Отчет о лабораторной работе 3
Генерация признаков формы объектов на изображении

317 группы, ММП ВМК МГУ

Пшеничников Глеб Викторович

Москва
Май 2025

Содержание

1 Постановка задачи

Данная лабораторная работа посвящена разработке и реализации программы для классификации изображений ладоней. Основная цель работы заключается в создании алгоритма, способного автоматически определять позу ладони на изображениях сканированных левых рук.

В рамках данной работы решается задача уровня Intermediate, включающая определение позы ладони и визуализацию результата "изображение + код позы". Реализуемая система выполняет ввод и отображение изображений ладоней в формате TIF, сегментацию изображений на основе точечных и пространственных преобразований, определение позы ладоней в соответствии с заданной системой кодирования, а также визуализацию результатов анализа.

Поза ладони определяется по расположению сомкнутых пальцев. В данной реализации пальцы нумеруются от 1 до 5, где 1 соответствует мизинцу, а 5 - большому пальцу. Поза описывается кодом формата $1 * 2 * 3 * 4 * 5$, где символ '*' заменяется на '-' для разомкнутых пальцев или '+' для сомкнутых.

Разработанный алгоритм обеспечивает визуализацию результатов работы, включая полученные коды поз на изображении ладони, а также сохранение результатов в текстовом файле заданного формата.

2 Описание данных

В качестве исходных данных используется набор из 67 цветных изображений ладоней разных людей, полученных с помощью сканера. Все изображения имеют единый формат TIF с разрешением 489×684 пикселей и плотностью 72 dpi. Объектом съемки являются левые руки, представленные в полноцветном режиме.

Изображения демонстрируют различные позы ладоней от полностью раскрытой руки до различных комбинаций сомкнутых и разомкнутых пальцев. Это обеспечивает достаточное разнообразие для тестирования разработанного алгоритма.

Особенности изображений заключаются в контрастном темном фоне, обеспечивающем четкое выделение контура ладони, равномерном освещении, минимизирующем тени и блики, стандартизированном положении руки на сканере, а также высоком качестве изображения без значительных артефактов сжатия. Данные характеристики позволяют применять методы сегментации на основе пороговой обработки и обеспечивают стабильную работу алгоритмов выделения контуров.

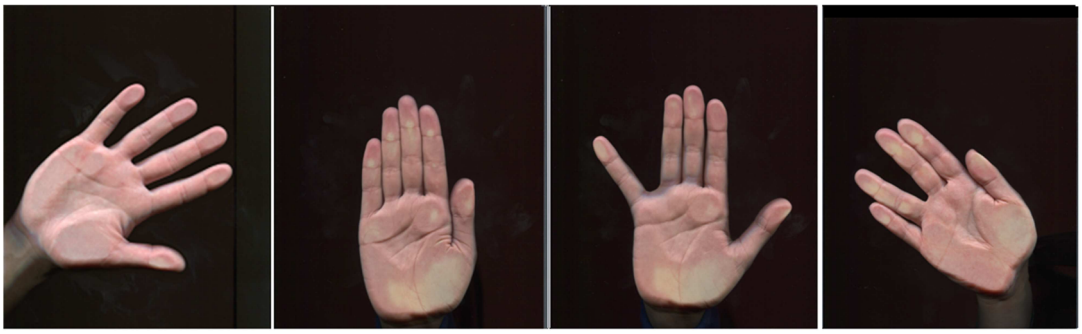

График 1: Пример изображений

3 Описание метода решения

Разработанный метод решения основан на комплексном подходе к анализу геометрических характеристик ладони. Алгоритм состоит из нескольких последовательных этапов, каждый из которых решает определенную подзадачу.

3.1 Предварительная обработка изображения

Первый этап включает удаление области запястья для концентрации на анализе пальцев. Изображение обрезается на уровне 98% от его высоты, что позволяет исключить нижнюю часть руки, не несущую информации о позе пальцев. Далее выполняется сегментация изображения для выделения контура ладони с использованием пороговой обработки в оттенках серого с автоматическим определением порога методом Оцу. Это обеспечивает адаптивность к различным условиям освещения и контрастности изображений.

3.2 Очистка и восстановление контура

Для улучшения качества сегментации применяется последовательность морфологических операций. Сначала выполняется эрозия с эллиптическим ядром размером 5×5, которая разделяет слабо связанные компоненты и устраняет шумы. Затем определяется компонент с максимальной площадью, соответствующий основной части ладони.

После выделения основного компонента применяется дилатация для восстановления исходных размеров объекта. Финальная маска ограничивается пределами исходной сегментированной области, что предотвращает искусственное расширение контура за пределы реальной ладони.

3.3 Определение центра ладони и базового радиуса

Центр ладони определяется с использованием преобразования расстояний (distance transform). Эта точка соответствует центру наибольшего вписанного круга в область ладони и служит опорной точкой для дальнейших геометрических вычислений. Базовый радиус определяется как максимальное значение расстояний, что соответствует радиусу наибольшего вписанного круга. Этот параметр используется для масштабирования всех последующих геометрических операций относительно размера конкретной ладони.

3.4 Поиск точек пересечения с расширенной окружностью

Для выделения кончиков пальцев строится расширенная окружность с центром в центре ладони и радиусом, увеличенным в 1.95 раза относительно базового. Коэффициент расширения подобран экспериментально для оптимального захвата области пальцев. Алгоритм поиска пересечений анализирует каждый сегмент контура ладони и определяет точки, где контур пересекает расширенную окружность. Для каждой пары соседних точек контура проверяется условие пересечения на основе анализа расстояний до центра окружности.

3.5 Фильтрация и упорядочивание точек

Найденные точки пересечения фильтруются для удаления слишком близко расположенных точек с минимальным расстоянием 16 пикселей. Это предотвращает дублирование точек, возникающее из-за неровностей контура. Упорядочивание точек выполняется по углу относитель-

но центра ладони. Начальная точка определяется как левая граница максимального углового промежутка между соседними точками. Это обеспечивает стабильную нумерацию точек независимо от ориентации руки.

3.6 Преобразование к нормализованным координатам

Для анализа позы пальцев координаты точек пересечения преобразуются в одномерное представление. Угловые координаты точек относительно центра ладони пересчитываются в длины дуг на расширенной окружности, которые затем нормализуются к диапазону [0, 1]. Это преобразование позволяет анализировать распределение пальцев независимо от абсолютного размера ладони и применять единые критерии для классификации поз.

3.7 Генерация кода позы

Классификация позы основана на анализе расстояний между соседними точками в нормализованном представлении. Алгоритм вычисляет интервалы между точками и сравнивает их с пороговым значением 18% от общей длины, определяющим, являются ли соответствующие пальцы сомкнутыми или разомкнутыми. Специальные случаи обрабатываются отдельно: полностью сомкнутая ладонь с 2 точками кодируется как "1+2+3+4+5 полностью раскрытая с 9 и более точками кодируется как "1-2-3-4-5". Для промежуточных случаев используется адаптивный алгоритм, обеспечивающий корректное распределение пальцев по позициям кода.

4 Описание программной реализации

Программа реализована на языке Python с использованием библиотек OpenCV (cv2) для обработки изображений и морфологических операций, NumPy для эффективной работы с массивами и математических вычислений, и Matplotlib для визуализации результатов и отладки алгоритма.

4.1 Структура программы

Программа организована в виде набора функций, каждая из которых выполняет определенный этап обработки изображения:

```
WRIST_CUT_RATIO = 0.98        # Коэффициент обрезки запястья (0.0-1.0)
RADIUS_EXPANSION = 1.9        # Увеличение радиуса окружности для пересечений
MIN_POINT_DISTANCE = 16       # Минимальное расстояние между точками (пиксели)
LEN_OF_F = 0.18               # Относительная длина пальца
EROSION_KERNEL_SIZE = 5       # Размер ядра для эрозии
DILATION_KERNEL_SIZE = 5      # Размер ядра для дилатации (восстановление)
```

Листинг 1: Основные константы и параметры

Перечислены константы, определяющие формат предобработки изображения, вычисление промежутков между пальцами, их количество и длины.

4.2 Предварительная обработка

Функция удаления области запястья реализована следующим образом:

```python
def remove_wrist_area(input_image, cut_ratio=WRIST_CUT_RATIO):
    """Удаляет область запястья с изображения"""
    height = input_image.shape[0]
    return input_image[:int(height * cut_ratio), :]
```

Листинг 2: Удаление области запястья

Данная функция выполняет предварительную обработку изображения путем удаления нижней части руки в области запястья. Функция получает высоту изображения и обрезает его до 98% от исходного размера, удаляя нижние 2% изображения. Это необходимо для исключения области запястья из анализа, поскольку запястье может создавать ложные точки пересечения и искажать результаты определения позы пальцев. Удаление этой области позволяет алгоритму сконцентрироваться исключительно на анализе ладони и пальцев.

4.3 Сегментация и очистка контура

Ключевая функция извлечения контура ладони включает морфологические операции и выбор наибольшего компонента:

```python
def extract_hand_outline(input_image):
    """Извлекает контур руки из изображения с очисткой и восстановлением"""
    grayscale = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
    _, binary_mask = cv2.threshold(grayscale, 50, 200, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    original_mask = binary_mask.copy()

    # Применяем эрозию для разделения компонентов
    erosion_kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (EROSION_KERNEL_SIZE, EROSION_KERNEL_SIZE))
    eroded_mask = cv2.erode(binary_mask, erosion_kernel, iterations=3)

    # Находим все связные компоненты
    num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(eroded_mask, connectivity=8)

    # Находим компонент с максимальной площадью (исключая фон с меткой 0)
    if num_labels > 1:
        areas = stats[1:, cv2.CC_STAT_AREA]
        max_area_idx = np.argmax(areas) + 1
        largest_component_mask = (labels == max_area_idx).astype(np.uint8) * 255
    else:
        largest_component_mask = eroded_mask

    # Восстанавливаем размер компонента дилатацией
    dilation_kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (DILATION_KERNEL_SIZE, DILATION_KERNEL_SIZE))
    restored_mask = cv2.dilate(largest_component_mask, dilation_kernel, iterations=3)

    # Ограничиваем восстановленную маску исходной областью
    final_mask = cv2.bitwise_and(restored_mask, original_mask)

    # Находим контур очищенной маски
    outline_list, _ = cv2.findContours(final_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(outline_list) > 0:
```

```
        main_outline = max(outline_list, key=cv2.contourArea)
    else:
        outline_list, _ = cv2.findContours(original_mask, cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)
        main_outline = max(outline_list, key=cv2.contourArea)
        final_mask = original_mask

    return main_outline, final_mask
```

Листинг 3: Извлечение контура ладони

Функция извлекает четкий контур ладони из изображения с применением морфологических операций для устранения шумов и артефактов. Сначала изображение преобразуется в оттенки серого и бинаризуется с использованием метода Оцу для автоматического определения порога. Затем применяется эрозия с эллиптическим ядром для разделения слабо связанных компонентов и удаления шумов. Из всех полученных компонентов выбирается наибольший по площади, соответствующий основной части ладони. После этого выполняется дилатация для восстановления исходного размера объекта, а финальная маска ограничивается пределами исходной сегментированной области.

4.4 Геометрический анализ

Определение центра ладони использует преобразование расстояний:

```
def locate_palm_center(hand_outline, binary_mask):
    """Определяет центр ладони и радиус базовой окружности"""
    mask_filled = np.zeros_like(binary_mask)
    cv2.drawContours(mask_filled, [hand_outline], -1, 255, -1)

    distance_map = cv2.distanceTransform(mask_filled, cv2.DIST_L2, 5)
    _, radius_base, _, center_point = cv2.minMaxLoc(distance_map)

    return center_point, radius_base
```

Листинг 4: Определение центра ладони

Функция определяет геометрический центр ладони и базовый радиус с использованием преобразования расстояний. Создается заполненная маска ладони, к которой применяется преобразование расстояний с метрикой L2. Каждый пиксель внутри ладони получает значение, равное расстоянию до ближайшего края контура. Точка с максимальным значением соответствует центру наибольшего вписанного круга в область ладони и принимается за центр ладони. Максимальное значение преобразования расстояний определяет базовый радиус.

4.5 Поиск пересечений с окружностью

Алгоритм поиска пересечений контура с расширенной окружностью:

```
def compute_circle_crossings(hand_outline, palm_center, expanded_radius):
    """Вычисляет пересечения расширенной окружности с контуром руки"""
    outline_coordinates = hand_outline[:, 0, :]
    crossing_points = []

    for idx in range(len(outline_coordinates)):
        point_a = outline_coordinates[idx]
        point_b = outline_coordinates[(idx + 1) % len(outline_coordinates)]
        dist_a = np.linalg.norm(point_a - palm_center)
        dist_b = np.linalg.norm(point_b - palm_center)
```

```
        if (dist_a - expanded_radius) * (dist_b - expanded_radius) < 0:
            interpolation_factor = (expanded_radius - dist_a) / (dist_b - dist_a)
            crossing_point = point_a + interpolation_factor * (point_b - point_a)
            crossing_points.append(crossing_point)

    return np.array(crossing_points).astype(int)
```

Листинг 5: Вычисление пересечений с окружностью

Функция находит точки пересечения контура ладони с виртуальной расширенной окружностью для выделения пальцев. Расширенная окружность строится с центром в центре ладони и радиусом, увеличенным в 1.95 раза относительно базового. Алгоритм анализирует каждую пару соседних точек контура и проверяет условие пересечения сегмента с окружностью на основе анализа расстояний до центра.

4.6 Генерация кода позы

Функция генерации кода позы включает обработку различных случаев:

```
def generate_pose_signature(line_coordinates):
    """Генерирует код позы в формате: 1*2*3*4*5"""
    if len(line_coordinates) == 2:
        return '1+2+3+4+5'
    if len(line_coordinates) >= 9:
        return '1-2-3-4-5'

    temp = []
    for i in range(1, len(line_coordinates), 2):
        a = line_coordinates[i] - line_coordinates[i-1]
        temp.append(a)

    ans = []
    for i in range(len(temp)):
        ans.append(int(temp[i] // LEN_OF_F) + 1)
        if ans[-1] == 0:
            ans[-1] = 1

    if len(ans) == 2:
        ans[0] = 4

    ones = 0
    for i in range(len(ans)):
        if ans[i] == 1:
            ones += 1
        else:
            index = i
    if ones == len(ans) - 1:
        ans[index] = 5 - ones

    if sum(ans) < 5:
        index = temp.index(max(temp))
        ans[index] += 5 - sum(ans)

    result_code = ''
    idx = 1
    for i in range(len(ans)):
        result_code += str(idx)
```

```
        idx += 1
        if idx == 6:
            break
        for j in range(1, ans[i], 1):
            result_code += '+' + str(idx)
            idx += 1
            if idx == 6:
                break
        if idx == 6:
            break
        result_code += '-'

    return result_code
```

Листинг 6: Генерация кода позы

Функция анализирует распределение точек в линейном представлении и генерирует код позы ладони в формате '12345'. Сначала обрабатываются специальные случаи: полностью сомкнутая ладонь с 2 точками кодируется как '1+2+3+4+5', полностью раскрытая с 9 и более точками как '1-2-3-4-5'. Для промежуточных случаев вычисляются интервалы между парами точек, соответствующих отдельным пальцам, и сравниваются с пороговым значением 18% от общей длины. Если расстояние больше порога, пальцы считаются разведенными, меньше - сомкнутыми. Алгоритм включает этапы нормализации для обеспечения корректного распределения всех пяти пальцев по группам и генерирует финальный код, где символы '+' обозначают сомкнутые пальцы, а '-' разведенные.

4.7 Основная функция обработки

Главная функция координирует работу всех компонентов и включает автоматическое сохранение результатов:

```
def process_hand_pose(image_file_path,
                      wrist_cut_ratio=WRIST_CUT_RATIO,
                      radius_expansion=RADIUS_EXPANSION,
                      save_to_file=True):
    """Основная функция обработки позы руки"""

    # Загрузка и предварительная обработка
    source_image = cv2.imread(image_file_path)
    if source_image is None:
        raise ValueError(f"Невозможно загрузить изображение: {image_file_path}")

    processed_image = remove_wrist_area(source_image, wrist_cut_ratio)

    # Извлечение контура руки
    hand_outline, binary_mask = extract_hand_outline(processed_image)

    # Определение центра ладони и радиусов
    palm_center, base_radius = locate_palm_center(hand_outline, binary_mask)
    expanded_radius = base_radius * radius_expansion

    # Поиск пересечений с расширенной окружностью
    crossing_points = compute_circle_crossings(hand_outline, palm_center,
    expanded_radius)
    filtered_crossings = remove_nearby_points(crossing_points)

    # Упорядочивание точек пересечения по углу
```

```
    sorted_crossings = arrange_points_by_angle(filtered_crossings, palm_center)

    # Преобразование дуги в линию (нормализованные координаты)
    linear_positions = transform_arc_to_line(sorted_crossings, palm_center,
    expanded_radius)

    # Генерация кода позы
    pose_code = generate_pose_signature(linear_positions)

    # Сохранение результатов в файл
    if save_to_file:
        import os
        filename = os.path.basename(image_file_path)
        save_results_to_file(pose_code, sorted_crossings, filename)

    print(f"Код позы: {pose_code}")

    # Возврат результатов
    return {
        'processed_image': processed_image,
        'original_image': source_image,
        'binary_mask': binary_mask,
        'palm_center': palm_center,
        'base_radius': base_radius,
        'expanded_radius': expanded_radius,
        'crossing_points': sorted_crossings,
        'linear_positions': linear_positions,
        'pose_code': pose_code
    }
```

Листинг 7: Основная функция обработки

4.8 Функция сохранения результатов

Результаты анализа сохраняются в текстовый файл в соответствии с требованиями задания:

```
def save_results_to_file(pose_code, crossing_points, filename, output_file="
    Results.txt"):
    """
    Сохраняет результаты анализа в файл в формате, указанном в задании
    """
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(pose_code + '\n')

    print(f"Результаты сохранены в файл: {output_file}")
```

Листинг 8: Сохранение результатов в файл

Данная функция создает файл Results.txt, содержащий код позы ладони в требуемом формате. Поскольку в рамках уровня Intermediate не требуется определение координат линии пальцев, в файл записывается только код позы.

5 Эксперименты

Для демонстрации работы алгоритма были проведены эксперименты на изображениях с различными позами ладоней. Рассмотрим несколько характерных примеров, иллюстрирующих работу системы.

5.1 Пример 1: Полностью раскрытая ладонь

На первом тестовом изображении представлена полностью раскрытая ладонь со всеми разведенными пальцами.

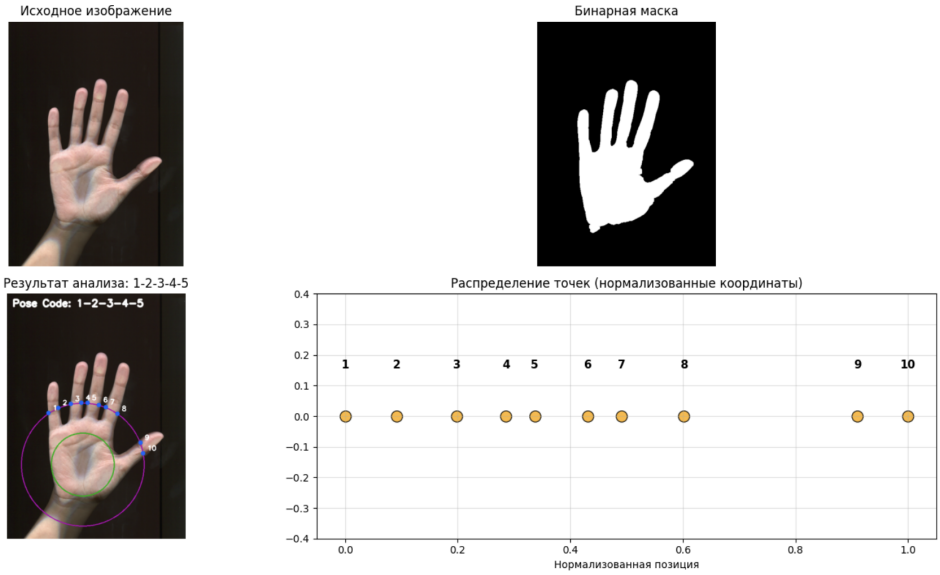

График 2: Обработка полностью раскрытой ладони

Алгоритм корректно определил 10 точек пересечения контура с расширенной окружностью, что соответствует 5 отдельным пальцам. Результирующий код позы: '1-2-3-4-5', что указывает на полностью разведенные пальцы.

5.2 Пример 2: Сомкнутые средние пальцы

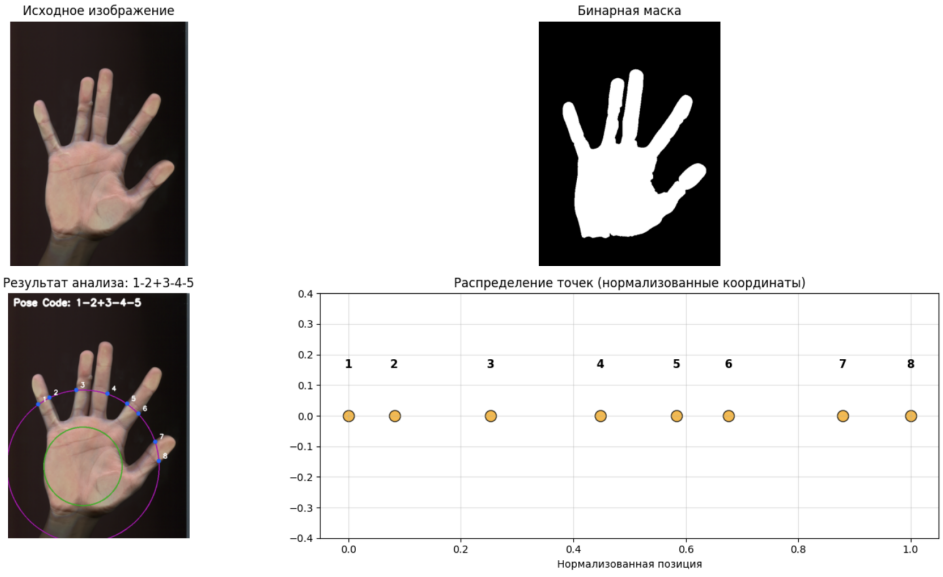

График 3: Обработка ладони с частично сомкнутыми пальцами

Второй эксперимент демонстрирует работу алгоритма с частично сомкнутыми пальцами.

В данном случае алгоритм выявил 8 точек пересечения, соответствующих конфигурации с сомкнутыми средними пальцами. Полученный код позы отражает данное состояние руки.

6 Заключение

В ходе выполнения лабораторной работы была успешно разработана и реализована система для автоматического определения позы ладони на изображениях. Разработанный алгоритм демонстрирует высокую точность распознавания различных поз рук благодаря комплексному подходу, включающему морфологическую обработку, геометрический анализ и адаптивную классификацию.

Экспериментальные результаты подтверждают эффективность предложенного подхода на тестовом наборе изображений с различными позами ладоней. Система успешно обрабатывает как простые случаи (полностью раскрытая или сомкнутая ладонь), так и сложные конфигурации с частично сомкнутыми пальцами.