Отчёт по выполнению второй лабораторной работы по курсу ОиР изображений

Сидоров Леонид 317 группа

15 мая 2021 г.

Постановка задачи

Данная работа состояла из двух частей:

- Задача класса **Intermediate** заключается в построении признакового описания изображений человеческих ладоней.
- Задача класса **Expert** заключается в поиске ближайших соседей и кластеризации полученных признаковых описаний.

В первой задаче алгоритм на вход получает изображение формата *tif*, а выходом является размеченное изображение того же формата. Под разметкой в данном случае понимается ломанная линия, проходящая через основания и кончики пальцев. Чтобы гарантировать сопоставимость ломаных у разных ладоней, зададим у неё направление. Началом линии всегда будем считать кончик большого пальца. Пример обработанного изображения можно увидеть на Рис. 1.

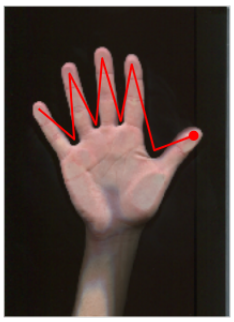

Рис. 1: Пример размеченного изображения.

Кроме того, программа должна выдавать восьмимерный вектор признаков — длины отрезков, составляющих ломаную линию (начиная с большого пальца).

Вторая задача заключается в кластеризации полученной выборки из восьмимерного признакового пространства с целью выяснения количества уникальных ладоней в выборке. Иначе говоря, необходимо определить число людей, чьи ладони представлены на изображениях, и составить списки ладоней для каждого. Стоит отметить, что все рассматриваемые ладони **правые**.

Также нужно найти для каждой ладони 3 наиболее похожих изображения и представить результат в виде таблицы «имя образца – имена ближайших соседей». В задании не указывалось никаких ограничений для используемых методов обработки и распознавания изображений.

Мною было решено разделить задачи классов **Intermediate** и **Expert** на два программных блока: первая задача будет реализована посредством цельной программы на языке *Python*, а вторая — через *Jupiter Notebook*, где будут последовательно проведены эксперименты. Признаковые описания объектов выборки и результаты кластеризации и поиска наиболее похожих объектов находятся в папке *output* в формате *csv*.

Описание данных

В качестве исходных данных прилагается набор из 99 цветных изображений ладоней разных людей в формате *tif*, полученных с помощью сканера, в формате 489×684 с разрешением 72 dpi. Однако одно из изображений уже размечено (Рис. 2), поэтому мы не можем использовать его при обучении и работаем в итоге с 98 цветными изображениями.

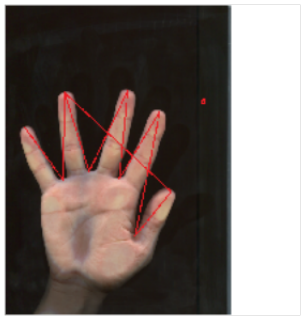

Рис. 2: Некорректный объект обучающей выборки

Как уже было отмечено, изображения получены при помощи сканера, поэтому нам гарантированы однородное освещение, однотонный фон и одинаковый масштаб. Кроме того, вспомним, о том, что все ладони в нашей выборке правые.

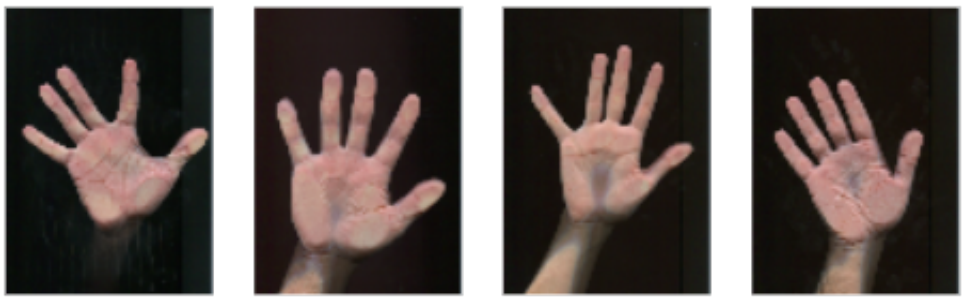

Рис. 3: Исходный набор изображений

Описание метода решения

Исходные задачи будем решать в последовательно и в изначальном порядке.

Задача класса Intermediate

Сегментация изображения

Как и в прошлой лабораторной работе, начнём с выделения сегментационной маски изображения.

Все ладони (даже людей других рас) имеют примерно однородный цвет из одного диапазона. Переведём изображение из цветовой схемы **RGB** в цветовую схему **HSV**, она позволяет гораздо эффективнее выделять цветовые диапазоны, ведь главный оттенок там определяется углом от 0 до 360 градусов (также мы можем отдельно задавать яркость и насыщенность цвета).

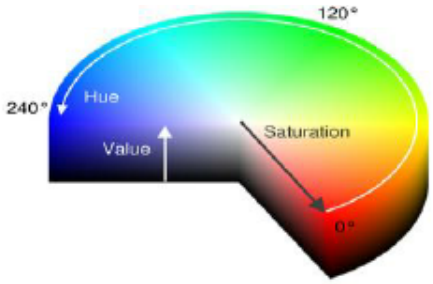

Рис. 4: Цветовая схема HSV

Такой подход действительно позволил выделить ладонь на фотографии. К полученной бинарной маске применяем операцию замыкания, чтобы избавиться от шума на фоне изображения. В этот раз объект интереса (ладонь) однороден в плане цвета, поэтому операция размыкания нам не понадобится. Также применяем медианный фильтр, чтобы сгладить изображение и избавиться от мелких неровностей на границе бинарной маски, этот шаг важен для следующего этапа обработки.

Выделение скелета изображения

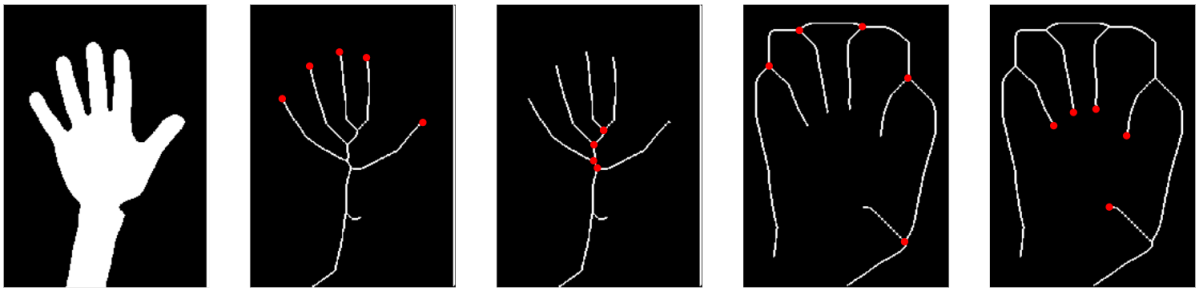

Рис. 5: Внешние и внутренние вершины скелета изображения

Вполне очевидной идеей по нахождению кончиков пальцев является построение скелета ладони и выделение его вершин. Для поиска оснований произведём аналогичную операцию: инвертируем маску ладони и опять построим скелет изображения.

Используемая мной библиотечная функция предоставляет исчерпывающую информацию о построенном скелете в виде таблицы (Рис. 6). Произведём первичную обработку этих данных. Сначала выделим внешние рёбра скелета, после чего исключим слишком длинные или слишком короткие рёбра. Последним шагом будет удаление вершин скелета, находящихся на краю изображения, и его лишних связных компонент.

	skeleton-id	node-id-src	node-id-dst	branch-distance	branch-type	image-coord-src-0	image-coord-src-1	image-coord-dst-0	image-coord-dst-1
0	1	1	2092	698.656854	0	0.0	502.0	697.000000	504.000000
1	2	118	949	212.531344	1	117.0	221.0	311.333333	264.666667
2	2	147	949	194.968879	1	131.0	288.0	311.333333	264.666667
3	2	209	950	234.242022	1	152.0	147.0	347.333333	240.666667
4	2	533	955	245.627200	1	233.0	79.0	387.000000	239.750000

Рис. 6: Таблица с данными о скелете изображения

В итоге, для каждого скелета изображения (внешнего и внутреннего) мы получим множество рёбер, из которого мы выделим ещё два множества вершин: внешние и внутренние. Примеры этих множеств изображены на (Рис. 5).

Для этой демонстрации было намерено подобрано "удачное"изображение, скелет которого почти не содержит лишних вершин, однако даже в этом случае мы видим на внешнем скелете снизу одно лишнее ребро, которое негативно скажется на построении линии пальцев. Обработкой подобных "лишних"рёбер и вершин мы займёмся в следующем пункте.

Построение линии пальцев

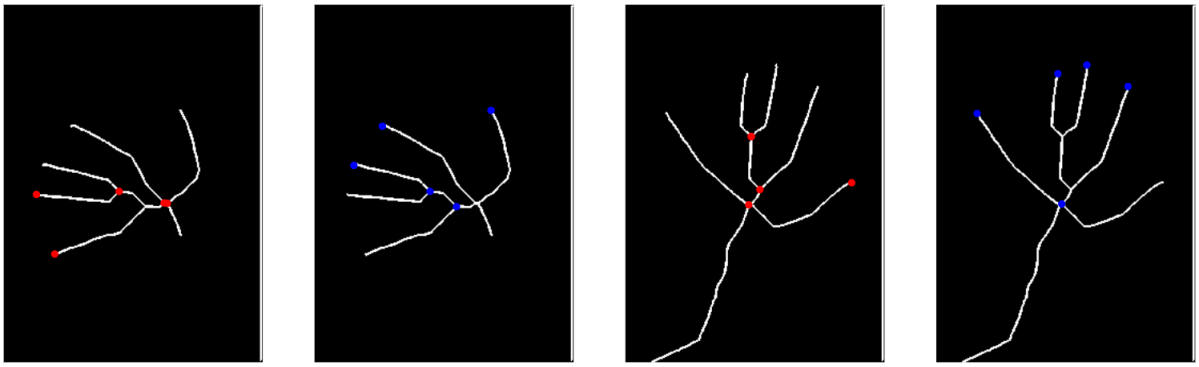

Рис. 7: Примеры не лучших инициализаций множеств вершин

Основной идеей этого пункта является поиск кончика большого пальца на изображении, а затем поочерёдное добавление к ломанной ближайших вершин скелета из множеств оснований и кончиков пальцев. На выходе должна получиться ломаная вида (Рис. 1).

Хоть ранее и было подмечено, что у скелета изображения одно множество отвечает за внешние вершины, а другое за внутренние, эти множества могут меняться местами или перемешиваться. Поэтому, чтобы решить первую проблему, мы вычисляем сумму попарных расстояний между элементами одного множества вершин и сравниваем её с заранее определённым порогом, чтобы понять, с каким множеством точек мы имеем дело.

Также нам могут попадаться вершины кисти, находящиеся на слишком большом расстоянии от основного скопления точек и не являющиеся элементами ладони. Избавиться от таких вершин мы можем при помощи анализа суммы расстояний до остальных точек. Если оно слишком велико, то это либо выброс, либо вершина кисти.

Кончиком большого пальца мы считаем одну из оставшихся вершин, наиболее удалённую от остальных. Однако даже так мы можем ошибиться и выбрать внутреннюю вершину ребра, соответствующего большому пальцу, а не внешнюю. Чтобы обойти эту проблему, применим ещё одну эвристику: если в данных координатах содержится не одна вершина, то заменим её на противоположный конец ребра (всю эту информацию можно получить из исходной таблицы скелета). Кроме того, при поиске кончика большого пальца стоит также обращать внимание на расстояние до других внешних и внутренних вершин (они должны быть максимальны).

Проверка на наличие вершин в тех же координатах легко обобщается на случай произвольного пальца. Примеры описанных выше ситуаций можно увидеть на (Рис. 7).

Получение признакового описания ладони

Поскольку в предыдущем пункте мы получили упорядоченные координаты точек ломаной, начинающейся на кончике больного пальца, придти к признаковому описанию совсем несложно. Направление ломаной позволяет говорить нам об однозначности полученного представления, а значит нам достаточно пройти по её звеньям и записать соответствующие расстояния. Однако не стоит пренебрегать выбросами — неудачно распознанными изображениями.

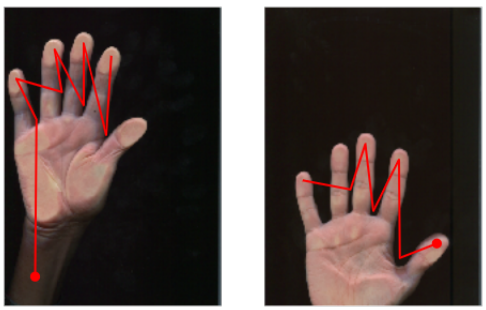

Рис. 8: Примеры простейших ошибок в построении ломаной

В подобных изображениях встречаются ошибки двух типов: недостаток звеньев ломанной и лишние звенья ломанной (Рис. 8). Переизбыток звеньев мы решаем с помощью жёсткого ограничения их количества, а недостающие с конца звенья мы заменяем медианой по соответствующему признаку.

Итогом этого этапа являются визуально размеченные изображения и таблица их признаков.

Задача класса Expert

В этом разделе мы решаем две подзадачи: поиск ближайших соседей и кластеризацию по полученным признаковым описаниям. Для обеих задач сначала нам следует определиться с понятием "похожести"двух объектов, то есть с используемой

мерой расстояния. Масштаб признаков является для нас важным признаком, ведь так мы сможем, например, отличить детскую или женскую руку от мужской. Поэтому наша метрика должна быть чувствительна к масштабу, а также учитывать индивидуальные различия признаков. Эти свойствами обладает стандартное евклидово расстояние. Мы будем использовать его во всех последующих экспериментах.

После выбора меры расстояния для решения первой подзадачи нам остаётся лишь воспользоваться функцией поиска ближайших соседей и получить результат, он хранится в файле *neighbors.csv*.

Решение второй задачи чуть менее тривиально. Нам заранее неизвестно количество уникальных ладоней на снимках. Поэтому я рассмотрел два подхода к решению этой задачи. Первый подход заключается в использовании алгоритма кластеризации **DBScan**, который самостоятельно определяет количество кластеров в исходных данных. Второй подход состоит в применении алгоритма **Kmeans** и последовательном переборе количества кластеров с целью оптимизации коэффициента силуэта $Silhouette = \frac{1}{N}\sum_{i=1}^{N}\frac{d_i - s_i}{max\{d_i, s_i\}}$, где N — количество объектов в выборке, s_i — среднее расстояние от рассматриваемого объекта до объектов того же кластера, а d_i — среднее расстояние от рассматриваемого объекта до объектов ближайшего соседнего кластера.

Как оказалось, **DBScan** плохо решает исходную задачу, потому что данные имеют примерно одинаковую плотность на всём признаковом пространстве в пределах облака точек. Однако для этого алгоритма нашлось другое применение. Известно, что **DBScan** способен определять определять выбросы. Воспользуемся этим его свойством, чтобы исключить из выборки неверно распознанные изображения. Подробное описание этого шага будет представлено в разделе **эксперименты**. Здесь и далее мы будем использовать **t-SNE** представление пространства, чтобы визуализировать результаты (Рис. 9).

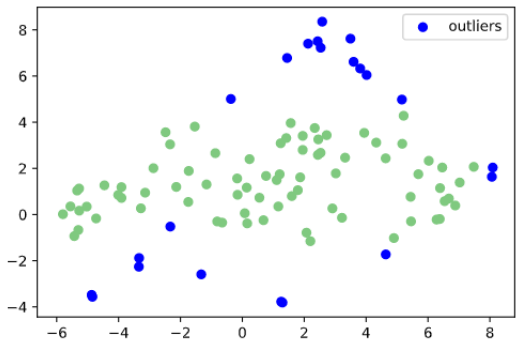

Рис. 9: Выбросы, размеченные алгоритмом DBScan, в t-SNE представлении

Как показали эксперименты (см. соответсвующий раздел), **Kmeans** следует применять именно после удаления выбросов. Значение количества кластеров перебиралось в диапазоне от 5 до 50, потому что именно такие ограничения показались мне разумными после зрительного анализа выборки (обычно мы имеем не менее двух фотографий одной ладони).

Итогом этого этапа являются количество кластеров (уникальных ладоней) и таблица с элементами этих кластеров. Поскольку причисление выбросов к какому-либо кластеру лишено смысла, мы будем хранить их в отдельной категории — -1 кластере.

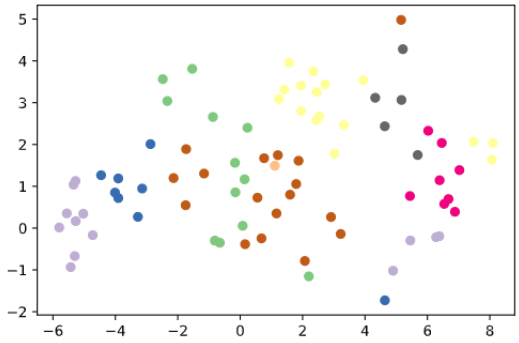

Рис. 10: Итоговое разбиение на кластеры в t-SNE представлении

Описание программой реализации

Как уже упоминалось выше, алгоритм структурно разделён на две части, это также отражается и на программной реализации.

Задача класса Intermediate

Программа была написана на языке **Python** с использованием таких библиотек как **OpenCV**, **PIL**, **skimage** и **scipy**. Полный список представлен в файле *requirements.txt*.

В папке с программой должны находиться папки *data*, в которой должны находиться изображения, подлежащие обработке, и *output*, в которой будут сохраняться размеченные изображения. В папке *data* следует располагать фотографии формата *tif*, файлы другого формата обрабатываться не будут, названия документов могут быть произвольными.

Кроме размеченных изображений в папке *output* будут находиться два *csv* файла: *dists.csv* и *labels.csv*. В них будут записаны признаковое описание фотографий и их номера соответственно. Первый признак имеет смысл расстояния от кончика до основания большого пальца, остальные признаки являются упорядоченными длинами последующих участков ломанной.

На тестовых данных обработка одного изображения в среднем занимает одну секунду. Программа реализует базовую систему логирования, позволяющую понять, какое изображение обрабатывается в данный момент и сколько времени потребовалось на обработку всех изображений. Эта информация выводится в консоль.

Задача класса Expert

Алгоритм реализован на базе *Jupiter Notebook*, для получения результата достаточно просто последовательно выполнить все ячейки кода. Код написан на языке **Python** с использованием библиотек **numpy** и **sklearn**.

На вход ноутбук принимает два *csv* файла: *dists.csv* и *labels.csv*. Эти файлы являются выходом программы для реализации задачи класса **Intermediate**, находится они должны в одной папке с самим ноутбуком. Для удобства проверки в архиве заранее хранятся оба требуемых файла, содержащих информацию о всей обучающей выборке.

Результаты работа алгоритма, как и на **Intermediate** этапе, будут сохраняться в папке *output*. Файл *neighbors.csv* содержит таблицу формата «имя образца – имена ближайших соседей», а файл *persons.csv* — таблицу формата «Персона № – имена изображений ладоней». Столбцы данных таблиц не подписаны, поскольку их смысл интуитивно понятен. Кроме того в случае *persons.csv* это сделать почти невозможно. Первый столбец этой таблицы содержит либо номер человека, которому принадлежат все ладони на данной строке, либо число -1, помечающее объекты данной строки как выбросы (нераспознанные фотографии).

Блокнот сообщает пользователю, какое число кластеров он выделил за время работы алгоритма.

Эксперименты

Начальная постановка задачи не предполагала наличия какой-либо метрики качества, которая позволила бы измерить удачность разметки. Также у нас нет примера уже размеченных данных, с которыми можно было бы сравнить результат. Поэтому остаётся только оценить правильность разметки визуально, благо тестовая выборка состоит всего из 98 изображений.

По итогу анализа я выяснил, что были неправильно размечены 20 файлов, а значит точность предложенного мною алгоритма составляет 79.6%. В некоторых изображениях неправильно распознано всего одно звено, в других же ломаная не имеет ничего общего с подразумеваемой линией пальцев. Исходя из этой информации, мы могли бы более точно оценить качество работы модели, но это потребовало бы либо строгой формализации, либо куда большего объёма монотонной работы по разметке.

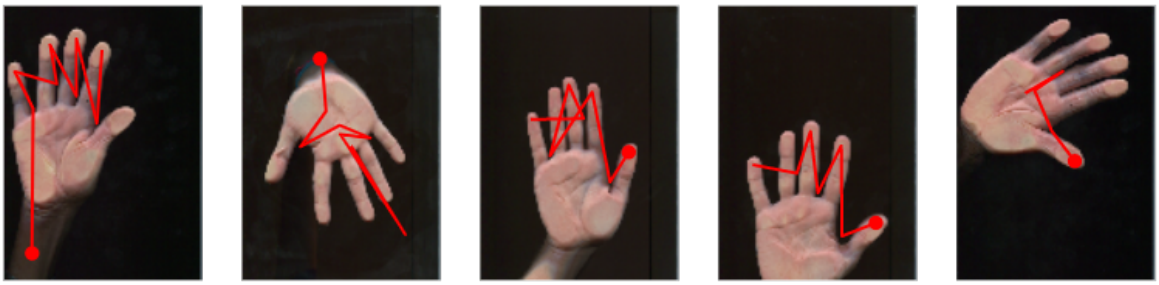

Рис. 11: Примеры различных ошибок построения линии пальцев

Теперь обратимся к задаче класса **Expert**, она носит более исследовательский характер, а потому требует большего количества экспериментов.

Как уже было подмечено в предыдущих разделах, алгоритм **DBScan** плохо делит исходную выборку на кластеры, однако весьма успешно определяет неверно распознанные изображения. При значении параметра $eps = 70$ алгоритм выявил 18 **выбросов**, всего один **(!)** из которых являлся правильно размеченным объектом. Почти все эти объекты имели значительные отклонения разметки от нормальной. Остальные три невыявленные DBScan ошибочные разметки почти не отличаются от корректно распознанной ладони (Рис. 12), и их наличие в выборке не сказывается на дальнейшем анализе.

Эксперименты показали, что если оставить выявленные выбросы в выборке, то

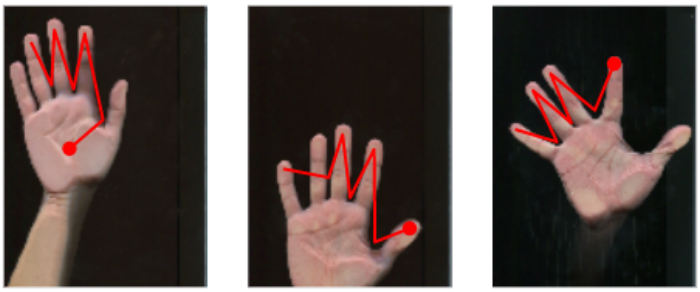

Рис. 12: Невыявленные DBScan ошибки

Kmeans просто не сможет обучиться. Коэффициент силуэта будет монотонно убывать с увеличением количества кластеров, а лучшее его значение будет соответствовать абсолютно бессмысленному разбиению (Рис. 13), в котором есть только два кластера, и один из них представлен всего одним объектом. Поэтому неудачно распознанные объекты необходимо удалять из обучающей выборки.

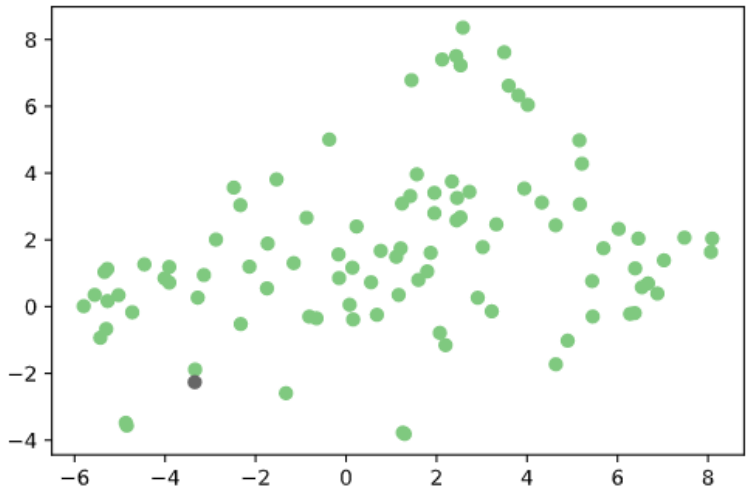

Рис. 13: Итог работы Kmeans на полной выборке в t-SNE представлении

Ранее уже упоминалось, что я перебираю количество кластеров от 5 до 50 штук, анализируя при этом значения коэффициента силуэта выборки. Я понимаю, что такой подход хорош только при наличии выпуклых кластеров, однако такой подход был выбран за неимением лучших альтернатив (DBScan показал крайне неудачный результат).

Значения коэффициента силуэта находятся в весьма узком диапазоне от 0.2 до 0.3, перебираемых значений весьма много, поэтому я не буду приводить их в данном докладе.

Метод **Kmeans** весьма сильно зависит от инициализации, поэтому нам важно проверить устойчивость наших прогнозов. При первом запуске перебор отдал предпочтение разбиению данных на 14 кластеров. Теперь для надёжности проведём ещё 100 симуляций, их результаты представлены на Рис. 14.

Алгоритм имеет простор для улучшений в плане стабильности, однако всё же разброс числа кластеров не так велик и лежит в диапазоне от 10 до 15. Среднее зна-

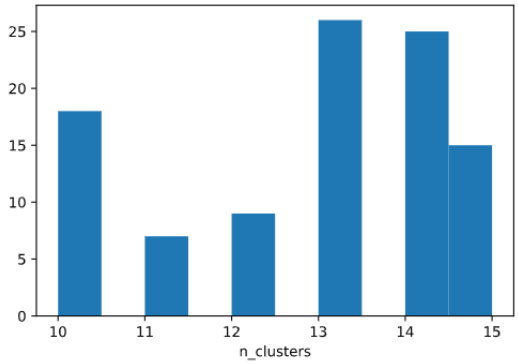

Рис. 14: Гистограмма значений параметра n_cluster, построенная по выборке из 100 испытаний.

чение этого параметра равно 12.78, что можно округлить до 13. Это число является оценкой количества уникальных ладоней людей на фотографиях.

Выводы

По итогам данной работы мною был предложен и программно реализован алгоритм по кодированию изображения человеческой ладони в восьмимерный вектор, а также проведён анализ закодированной таким образом информации с целью поиска похожих объектов и выделения групп одинаковых ладоней (принадлежащих одному человеку).

Алгоритм кодирования успешно справляется со своей задачей на 80% тренировочной выборки, хотя и представляет из себя набор эвристик, применённых к скелету изображения. Нераспознанные 20% датасета легко отслеживаются при помощи алгоритмов кластеризации. Я уверен, что при более точной настройке алгоритм способен повысить долю распознаваемых объектов или даже обработать выборку целиком.

Мне сложно судить об успехе второй части задания, поскольку у нас нет объективных критериев качества для этой задачи. Тем не менее был проведён ряд экспериментов, раскрывающих свойства полученного векторного представления. Также мы получили некоторое разбиение исходных объектов на кластеры, не противоречащее здравому смыслу, и группы ближайших соседей для каждого объекта в датасете.