УДК 004.855.5
DOI 10.25205/1818-7900-2019-17-3-73-92

Разработка мобильного приложения визуализации технических характеристик устройств под ОС Android на базе технологий дополненной реальности и методов машинного обучения

И. С. Прокопьев [1], **Е. С. Шмаков** [2]

[1] *Новосибирский государственный университет*
Новосибирск, Россия

[2] *ООО «БНС»*
Новосибирск, Россия

Аннотация

В последнее время увеличивается количество окружающих нас электронных приборов, возрастает и их техническая сложность. В то же время инструкции к данным приборам читает только малая часть людей. Также возможны ситуации, когда инструкции или специалиста нет рядом с устройством. Поэтому часто встречаются случаи неправильного использования техники. В связи с этим закономерно встает задача быстрого распознавания типа устройства с последующим выводом информации о данном устройстве. В работе описывается процесс реализации системы распознавания типа устройства с использованием методов машинного обучения, с последующим выводом на экран его основных заранее отобранных характеристик и отображением пользователю набора вероятных инструкций для данного типа устройств.

Поскольку направлением данной работы является нахождение похожих объектов, а также их классификация на основе информации о внешнем представлении объекта, то рассматриваются алгоритмы получения признаков объекта из его изображения. В статье предлагается подход получения определяющих признаков объекта на основе его внешнего представления (контуров).

Ключевые слова

распознавание объектов, машинное обучение, дополненная реальность, контуры объекта, признаки изображения

Для цитирования

Прокопьев И. С., *Шмаков Е. С.* Разработка мобильного приложения визуализации технических характеристик устройств под ОС Android на базе технологий дополненной реальности и методов машинного обучения // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 73–92. DOI 10.25205/1818-7900-2019-17-3-73-92

Development Android Application for Visualization of Technical Characteristics Based on Augmented Reality and Machine Learning Methods

I. S. Prokopyev [1], **E. S. Shmakov** [2]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation

[2] *LLC "BNS"*
Novosibirsk, Russian Federation

Abstract

Nowadays the number of electronic devices has increased as well as their complexity. At the same time only few people are reading user guides for these devices. Moreover, in certain cases even no guides are provided by the manufacturer. Therefore there is a large number of cases when people use such devices incorrectly. This issue could be possi-

bly solved with a system automatically recognizing the type of the device. Such system could provide all necessary user information about that device. This article suggests one of possible implementations of such device type recognition system. It recognizes type of device in an unsupevised way and shows main characteristics and user guides for the recognized gadget.

Our approach for constructing such system relies on machine learning methods since greedy search for an object pattern is not efficient, as it was found out by recent scholarly works. Moreover, automatic object patterns classifiers show higher performance in this task and allow to scale the system to various kinds of input data. The algorithms that we are using for object classification are based on feature extraction from an graphical representation of the object look. This representation is usually proposed in an digital photography format. We consider our study as the first work towards automated defining characteristics of a device based on its graphical representation.

Keywords

object recognition, machine learning, augmented reality, object contours, image features

For citation

Prokopyev I. S., Shmakov E. S. Development Android Application for Visualization of Technical Characteristics Based on Augmented Reality and Machine Learning Methods. *Vestnik NSU. Series: Information Technologies*, 2019, vol. 17, no. 3, p. 73–92. (in Russ.) DOI 10.25205/1818-7900-2019-17-3-73-92

В последнее время увеличивается количество окружающих нас электронных приборов, возрастает и их техническая сложность. В то же время инструкции к данным приборам читает только малая часть людей. Возможны ситуации, когда инструкции или специалиста нет рядом с устройством, поэтому часто встречаются случаи неправильного использования техники. В связи с этим закономерно встает задача быстрого распознавания типа устройства с последующим выводом информации о данном устройстве. Но для реализации системы, хранящей информацию обо всех устройствах, требуется огромный объем данных, который обработать вручную не представляется возможным. Поиск объекта по шаблону в данной задаче является нерациональным, так как любой такой алгоритм составляется вручную с использованием перебора большого количества вариантов классификации объектов. Сложности возникают при распознавании объектов с разных ракурсов, при различном освещении и при деформации. Методы машинного обучения позволяют обучить модель с использованием различных видов входных данных, что дает высокий уровень распознавания объектов. Поэтому решением данной проблемы является использование технологий машинного обучения.

В работе описывается процесс реализации системы распознавания типа устройства, с последующим выводом на экран его основных заранее отобранных характеристик и отображением пользователю набора вероятных инструкций для данного типа устройств.

Поскольку направлением данной работы является нахождение похожих объектов, а также их классификация на основе информации о внешнем представлении объекта, то рассматриваются алгоритмы получения признаков объекта из его изображения. В статье предлагается подход к получению определяющих признаков объекта на основе его внешнего представления. При этом должна иметься возможность корректировки признаков, влияющих на результат распознавания объекта, и корректировка вклада отдельно взятых признаков.

Одним из наиболее популярных методов выделения признаков объекта на основе его внешнего представления является метод SURF. Этот метод основан на поиске ключевых точек изображения. Ключевая точка – это точка, имеющая определенные признаки, существенно отличающие ее от остальных точек изображения. Имея информацию о ключевых точках объекта, можно выполнять сравнение двух объектов или классифицировать объект, имея информацию о том, какой набор особых точек характерен для того или иного типа объектов. После получения ключевых точек изображения для каждой из них высчитывается ее дескриптор. Дескриптор – это вектор, характеризующий особую точку.

Данный алгоритм не подходит к решению задачи, поскольку, имея только ключевые точки (рис. 1), которые являются более характеристикой изображения, нежели объекта, нельзя предсказать признаки устройств. Поэтому в данной работе используется метод, основанный на использовании контурного анализа [1. С. 224].

Рис. 1. Пример работы алгоритма SURF
Fig. 1. SURF Algorithm Example

Контурный анализ представляет собой совокупность методов описания, преобразования, выделения контуров объекта на изображении. Контур в данном контексте – это граница объектов, т. е. совокупность точек (пикселей). Контуры изображения являются областями, которые хранят большое количество информации об объекте и не зависят при этом от уровня освещенности и яркости. Контурный анализ требует минимизацию количества шумов на изображении, поэтому до выделения контуров обычно применяются операции, которые уменьшают количество шумов (размытие, анализ гистограммы и т. д.). Одними из наиболее распространенных алгоритмов выделения контуров являются алгоритмы Adaptive Thresholding и Canny. Рассмотрим их подробнее.

Adaptive Thresholding часто применяют в задаче сегментации изображений. Подход является пороговым методом, при котором выбирается определенное значение яркости T, обычно полученное из анализа гистограммы изображения $f(x, y)$, затем каждый пиксель поочередно сравнивают с выбранным значением, и получившиеся два разбиения соответствуют либо принадлежности точки фону, либо объекту:

.

В алгоритме, использующем адаптивный порог, пороговое значение каждого пикселя зависит от соседних. Берется либо среднее значение окружающих пикселей, либо значение, в которое пиксели, лежащие ближе к исходному пикселю, дают больший вклад, а более удаленные – меньший. Затем для каждого пикселя из получившегося значения для области вычитают заранее заданную константу. Получившееся число является пороговым значением для области. Исходя из того, что данный алгоритм опирается на значения яркости изображения, итоговый результат становится чувствительным к перепадам яркости, теням и другим световым артефактам.

Алгоритм Canny состоит из пяти шагов.

1. Сглаживание. На данном этапе происходит фильтрация шума изображения с помощью фильтра Гаусса размером 5×5. Влияние пикселей при использовании фильтра Гаусса друг на друга обратно пропорционально квадрату расстояния между ними.

2. Нахождение градиентов для каждого пикселя изображения. На данном этапе для вычисления приближенного значения градиентов применяется оператор Собеля.

3. Подавление немаксимумов (Non-maximum suppression). Пикселями границ становятся точки, в которых в направлении вектора градиента наблюдается локальный максимум.

4. Выполнение двойной пороговой фильтрации. На данном шаге потенциальные границы определяются пороговыми значениями.

5. Трассировка области неоднозначности. Происходит подавление границ, не связанных с выделенными на шаге 4.

В результате анализа описанных подходов предложен алгоритм выделения признаков устройств на основе его контуров, а также их связи с техническими характеристиками устройств. Реализованный алгоритм является более гибким и «прозрачным», так как поддерживает настройку вклада каждого контура в получение его признаков и удаление ненужных контуров.

Поскольку не существует отдельного набора данных, подходящего под начальные условия задачи, было решено воспользоваться платформой “Open Image Dataset”. Вторым способом получения данных является использование поискового робота (так называемый скрапер) для обхода сайта производителя или поставщика техники. Полученные данные представляют собой таблицу в виде списка URL-адресов для загрузки данных, а также таблицу соответствий адресов классам изображений. Но в дальнейшем данные, полученные таким способом, не использовались, поскольку невозможно подобрать инструкции к моделям, представленным на изображениях. Поэтому в работе используется скрапер для обхода сайта – реселлера техники, который получает изображения техники вместе с его инструкцией в формате PDF.

Для того чтобы получить технические характеристики объекта, используется поисковый скрапер “Web Scraper”, который обходит сайт-реселлер, получая с каждой страницы товара нужную информацию. Для обхода сайта строится граф переходов между страницами, пагинацией сайта. Также задаются поля веб-страницы, откуда берутся параметры объекта, а также ссылка на его изображение (рис. 2). Результат обхода сайта сохраняется в таблице в формате CSV для последующего использования при кластеризации. Пример полученных данных показан на рис. 3.

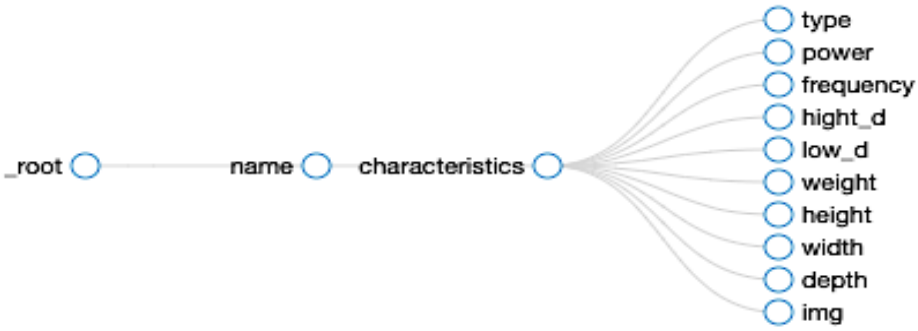

Рис. 2. Граф переходов по сайту

Fig. 2. Site’s Data Graph

name	type	power	power_m	freq	freq_m	hight_d	low_d	weight	height	width	depth
Полочные колонки Dali	полочные					21		2.6	237.0	140.0	195.0
Полочные колонки Vect	полочные	15	100	45	33000.0			9.5	310.0	170.0	220.0
Полочные колонки Wha	полочные	25	125	48	20000.0	25		9	355.0	221.0	290.0
Напольные колонки Auc	напольные		200	40	22000.0	25	120.0	27	980.0	140.0	250.0
Напольные колонки Poll	напольные	20	100	40	20000.0	25.4	165.0	19	920.0	235.0	260.0

Рис. 3. Результат выполнения скрапера

Fig. 3. Scrapper’s Search Result

Чтобы выделить контуры из исходного изображения, необходимы предварительные преобразования. Сначала изображение переводится из цветного в монохромное, соответственно значение каждого пикселя становится равным числу из множества [0, ..., 255]. Далее необходимо выделить объект на исходном изображении, но поскольку изображения взяты из каталога сайта-реселлера, объекты зачастую находятся по центру на белом фоне, без лишних объектов, поэтому задача распознавания объекта на изображении на данном этапе не рассматривается. На следующем шаге необходимо выделить границы объекта. Для решения задачи выделения границ объекта применяются алгоритмы Canny или Adaptive Thresholding (адаптивная пороговая обработка). Проанализируем их применимость к задаче.

В данной задаче перед алгоритмом Canny также было выполнено размытие по Гауссу с размером окна, равным 5. Данное значение обусловлено тем, что при бóльших значениях наблюдается снижение уровня шума, но также и потеря значимых контуров. Алгоритм Adaptive Thresholiding реализуется методом *adaptiveThreshold* из библиотеки OpenCV. Размер фильтра составляет 5 пикселей. Для того чтобы избавиться от шумов, перед выполнением алгоритма изображение обрабатывается с помощью медианного фильтра. Алгоритм его работы заключается в выборке среднего значения пикселей, попавших в окно фильтра на текущей итерации. Получившееся значение и будет выходным для рассматриваемого пикселя.

Чтобы сузить толщину линий контура и восстановить разрывы в контуре, на бинарном изображении применяются морфологические преобразования. В программной реализации, использующей алгоритм Canny, применяются операции “Erosion” и “Closing” [2. P. 137]. Erosion позволяет сузить контур. Принцип работы заключается в последовательном скольжении ядра по изображению: если все пиксели в окне равны единице, то и результирующий пиксель приравнивается к 1, иначе – к 0. Операция “Closing” является последовательным расширением и сужением до исходного состояния контура, что позволяет избавиться от пробелов внутри контура. Программно это реализуется функцией *morphologyEx* из библиотеки OpenCV. В реализации, использующей алгоритм Adaptive Thresholding, применяется операция “Opening”, которая помогает избавиться от мелких шумов на изображении, что, в свою очередь, минимизирует появление разрывных мелких контуров.

Поскольку сравнение двух объектов будет производиться в том числе и на основании иерархии контуров и их моментов, даже незначительные изменения положения на фотографии или уровня освещенности может снизить процент распознавания, а также повлиять на классификацию устройства. К тому же, поскольку внешний вид в пределах типа распознаваемого устройства может варьироваться, необходимо сохранить только значимые детали. Для решения этой задачи был написан метод, который выполняет аппроксимацию контура. Аппроксимация контура достигается путем удаления из контура компонент, находящихся на расстоянии меньше заданного. Также контуры, состоящие из количества точек меньше заданного, не включаются в получившийся список. Иерархия контуров при этом перестраивается с учетом удаленных контуров. Результат выполнения можно видеть на рис. 4, где справа находится объект с оптимизированным списком контуров, а слева – без оптимизации. На различных изображениях при использовании данного алгоритма количество контуров уменьшается в 2–5 раз, существенно не меняя представление объекта.

При использовании алгоритма Adaptive Thresholding наблюдается более сильное акцентирование мелких деталей, но также большие разрывы контуров (рис. 5). Границы объектов распознаются как отдельные контуры. С учетом этого в конечной реализации используется выделение контуров на основе алгоритма Canny.

Сравнение схожести объектов, полученных с фотографии пользователя, и исходной выборки, опирается на сравнение контуров. Для того чтобы определить положение контуров относительно друг друга, используется «иерархия контуров». Библиотека OpenCV имеет встроенный интерфейс для выделения контуров из предварительно обработанных изображений, реализующийся с помощью метода *cv2.findContours*(*input_image*, *mode*, *method*), где:

1) *input_image* – исходное черно-белое изображение;

2) *mode* – режим поиска контура. В данном случае используется извлечение контуров вместе с полной иерархией (RETR_TREE);

3) *method* – метод контурной аппроксимации. Используется метод, при котором каждый контур хранит только угловые точки (CHAIN_APPROX_SIMPLE). Использование такого метода обусловлено тем, что слишком большое количество точек может увеличить скорость сравнения и привести к снижению результатов.

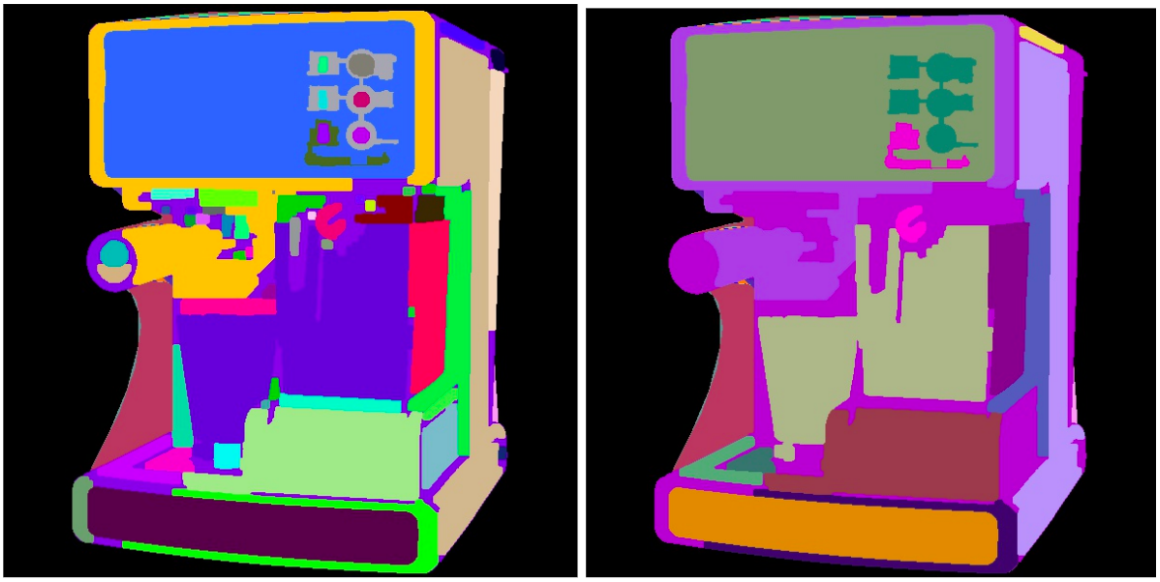

Рис. 4. Выделение контуров без применения (слева) и с применением (справа) оптимизации контуров

Fig. 4. Contours with (right) and without (left) Optimization

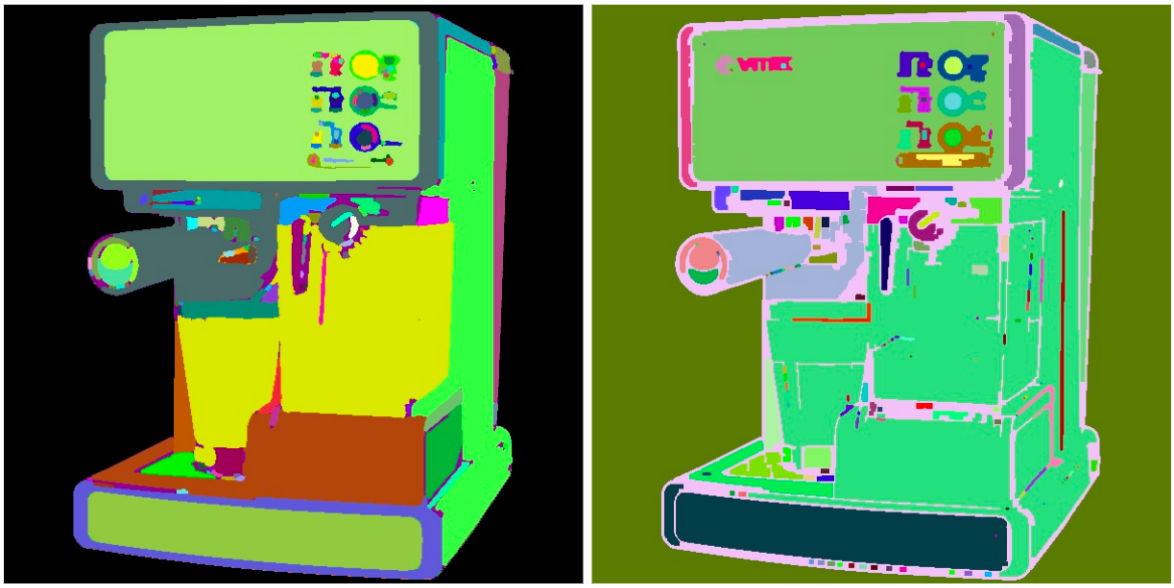

Рис. 5. Результат выделения границ методами Canny (справа) и Adaptive Thresholding (слева)

Fig. 5. Canny (right) and Adaptive Thresholding (left) Algorithms Results

Для каждого контура существует родительский контур и дочерние контуры. Структура контуров, которую предоставляет библиотека OpenCV, выглядит следующим образом:

$$\mathbf{K = [k1, k2, k3, k4]},$$

где

k1 – индекс следующего контура на текущем слое;
k2 – индекс предыдущего контура на текущем слое;
k3 – индекс первого дочернего контура;
k4 – индекс родительского контура.

Таким образом, иерархией контуров является граф (рис. 6).

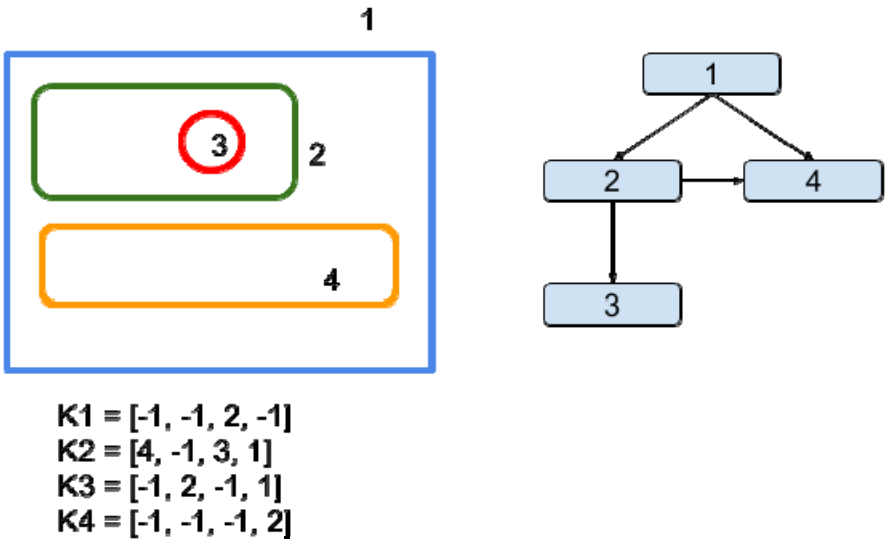

Рис. 6. Пример структуры контуров
Fig. 6. Contours Structure Example

Для оптимизации времени сравнения и для большей прозрачности самой операции сравнения, рассматривание контуров происходит от большего к меньшему, т. е. от родительского к дочернему. Соответственно при больших отклонениях мер родительских контуров операция завершается неудачей, и дочерние контуры сравниваться уже не будут. Чтобы сравнивать контуры между собой, необходимо выделить количественные коэффициенты контуров, а также задать сами правила сравнения. В условиях текущей задачи точное соответствие контуров при сравнении не требуется, поскольку тогда даже небольшие различия, возникающие из-за разных ракурсов, уровня освещения и большей предметности, например, моделей техники, может приводить к ухудшению качества распознавания.

Сравнение получившихся контуров происходит с помощью сравнения их моментов, что позволяет не учитывать их размер и угол поворота в пространстве. Момент – это численная характеристика контура, момент контура складывается из моментов его точек. Различают центральные, нормализованные и инвариантные моменты. Момент точки (x, y) порядка $(p + q)$ определяется как

,

где $I(x, y)$ – интенсивность пикселя с координатой (x, y); p – порядок x; q – порядок y. Порядок – степень, в которую возводят компоненты суммы. В случае с контурами момент, в кото-

ром $p = q = 0$, равен числу точек в контуре (поскольку рассматриваемое на данном шаге изображение бинарное, то интенсивность для точек контура равна 1).

Но моменты, вычисленные по такой формуле, зависят от положения в пространстве, поэтому используются центральные моменты, которые лишены этого недостатка, а также являются инвариантными относительно масштабирования. В вычислении центральных моментов применяются центроиды $(\bar{x}, \bar{y})$, где $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$. В таком случае формула принимает вид

$$\mu_{p,q} = \sum_x \sum_y I(x,y)(x-\bar{x})^p (y-\bar{y})^p.$$

Для независимости моментов от масштабирования моменты нормализуются:

$$\eta_{ij} = \mu_{i,j} / \mu_{00}^{(i+j)/2+1}.$$

Данный тип моментов не применяется при сравнении моментов, так как он не является инвариантным относительно поворотов. Поэтому для этих целей применяются Hu-моменты [3]. Инвариантный момент является линейной комбинацией центральных моментов. Hu-моменты – это набор из семи моментов, первые шесть инвариантны относительно сдвига, масштабирования и поворота, последний момент меняет знак при зеркальном отражении:

$$h_0 = \eta_{20} + \eta_{02},$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$

$$h_3 = (\eta_{30} + 3\eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$

$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2) + 3(\eta_{21} - \eta_{03}) \times \\ \times (3(\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2),$$

$$h_5 = (\eta_{21} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{12} + \eta_{03})^2 + (4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{12} + \eta_{03})),$$

$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2) + (\eta_{30} - 3\eta_{12}) \times \\ \times (\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{12} + \eta_{03})^2).$$

В работах [4; 5] приводится метод сравнения контуров на основе встроенных функций сравнения контуров (которые используют Hu-моменты) из библиотеки OpenCV, но неприменимость этого подхода в данной задаче обусловливается тем, что получаемый объект сравнивается не с одним объектом, а участвует в классификации, из чего следует необходимость хранения множества характеристик для каждого объекта выборки.

Hu-моменты вычисляются функцией *moment.HuMoments*() из библиотеки OpenCV. Для сравнения меры схожести моментов применяется несколько метрик. Зачастую применяется метрика вида $M(A,B) = \sum_{i=0}^{6} |H_i^B - H_i^A|$, где H – логарифм Hu-момента по основанию 10, взятый со знаком этого момента.

Использование данной метрики обусловлено также тем, что при составлении таблицы признаков для кластеризации необходимо хранить данные о каждом контуре, используя которые можно сравнивать контуры между собой. Таким образом, для каждого контура хранится алгебраическая сумма его моментов, имея которые и используя свойства суммы $\sum_{i=0}^{I}(a_i + b_i) = \sum_{i=0}^{I} a_i + \sum_{i=0}^{I} b_i$, можно вычислить меру сходства между контурами.

Поскольку категории, извлеченные с сайта, находятся в разном представлении (текстовая, численная информация), то необходимо нормализовать признаки, т. е. привести их к одному виду, в одну форму. Рассмотрим признаки, полученные с сайта, для выборки аудиоколонок:

1) тип колонок (напольные и т. д.);
2) материал корпуса (дерево, металл и т. д.).
3) мощность;
4) частотный диапазон;

5) габариты;
6) вес;
7) диаметр высокочастотных излучателей;
8) диаметр низкочастотных излучателей.

Выбор признаков обусловлен наличием связи между внешним видом устройства и его характеристиками. Например, имея данные о габаритах устройства и информацию о габаритах, полученную из анализа изображения, можно предсказать габариты объекта, полученного с камеры смартфона при распознавании.

Поскольку в полученной в результате обхода сайта таблице данные представления в столбцах в текстовой и числовой информации, необходимо привести данные в столбцах только к одному виду, опустив единицы изменения, предлоги, а также разбить столбцы, содержащие интервалы на два – минимальные и максимальные значения (рис. 7). Также в таблицу вносится информация о контурах, расположенная в порядке уменьшения периметра контуров.

Type	power	power_ma	freq	freq_m	hight_d	low_d	weight	height	width	depth	contour_0	contour_1
Полочные					21		2.6	237.0	140.0	195.0	16.663920881268623	-0.5297403
Полочные	15	100	45	33000.0			9.5	310.0	170.0	220.0	7.3507366452908975	55.7952324
Полочные	25	125	48	20000.0	25		9	355.0	221.0	290.0	-2.64724015162624	-0.8186475
Напольные		200	40	22000.0	25	120.0	27	980.0	140.0	250.0		-28.011966
Напольные	20	100	40	20000.0	25.4	165.0	19	920.0	235.0	260.0	-5.347270012066903	-3.8756198

Рис. 7. Данные после операции преобразования
Fig. 7. Data after Transformations

Поскольку в дальнейшем данные будут использоваться в алгоритме классификации, а в большинстве методов классификации используются евклидово или метрические пространства, то данные представляются в виде векторов. Метрикой является евклидова метрика ($\sqrt{m}$, где $m = \sum_{k=1}^{n}(p_k - q_k)^2$, $p = (p_1,...,p_n)$, $q = (q_1,...,q_n)$ – векторы размерностью n). Категориальные данные преобразуются в числовые. В таких случаях можно сопоставить с каждой категорией числовое представление, например: “catagoria_a” = 1, “catagoria_b” = 2 и т. д. Но проблема данного подхода заключается в том, что числовое представление категорий создает евклидовое представление. На нем становятся доступны операции, не имеющие смысла в категориальных признаках, такие как вычитание, умножение и т. д. Для того чтобы избежать этого, используется подход, в котором каждая категория заменяется на отдельный признак. На позиции, соответствующие численному значению, ставится 1, иначе – 0 (рис. 8).

contour_29	contour_30	type_floor	type_shelving
15.217872473771653	-14.894815648454125	0	1
13.692011951099424	3.790743824951145	0	1
27.824993381208337	27.77607981130654	1	0

Рис. 8. Результат выполнение OneHotEncoder
Fig. 8. OneHotEncoder Execution Result

В выборке встречаются недостающие данные, которые заменяются средним значением признака. Поскольку значения признаков могут сильно варьироваться – от пары десятков до нескольких десятков тысяч (например, в столбцах, отвечающих за моменты), разные признаки дают разный вклад при определении объекта. Данная проблема решается нормализацией значений признаков. В результате нормализации значение каждого признака принимает значение в интервале [–1, 1].

Поскольку для каждого класса (в условиях задачи классов техники) существует свое число внешних определяющих признаков (которое может не совпадать между классами), в качестве проверки работоспособности алгоритма решается задача бинарной классификации аудиоколонок.

Для того чтобы выделить признаки из контуров, необходимо их проанализировать. Для каждого класса коэффициенты, описанные ниже, могут отличаться и настраиваться исходя из выборки. Рассмотрим изображение из полученной выборки и его контуры (рис. 9).

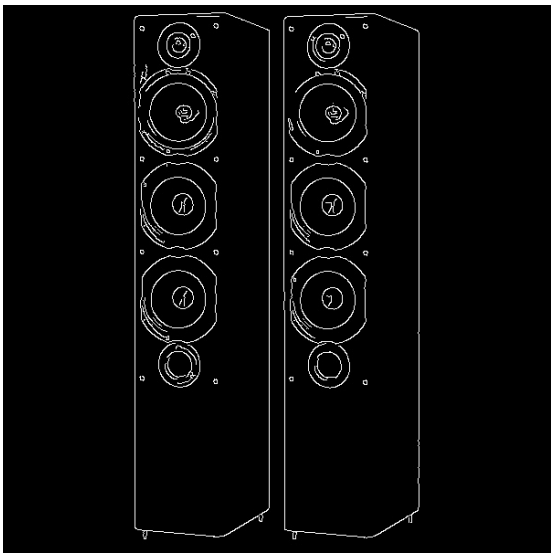

Рис. 9. Контуры объекта
Fig. 9. Object's Contours

На изображении присутствуют незначительные контуры либо помехи, которые при дальнейшей кластеризации и сравнении объектов могут вносить погрешности. Для того чтобы минимизировать эти эффекты, применяется фильтрация по периметру, а также по площади контура. Поскольку площадь объекта может меняться, данная фильтрация нужна прежде всего, чтобы явно выделить внешние грани объектов, а также большие ограниченные области внутри самих объектов. Рассмотрим распределение контуров по площади и периметру.

Общее количество контуров рассматриваемого объекта – 134. Преобладают контуры, площадь которых близка к 0 (рис. 10), такие контуры не являются определяющими для объекта и, скорее всего, являются шумом. Но поскольку это может быть площадь незамкнутых контуров, то также следует рассмотреть периметр контуров объекта.

Больше всего контуров, периметры которых близки к 0 (рис. 11). Также есть контуры, длина которых больше тысячи. Можно предположить, что это грани объекта и грани динамиков – определяющие признаки. В данном случае из набора контуров удаляются контуры, длина которых меньше 50 и площадь которых меньше 1 000. Данные числа получены экспериментально. На рис. 12 показаны оставшиеся контуры, по моментам которых будет происходить последующее сравнение объектов.

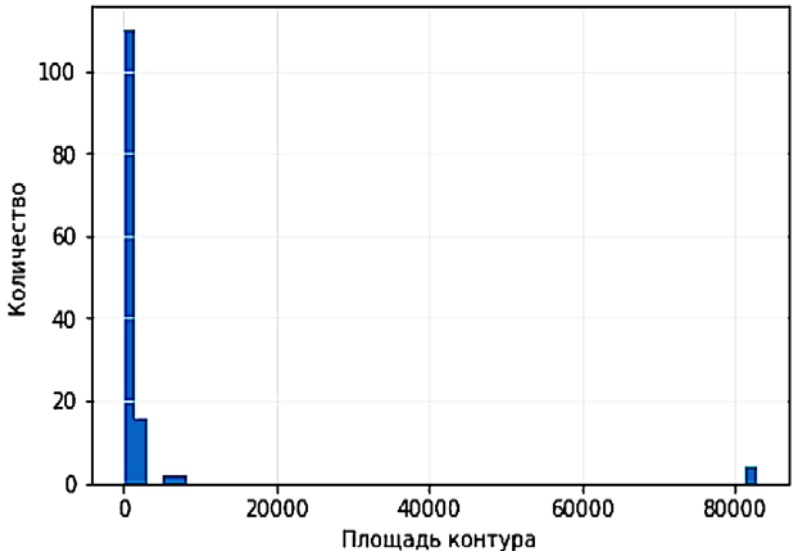

Рис. 10. Распределение контуров по площади
Fig. 10. Areas of Contours

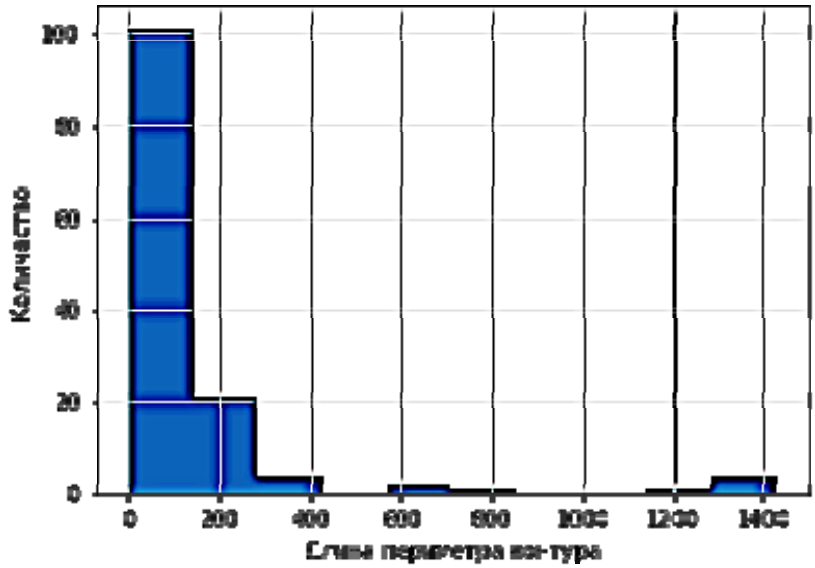

Рис. 11. Распределение контуров по периметру
Fig. 11. Perimeters of Contours

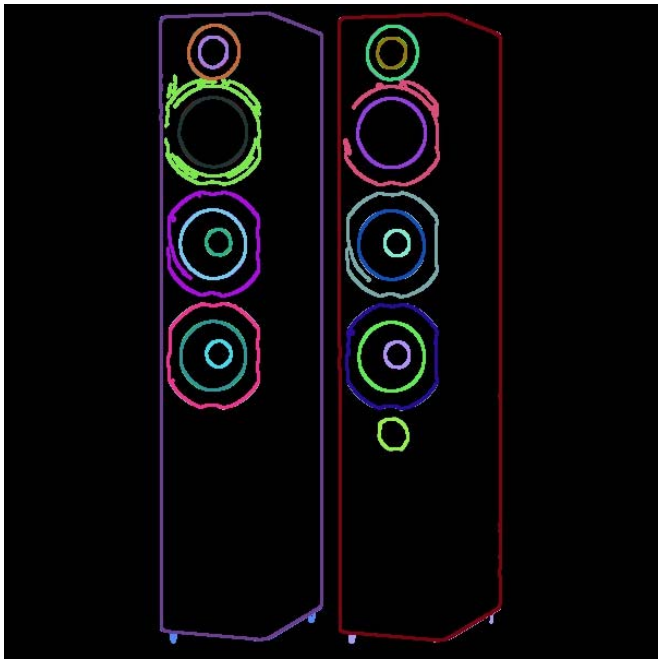

Рис. 12. Выделенные признаки изображения
Fig. 12. Object's Features

Примем гипотезу, что наиболее точно определяют объект его наибольшие компоненты. Для того чтобы построить таблицу признаков, выделим первые N контуров на основе их периметра, в таблицу признаков записываются соответствующие им моменты.

В качестве обоснования состоятельности подхода связи внешнего представления изображения с его характеристиками решается задача бинарной кластеризации. Набор данных включает в себя объекты двух типов – напольные и полочные аудиоколонки. В качестве алгоритма кластеризации используется алгоритм K-means (рис. 13). Принцип работы данного алгоритма заключается в перерасчете центров масс кластеров, изначально заданных произвольным образом, до тех пор, пока с каждой итерацией внутрикластерное расстояние не перестанет изменяться. В качестве метрики используется евклидово расстояние. Точность кластеризации составила 0,95.

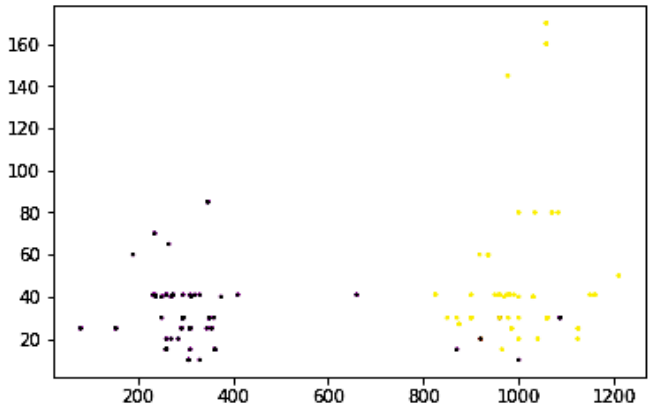

Рис. 13. Кластеризация с помощью метода k-средних
Fig. 13. Clustering with Using K-Means Algorithm

Алгоритм работы распознавания характеристик объекта состоит из следующих шагов:

1) по полученной выборке с отсутствующими данными строится таблица признаков;
2) таблица признаков нормализуется (все признаки приводятся к одному виду);
3) происходит заполнение недостающих данных исходя из значений данного признака для других объектов класса;
4) при получении объекта с неполными данными заполняются отсутствующие данные, и объект классифицируется.

Как уже было сказано, обучающая выборка состоит из объектов, принадлежащих двум классам – напольные и полочные аудиоколонки. Поскольку при фотографировании объекта модели на вход подается вектор, имеющий только часть, полученную из анализа изображения, требуется предсказать первую часть вектора, включающую в себя характеристики устройства. Вычисление всего вектора неизвестных признаков целиком одной моделью является нетривиальной задачей и может давать плохой результат, поэтому для каждого вычисляемого признака создается своя модель. Соответственно, количество моделей равно количеству неизвестных признаков. Результатом же выполнения будет класс – численная характеристика неизвестного параметра. Например, известно, что мощность устройства может быть одним значением из множества [50Вт, 100Вт, 300Вт]. В таком случае каждое значение является отдельным классом, и результатом классификации будет одно из этих значений. Такой подход также позволяет выбирать только определенные известные признаки для получения значения неизвестного признака и предугадывать значения, используя заранее вычисленные значения. В рассматриваемой выборке создается 10 моделей для получения неизвестных десяти характеристик. Но, поскольку данный подход является операцией над категориальными признаками, возможные значения которых должны быть заранее известны, он не подходит под признаки конечного вида, например габариты и т. д. Для того чтобы вычислить значения таких признаков, решается задача регрессии [6. С. 18]. Для решения данной задачи при классификации были использованы следующие алгоритмы:

1) дерево решений, точность 0,96;
2) *k* ближайших соседей, точность составила 0,79;
3) метод опорных векторов, точность 0,67.

Точность рассчитывается следующей формулой:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

В данном случае TP, TN – количество верно распознанных объектов первого или второго класса, FP, FN – количество неверно распознанных объектов первого или второго класса. Для того чтобы узнать точность предсказания каждого класса в отдельности, нужно использовать понятия полноты (recall) – доля объектов класса, из всех объектов класса, которую нашел алгоритм, и точности (precision) – доля объектов, которую алгоритм назвал положительными и которые действительно таковыми являются, поэтому используется *F*-метрика:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}.$$

F-метрика является средним гармоническим величин precision и recall, где $\beta = 1$. С использованием данной метрики распознавание напольных колонок составило 0,92, а полочных – 0,93.

При распознавании массы колонок выборка была разбита на классы до 10 кг, от 10 до 20 и выше 20 кг. Такое разбиение обосновывается на частоте встречаемости данных масс. При данном разбиении доля распознавания составила 0,74.

Как уже было сказано, для предсказания некатегориальных признаков (таких как длина и т. д.) используется регрессия. Поскольку данные в разных классах могут сильно отличаться, для предсказания параметра используются выборки, включающие только свой класс (в данном случае, например, отдельно класс напольных колонок и отдельно полочных). В качестве алгоритмов регрессии выбраны алгоритмы линейной регрессии, регрессия на ос-

нове случайного леса k ближайших соседей. Для вычисления каждого из неизвестных признаков из исходного списка признаков выбираются те, которые имеют зависимость от искомого признака. Например, при поиске габаритов такими данными являются те признаки, которые задают моменты первых контуров, вес, высота отдельных известных компонентов объекта.

После первого прогона алгоритмов результаты выглядят следующим образом:

1) линейная регрессия – 0,6;
2) случайный лес (150 деревьев) – 0,76;
3) k ближайших соседей (6 соседей) – 0,36.

В данной задаче в качестве метрики используется коэффициент детерминации (R-квадрат):

$$R^2 = 1 - \frac{V(y|x)}{V(y)} = 1 - \frac{\sigma^2}{\sigma_y^2}.$$

Данная метрика показывает, насколько условная дисперсия модели отличается от реальной дисперсии модели.

Поскольку второй алгоритм дал наиболее точный результат, модель, его реализующая, была выбрана в качестве основной для дальнейшего обучения. Также остальные алгоритмы при изменении параметров давали не сильно отличающиеся результаты.

После преобразований наилучший полученный результат стал равен 0,82. Данный результат достигнут на конфигурации, состоящей из 200 деревьев; количество выделяемых признаков, равное количеству выбранных заранее; максимальная глубина дерева = 20 (так как в данных возможны шумовые выбросы, что при большой глубине давало бы ненужные взаимосвязи).

Рассмотрим применение алгоритма к другим обучающим выборкам. Для этого была извлечена выборка, состоящая из обычных и двухстворчатых холодильников (рис. 14). Характеристиками объекта были выбраны:

1) тип;
2) вес;
3) количество камер (от 2 до 4);
4) ширина;
5) высота;
6) глубина.

Рис. 14. Пример данных выборки холодильников

Fig. 14. Samples of Fridges Data

В качестве пороговых значений для минимальной площади конура было выбрано значение $S = 1\,000$, для периметра – $P = 500$. Точность классификации объекта составила:

1) дерево решений, точность 0,94;
2) k ближайших соседей, точность 0,73;
3) метод опорных векторов, точность 0,68.

Результат распознавания количества камер 0,75. Для получения результата брались характеристики, отвечающие за ширину объекта, а также значения первых пяти контуров (данное количество получено опытным путем). Из полученных результатов видно, что алгоритм показывает хорошие результаты распознавания характеристик для различных выборок объектов. Качество зависит от анализа признаков контуров, таких как значения коэффициентов при выделении контуров, выбор минимальных значений для площади и периметра контуров.

Разработка системы

Алгоритм работы системы состоит из следующих шагов (рис. 15):

1) пользователь открывает приложение;
2) при фотографировании объекта изображение отсылается на сервер;
3) сервер классифицирует объект;
4) клиенту возвращается класс устройства, данные о нем и список возможных для загрузки инструкций;
5) на экране телефона показывается информация о найденном объекте с применением дополненной реальности;
6) при нажатии на список инструкций на сервер передается запрос, результатом которого является выбранная инструкция в формате PDF;
7) при получении мобильным приложением инструкции она отображается на экране смартфона.

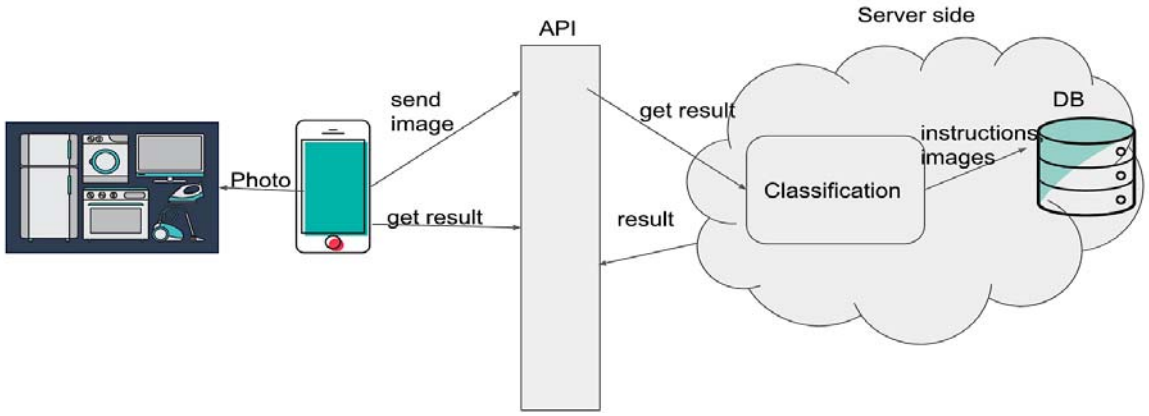

Рис. 15. Схема работы приложения
Fig. 15. Application Workflow Scheme

Так как распознавание непосредственно на устройстве является ресурсоемким, то было принято решение реализовать архитектуру «тонкий клиент», когда устройство ответственно только за получение и отправку данных. При старте приложения открывается экран, где показывается изображение с камеры устройства. При наведении смартфона на объект серверу передается текущее изображение. Для того чтобы не учитывать контуры фона, изображение обрезается (выделяется квадрат со стороной, равной трети исходного изображения). Затем клиентское приложение получает список вероятных характеристик устройства, а также иден-

тификаторы, присущие инструкциям устройств данного класса. При выборе одной из инструкций на сервер передается запрос, результатом которого является выбранная инструкция.

Мобильное приложение написано под OS “Android” на языке Kotlin. При написании приложения используются принципы ООП. Приложение спроектировано с использованием подхода “Clean Architecture” [7]. Этот подход является одним из популярных архитектурных решений на данный момент при проектировании мобильных приложений из-за простоты обучения, четкой интерпретации и большого количества обучающих источников. Также множество современных библиотек приспособлены для использования именно данного решения. Clean Architecture позволяет добиться таких важных моментов:

1) масштабируемость;
2) независимость от фреймворков;
3) хорошая тестируемость;
4) независимость от реализации пользовательского интерфейса (UI);
5) независимость от выбора базы данных.

Подход заключается в разбиении приложения по следующим уровням (рис. 16).

1. Слой представления. На данном слое логика связывается с интерфейсом приложения. Фрагменты и Активности, которые поставляет фреймворк Android, не имеют собственной логики, а ответственны только за отображение данных, полученных от презентера, – компоненты приложения, связывающей логику отображения информации с бизнес-логикой приложения, и передачи команд, таких как ввод текста пользователем или нажатие на кнопки презентеру. Презентеры на этом слое связываются с интеракторами из слоя бизнес-логики. Интерактор – это логика, разбитая на пользовательские сценарии, например: интерактор для авторизации на сервере, интерактор для отправления данных и т. д. Зачастую в этом месте происходит смена потока выполнения с интерфейсного на фоновый.

2. Слой бизнес-логики. Данный слой является модулем, лишенным зависимостей от фреймворка Android. На данном слое находится вся логика приложения. Логика разбита на интеракторы. Интеракторы связаны со слоем данных (Data-слой).

3. Слой данных. На этом слое находятся источники данных (база данных, кэш и т. д.), также на данном слое происходит соединение с сервером.

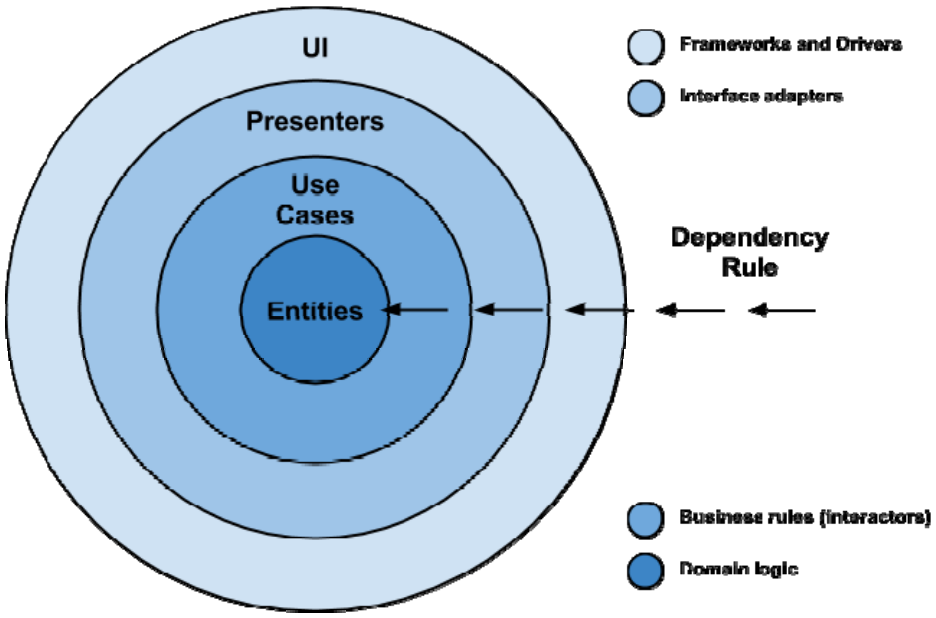

Рис. 16. Слои Clean Architecture
Fig. 16. Clean Architecture Layers

При включении приложения происходит регистрация клиента на сервере – клиентское приложение получает токен, по которому в дальнейшем происходит его аутентификация на сервере. Взаимодействие с сервером происходит по технологии REST, для соединения с сервером используется библиотека Retrofit. На главном экране приложения отображается полученное с камеры устройства изображение. При наведении камеры на распознаваемый объект полученное изображение отсылается на сервер для дальнейшей классификации. После того как сервер вычислил значение признаков для классифицируемого объекта, клиентское приложение извлекает их из полученного файла в формате JSON и отображает на экране устройства. Также во всплывающем окне появляется список инструкций, связанный с данным кластером устройств. При выборе нужной инструкции на сервер посылается GET-запрос с ее идентификатором. Результатом выполнения запроса является инструкция в формате PDF, которая по окончании загрузки сохраняется в память устройства и отображается на экране. Для отображения информации на экране используется библиотека ARCore.

Когда фреймворк ARCore распознал поверхность (вертикальную или горизонтальную), через заранее заданный промежуток времени выполняется отправка на сервер изображения с камеры. Если серверное приложение распознало один из типов устройств, то клиентское приложение получает информацию об этом устройстве, которая показывается поверх распознанного устройства (рис. 17) в виде списка.

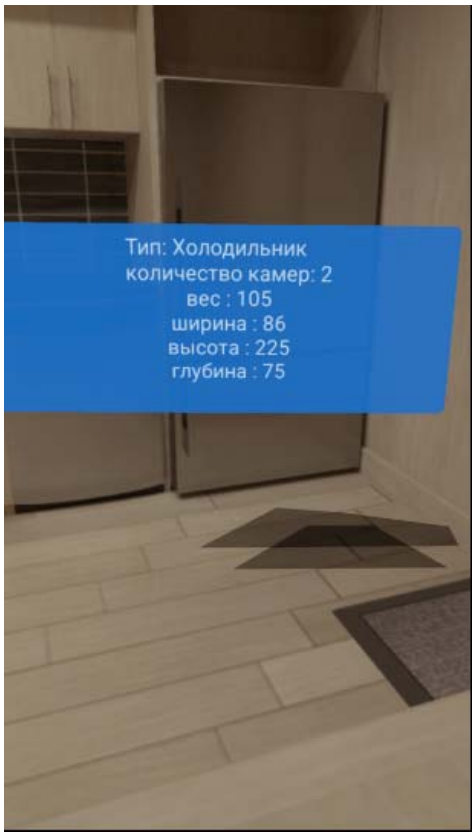

Рис. 17. Отображение характеристик устройства на примере холодильника

Fig. 17. Displaying of Fridge Characteristics

Сервер хранит заранее обученную модель, на вход которой при запросе передается информация, полученная из анализа переданного клиентом изображения. Поскольку полученное изображение цветное, то сначала оно переводится в черно-белое, затем из него выделя-

ются контуры и их моменты по алгоритму Canny, описанному ранее. После успешной классификации и вычисления характеристик входящего устройства в базе данных инструкций ищутся все ассоциированные с данным классом устройств инструкции. Затем идентификаторы инструкций передаются клиентскому приложению. При получении запроса от клиента на скачивание определенной инструкции в базе данных ищется искомая инструкция по идентификатору и отсылается клиенту. Серверная компонента приложения связывается с приложением-клиентом с помощью архитектурного подхода REST API. Запрос, реализующий данный подход, выглядит следующим образом:

1) маршрут отправки (адрес, по которому направляется запрос);
2) тип метода (GET, POST, PUT);
3) заголовки запроса;
4) тело запроса (данные).

Метод GET используется для получения с сервера определенных данных, результат на запрос данных отправляется приложению-клиенту (рис. 18).

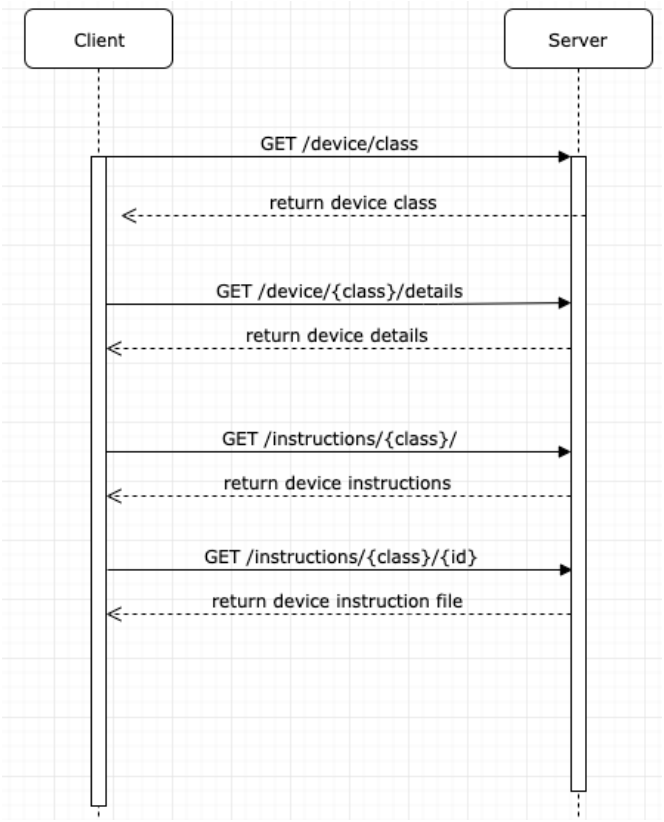

Рис. 18. Схема взаимодействия клиента и сервера
Fig. 18. Client-Server Connection Scheme

Запрос **GET /device/class/** отвечает за вычисление класса устройства, изображение которого присылает клиент в теле запроса. Результатом выполнения запроса является класс девайса. Запрос **GET /device/{class}/details** возвращает характеристики устройства. Запрос **GET /instructions/{class}/** запускает поиск инструкций (краткой инструкции и списка доступных для загрузки инструкций) для конкретного типа устройств. Загрузка определенной инструкции производится по запросу **GET /instructions/{class}/id**.

Заключение

Разработанный подход получения признаков объекта показал хорошую точность на исходной выборке. Данный подход позволяет связывать отдельные части объекта (ограниченные контуром) с каким-либо набором его признаков, что позволяет рассматривать изображение объекта как набор составляющих его частей, что в дальнейшем можно использовать задаче кластеризации объектов, зная характеристики отдельных компонент объектов.

Список литературы

1. **Фурман Я. А., Кревецкий А. В., Передреев А. К., Роженцов А. А., Хафизов Р. Г., Егошина И. Л., Леухин А. Н.** Введение в контурный анализ и его приложение к обработке изображений и сигналов. М.: Физматлит, 2002. 592 с.
2. **Garcia G. B., Suarez O. D., Aranda J. L. E.** Learning Image Processing with OpenCV. Birmingham, Packt Publ., 2015, 319 p.
3. **Ming-Kuei Hu**. Visual Pattern Recognition by Moments Invariants. *IRE Transactions on Information Theory*, 1962, vol. 8, no. 2, p. 179–187.
4. **Belongie S., Malik J., Puzicha J.** Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, vol. 24, no. 4, p. 509–522. DOI 10.1109/34.993558
5. **Коробов Д. В., Патин М. В.** Метод распознавания шрифта текста с изображения // Молодой ученый. 2016. № 12. С. 161–165.
6. **Вапник В. Н.** Восстановление зависимостей по эмпирическим данным. М.: Наука, 1979. 448 с.
7. **Robert C. Martin.** The Clean Architecture. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 15.02.2019)

References

1. **Furman Ya. A., Krevetskiy A. V., Peredreev A. K., Rozhentsov A. A., Khafizov R. G., Egoshina I. L., Leukhin A. N.** Vvedenie v konturnyj analiz i ego prilozhenie k obrabotke izobrazhenij i signalov. Moscow, Fizmatlit Publ., 2002, 492 p. (in Russ.)
2. **Garcia G. B., Suarez O. D., Aranda J. L. E.** Learning Image Processing with OpenCV. Birmingham, Packt Publ., 2015, 319 p.
3. **Ming-Kuei Hu**. Visual Pattern Recognition by Moments Invariants. *IRE Transactions on Information Theory*, 1962, vol. 8, no. 2, p. 179–187.
4. **Belongie S., Malik J., Puzicha J.** Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, vol. 24, no. 4, p. 509–522. DOI 10.1109/34.993558
5. **Korobov D. V., Patin M. V.** Metod raspoznavaniya shrifta teksta s izobrazheniya. *Molodoj uchenyj*, 2016, no. 12, p. 161–165. (in Russ.)
6. **Vapnik V. N.** Vosstanovlenie zavisimostej po empiricheskim dannym. Moscow, Nauka, 1979, 448 p. (in Russ.)
7. **Robert C. Martin.** The Clean Architecture. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 15.02.2019)

Материал поступил в редколлегию
Received
30.04.2019

Сведения об авторах / Information about the Authors

Прокопьев Иван Сергеевич, магистрант факультета информационных технологий Новосибирского государственного университета (ул. Пирогова, 1, Новосибирск, 630090, Россия)

Ivan S. Prokopyev, Master's Student, Faculty of Information Technologies, Novosibirsk State University (1 Pirogov Str., Novosibirsk, 630090, Russian Federation)

i.prokopev3@g.nsu.ru

Шмаков Евгений Сергеевич, Software Architect / Tech lead, ООО "БНС" (ул. Фрунзе, 49, Новосибирск, Россия)

Evgeny S. Shmakov, Software Architect / Tech lead, LLC "BNS" (49 Frunze St., Novosibirsk, Russian Federation)

sjs-master@yandex.ru