(HSV_victor, (111,0,0), (210,200,255))
mask4 = cv2.inRange(HSV_victor, (0,0,0), (0,0,8))
mask = mask0 + mask1 + mask2 + mask3 + mask4 + mask5
masked_victor = cv2.bitwise_and(HSV_victor, HSV_victor, mask = mask)


# RESIZING victor
resized_victor = cv2.resize(masked_victor, (int(w/8), int(h/8)))
resized_mask = cv2.resize(mask, (int(w/8), int(h/8)))
# TRANSLATING victor
translation_matrix = np.float32([[1,0,390], [0,1,890]])
translated_victor = cv2.warpAffine(resized_victor, translation_matrix, (w, h))
translated_mask = cv2.warpAffine(resized_mask, translation_matrix, (w, h))
# ROTATING victor
rotation_matrix = cv2.getRotationMatrix2D((w/2,h/2), -90, 1)
rotated_victor = cv2.warpAffine(translated_victor, rotation_matrix, (w, h))
rotated_mask = cv2.warpAffine(translated_mask, rotation_matrix, (w, h))
```

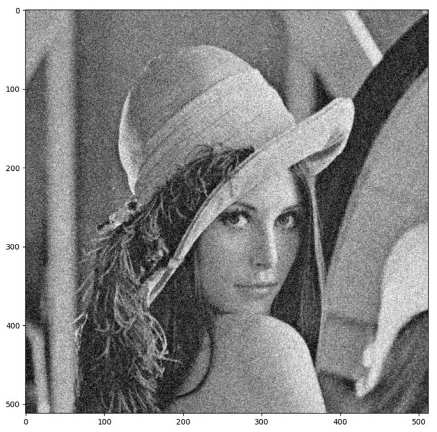
```
# INVERSE MASKING poster
HSV_poster = cv2.cvtColor(img_poster, cv2.COLOR_BGR2HSV)
inv_mask = cv2.bitwise_not(rotated_mask)
masked_poster = cv2.bitwise_and(HSV_poster, HSV_poster, mask = inv_mask)

# ADDING MASKS
result = cv2.addWeighted(rotated_victor, 1, masked_poster, 1, 0)
result = cv2.cvtColor(result,cv2.COLOR_HSV2RGB)

plt.figure()
plt.imshow(result)
```

Exercise 2

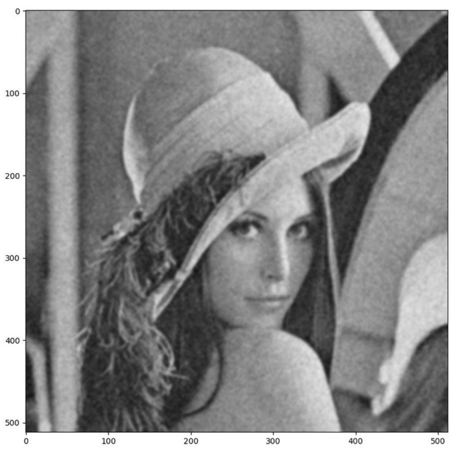
For this exercise, I tried different combinations of kernel sizes and values for sigma (I used every time the same sigma for x and y) and got this results:

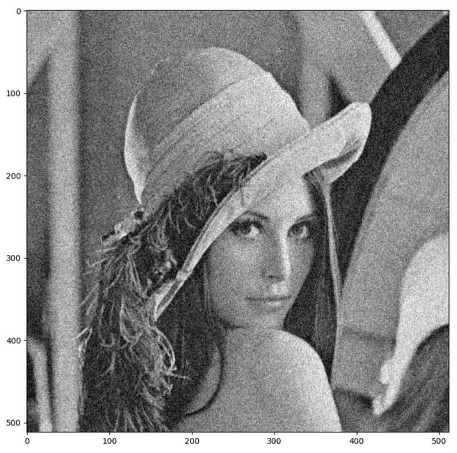
Ksize = (3,3)   Sigma = biggest float64 integer



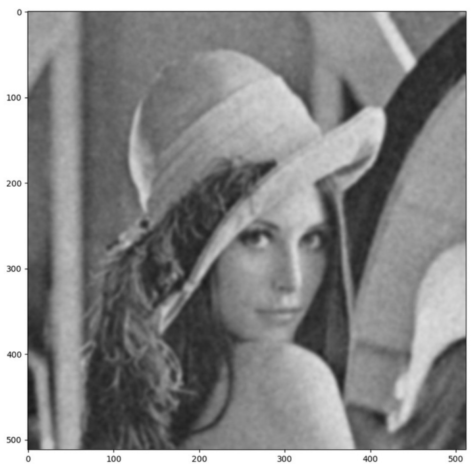


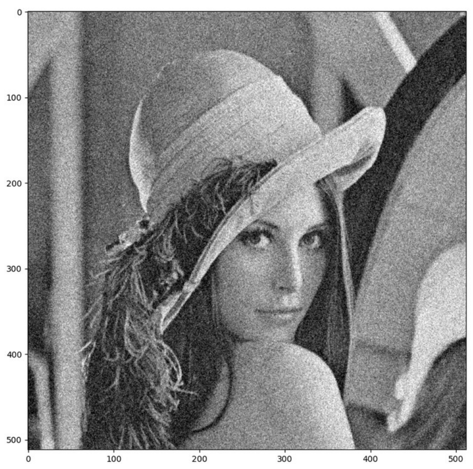
Ksize = (7,7)   Sigma = 1.5





Ksize = (11,11)   Sigma = 2





In all cases, the right picture is the original one for comparison. In my opininon the best solution is the second one, as it balances the compromise of the third one of being to noisy, and the ineffectiveness of the kernel size = 3 while reducing considerably the noise of the image

Code:

```
path=r'/home/victor/fisicaua/tercero/SIUE/robotic_vision/entregas/hw5/lena_noisy.png'

img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
gray = cv2.cvtColor(img,cv2.COLOR_GRAY2RGB) # gray image with vectorsize = 3

gauss = cv2.GaussianBlur(gray,ksize=(7,7),sigmaX=1.5,sigmaY=1.5)

plt.figure()
plt.tight_layout()
plt.imshow(gauss)

plt.figure()
plt.tight_layout()
plt.imshow(gray)

plt.show()
```