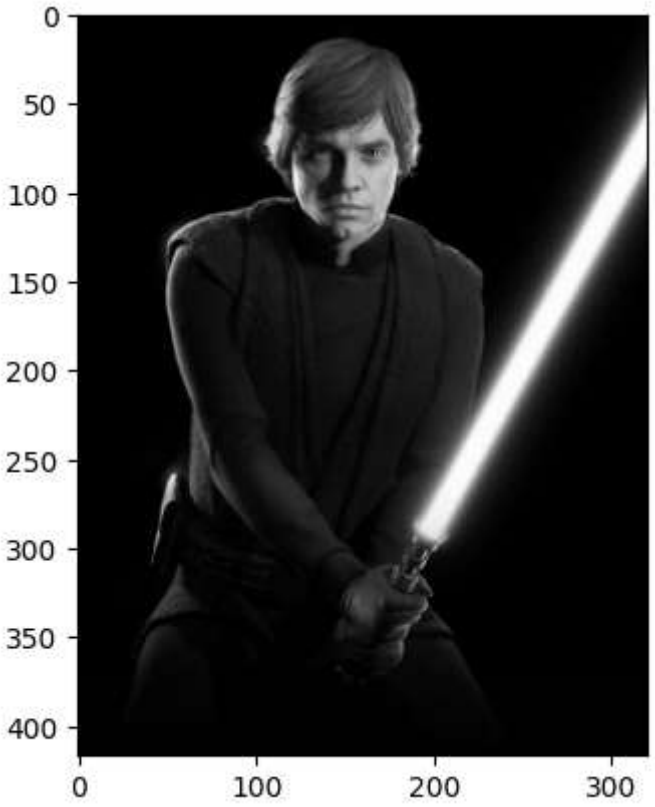
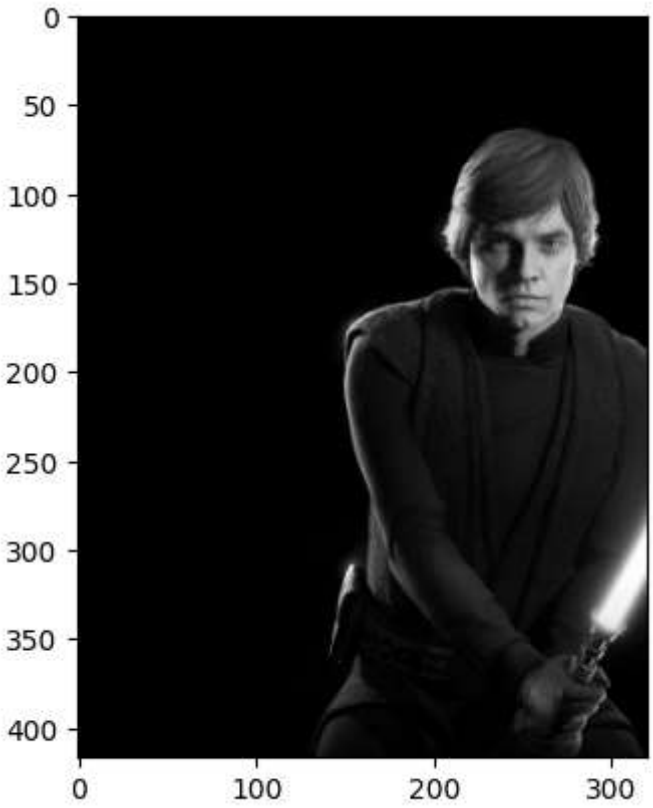
In [ ]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

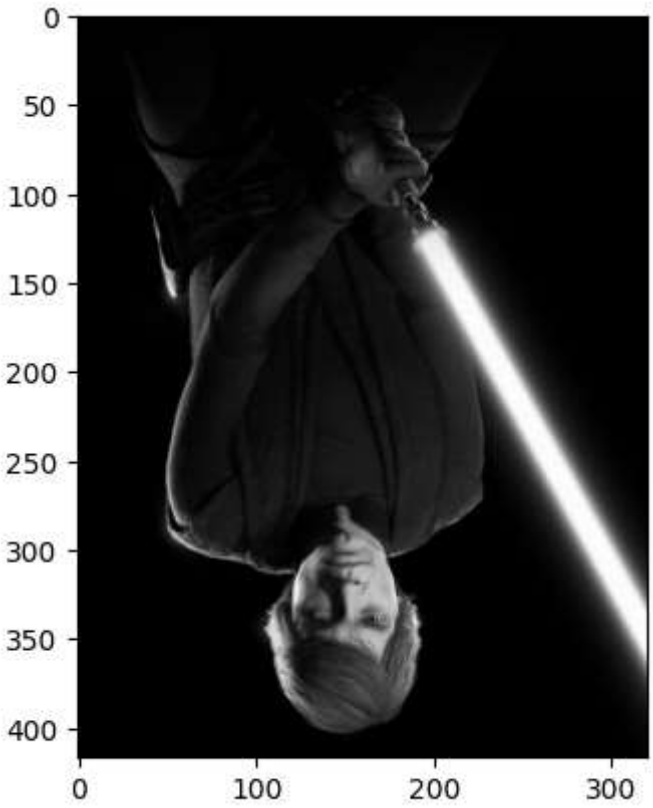
In [ ]:
```
img = cv2.imread('photo.jpg',0) # the '0' means read as grayscale
rows, cols = img.shape
plt.imshow(img, cmap='gray')
plt.show()
```



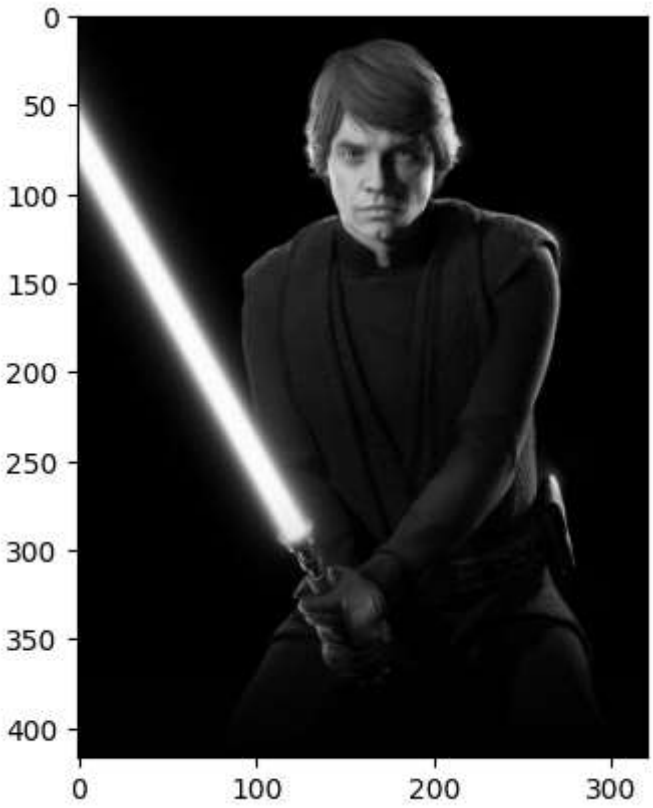
In [ ]:
```
M = np.float32([[1, 0, 100],
                [0, 1, 50]]) # where tx = 100 and ty = 50
translated_image = cv2.warpAffine(img, M, (cols, rows))
plt.imshow(translated_image, cmap='gray')
plt.show()
```



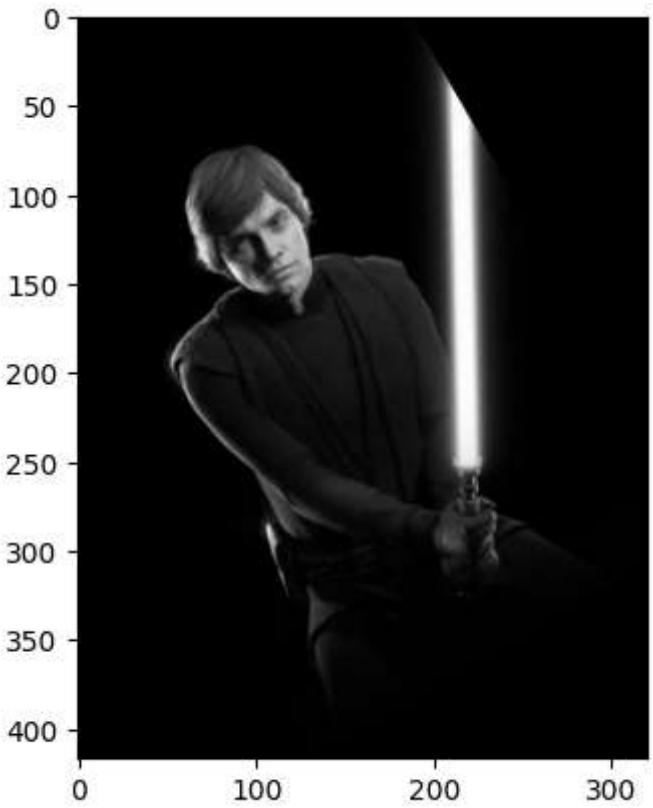
In [ ]:
```
M = np.float32([[1,  0, 0],
                [0, -1, rows], #translated down by 'rows' so we can see it in the d
                [0,  0, 1]])
X_reflected_img = cv2.warpPerspective(img, M,(int(cols),int(rows)))
plt.imshow(X_reflected_img, cmap='gray')
plt.show()
```



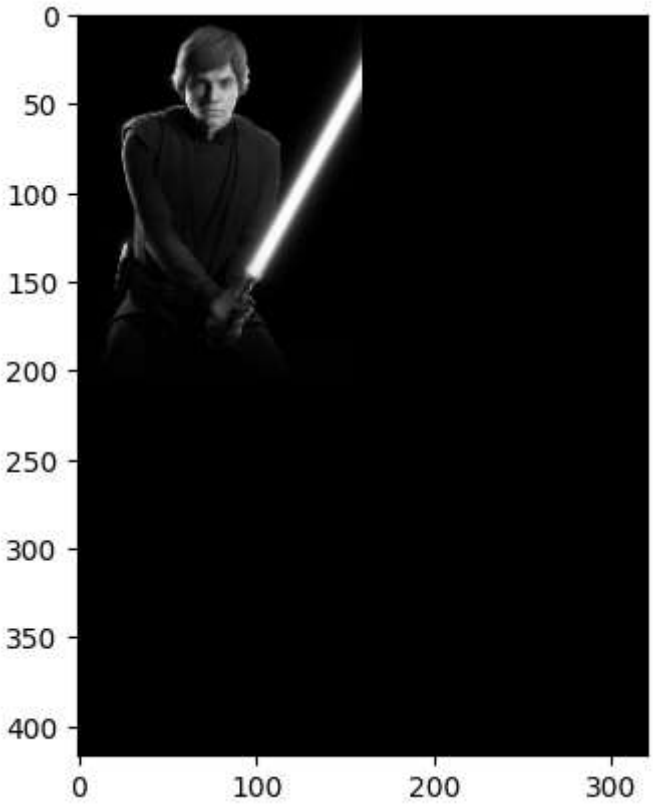
In [ ]:
```
M = np.float32([[-1, 0, cols], #translated right by 'cols' so we can see it in the
                [0, 1, 0],
                [0, 0, 1]])
Y_reflected_img = cv2.warpPerspective(img, M,(int(cols),int(rows)))
plt.imshow(Y_reflected_img, cmap='gray')
plt.show()
```



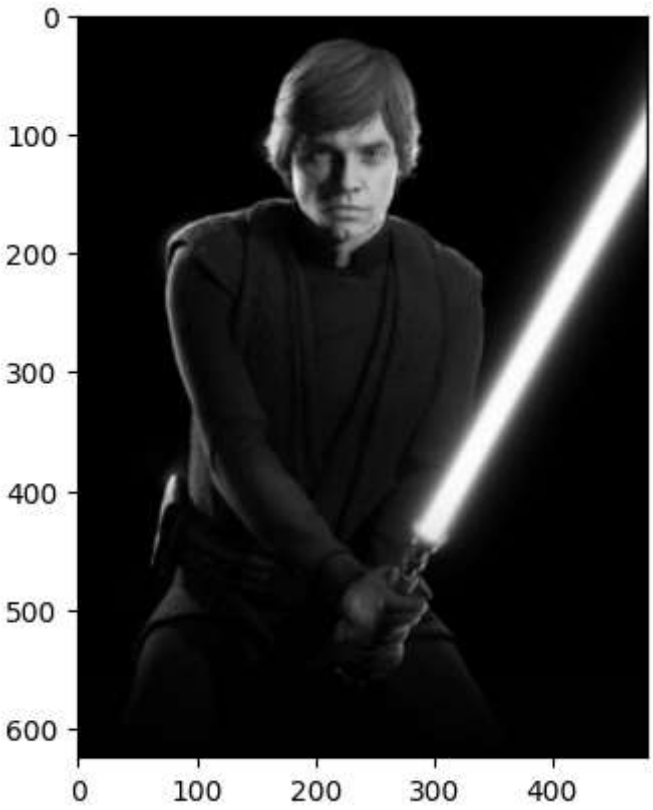
In [ ]:
```
img_rotation = cv2.warpAffine(img,cv2.getRotationMatrix2D((cols/2, rows/2),30,0.8),
plt.imshow(img_rotation, cmap='gray')
plt.show()
```



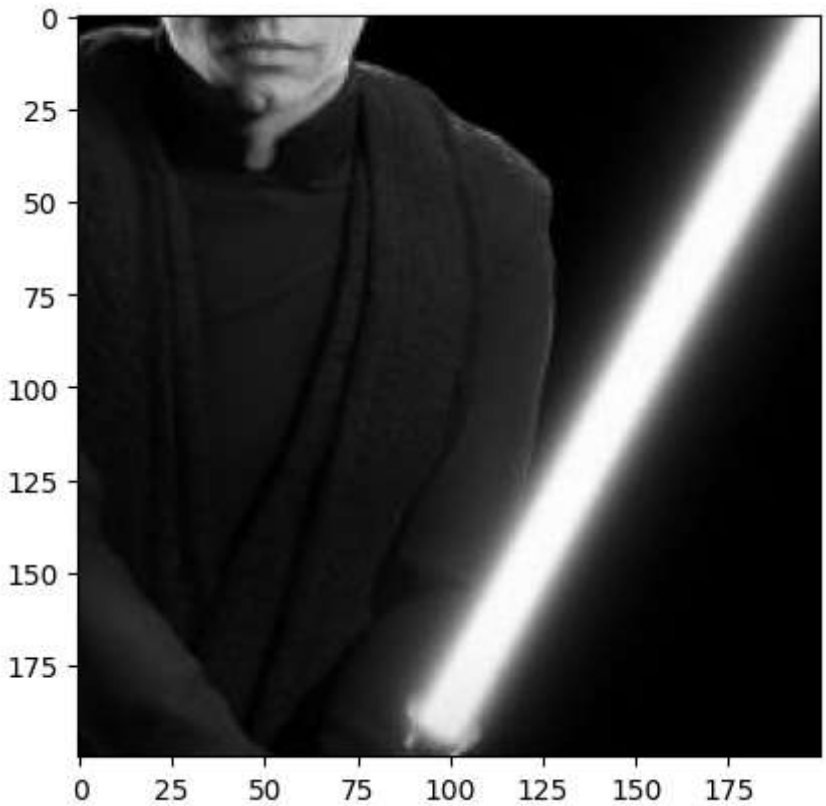
In [ ]:
```
M=np.float32([[0.5,0,0],
              [0,0.5,0]])
img_shrinked = cv2.warpAffine(img,M,(int(cols), int(rows)))
plt.imshow(img_shrinked,cmap='gray')
plt.show()
```



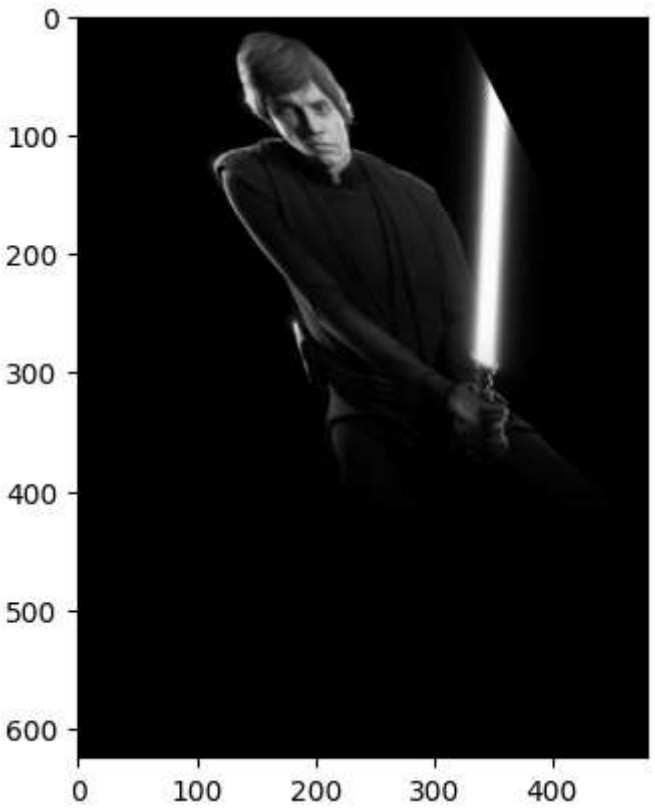
In [ ]:
```
M=np.float32([[1.5,0,0],[0,1.5,0]])
img_enlarged = cv2.warpAffine(img,M,(int(cols*1.5), int(rows*1.5)))
plt.imshow(img_enlarged,cmap='gray')
plt.show()
```



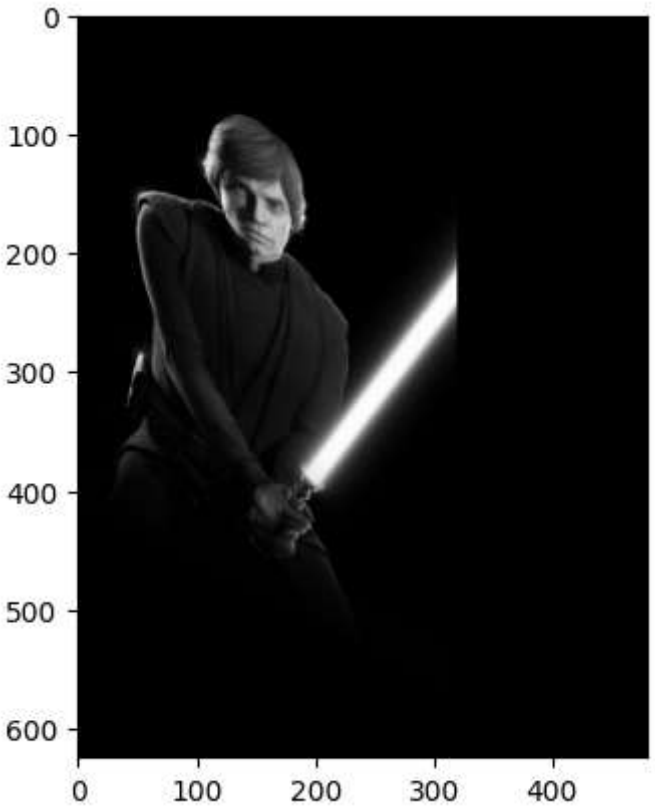
In [ ]:
```
cropped_img = img[100:300, 100:300]
plt.imshow(cropped_img,cmap='gray')
plt.show()
```



In [ ]:
```
M = np.float32([[1, 0.5, 0], [0, 1, 0], [0, 0, 1]])
x_sheared_img = cv2.warpPerspective(img, M, (int(cols*1.5), int(rows*1.5)))
plt.imshow(x_sheared_img,cmap='gray')
plt.show()
```



In [ ]:
```
M = np.float32([[1,   0, 0], [0.5, 1, 0], [0,   0, 1]])
y_sheared_img = cv2.warpPerspective(img, M, (int(cols*1.5), int(rows*1.5)))
plt.imshow(y_sheared_img,cmap='gray')
plt.show()
```



In [ ]:
```
fig, ax = plt.subplots(2,5, figsize=(10, 5))
ax[0][0].imshow(img, cmap='gray')
ax[0][1].imshow(translated_image, cmap='gray')
ax[0][2].imshow(X_reflected_img, cmap='gray')
ax[0][3].imshow(Y_reflected_img, cmap='gray')
ax[0][4].imshow(img_rotation, cmap='gray')
ax[1][0].imshow(img_shrinked, cmap='gray')
ax[1][1].imshow(img_enlarged, cmap='gray')
ax[1][2].imshow(cropped_img, cmap='gray')
ax[1][3].imshow(x_sheared_img, cmap='gray')
ax[1][4].imshow(y_sheared_img, cmap='gray')

ax[0][0].set_title('Original Image')
ax[0][1].set_title('Translated Image')
ax[0][2].set_title('X-Reflected Image')
ax[0][3].set_title('Y-Reflected Image')
ax[0][4].set_title('Rotated Image')
ax[1][0].set_title('Shrunken Image')
ax[1][1].set_title('Enlarged Image')
ax[1][2].set_title('Cropped Image')
ax[1][3].set_title('X-Shared Image')
ax[1][4].set_title('Y-Sheared Image')

for ax in ax.flat:
    # Remove ticks on both axes
    ax.set_xticks([])
    ax.set_yticks([])

# plt.axis('off')
```

```
plt.tight_layout() # to make the images fit better in the display window, preventin
plt.show()
```