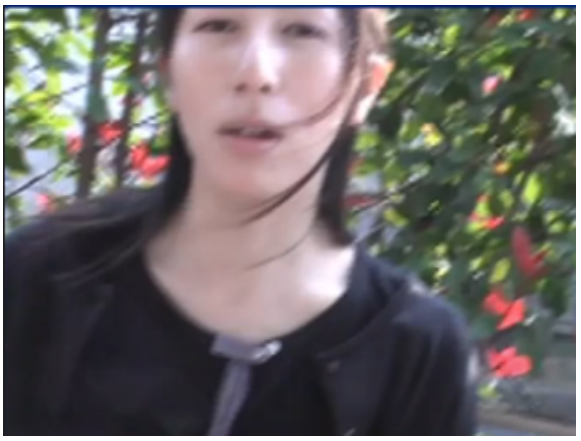
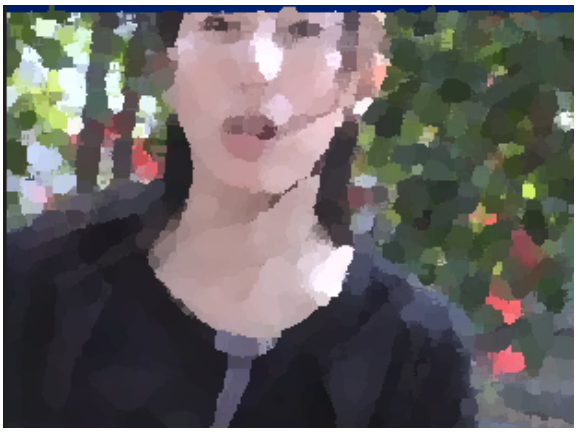
openCV

Watercolor

```
import cv2
# read image
img = cv2.imread('image/Watercolor.png')
# Use the oilPainting function in the xphoto module to proces
res = cv2.xphoto.oilPainting(img, 4, 1)
cv2.imshow('cartoon_image', res)
# save
cv2.imwrite('save/cartoon_image.png', res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





move image

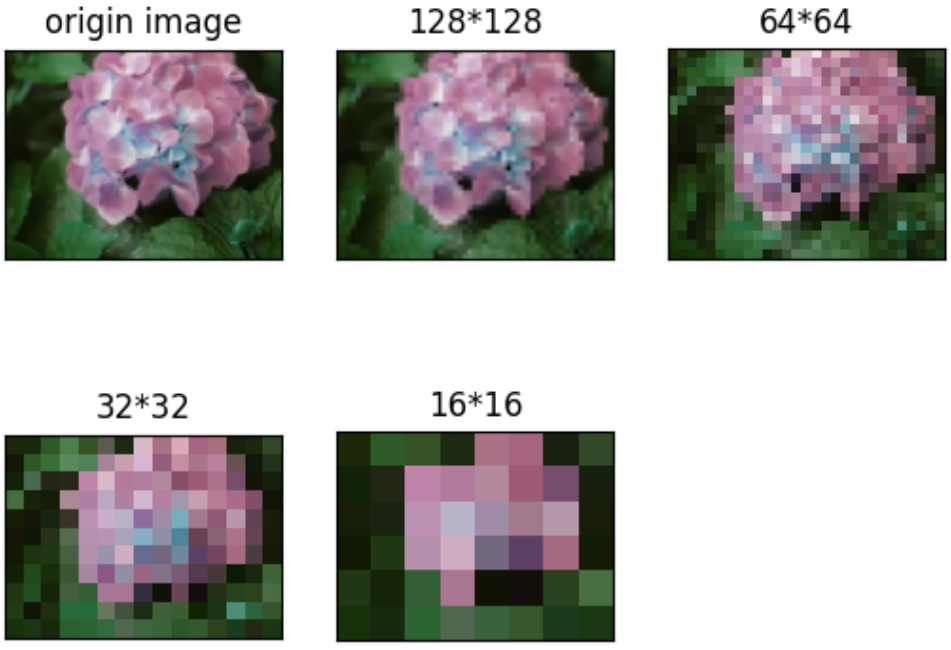
```
import cv2
import numpy as np
img = cv2.imread('image/move.png')
rows = len(img)
cols = len(img[0])
p1 = np.array([[0, 0], [cols - 1, 0], [0, rows - 1]], dtype=n
p2 = np.array([[150, 0], [cols - 1, 0], [0, rows - 1]], dtype
```

```
M = cv2.getAffineTransform(p1, p2)
dst = cv2.warpAffine(img, M, (cols, rows))
cv2.imshow('img', img)
cv2.imshow('dst', dst)
cv2.imwrite('save/move_output.png', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

Quantization

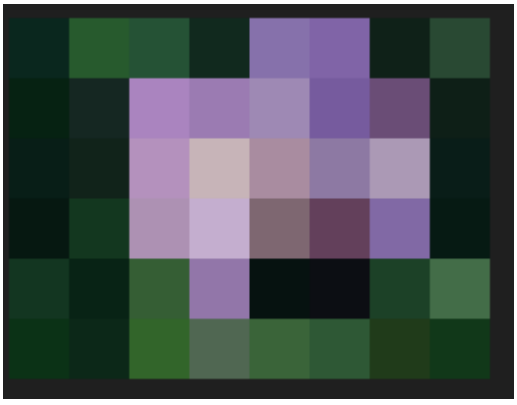
```
import matplotlib.pyplot as plt
import cv2
import numpy as np

img0 = cv2.imread('image/flaw.png')
img1 = cv2.resize(img0, fx = 0.5, fy = 0.5, dsize = None)
# img2 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
height = img1.shape[0]
width = img1.shape[1]
print(img1.shape)
print(width, height)
cv2.namedWindow("W0")
cv2.imshow("W0", img1)
cv2.waitKey(delay = 0)

img8 = img1[0:-1:2, 0:-1:2]
img9 = img1[0:-1:4, 0:-1:4]
img10 = img1[0:-1:8, 0:-1:8]
```

```
img11 = img1[0:-1:16, 0:-1:16]
titles = ['origin image', '128*128', '64*64', '32*32', '16*16
image = [img1, img8, img9, img10, img11]
for j in range(5):
    plt.subplot(2, 3, j + 1)
    if j == 0:
        plt.imshow(image[j])
    else:
        plt.imshow(image[j], 'gray')
    plt.title(titles[j])
    plt.xticks([]), plt.yticks([])
plt.savefig('save/quan.png')
plt.show()
```




```
import cv2
import numpy as np
img = cv2.imread('image/tonumber16.png', cv2.IMREAD_GRAYSCA
```

```
LE)
resized_img = cv2.resize(img, (16, 16), interpolation=cv2.I
NTER_NEAREST)
def pixel_to_value(pixel):
    if pixel < 50:
        return 5
    elif pixel < 100:
        return 3
    elif pixel < 150:
        return 2
    elif pixel < 200:
        return 1
    else:
        return 0
value_matrix = np.vectorize(pixel_to_value)(resized_img)
print(value_matrix)
```



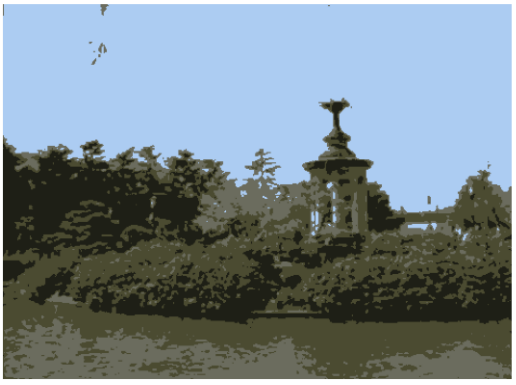
```
[[5 5 3 3 3 3 5 5 2 2 2 2 5 5 3 3]
 [5 5 3 3 3 3 5 5 2 2 2 2 5 5 3 3]
 [5 5 3 3 3 3 5 5 2 2 2 2 5 5 3 3]
 [5 5 5 5 1 1 2 2 2 2 2 2 3 3 5 5]
 [5 5 5 5 1 1 2 2 2 2 2 2 3 3 5 5]
 [5 5 5 5 1 1 2 2 2 2 2 2 3 3 5 5]
 [5 5 5 5 1 1 1 1 1 1 2 2 1 1 5 5]
 [5 5 5 5 1 1 1 1 1 1 2 2 1 1 5 5]
 [5 5 5 5 1 1 1 1 2 2 3 3 2 2 5 5]
 [5 5 5 5 1 1 1 1 2 2 3 3 2 2 5 5]
 [5 5 5 5 1 1 1 1 2 2 3 3 2 2 5 5]
 [5 5 5 5 3 3 2 2 5 5 5 5 3 3 3 3]
 [5 5 5 5 3 3 2 2 5 5 5 5 3 3 3 3]
 [5 5 5 5 3 3 2 2 5 5 5 5 3 3 3 3]
 [5 5 5 5 3 3 3 3 3 3 3 3 5 5 5 5]
 [5 5 5 5 3 3 3 3 3 3 3 3 5 5 5 5]]
```

```
import cv2
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
img = cv2.imread('image/color.png')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
pixels = img_rgb.reshape(-1, 3)
kmeans = KMeans(n_clusters=4)
kmeans.fit(pixels)
centers = kmeans.cluster_centers_.astype('uint8')
new_pixels = centers[kmeans.labels_]
new_img = new_pixels.reshape(img_rgb.shape)
plt.figure(figsize=(10,5))
plt.subplot(1, 2, 1)
plt.imshow(img_rgb)
plt.title('Original Image ')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(new_img)
plt.title('Quantized Image (4 colors)')
plt.axis('off')
plt.show()
```

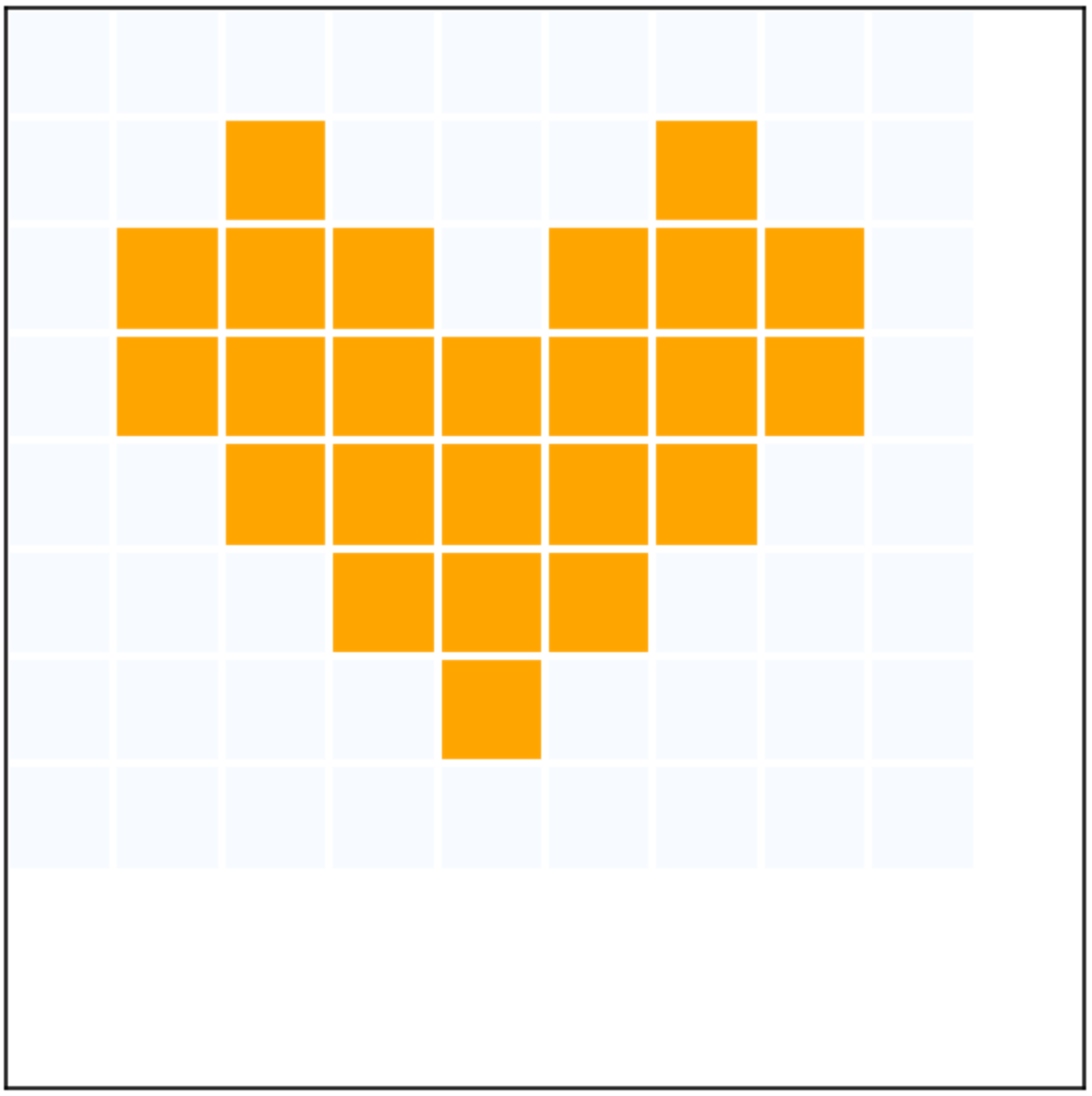
Original Image

Quantized Image (4 colors)



```
import matplotlib.pyplot as plt
import numpy as np
grid = np.zeros((9, 8))
heart_shape = [
    (1, 2), (1, 6),
    (2, 1), (2, 2), (2, 3), (2, 5), (2, 6), (2, 7),
    (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (3, 7),
    (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),
```

```
    (5, 3), (5, 4), (5, 5),
    (6, 4)
]
for pos in heart_shape:
    grid[pos] = 1
fig, ax = plt.subplots()
ax.imshow(grid, cmap="Blues", origin="upper")
for pos in heart_shape:
    ax.add_patch(plt.Rectangle((pos[1]-0.5, pos[0]-0.5), 1,
1, fill=True, color="orange", edgecolor="white"))
ax.set_xticks(np.arange(-.5, 10, 1), minor=True)
ax.set_yticks(np.arange(-.5, 10, 1), minor=True)
ax.grid(which="minor", color="white", linestyle='-', linewi
dth=2)
ax.tick_params(which="both", bottom=False, left=False, labe
lbottom=False, labelleft=False)
plt.show()
```



```
import cv2
import numpy as np
image_path = "D:\WorkSpace\class\computer vision\image\\2D
2.png"
image = cv2.imread(image_path, cv2.IMREAD_COLOR)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
brightness = np.mean(gray)

texture = cv2.Laplacian(gray, cv2.CV_64F)
texture_variance = np.var(texture)
```

```
edges = cv2.Canny(gray, 100, 200)
cv2.imshow('oringe image', image)
cv2.imshow('Edge Detection) ', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```