МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра: Программной инженерии**

Направление подготовки: «Программная инженерия»

«Обработка изображений»

**Отчёт по лабораторной работе**

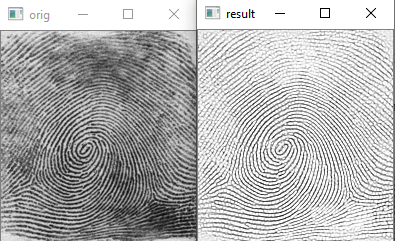
|  |  |
| --- | --- |
|  | Выполнил:  студент группы 381908-3  Имя  Новожилов Александр Юрьевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Проверила:  Гетманская А.А  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Нижний Новгород  
2021 г.

# Задача

Сделать обработку отпечатка пальца с помощью фильтров габора

# Результаты



Проход фильтра Габора под 6-ю разными углами (0, 30, 60, 90, 120, 150)

# Код:

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
def BGR2GRAY(img):  
 gray = 0.2126 \* img[..., 2] + 0.7152 \* img[..., 1] + 0.0722 \* img[..., 0]  
 return gray  
  
def Gabor\_filter(K\_size=111, Sigma=7, Gamma=1.2, Lambda=10, Psi=0, angle=0):  
 d = K\_size // 2  
 gabor = np.zeros((K\_size, K\_size), dtype=np.float32)  
  
 for y in range(K\_size):  
 for x in range(K\_size):  
 px = x - d  
 py = y - d  
 theta = angle / 180. \* np.pi  
 \_x = np.cos(theta) \* px + np.sin(theta) \* py  
 \_y = -np.sin(theta) \* px + np.cos(theta) \* py  
 gabor[y, x] = np.exp(-(\_x \*\* 2 + Gamma \*\* 2 \* \_y \*\* 2) / (2 \* Sigma \*\* 2)) \* np.cos(  
 2 \* np.pi \* \_x / Lambda + Psi)  
 gabor /= np.sum(np.abs(gabor))  
 return gabor  
  
def Gabor\_filtering(gray, K\_size=111, Sigma=7, Gamma=1.2, Lambda=10, Psi=0, angle=0):  
 H, W = gray.shape  
 out = np.zeros((H, W), dtype=np.float32)  
 gabor = Gabor\_filter(K\_size=K\_size, Sigma=Sigma, Gamma=Gamma, Lambda=Lambda, Psi=0, angle=angle)  
 out = cv2.filter2D(gray, -1, gabor)  
 out = np.clip(out, 0, 255)  
 out = out.astype(np.uint8)  
 return out  
  
def Gabor\_process(img):  
 H, W, \_ = img.shape  
 gray = BGR2GRAY(img).astype(np.float32)  
 As = [0, 30, 60, 90, 120, 150]  
 plt.subplots\_adjust(left=0, right=1, top=1, bottom=0, hspace=0, wspace=0.2)  
 out = np.zeros([H, W], dtype=np.float32)  
 for i, A in enumerate(As):  
 \_out = Gabor\_filtering(gray, K\_size=11, Sigma=1.5, Gamma=1.2, Lambda=3, angle=A)  
 out += \_out  
 plt.imshow(\_out, cmap='gray')  
 plt.show()  
  
 out = out / out.max() \* 255  
 out = out.astype(np.uint8)  
 return out  
  
img = cv2.imread(r'D:\hm\otp.PNG').astype(np.float32)  
img1 = cv2.imread(r'D:\hm\otp.PNG')  
def main():  
 cv2.imshow("orig", img1)  
  
 out = Gabor\_process(img)  
 cv2.imshow("result",cv2.bitwise\_not( out))  
  
 thresh = 25  
 img\_binary = cv2.threshold(out, thresh, 255, cv2.THRESH\_BINARY)[1]  
 cv2.imshow("result\_b-w", img\_binary)  
  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()