ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2

за курсом «Алгоритми аналізу та методи обробки зображень»

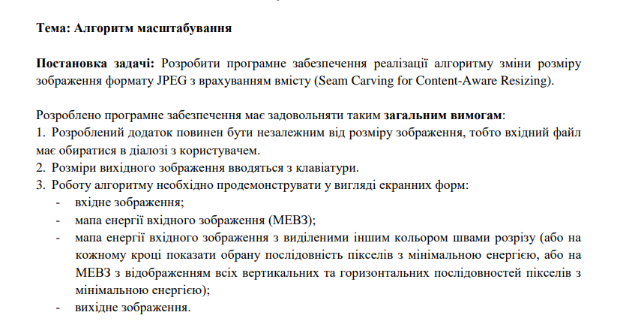
студента групи ПК-24М-1

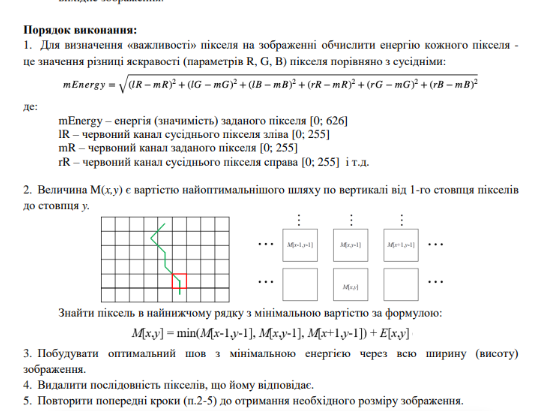
Афанасьєва Дениса

кафедра комп’ютерних технологій, ДНУ

2024/2025 н.р.

Постановка задачі



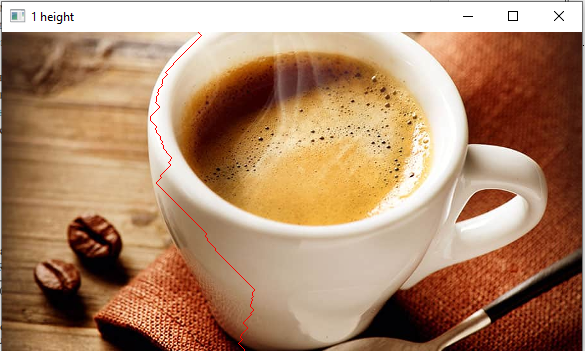


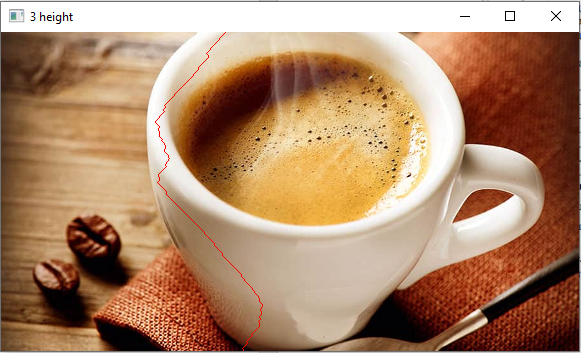
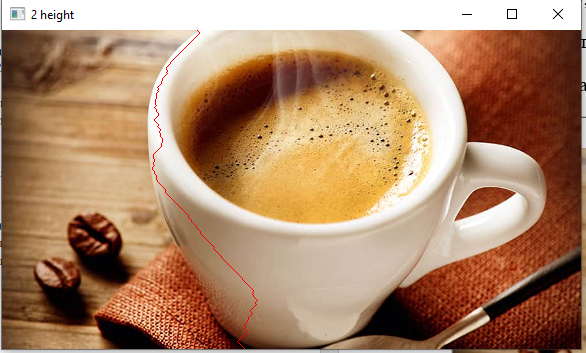
Опис розв’язку

Використовував мову програмування Python, з використанням бібліотек OpenCV, NumPy, Matplotlib та SciPy.

У результаті застосування програми до фото, отримав такі результати:

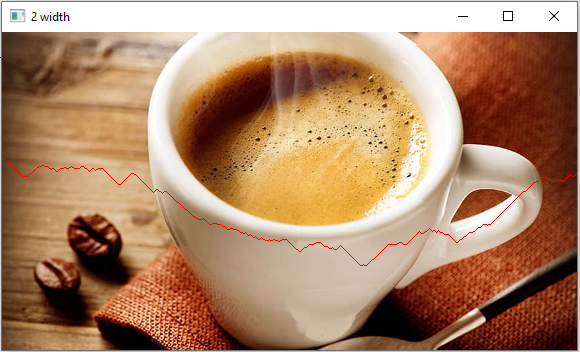
Шви для вертикальних зрізів

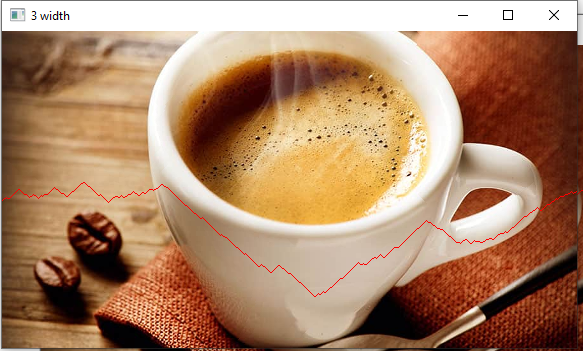




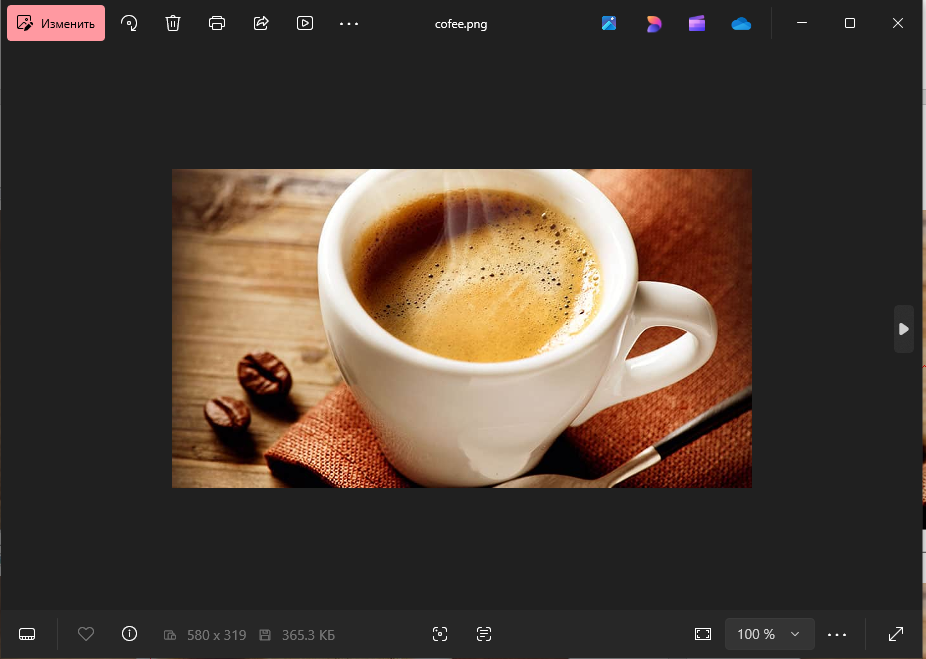
Шви для горизонтальних зрізів



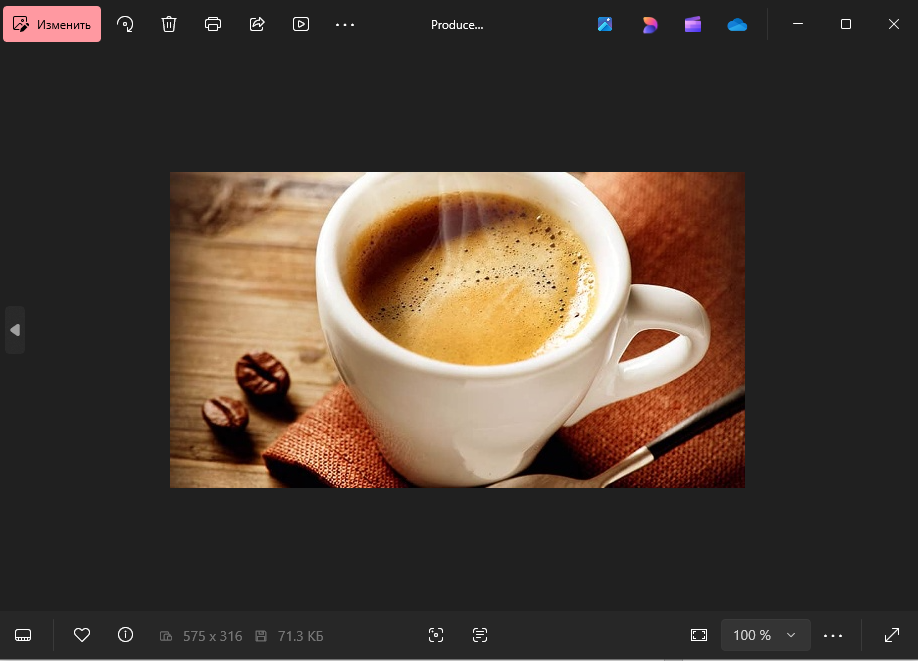




Початкове зображення:



Вихідне зображення:



Код програми

import cv2  
import numpy as np  
from enum import Enum  
import matplotlib.pyplot as plt  
from math import sqrt  
  
  
def open\_img(path=""):  
 path\_to\_file = path or input("Enter path to image: \t")  
 img = cv2.imread(path\_to\_file, cv2.IMREAD\_COLOR)  
 if img is None:  
 print("Can't open image:(")  
 return None  
 # cv2.imshow("Your image", img)  
 # cv2.waitKey(0)  
 return img  
  
  
def calc\_energy(img: np.ndarray) -> np.ndarray:  
 *"""  
 :param img: изначальное изображение (трехцветное)  
 :return: out\_arr - карта энергий кажого пикселя  
 """* hl, vl, \_ = img.shape  
  
 out\_arr: np.ndarray = np.zeros((hl, vl))  
  
 for i in range(hl):  
 for j in range(vl):  
 if i == 0:  
 (mb, mg, mr) = img[i, j].astype(np.int32)  
 (rb, rg, rr) = img[i + 1, j].astype(np.int32)  
 out\_arr[i, j] = sqrt(  
 sum([pow(mr, 2), pow(mg, 2), pow(mb, 2), pow(rr - mr, 2), pow(rg - mg, 2), pow(rb - mb, 2)]))  
 elif i == hl - 1:  
 (mb, mg, mr) = img[i, j].astype(np.int32)  
 (lb, lg, lr) = img[i - 1, j].astype(np.int32)  
 out\_arr[i, j] = sqrt(  
 sum([pow(lr - mr, 2), pow(lg - mg, 2), pow(lb - mb, 2), pow(mr, 2), pow(mg, 2), pow(mb, 2)]))  
 else:  
 (mb, mg, mr) = img[i, j].astype(np.int32)  
 (rb, rg, rr) = img[i + 1, j].astype(np.int32)  
 (lb, lg, lr) = img[i - 1, j].astype(np.int32)  
 out\_arr[i, j] = sqrt(  
 sum([pow(lr - mr, 2), pow(lg - mg, 2), pow(lb - mb, 2), pow(rr - mr, 2), pow(rg - mg, 2),  
 pow(rb - mb, 2)]))  
  
 return out\_arr  
  
  
def camulative\_map(energy: np.ndarray) -> tuple:  
 rows, cols = energy.shape  
 M = np.copy(energy)  
 backtrack = np.zeros\_like(M, dtype=np.int32)  
 for i in range(1, rows):  
 for j in range(cols):  
 if j == 0:  
 idx = np.argmin(M[i - 1, j:j + 2])  
 backtrack[i, j] = idx + j  
 min\_energy = M[i - 1, idx + j]  
 elif j == cols - 1:  
 idx = np.argmin(M[i - 1, j - 1:j + 1])  
 backtrack[i, j] = idx + j - 1  
 min\_energy = M[i - 1, idx + j - 1]  
 else:  
 idx = np.argmin(M[i - 1, j - 1:j + 2])  
 backtrack[i, j] = idx + j - 1  
 min\_energy = M[i - 1, idx + j - 1]  
  
 M[i, j] += min\_energy  
  
 return M, backtrack  
  
  
def find\_seam(M, backtrack) -> np.ndarray:  
 rows, cols = M.shape  
 seam = np.zeros(rows, dtype=np.int32)  
 seam[-1] = np.argmin(M[-1])  
 for i in range(rows - 2, -1, -1):  
 seam[i] = backtrack[i + 1, seam[i + 1]]  
 return seam  
  
  
def remove\_seam(img, seam) -> np.ndarray:  
 rows, cols, channels = img.shape  
 new\_img = np.zeros((rows, cols - 1, channels), dtype=img.dtype)  
 for i in range(rows):  
 if i >= len(seam):  
 break  
 j = seam[i]  
 new\_img[i, :, :] = np.delete(img[i, :, :], j, axis=0)  
 return new\_img  
  
  
def paint\_seam(img: np.ndarray, seam: np.ndarray, is\_horizontal: bool = True) -> np.ndarray:  
 painted\_img = img.copy()  
 rows = seam.shape[0]  
 if is\_horizontal:  
 for i in range(rows):  
 painted\_img[seam[i], i] = (0, 0, 255)  
 else:  
 for i in range(rows):  
 painted\_img[i, seam[i]] = (0, 0, 255)  
 return painted\_img  
  
  
def client():  
 # Открываем изображение  
 img = open\_img("cofee.png")  
 h, w, \_ = img.shape  
 print(f"Orginal image resolution {w}x{h}")  
  
 # Вводим желаемый размер изображения  
 width = int(input("Введите желаемую ширину картинки: "))  
 height = int(input("Введите желаемую высоту картинки: "))  
  
 # Уменьшаем высоту  
 modified\_img = img.copy()  
 for i in range(w - width):  
 energy = calc\_energy(modified\_img)  
 M, backtrack = camulative\_map(energy)  
 seam = find\_seam(M, backtrack)  
 if i < 3:  
 img\_with\_seam = paint\_seam(modified\_img, seam, False)  
 cv2.imshow(f"{i + 1} height", img\_with\_seam)  
 modified\_img = remove\_seam(modified\_img, seam)  
  
 # Уменьшаем ширину  
 modified\_img = modified\_img.transpose((1, 0, 2))  
 for i in range(h - height):  
 energy = calc\_energy(modified\_img)  
 M, backtrack = camulative\_map(energy)  
 seam = find\_seam(M, backtrack)  
 if i < 3:  
 img\_with\_seam = paint\_seam(modified\_img.transpose((1,0,2)), seam)  
 cv2.imshow(f"{i + 1} width", img\_with\_seam)  
 modified\_img = remove\_seam(modified\_img, seam)  
 modified\_img = modified\_img.transpose((1, 0, 2))  
  
 print(f"Produced image resolution"  
 f" {modified\_img.shape[1]}x{modified\_img.shape[0]}")  
  
 cv2.imwrite("Produced\_image.jpg", modified\_img)  
 cv2.imshow("Original image", img)  
 cv2.imshow("Produced image", modified\_img)  
 cv2.waitKey(0)  
  
  
class Tests:  
 pass  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 client()

Висновок

У ході виконання лабораторної роботи навчився працювати з бібліотеками opencv, numpy, scypy, matplotlib для мови програмування Python. Зміг реалізувати алгоритм зміни розміру зображення з врахуванням вмісту, та застосував його до мого зображення.