Лабораторна робота 3

Мета: ознайомитись з основними можливостями та функціями бібліотеки OpenCV по обробці зображень

Виконання роботи

1)Обрати зображення відповідно до варіанту;

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

def display_images(images, titles, cols=3, figsize=(15, 5)):
    """Відображає список зображень у сітці."""
    rows = int(np.ceil(len(images) / cols))
    fig, axes = plt.subplots(rows, cols, figsize=figsize)

    if rows == 1 and cols == 1:
        axes = np.array([axes])
    elif rows == 1 or cols == 1:
        axes = axes.flatten()
    else:
        axes = axes.flatten()

    for i, (img, title) in enumerate(zip(images, titles)):
        ax = axes[i]

        if len(img.shape) == 3:
            img_display = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        else:
            img_display = img

        ax.imshow(img_display, cmap='gray' if len(img.shape) == 2 else None)
        ax.set_title(title, fontsize=10)
        ax.axis('off')

    for j in range(len(images), len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()

IMAGE_PATH = 'cat1.jpg'

try:
    img_bgr = cv2.imread(IMAGE_PATH)
    if img_bgr is None:
        raise FileNotFoundError(f"Зображення '{IMAGE_PATH}' не знайдено.")

    gray_img = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
    print(f"Зображення '{IMAGE_PATH}' завантажено. Розмір: {img_bgr.shape}")

except FileNotFoundError as e:
    print(e)
    print("Використовується штучне тестове зображення.")
    img_bgr = np.zeros((300, 400, 3), dtype=np.uint8)
    img_bgr[50:150, 50:150] = [100, 100, 100]
    img_bgr[150:250, 250:350] = [250, 250, 250]
    img_bgr[50:150, 250:350] = [5, 5, 5]
    cv2.putText(img_bgr, 'TEST IMAGE', (100, 280), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    gray_img = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
```

Рис.3.1 Допоміжна функція



Рис.3.2 Оригінальне забраження

2) Побудувати гістограму яскравості до зображення;

```
print("\n--- Виконання п.2: Побудова гістограми ---")

hist = cv2.calcHist([gray_img], [0], None, [256], [0, 256])

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
display_images([img_bgr], ["Оригінальне зображення (BGR)"], cols=1, figsize=(6, 4))
plt.close()

plt.subplot(1, 2, 2)
plt.title("Гістограма інтенсивності (Яскравість)")
plt.xlabel("Рівень яскравості i [0, 255]")
plt.ylabel("Кількість пікселів h(i)")
plt.plot(hist, color='black')
plt.xlim([0, 256])
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Рис.3.3 Код для гістограми

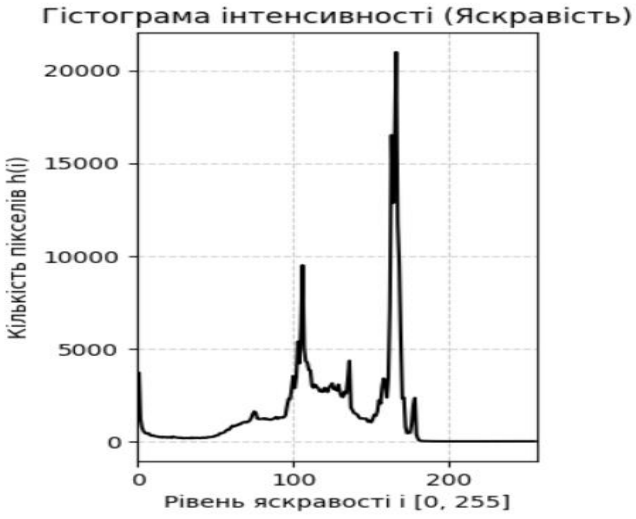


Рис 3.4 Гістограма

3) Застосувати фільтри blur, GaussianBlur, medianBlur, bilateralFilter та порівняти вплив на зображення;

```
print("\n--- Виконання п.3: Застосування фільтрів розмиття ---")

ksize = (7, 7)
sigma = 1.0
median_k = 7
bilat_d = 9
bilat_sc = 75
bilat_ss = 75

blur_img = cv2.blur(img_bgr, ksize)

gaussian_img = cv2.GaussianBlur(img_bgr, ksize, sigma)

median_img = cv2.medianBlur(img_bgr, median_k)

bilateral_img = cv2.bilateralFilter(img_bgr, bilat_d, bilat_sc, bilat_ss)

display_images(
    [img_bgr, blur_img, gaussian_img, median_img, bilateral_img],
    ["Оригінал", f"Blur (Усереднення {ksize})", f"Gaussian $\sigma={sigma}$", f"Median $k={median_k}$", "Bilateral (Збереження країв)"],
    cols=5, figsize=(18, 5)
)
```

Рис.3.5 Код для фільтрів розмиття



Рис.3.6 Фото з фільтрами

4) До обраного зображення застосувати детектор контурів Sobel, Canny знайти контури за допомогою findContours;

```python
print("\n--- Виконання п.4: Детекція та пошук контурів ---")

sobelx = cv2.Sobel(gray_img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(gray_img, cv2.CV_64F, 0, 1, ksize=3)

sobel_mag = cv2.magnitude(sobelx, sobely)
sobel_output = cv2.convertScaleAbs(sobel_mag)

canny_threshold1 = 50
canny_threshold2 = 150
canny_img = cv2.Canny(gray_img, canny_threshold1, canny_threshold2)

contours, hierarchy = cv2.findContours(
    canny_img.copy(),
    cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE
)

contour_img = np.zeros_like(img_bgr)

cv2.drawContours(contour_img, contours, -1, (0, 255, 0), 2)

print(f"Загальна кількість знайдених контурів (findContours): {len(contours)}")

display_images(
    [gray_img, sobel_output, canny_img, contour_img],
    ["Сірий оригінал", "Sobel (Магнітуда градієнта)", f"Canny (T={canny_threshold1}, {canny_threshold2})", "Знайдені контури (drawContours)"],
    cols=4, figsize=(16, 4)
)
```

Рис.3.7 Код для контурів



Рис.3.8 Фото з контурами

5) Обрати нове зображення де присутні люди та видно їх обличчя

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

def display_images(images, titles, cols=3, figsize=(15, 5)):
    """Відображає список зображень у сітці."""
    if not images:
        print("Немає зображень для відображення.")
        return

    num_images = len(images)
    rows = int(np.ceil(num_images / cols))

    fig, axes = plt.subplots(rows, cols, figsize=figsize)

    if num_images == 1:
        axes = np.array([axes])
    else:
        axes = axes.flatten()

    for i, (img, title) in enumerate(zip(images, titles)):
        if img is None or img.size == 0: continue
        ax = axes[i]

        if len(img.shape) == 3:
            img_display = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        else:
            img_display = img

        cmap_val = 'gray' if len(img.shape) == 2 and 'Карта' not in title else None
        if 'Карта' in title:
             cmap_val = 'viridis'

        ax.imshow(img_display, cmap=cmap_val)
        ax.set_title(title, fontsize=10)
        ax.axis('off')

    for j in range(num_images, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()
```

Рис.3.9 Допоміжна функція



Рис.3.10 Оригінальне зображення

6) Завантажити haarcascade_frontalface_default.xml файл та за допомогою нього виявити обличчя та очі людей які зображенні на фото. Після чого порахувати кількість виявлених обличь та нанести рамки

```
FACE_IMAGE_PATH = 'people1.png'
img_face = None

print(f"\n--- Виконання п.6: Детекція обличчя та очей ---")

try:
    img_face = cv2.imread(FACE_IMAGE_PATH)
    if img_face is None:
        raise FileNotFoundError(f"Зображення '{FACE_IMAGE_PATH}' не знайдено.")

    gray_face = cv2.cvtColor(img_face, cv2.COLOR_BGR2GRAY)

    face_xml = cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
    eye_xml = cv2.data.haarcascades + 'haarcascade_eye.xml'

    face_cascade = cv2.CascadeClassifier(face_xml)
    eye_cascade = cv2.CascadeClassifier(eye_xml)

    if face_cascade.empty() or eye_cascade.empty():
         raise IOError("Помилка завантаження XML-каскадів. Перевірте, чи правильно встановлено OpenCV, або вкажіть повний шлях до файлів.")

    faces = face_cascade.detectMultiScale(
        gray_face,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30)
    )

    face_count = 0
    img_result = img_face.copy()

    for (x, y, w, h) in faces:
        face_count += 1
        cv2.rectangle(img_result, (x, y), (x + w, y + h), (255, 0, 0), 2)

        roi_gray = gray_face[y:y + h // 2, x:x + w]
        roi_color = img_result[y:y + h // 2, x:x + w]
        eyes = eye_cascade.detectMultiScale(roi_gray)

        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)

    print(f"Кількість виявлених облич: {face_count}")
    display_images([img_result], [f"Детекція облич та очей на {FACE_IMAGE_PATH} (Знайдено: {face_count})"], cols=1, figsize=(8, 6))

except FileNotFoundError as e:
    print(f"ПОМИЛКА: {e}. ПЕРЕВІРТЕ: Файл 'people1.png' має бути у робочій директорії.")
except IOError as e:
    print(f"ПОМИЛКА: {e}")
except Exception as e:
    print(f"Непередбачена помилка: {e}")
```

Рис.3.11 Код для виявлення обличчя та очей



Рис.3.12 Детекція обличчя та очей

7. Обрати стереопару зображень та визначити ключові точки за допомогою доступних алгоритмів та з’єднати їх

```python
LEFT_IMG_PATH = 'stereoL.png'
RIGHT_IMG_PATH = 'stereoR.png'
img1, img2 = None, None

print(f"\n--- Виконання п.7-8: Стереозір та Карта глибини ---")
try:
    img1 = cv2.imread(LEFT_IMG_PATH)
    img2 = cv2.imread(RIGHT_IMG_PATH)

    if img1 is None or img2 is None:
        raise FileNotFoundError(f"Стереопара не знайдена. Необхідні файли: {LEFT_IMG_PATH} та {RIGHT_IMG_PATH}")

    gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    print("Стереопара завантажена. Пошук ключових точок SIFT...")

    sift = cv2.SIFT_create()
    kp1, des1 = sift.detectAndCompute(gray1, None)
    kp2, des2 = sift.detectAndCompute(gray2, None)

    FLANN_INDEX_KDTREE = 1
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(des1, des2, k=2)

    good_matches = []
    ratio_threshold = 0.7
    for m, n in matches:
        if m.distance < ratio_threshold * n.distance:
            good_matches.append(m)

    print(f"Найдено надійних відповідностей (після Ratio Test): {len(good_matches)}")

    img_matches = cv2.drawMatches(img1, kp1, img2, kp2, good_matches, None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    display_images(
        [img1, img2, img_matches],
        ["Зображення 1 (Ліве)", "Зображення 2 (Праве)", f"З'єднані SIFT точки ({len(good_matches)} шт.)"],
        cols=3, figsize=(18, 6)
    )
```

Рис.3.13 Код для стереопари зображень та її ключових точок



Рис.3.14 Стереопари зображень та її ключові точки

8. На основі п.7 побудувати карту глибини використовуючи StereoBM і StereoSGBM.

```python
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)

good = []
pts1 = []
pts2 = []

matches = sorted(matches, key=lambda x: x[0].distance)

for i, (m, n) in enumerate(matches):
    if m.distance < 0.6 * n.distance:
        good.append(m)
        pts2.append(kp2[m.trainIdx].pt)
        pts1.append(kp1[m.queryIdx].pt)

print(f"Кількість гарних збігів: {len(good)}")

img_matches = cv2.drawMatches(img1, kp1, img2, kp2, good[:100], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(img_matches, "jet")
plt.title("Відповідність ознак (SIFT + FLANN)")
plt.axis("off")
plt.show()

stereo = cv2.StereoBM_create(numDisparities=16*6, blockSize=11)
disparity2 = stereo.compute(img2, img1)
plt.imshow(disparity2, "jet")
plt.title("Карта глибини (Disparity Map)")
plt.axis("off")
plt.show()
```

Рис.3.15 Код для карти глибини

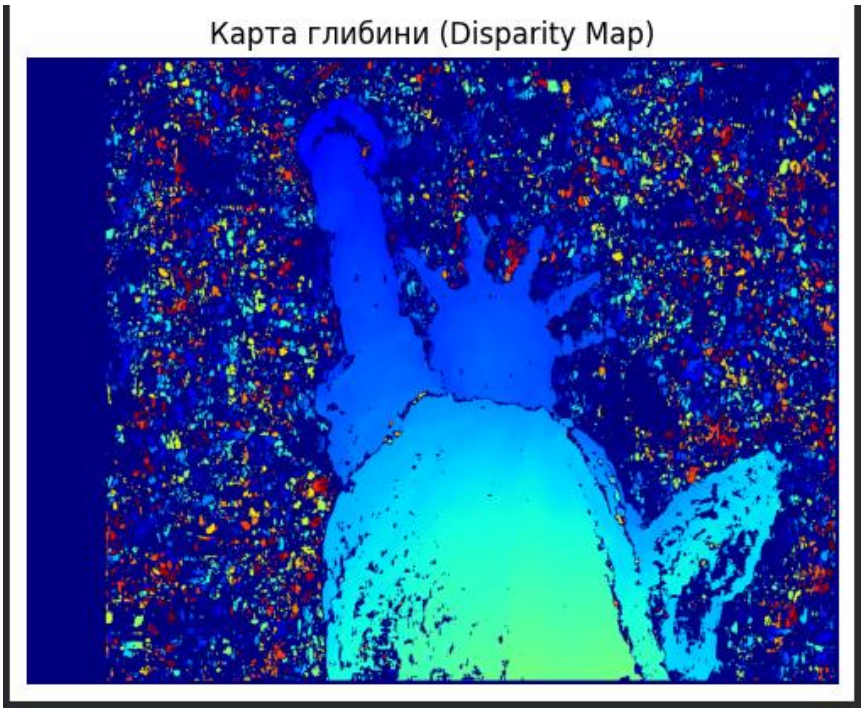


Рис.3.16 Карта глибини

Висновок: ознайомилась з основними можливостями та функціями бібліотеки OpenCV по обробці зображень