КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт физики

Кафедра Радиофизики

П. А. Корчагин, И. П. Корчагин

ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

Методические указания к выполнению лабораторных работ

Казань – 2025

УДК 681.322

ББК 32.973.26-018.2я73

К70

Принято на заседании кафедры радиофизики

Протокол № 1 от 2 сентября 2025 года

Рецензент

кандидат физико-математических наук,

доцент кафедры радиоастрономии КФУ

Е. Ю. Зыков

Корчагин П. А., Корчагин И. П.

К70 Цифровая обработка изображений. Методические указания к выполнению лабораторных работ/ П. А.Корчагин, И. П. Корчагин. – Казань: Казан. ун-т, 2025. – 42 с.

Пособие охватывает ключевые аспекты ЦОИ – от базовых понятий, таких как типы изображений, гистограммы и разрешение, до сложных алгоритмов: фильтрации, морфологических операций, сегментации и анализа объектов. Особое внимание уделено практической реализации методов с использованием языка Python и библиотек OpenCV, Pillow и NumPy, что делает материал доступным и применимым в реальных задачах.

Для студентов направления «Радиофизика» пособие формирует основы для обработки данных дистанционного зондирования, анализа сигналов визуального диапазона и работы с изображениями из физических экспериментов. Для студентов «Информационная безопасность» представленные методы открывают возможности в таких актуальных областях, как анализ визуального контента, мониторинг экранов, обнаружение утечек данных и цифровая криминалистика.

Оглавление

1. Введение в цифровую обработку изображений

Цифровая обработка изображений (ЦОИ) – это область, которая занимается манипуляцией и анализом цифровых изображений с использованием компьютерных алгоритмов. Она является междисциплинарной областью, находящейся на пересечении информатики, математики, инженерии и других наук. ЦОИ играет ключевую роль во многих современных технологиях, от медицинских систем визуализации и спутниковой съемки до систем безопасности и развлекательной индустрии.

Целью ЦОИ может быть улучшение визуального качества изображения, извлечение информации из изображений, сжатие данных, восстановление изображений, а также подготовка изображений для машинного зрения и искусственного интеллекта.

Python является одним из наиболее популярных языков программирования для цифровой обработки изображений благодаря богатому набору специализированных библиотек, простоте освоения и мощным возможностям для научных вычислений.

2. Основные понятия и термины

2.1. Цифровое изображение

Цифровое изображение представляет собой двумерную матрицу (или трехмерную для цветных изображений) числовых значений, называемых пикселями. Каждый пиксель содержит информацию о яркости или цвете в соответствующей точке изображения.

2.2. Типы изображений

Бинарные изображения – содержат только два значения (0 и 1, черный и белый).

Полутоновые изображения – содержат оттенки серого (обычно от 0 до 255).

Цветные изображения – содержат информацию о цвете в различных цветовых моделях (RGB, HSV, Lab и др.).

2.3. Разрешение изображения

Разрешение определяется количеством пикселей по горизонтали и вертикали (например, 1920×1080). Чем выше разрешение, тем больше деталей содержит изображение.

2.4. Гистограмма изображения

Гистограмма показывает распределение интенсивности пикселей в изображении и является важным инструментом анализа изображений.

3. Обзор библиотек Python для ЦОИ

3.1. OpenCV

OpenCV (Open Source Computer Vision Library) – основная библиотека для компьютерного зрения и обработки изображений. Содержит более 2500 алгоритмов для различных задач обработки изображений.

3.2. Pillow (PIL)

Pillow – дружественная библиотека для базовых операций с изображениями: открытие, сохранение, изменение размера, поворот и базовые фильтры.

3.3. NumPy

NumPy предоставляет мощные инструменты для работы с многомерными массивами, которые используются для представления изображений в Python.

4. Лабораторная работа 1: Основы работы с изображениями

4.1. Цель работы

Изучение базовых операций с изображениями: чтение, отображение, сохранение, изменение размера, обрезка и преобразование цветовых пространств.

4.2. Теоретические основы

Основные функции для работы с изображениями включают:

- cv2.imread() – чтение изображения из файла;
- cv2.imshow() – отображение изображения;
- cv2.imwrite() – сохранение изображения в файл;
- cv2.resize() – изменение размера изображения;
- cv2.cvtColor() – преобразование цветового пространства.

4.3. Варианты заданий

Вариант 1: Обработка фотографии пейзажа

Задание: Загрузите изображение природного пейзажа, преобразуйте его в оттенки серого, измените размер до 800×600 пикселей, обрежьте центральную часть размером 400×300 и сохраните результат.

Методические указания:

1. Используйте cv2.imread() с флагом cv2.IMREAD_COLOR для загрузки;

2.Для преобразования в серый цвет используйте cv2.COLOR_BGR2GRAY;

3. При изменении размера используйте интерполяцию cv2.INTER_AREA для уменьшения;

4. Для обрезки вычислите центральные координаты: center_x = width//2, center_y = height//2;

5. Проверьте качество сохранения с параметром качества для JPEG.

Вариант 2: Обработка портретного изображения

Задание: Загрузите портретное изображение, создайте его цветную и черно-белую версии, измените размер до квадратного формата 512×512, примените обрезку с сохранением лица в центре.

Методические указания:

1. Сохраните оригинал перед преобразованиями для сравнения;

2. Используйте cv2.INTER_CUBIC для интерполяции при увеличении;

3. При создании квадратного формата учитывайте соотношение сторон;

4. Для центрирования лица используйте симметричную обрезку;

5. Сравните результаты разных методов интерполяции.

Вариант 3: Обработка изображения архитектуры

Задание: Загрузите изображение здания или архитектурного объекта, создайте миниатюру размером 150×150, сохраните в трех форматах: PNG, JPEG и WEBP.

Методические указания:

1. Изучите влияние сжатия на архитектурные детали;

2. Используйте разные уровни качества JPEG (50, 75, 95);

3. Сравните размеры файлов различных форматов;

4. Обратите внимание на четкость линий и краев в миниатюре;

5. Используйте cv2.INTER_AREA для создания качественных миниатюр.

Вариант 4: Цветовые преобразования

Задание: Загрузите цветное изображение и создайте его версии в различных цветовых пространствах: RGB, HSV, LAB, YUV. Сохраните каждый канал отдельно.

Методические указания:

1. Изучите особенности каждого цветового пространства;

2. Используйте cv2.split() для разделения каналов;

3. Визуализируйте каналы с помощью cv2.imshow();

4. Обратите внимание на информативность разных каналов;

5. Проведите обратное преобразование с cv2.merge().

Вариант 5: Комплексная обработка

Задание: Создайте программу-конвертер, которая принимает изображение любого размера и формата, нормализует его до стандартного размера 640×480, применяет автокоррекцию яркости и сохраняет в оптимальном формате.

Методические указания:

1. Определите исходные параметры изображения с помощью shape;

2. Вычислите коэффициент масштабирования для сохранения пропорций;

3. Используйте гистограмму для анализа яркости;

4. Примените cv2.equalizeHist() для коррекции яркости;

5. Выберите формат сохранения на основе содержимого изображения.

4.4. Базовый пример кода

```
import cv2
import numpy as np

# Загрузка изображения
img = cv2.imread('image.jpg')
if img is None:
    print("Ошибка загрузки изображения")
    exit()

# Отображение оригинала
cv2.imshow('Original', img)
```

```
# Преобразование в оттенки серого
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscale', gray)

# Изменение размера
resized = cv2.resize(img, (800, 600), interpolation=cv2.INTER_AREA)

# Обрезка центральной части
height, width = resized.shape[:2]
center_x, center_y = width // 2, height // 2
crop_width, crop_height = 400, 300
x1 = center_x - crop_width // 2
y1 = center_y - crop_height // 2
x2 = center_x + crop_width // 2
y2 = center_y + crop_height // 2
cropped = resized[y1:y2, x1:x2]

# Сохранение результата
cv2.imwrite('result.jpg', cropped)

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

5. Лабораторная работа 2: Геометрические преобразования

5.1. Цель работы

Изучение методов геометрических преобразований изображений: поворотов, сдвигов, аффинных и перспективных преобразований.

5.2. Теоретические основы

Геометрические преобразования изменяют пространственное расположение пикселей в изображении:

- **Поворот** – вращение изображения на заданный угол относительно центра;
- **Сдвиг** – смещение изображения по осям X и Y;
- **Аффинные преобразования** – сохраняют параллельные линии;
- **Перспективные преобразования** – изменяют перспективу изображения.

5.3. Варианты заданий

Вариант 1: Коррекция наклона документа

Задание: Загрузите изображение наклонного текстового документа, определите угол наклона и примените поворот для выравнивания текста горизонтально.

Методические указания:

1. Используйте cv2.HoughLines() для определения угла наклона линий текста;

2. Вычислите среднее значение углов для определения общего наклона;

3. Примените cv2.getRotationMatrix2D() с найденным углом;

4. Используйте cv2.warpAffine() с увеличенным размером выходного изображения;

5. Обрежьте лишние области после поворота.

Вариант 2: Стабилизация изображения

Задание: Создайте последовательность из 5 слегка смещенных версий одного изображения, затем примените обратные сдвиги для стабилизации.

Методические указания:

1. Создайте случайные сдвиги в диапазоне ± 20 пикселей;

2. Сохраните параметры сдвигов для последующей коррекции;

3. Используйте cv2.getAffineTransform() для создания матриц сдвига;

4. Примените обратные преобразования для стабилизации;

5. Сравните качество исходного и стабилизированного изображения.

Вариант 3: Коррекция перспективы книги

Задание: Загрузите фотографию открытой книги, снятой под углом, и примените перспективную коррекцию для получения вида "сверху".

Методические указания:

1. Найдите углы страницы книги (можно задать вручную);

2. Определите целевые координаты прямоугольника;

3. Используйте cv2.getPerspectiveTransform() для создания матрицы;

4. Примените cv2.warpPerspective() с подходящим размером выходного изображения;

5. Проверьте результат на читаемость текста.

Вариант 4: Создание панорамы

Задание: Возьмите два перекрывающихся изображения одной сцены и объедините их в панораму, используя аффинные преобразования.

Методические указания:

1. Найдите ключевые точки на обоих изображениях (можно использовать SIFT);

2. Сопоставьте точки между изображениями;

3. Вычислите матрицу аффинного преобразования;

4. Примените преобразование к одному из изображений;

5. Объедините изображения с учетом перекрытия.

Вариант 5: Мульти-преобразования

Задание: Создайте интерактивную программу для последовательного применения нескольких геометрических преобразований с сохранением истории и возможностью отмены.

Методические указания:

1. Реализуйте стек преобразований для хранения истории;

2. Создайте функции для каждого типа преобразования;

3. Добавьте возможность просмотра промежуточных результатов;

4. Реализуйте функцию отмены последнего преобразования;

5. Обеспечьте сохранение финальной матрицы преобразования.

5.4. Пример кода для поворота и сдвига

```
import cv2
import numpy as np

# Загрузка изображения
img = cv2.imread('image.jpg')
height, width = img.shape[:2]

# Поворот изображения на 45 градусов
center = (width // 2, height // 2)
rotation_matrix = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated = cv2.warpAffine(img, rotation_matrix, (width, height))

# Сдвиг изображения
translation_matrix = np.float32([[1, 0, 100], [0, 1, 50]])
translated = cv2.warpAffine(img, translation_matrix, (width, height))

# Аффинное преобразование
```

```
src_points = np.float32([[50, 50], [200, 50], [50, 200]])
dst_points = np.float32([[70, 70], [180, 30], [30, 220]])
affine_matrix = cv2.getAffineTransform(src_points, dst_points)
affine_transformed = cv2.warpAffine(img, affine_matrix, (width, height))

# Отображение результатов
cv2.imshow('Original', img)
cv2.imshow('Rotated', rotated)
cv2.imshow('Translated', translated)
cv2.imshow('Affine', affine_transformed)

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

6. Лабораторная работа 3: Фильтрация изображений

6.1. Цель работы

Изучение методов фильтрации изображений для сглаживания, устранения шума, повышения резкости и выделения границ.

6.2. Теоретические основы

Фильтрация изображений основана на операции свертки. Основные типы фильтров:

- **Линейные фильтры** – усредняющий, гауссовский, для повышения резкости;
- **Нелинейные фильтры** – медианный, билатеральный;
- **Фильтры выделения границ** – Собеля, Лапласа, Кэнни.

6.3. Варианты заданий

Вариант 1: Реставрация старой фотографии

Задание: Загрузите изображение старой фотографии с шумом, примените последовательность фильтров для улучшения качества: устранение шума, повышение резкости, улучшение контраста.

Методические указания:

1. Начните с медианного фильтра для устранения соляного шума;
2. Примените гауссовское сглаживание для общего устранения шума;
3. Используйте нерезкое маскирование для повышения резкости;
4. Примените эквализацию гистограммы для улучшения контраста;
5. Сравните результаты после каждого этапа обработки.

Вариант 2: Подготовка изображения для OCR

Задание: Подготовьте текстовое изображение для распознавания символов: устраните шум, повысьте контрастность текста, выделите границы символов.

Методические указания:

1. Преобразуйте изображение в оттенки серого;
2. Используйте морфологические операции для очистки;
3. Примените адаптивную пороговую обработку;
4. Используйте фильтр Собеля для выделения границ символов;
5. Проведите постобработку для удаления артефактов.

Вариант 3: Художественная обработка пейзажа

Задание: Создайте художественный эффект для пейзажного изображения, используя различные фильтры: размытие для создания глубины резкости, выделение границ для эффекта рисунка.

Методические указания:

1. Создайте маску глубины на основе цвета или яркости;

2. Примените переменное размытие по Гауссу;

3. Используйте билатеральный фильтр для сохранения краев;

4. Примените детектор границ Кэнни с мягкими параметрами;

5. Объедините результаты для создания художественного эффекта.

Вариант 4: Медицинская визуализация

Задание: Обработайте медицинское изображение (рентген, МРТ) для улучшения видимости структур: устранение шума, повышение контраста, выделение границ анатомических структур.

Методические указания:

1. Используйте анизотропную диффузию для сохранения границ;

2. Примените CLAHE (адаптивная эквализация гистограммы);

3. Используйте градиентные фильтры для выделения границ;

4. Примените морфологические операции для очистки;

5. Создайте составное изображение с улучшенной визуализацией.

Вариант 5: Система адаптивной фильтрации

Задание: Разработайте систему, которая автоматически выбирает оптимальные параметры фильтрации на основе анализа характеристик изображения (шум, резкость, контрастность).

Методические указания:

1. Реализуйте функции оценки качества изображения;

2. Создайте алгоритм определения типа и уровня шума;

3. Разработайте правила выбора фильтров;

4. Реализуйте итеративное улучшение с оценкой результата;

5. Добавьте возможность ручной коррекции параметров.

6.4. Пример кода фильтрации

```
import cv2
import numpy as np

# Загрузка изображения
img = cv2.imread('image.jpg')

# Добавление шума для демонстрации
noise = np.random.normal(0, 25, img.shape).astype(np.uint8)
noisy_img = cv2.add(img, noise)

# Применение различных фильтров
# Усредняющий фильтр
blur = cv2.blur(noisy_img, (5, 5))

# Гауссовское размытие
gaussian = cv2.GaussianBlur(noisy_img, (5, 5), 0)

# Медианный фильтр
median = cv2.medianBlur(noisy_img, 5)

# Билатеральный фильтр
bilateral = cv2.bilateralFilter(noisy_img, 9, 75, 75)

# Повышение резкости
kernel_sharpen = np.array([
    [-1, -1, -1],
    [-1,  9, -1],
    [-1, -1, -1]
])
```

```
sharpened = cv2.filter2D(img, -1, kernel_sharpen)

# Выделение границ
# Оператор Собеля
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)
sobel = np.sqrt(sobelx**2 + sobely**2)

# Детектор границ Кэнни
edges = cv2.Canny(gray, 100, 200)

# Отображение результатов
cv2.imshow('Original', img)
cv2.imshow('Noisy', noisy_img)
cv2.imshow('Blur', blur)
cv2.imshow('Gaussian Blur', gaussian)
cv2.imshow('Median Filter', median)
cv2.imshow('Bilateral', bilateral)
cv2.imshow('Sharpened', sharpened)
cv2.imshow('Canny Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

7. Лабораторная работа 4: Морфологические операции

7.1. Цель работы

Изучение морфологических операций для обработки бинарных и полутоновых изображений: эрозия, дилатация, открытие, закрытие, морфологический градиент.

7.2. Теоретические основы

Морфологические операции основаны на теории множеств и используют структурирующий элемент:

- **Эрозия** – уменьшение размеров белых областей;
- **Дилатация** – увеличение размеров белых областей;
- **Открытие** – эрозия, затем дилатация;
- **Закрытие** – дилатация, затем эрозия;
- **Морфологический градиент** – разность дилатации и эрозии.

7.3. Варианты заданий

Вариант 1: Анализ микроскопических изображений

Задание: Обработайте изображение биологических клеток: выделите отдельные клетки, устраните шум, измерьте размеры клеток.

Методические указания:

1. Преобразуйте изображение в бинарный формат с пороговой обработкой;
2. Используйте открытие для устранения мелкого шума;
3. Примените закрытие для заполнения разрывов в контурах клеток;
4. Используйте watershed для разделения соприкасающихся клеток;
5. Измерьте площадь каждой клетки с помощью contourArea.

Вариант 2: Обработка отпечатков пальцев

Задание: Улучшите качество изображения отпечатка пальца: устраните разрывы в папиллярных линиях, удалите шум, выделите особые точки (окончания и разветвления линий).

Методические указания:

1. Используйте скелетизацию для утончения папиллярных линий;
2. Примените морфологическое закрытие для устранения разрывов;
3. Используйте hit-or-miss преобразование для поиска особых точек;

4. Создайте различные структурирующие элементы для разных типов особенностей;

5. Проведите постобработку для устранения ложных точек.

Вариант 3: Анализ текстовых документов

Задание: Обработайте сканированное изображение текста: разделите строки и символы, устраните шум печати, восстановите поврежденные символы.

Методические указания:

1. Используйте горизонтальные и вертикальные структурирующие элементы;

2. Примените opening для удаления точек и мелких артефактов;

3. Используйте closing для восстановления разорванных линий в символах;

4. Примените проекцию гистограммы для разделения строк;

5. Используйте морфологические операции для улучшения качества символов.

Вариант 4: Обработка промышленных изображений

Задание: Проанализируйте изображение промышленных деталей: выделите дефекты поверхности, измерьте размеры отверстий, определите качество обработки краев.

Методические указания:

1. Создайте эталонную маску детали;

2. Используйте морфологические операции для выделения дефектов;

3. Примените различные размеры структурирующих элементов;

4. Используйте расстояние-трансформацию для измерения размеров;

5. Реализуйте автоматическую классификацию типов дефектов.

Вариант 5: Интерактивная морфологическая лаборатория

Задание: Создайте интерактивную программу для исследования влияния различных морфологических операций и структурирующих элементов на изображение.

Методические указания:

1. Реализуйте возможность изменения размера и формы структурирующего элемента;

2. Добавьте ползунки для регулировки количества итераций;

3. Создайте предустановки для типичных задач;

4. Добавьте возможность комбинирования операций;

5. Реализуйте сохранение и загрузку настроек.

7.4. Пример кода морфологических операций

```
import cv2
import numpy as np

# Загрузка и преобразование в бинарное изображение
img = cv2.imread('image.jpg', 0)
_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# Определение структурирующих элементов
kernel_small = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
kernel_large = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
kernel_ellipse = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))

# Базовые морфологические операции
erosion = cv2.erode(binary, kernel_small, iterations=1)  # Эрозия
dilation = cv2.dilate(binary, kernel_small, iterations=1)  # Дилатация

# Составные морфологические операции
```

```
    opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel_small)  # Открытие (устранение шума)
    closing = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel_small)  # Закрытие (заполнение разрывов)
    gradient = cv2.morphologyEx(binary, cv2.MORPH_GRADIENT, kernel_small)  # Морфологический градиент

    # Специальные морфологические операции
    tophat = cv2.morphologyEx(binary, cv2.MORPH_TOPHAT, kernel_large)  # Top Hat (светлые детали)
    blackhat = cv2.morphologyEx(binary, cv2.MORPH_BLACKHAT, kernel_large)  # Black Hat (темные детали)

    # Hit-or-Miss преобразование
    hitmiss_kernel = np.array([
        [0, 1, 0],
        [1, -1, 1],
        [0, 1, 0]
    ], dtype=np.int8)
    hitmiss = cv2.morphologyEx(binary, cv2.MORPH_HITMISS, hitmiss_kernel)

    # Отображение результатов
    cv2.imshow('Original', binary)
    cv2.imshow('Erosion', erosion)
    cv2.imshow('Dilation', dilation)
    cv2.imshow('Opening', opening)
    cv2.imshow('Closing', closing)
    cv2.imshow('Gradient', gradient)
    cv2.imshow('Top Hat', tophat)
    cv2.imshow('Black Hat', blackhat)
```

```
cv2.imshow('Hit or Miss', hitmiss)


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

8. Лабораторная работа 5: Сегментация изображений

8.1. Цель работы

Изучение различных методов сегментации изображений для разделения изображения на области с похожими характеристиками.

8.2. Теоретические основы

Сегментация изображений включает следующие основные методы:

- **Пороговая сегментация** – разделение по яркости;
- **Цветовая сегментация** – разделение по цветовым характеристикам;
- **Сегментация методом водораздела** – разделение на основе топографической аналогии;
- **Кластеризация** – группировка пикселей по схожести.

8.3. Варианты заданий

Вариант 1: Медицинская сегментация

Задание: Выполните сегментацию медицинского изображения (МРТ мозга) для выделения различных тканей: серое вещество, белое вещество, спинномозговая жидкость.

Методические указания:

1. Примените предварительную фильтрацию для устранения шума;
2. Используйте алгоритм Оцу для автоматического выбора порогов;
3. Примените многопороговую сегментацию;
4. Используйте морфологические операции для постобработки;

5. Проведите валидацию результатов с экспертной разметкой.

Вариант 2: Сегментация дорожной сцены

Задание: Проведите сегментацию изображения дорожной сцены: выделите дорогу, автомобили, знаки, пешеходов, здания, небо.

Методические указания:

1. Используйте цветовое пространство HSV для выделения неба и дороги;

2. Примените детектор границ для выделения объектов;

3. Используйте метод водораздела для разделения объектов;

4. Примените морфологические операции для улучшения сегментации;

5. Проведите классификацию сегментированных областей.

Вариант 3: Сегментация растительности

Задание: Выделите различные типы растительности на спутниковом снимке: лес, трава, сельскохозяйственные культуры, водоемы.

Методические указания:

1. Вычислите вегетационные индексы (NDVI, EVI);

2. Используйте кластеризацию K-means в цветовом пространстве;

3. Примените адаптивную пороговую обработку;

4. Используйте текстурные признаки для классификации;

5. Проведите постобработку для устранения шума классификации.

Вариант 4: Промышленная инспекция

Задание: Проведите сегментацию изображения для контроля качества продукции: выделите дефекты поверхности, измерьте размеры деталей.

Методические указания:

1. Создайте эталонную модель изделия;

2. Используйте разностную сегментацию для поиска отклонений;

3. Примените адаптивную пороговую обработку;

4. Используйте морфологические операции для выделения дефектов;

5. Реализуйте автоматическую классификацию типов дефектов.

Вариант 5: Интерактивная сегментация

Задание: Разработайте систему интерактивной сегментации, где пользователь может указать области интереса, и система автоматически дорисует сегментацию.

Методические указания:

1. Реализуйте алгоритм GrabCut для интерактивной сегментации;

2. Добавьте возможность рисования масок переднего и заднего плана;

3. Используйте Graph-Cut оптимизацию;

4. Добавьте возможность корректировки результатов;

5. Реализуйте сохранение и загрузку сессий сегментации.

8.4. Пример кода сегментации

```
import cv2
import numpy as np
from sklearn.cluster import KMeans

# Загрузка и предварительная обработка изображения
img = cv2.imread('image.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 1. Простая пороговая сегментация
_, thresh_binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# 2. Пороговая сегментация по методу Оцу
_, thresh_otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

```
    # 3. Адаптивная пороговая сегментация
    adaptive_thresh = cv2.adaptiveThreshold(
        gray, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY, 11, 2
    )

    # 4. Сегментация по цвету в HSV пространстве
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Диапазон для красного цвета
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red, upper_red)

    # 5. K-means кластеризация
    def kmeans_segmentation(image, clusters=3):
        data = image.reshape((-1, 3))
        data = np.float32(data)

        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITE-
RIA_MAX_ITER, 20, 1.0)
        _, labels, centers = cv2.kmeans(
            data, clusters, None,
            criteria, 10, cv2.KMEANS_RANDOM_CENTERS
        )

        centers = np.uint8(centers)
        segmented_data = centers[labels.flatten()]
        return segmented_data.reshape(image.shape)
```

```
segmented_image = kmeans_segmentation(img, 3)

# 6. Сегментация методом водораздела
def watershed_segmentation(gray_image):
    # Бинаризация
    _, binary = cv2.threshold(
        gray_image, 0, 255,
        cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU
    )

    # Удаление шума
    kernel = np.ones((3, 3), np.uint8)
    opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel, iterations=2)

    # Определение фона
    sure_bg = cv2.dilate(opening, kernel, iterations=3)

    # Определение переднего плана
    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
    _, sure_fg = cv2.threshold(
        dist_transform,
        0.7 * dist_transform.max(),
        255, 0
    )

    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(sure_bg, sure_fg)

    # Маркировка компонентов
```

```
    _, markers = cv2.connectedComponents(sure_fg)
    markers = markers + 1
    markers[unknown == 255] = 0

    # Применение алгоритма водораздела
    markers = cv2.watershed(img, markers)
    img[markers == -1] = [255, 0, 0]  # Помечаем границы

    return img

watershed_result = watershed_segmentation(gray)

# Отображение результатов
cv2.imshow('Original', img)
cv2.imshow('Binary Threshold', thresh_binary)
cv2.imshow('Otsu Threshold', thresh_otsu)
cv2.imshow('Adaptive Threshold', adaptive_thresh)
cv2.imshow('Color Segmentation', mask_red)
cv2.imshow('K-means (3 clusters)', segmented_image)
cv2.imshow('Watershed Segmentation', watershed_result)

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

9. Лабораторная работа 6: Анализ и распознавание объектов

9.1. Цель работы

Изучение методов поиска контуров, вычисления характеристик объектов и распознавания простых геометрических форм.

9.2. Теоретические основы

Анализ объектов включает несколько этапов:

- **Поиск контуров** – обнаружение границ объектов;
- **Вычисление характеристик** – площадь, периметр, центроид, моменты;
- **Распознавание форм** – классификация объектов по геометрическим признакам;
- **Анализ дескрипторов** – использование инвариантных признаков.

9.3. Варианты заданий

Вариант 1: Счетчик объектов на конвейере

Задание: Создайте систему подсчета различных деталей на конвейере: болты, гайки, шайбы. Система должна классифицировать объекты по форме и подсчитывать количество каждого типа.

Методические указания:

1. Используйте фоновое вычитание для выделения движущихся объектов;

2. Примените морфологические операции для улучшения контуров;

3. Вычислите дескрипторы формы: компактность, удлиненность, выпуклость;

4. Создайте классификатор на основе геометрических признаков;

5. Реализуйте трекинг объектов для предотвращения повторного подсчета.

Вариант 2: Анализ биологических образцов

Задание: Проанализируйте микроскопическое изображение клеток: подсчитайте количество клеток, измерьте их размеры, определите стадию деления клеток.

Методические указания:

1. Используйте сегментацию методом водораздела для разделения клеток;

2. Вычислите морфологические характеристики каждой клетки;

3. Используйте соотношение периметра к площади для анализа формы;

4. Примените анализ текстуры для определения внутренней структуры;

5. Создайте систему классификации стадий клеточного цикла.

Вариант 3: Распознавание дорожных знаков

Задание: Разработайте систему распознавания дорожных знаков: обнаружение знаков на изображении, классификация по форме (круглые, треугольные, восьмиугольные), определение типа знака.

Методические указания:

1. Используйте цветовую сегментацию для выделения знаков;

2. Примените преобразование Хо для обнаружения кругов и прямых линий;

3. Вычислите дескрипторы формы для классификации;

4. Используйте аппроксимацию контуров для определения количества вершин;

5. Реализуйте проверку на стандартные пропорции дорожных знаков.

Вариант 4: Контроль качества печатных плат

Задание: Создайте систему контроля качества печатных плат: обнаружение отсутствующих компонентов, проверка правильности пайки, измерение размеров дорожек.

Методические указания:

1. Создайте эталонное изображение для сравнения;

2. Используйте регистрацию изображений для выравнивания;

3. Примените разностное изображение для поиска отклонений;

4. Используйте морфологические операции для анализа дорожек;

5. Реализуйте автоматическую генерацию отчета о дефектах.

Вариант 5: Интеллектуальная система сортировки

Задание: Разработайте систему для автоматической сортировки различных объектов (фрукты, детали, документы) на основе их визуальных характеристик.

Методические указания:

1. Реализуйте многоклассовую систему распознавания;

2. Используйте комбинацию цветовых, текстурных и формовых признаков;

3. Примените машинное обучение для улучшения классификации;

4. Добавьте возможность обучения новым классам объектов;

5. Реализуйте систему контроля качества сортировки.

9.4. Пример кода анализа объектов

```
import cv2
import numpy as np

def analyze_objects(image_path):
    """Анализирует объекты на изображении и определяет их характеристики."""
    # Загрузка и предобработка изображения
    img = cv2.imread(image_path)
    if img is None:
        raise ValueError("Не удалось загрузить изображение")

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Бинаризация с адаптивным порогом
    binary = cv2.adaptiveThreshold(
        blurred, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
```

```
    cv2.THRESH_BINARY_INV, 11, 2
)

# Поиск контуров
contours, _ = cv2.findContours(
    binary, cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE
)

# Словарь для классификации форм
SHAPE_NAMES = {
    3: "triangle",
    4: "rectangle",
    "circle": "circle",
    "default": "polygon"
}

# Анализ каждого контура
for i, contour in enumerate(contours):
    # Фильтрация по площади
    area = cv2.contourArea(contour)
    if area < 100:  # Игнорируем слишком маленькие объекты
        continue

    # Вычисление характеристик
    perimeter = cv2.arcLength(contour, True)
    epsilon = 0.02 * perimeter
    approx = cv2.approxPolyDP(contour, epsilon, True)
    vertices = len(approx)
```

```
# Геометрические характеристики
x, y, w, h = cv2.boundingRect(contour)
aspect_ratio = float(w) / h

# Определение формы
shape = SHAPE_NAMES.get(vertices, SHAPE_NAMES["default"])
if vertices == 4:
    shape = "square" if 0.95 <= aspect_ratio <= 1.05 else "rectangle"
elif vertices > 4:
    circularity = (4 * np.pi * area) / (perimeter ** 2)
    if circularity > 0.7:
        shape = SHAPE_NAMES["circle"]

# Дополнительные метрики
hull = cv2.convexHull(contour)
hull_area = cv2.contourArea(hull)
solidity = float(area) / hull_area if hull_area > 0 else 0
compactness = (perimeter ** 2) / (4 * np.pi * area) if area > 0 else 0

# Отрисовка результатов
cv2.drawContours(img, [contour], -1, (0, 255, 0), 2)
cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 1)

# Добавление текста с информацией
info_text = [
    f"Obj {i+1}: {shape}",
    f"Area: {int(area)}",
    f"Verts: {vertices}"
]
```

```
        for j, text in enumerate(info_text):
            cv2.putText(
                img, text, (x, y - 10 - j*15),
                cv2.FONT_HERSHEY_SIMPLEX, 0.4,
                (0, 255, 255), 1
            )

        # Вывод характеристик в консоль
        print(f"\nObject {i+1}: {shape}")
        print(f"  Area: {area:.2f}")
        print(f"  Perimeter: {perimeter:.2f}")
        print(f"  Vertices: {vertices}")
        print(f"  Aspect Ratio: {aspect_ratio:.2f}")
        print(f"  Solidity: {solidity:.2f}")
        print(f"  Compactness: {compactness:.2f}")

    return img


if __name__ == "__main__":
    try:
        result_img = analyze_objects('objects.jpg')
        cv2.imshow('Object Analysis', result_img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    except Exception as e:
        print(f"Error: {str(e)}")
```

9.5. Дополнительные функции для анализа

```
import cv2
```

```
import numpy as np
from typing import Dict, Optional, Union

def calculate_shape_descriptors(contour: np.ndarray) -> Optional[Dict[str, Union[float, np.ndarray]]]:
    """
    Вычисляет геометрические дескрипторы формы для заданного контура.

    Параметры:
        contour : np.ndarray
            Входной контур (массив точек)

    Возвращает:
        Словарь с вычисленными характеристиками формы или None, если площадь контура равна 0
    """
    # Базовые характеристики
    area = cv2.contourArea(contour)
    perimeter = cv2.arcLength(contour, True)

    if area == 0:
        return None

    # Моменты (включая инвариантные моменты Ху)
    moments = cv2.moments(contour)
    hu_moments = cv2.HuMoments(moments)

    # Ограничивающий прямоугольник
    x, y, w, h = cv2.boundingRect(contour)
```

```
    # Выпуклая оболочка
    hull = cv2.convexHull(contour)
    hull_area = cv2.contourArea(hull)

    # Вычисление метрик формы
    descriptors = {
        'area': area,
        'perimeter': perimeter,
        'compactness': (perimeter ** 2) / (4 * np.pi * area),
        'aspect_ratio': float(w) / h if h != 0 else 0,
        'solidity': float(area) / hull_area if hull_area > 0 else 0,
        'circularity': (4 * np.pi * area) / (perimeter ** 2) if perimeter > 0 else 0,
        'hu_moments': hu_moments.flatten(),
        'rectangularity': area / (w * h) if (w * h) > 0 else 0,
        'convexity': cv2.arcLength(hull, True) / perimeter if perimeter > 0 else 0
    }

    return descriptors

def classify_shape(descriptors: Dict[str, Union[float, np.ndarray]]) -> str:
    """
    Классифицирует форму объекта на основе вычисленных дескрипторов.

    Параметры:
        descriptors : dict
            Словарь с характеристиками формы

    Возвращает:
        Строку с названием формы
    """
```

```
    if descriptors is None:
        return "unknown"

    # Извлечение ключевых параметров
    circularity = descriptors['circularity']
    aspect_ratio = descriptors['aspect_ratio']
    solidity = descriptors['solidity']
    rectangularity = descriptors['rectangularity']
    convexity = descriptors['convexity']

    # Правила классификации
    if circularity > 0.85 and solidity > 0.9:
        return "circle"
    elif 0.9 < aspect_ratio < 1.1 and rectangularity > 0.85:
        return "square"
    elif (aspect_ratio < 0.8 or aspect_ratio > 1.2) and rectangularity > 0.8:
        return "rectangle"
    elif convexity < 0.85 and solidity < 0.7:
        return "star_or_cross"
    elif 0.7 < circularity < 0.85:
        return "ellipse"
    elif descriptors['hu_moments'][0] > 0.2:  # Использование моментов Ху
        return "triangle"

    return "irregular"

# Пример использования:
if __name__ == "__main__":
    # Создаем тестовые контуры
    circle_contour = np.array([[[x, y]] for x, y in
```

```
                [(100 + 50*np.cos(t), 100 + 50*np.sin(t))
                 for t in np.linspace(0, 2*np.pi, 32)]])

    # Анализ формы
    descriptors = calculate_shape_descriptors(circle_contour)
    shape_type = classify_shape(descriptors)

    print(f"Detected shape: {shape_type}")
    print("Shape descriptors:", descriptors)
```

10. Заключение

Данное методическое пособие представляет собой комплексное введение в цифровую обработку изображений с использованием языка программирования Python. В ходе изучения материала студенты получили практические навыки работы с основными библиотеками для обработки изображений и освоили ключевые алгоритмы компьютерного зрения.

10.1. Рекомендации по дальнейшему изучению

Для углубления знаний в области цифровой обработки изображений рекомендуется:

1. **Изучение современных методов машинного обучения** – нейронные сети, сверточные нейронные сети (CNN), трансферное обучение;
2. **Освоение дополнительных библиотек** – scikit-image, TensorFlow, PyTorch, YOLO для детекции объектов;
3. **Изучение специализированных областей** – медицинская визуализация, дистанционное зондирование, промышленное машинное зрение;
4. **Практическая работа с реальными проектами** – участие в соревнованиях Kaggle, создание собственных проектов.

10.2. Методические рекомендации для преподавателей

Организация учебного процесса:

1. **Последовательность изучения:** рекомендуется строго следовать порядку лабораторных работ, так как каждая следующая опирается на знания предыдущих.

2. **Распределение вариантов:** варианты заданий можно распределять случайным образом или в соответствии с интересами студентов.

3. **Контроль выполнения:** каждая лабораторная работа должна завершаться демонстрацией результатов и защитой кода.

4. **Групповые проекты:** рекомендуется организовать итоговый проект, объединяющий знания из разных лабораторных работ.

10.3. Критерии оценки выполнения заданий

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)
Правильность алгоритма	Алгоритм реализован корректно, результаты соответствуют ожидаемым	Алгоритм работает с незначительными неточностями	Алгоритм работает, но с существенными ограничениями
Качество кода	Код хорошо структурирован, содержит комментарии, следует стандартам	Код читаем, содержит базовые комментарии	Код работает, но плохо структурирован

Анализ результатов	Проведен глубокий анализ, сделаны обоснованные выводы	Результаты проанализированы, выводы частично обоснованы	Базовый анализ результатов
Творческий подход	Предложены улучшения, дополнительные эксперименты	Показано понимание возможностей развития	Выполнено базовое задание

Изучение цифровой обработки изображений открывает широкие возможности для применения полученных знаний в различных областях: от медицины и безопасности до развлечений и искусства. Успешное освоение представленного материала станет прочной основой для дальнейшего профессионального развития в области компьютерного зрения и искусственного интеллекта.

11. Список рекомендованной литературы

1. **Гонсалес Р., Вудс Р.** Цифровая обработка изображений. – 3-е изд. –М.: Техносфера, 2012. – 1104 с. – ISBN 978-5-94836-326-7.
2. **Гонсалес Р., Вудс Р., Эддинс С.** Цифровая обработка изображений в среде MATLAB. – М.: Техносфера, 2006. – 616 с. – ISBN 5-94836-098-9.
3. **Петру М., Петру К.** Обработка изображений: основы. – 2-е изд. – М.: Техносфера, 2011. – 784 с. – ISBN 978-5-94836-299-4.
4. **Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н.** Цифровая обработка сигналов. – М.: Радио и связь, 1985. – 312 с.

5. **Федорков Б.Г., Телец В.А.** Микросхемы ЦАП и АЦП: функционирование, параметры, применение. – М.: Энергоатомиздат, 1990. – 320 с.

6. **Прэтт У.** Цифровая обработка изображений. – М.: Мир, 1982. – 790 с.

7. **Яне Б.** Цифровая обработка изображений. – М.: Техносфера, 2007. – 584 с. – ISBN 978-5-94836-142-3.

8. **Сойер А., Смит П.** Обработка изображений на языке С. – М.: Радио и связь, 1996. – 320 с. – ISBN 5-256-01495-8.

9. **Шапиро Л., Стокман Дж.** Компьютерное зрение. – М.: БИНОМ, 2006. – 752 с. – ISBN 5-94774-384-3.

10. **Браунинг Д.** OpenCV: практика обработки изображений на Python. – СПб.: Питер, 2022. – 288 с. – ISBN 978-5-4461-2040-6.

Учебное издание

Корчагин Павел Анатольевич, **Корчагин** Илья Павлович

ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

Методические указания к выполнению лабораторных работ