Додаток А

Код для піксельного порівняння зображень

```
python
import cv2
importnumpy as np

# Завантаження зображень
img1 = cv2.imread('image1.jpg')
img2 = cv2.imread('image2.jpg')

# Перевірка, що зображення однакового розміру
if img1.shape != img2.shape:
print("Зображення мають різні розміри!")
exit()

# Перетворення зображень у градації сірого (якщо зображення кольорові)
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# Обчислення абсолютної різниці між зображеннями
diff = cv2.absdiff(gray1, gray2)

# Застосування порогового значення для виділення відмінностей
_, thresh = cv2.threshold(diff, 50, 255, cv2.THRESH_BINARY)

# Показати результат
cv2.imshow('Differences', thresh)

# Очікування клавіші для закриття вікна
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Додаток Б

Код для підключення бібліотеки scikit-image

```
python
fromskimage.metricsimportstructural_similarity as ssim
import cv2
importnumpy as np

# Завантаження зображень
img1 = cv2.imread('image1.jpg', cv2.IMREAD_GRAYSCALE)  # Перетворення в градації
сірого
img2 = cv2.imread('image2.jpg', cv2.IMREAD_GRAYSCALE)

# Перевірка, що зображення однакового розміру
if img1.shape != img2.shape:
print("Зображення мають різні розміри!")
exit()

# Обчислення SSIM між зображеннями
score, diff = ssim(img1, img2, full=True)

print(f"SSIM: {score}")

# diff - це зображення відмінностей, яке можна візуалізувати
# Перетворення різниці у 8-бітове зображення
diff = (diff * 255).astype("uint8")

# Відображення відмінностей
cv2.imshow('Difference', diff)

# Показати результат
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Додаток В

Код для підключення операторів Собеля та Канні

```
python
import cv2
importnumpy as np

# Завантаження зображення
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)  # Перетворюємо зображення в
градації сірого

# Застосування оператора Собеля
sobel_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)  # Градієнт по осі X
sobel_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)  # Градієнт по осі Y
sobel_edges = cv2.magnitude(sobel_x, sobel_y)  # Знаходження амплітуди градієнта

# Застосування порогового значення для виділення контурів
_, sobel_thresh = cv2.threshold(np.abs(sobel_edges), 50, 255, cv2.THRESH_BINARY)

# Застосування оператора Канні для виділення контурів
canny_edges = cv2.Canny(img, 100, 200)

# Відображення результатів
cv2.imshow('Original Image', img)
cv2.imshow('SobelEdges', sobel_thresh)
cv2.imshow('CannyEdges', canny_edges)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Додаток Г

Код для порівняння гістограм

```
python
import cv2
importnumpy as np
frommatplotlibimportpyplot as plt

# Завантаження зображень
img1 = cv2.imread('image1.jpg')
img2 = cv2.imread('image2.jpg')

# Перетворення зображень у градації сірого (якщо вони кольорові)
img1_gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2_gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# Побудова гістограм
hist1 = cv2.calcHist([img1_gray], [0], None, [256], [0, 256])
hist2 = cv2.calcHist([img2_gray], [0], None, [256], [0, 256])

# Нормалізація гістограм (для коректного порівняння)
hist1 /= hist1.sum()
hist2 /= hist2.sum()

# Порівняння гістограм з використанням різних метрик
correlation = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CORREL)
print(f"Кореляція: {correlation}")

# Евклідова відстань
euclidean_distance = cv2.compareHist(hist1, hist2, cv2.HISTCMP_EUCLIDEAN)
print(f"Евклідова відстань: {euclidean_distance}")

# Хі-квадрат
chi_square = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CHISQR)
print(f"Хі-квадрат: {chi_square}")

# Взаємна інформація
intersect = cv2.compareHist(hist1, hist2, cv2.HISTCMP_INTERSECT)
print(f"Перетин: {intersect}")

# Візуалізація гістограм
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.title('гістограма Image 1')
plt.plot(hist1)
plt.xlim([0, 256])

plt.subplot(1, 2, 2)
plt.title('гістограма Image 2')
plt.plot(hist2)
plt.xlim([0, 256])

plt.show()
```

Додаток Д

Код для підключення бібліотеки ORB

```
python
import cv2
importnumpy as np

# Завантажуємо зображення
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Створюємо детектор ORB
orb = cv2.ORB_create()

# Знаходимо ключові точки та дескриптори
keypoints, descriptors = orb.detectAndCompute(image, None)

# Візуалізуємо ключові точки на зображенні
output_image = cv2.drawKeypoints(image, keypoints, None, color=(0, 255, 0))

# Показуємо результат
cv2.imshow('ORB Keypoints', output_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
# Створюємо детектор ORB з додатковими параметрами
orb = cv2.ORB_create(nfeatures=1000, scaleFactor=1.2, nlevels=8)

# Знаходимо ключові точки та дескриптори
keypoints, descriptors = orb.detectAndCompute(image, None)

# Візуалізуємо ключові точки
output_image = cv2.drawKeypoints(image, keypoints, None, color=(0, 255, 0))

cv2.imshow('ORB Keypoints', output_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Додаток Е

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК 503320.022ПЗ	Пояснювальна записка	68	с.
			Інші документи		
А4	2		Код для піксельного		
			порівняння зображень	1	с.
А4	3		Код для підключення		
			бібліотеки scikit-image	1	с.
А4	4		Код для підключення		
			операторів Собеля та Канні	1	с.
А4	5		Код для порівняння гістограм	1	с.
А4	6		Код для підключення		
			бібліотеки ORB	1	с.
А4	7		Демонстраційний матеріал	16	с.

Змін.	Арк	Номер докум.	Підпис	Дата	ГЮІК. 503320.022ВД			
Розроб.		Гавриков Є.Ю.			Розроблення системи автоматизації для контролю точності виготовлення деталей на 3D принтері Відомість кваліфікаційної роботи	Літера	Арку	Арк
Переві		Аллахверанов Р.Ю.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.конт		Бронніков А.І.						
Затв.		Невлюдов І.Ш.						