## 2 АРХИТЕКТУРА ПОРТЫ И АДАПТЕРЫ ПРОГРАММНОГО КОМПЛЕКСА СЕГМЕНТАЦИИ И ОКРАСА СТЕН НА ИЗОБРАЖЕНИЯХ

* 1. **Функциональная модель программного комплекса сегментации и окраса стен на изображениях**

Разрабатываемое мобильное приложение должно решать одну основную задачу: окрас стены на изображении. Дополнениями данной задачи являются: режим реального времени, возможность выбора цвета окрашивания, возможность окрашивания текстурами.

Основная задача включает в себя две составляющих: локализация и сегментация стены на изображении, окрашивание той части изображения, что была обозначена как стена.

Из обзора сегментации – известно, что каждому пикселю изображения, в данном случае, присваивается метка класса, следовательно, результат работы первой функции это массив значений размером с число пикселей изображения, в котором пиксели, распознанные как стена, обозначены одним значением, а остальное – другим. Подобный массив является своеобразной маской.

Входными же данными первой функции будет исходное изображение, также представленное в виде массива.

Вторая функция, функция окраса, должна принимать результат работы первой – маску, исходное изображение и желаемый цвет. Данная функция вернёт уже окрашенное изображение также в виде массива.

Две вышеописанные функции выступают в виде первостепенных бизнес правил. Остальной функционал станет дополнением к бизнес логике.

В дополнительный функционал входит:

* графический вывод результата;
* импорт изображения из галереи;
* захват изображения через камеру;
* дополнительная постобработка маски.

Если функция окраски использует функционал библиотеки компьютерного зрения, то функция локализации стен на изображениях совмещает два разных подхода к решению задачи локализации и сегментации стен:

* локализация стен методами компьютерного зрения без нейронных сетей;
* локализация и сегментация стен с помощью нейронной сети.

Подобный функционал позволит провести исследование и сравнительный анализ локализации стен обоими методами, а также даст пользователю возможность выбирать, какой из методов использовать при эксплуатации приложения.

Для достижения наилучшей точности сегментации и достаточной скорости обработки изображений, непосредственно, алгоритмы решающие данную задачу должны быть расположены на машине с соответствующими техническими характеристиками. Мобильные устройства по производительности проигрывают персональным компьютерам с хорошей сборкой. Учитывая данный факт, исследование качества сегментации нейронными сетями и без них дополняется исследованием скорости сегментации. В последнем примут участие три алгоритма:

* нейронная сеть;
* облегчённая нейронная сеть для мобильных устройств;
* методы компьютерного зрения без нейронной сети.

На изображении 2.1 показана схема функционального разделения приложения на данном этапе.

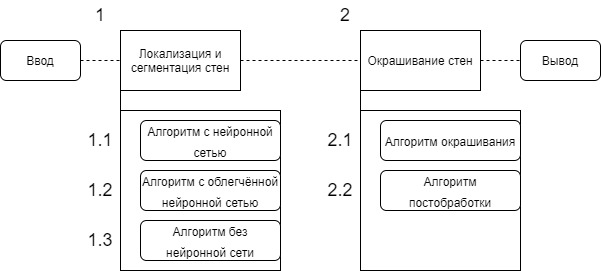


Рисунок 2.1 – Визуализация основных функций приложения

Пунктирным подчёркиванием на схеме указано направления потока управления.

В сносках указаны подфункции двух главных функций приложения. Подфункции 1.1, 1.2 и 1.3 подлежат сравнению по скорости обработки данных, а также по качеству выходного результата.

Нейронные сети с облегченной архитектурой более быстрые, но заведомо выдают менее точный результат. Для реализации качественной сегментации стен основной алгоритм не должен располагаться на мобильном устройстве. Здесь существует несколько подходов: реализация стороннего сервиса локально на персональном компьютере (доступ по локальной сети) или хостинг разработанного серверного *API* (*Application Programming Interface*) на удалённую машину (и доступ по глобальной сети интернет).

В обоих случаях мобильное приложение становится клиентом, и должно реализовывать сетевые программные интерфейсы.

На рисунке 2.2 приведена схема структуры программного комплекса с указанным распределением функциональных обязанностей.



Рисунок 2.2 – Клиент серверное разграничение функций в программном комплексе

Как видно из схемы, вся бизнес логика, связанная с обработкой изображения, была вынесена на сервер. Мобильное приложение получает исходное изображение из вне, отправляет его и дополнительные параметры на сервер по сети, и ожидает результирующего изображения от сервера.

* 1. **Архитектура компонентов программного комплекса локализации стен на изображениях**

***Схемы подтыбженные их чистой архитектуры.***

***Взаимодействие классов и интерфейсом вдург с другом***

Раздел 2. Архитектура программного обеспечения.

определены структурные элементы программы,

спроектирована внутренняя архитектура

заданы схемы разделения данных.

В результате моделирования предметной области

должны быть приведены основные сущности и связи между ними,

создана информационная модель предметной области.

должны быть описаны внутренние и внешние интерфейсы взаимодействия каждой программной части,

***…***

Данные классы описывают все функциональные особенности разрабатываемого приложения.

10-12 листов.

* 1. **Методы анализа и обработки изображений**
  2. **Обзор математических алгоритмов *Canny* и Хафа**

Со стороны компьютерного зрения, без использования нейронных сетей, локализация стен может быть осуществлена стандартными методами обработки контуров. Достаточное число математических методов в реализованном виде можно найти в библиотеке компьютерного зрения *OpenCv* [4]*.*

*OpenCV,* как библиотека компьютерного зрения, хорошо подходит для предварительной обработки изображения за счёт развитого функционала и обширной докумментации.

Основными встроенными методами, подходящими для решения задачи, выступают метод *Canny* и метод преобразования Хафа.

Метод *Canny* – метод обнаружения края элементов на изображении.

Метод *Canny* является многоэтапным алгоритмом.

На первом этапе алгоритма происходит подавление шума методом Гаусса. Так как обнаружение контуров чувствительно к шуму. Дальше следует нахождение интенсивности изображения за счёт фильтрации ядром Собеля по вертикали и по горизонтали. *OpenCV* имеет отдельную функцию Собеля, но в совокупности с Canny она работает эффективнее. Этот же этап включает нахождение градиента. После получения величины и направления градиента выполняется полное сканирование изображения для удаления любых нежелательных пикселей, которые могут не составлять края. Для этого на каждом пикселе пиксель проверяется, является ли он локальным максимумом в его окрестности в направлении градиента. После этого этапа на выходе образуется двоичное изображение с тонкими краями.

Завершающим этапом является нахождение пороговое значение гистерезиса. На этом этапе решается, какие из ребер на самом деле являются ребрами, а какие нет. Создаются два значения *maxVal* и *minVal*.

Любые края с градиентом интенсивности больше *maxVal* обязательно будут краями, а те, что ниже *minVal*, обязательно не будут краями. На этом этапе также удаляются небольшие шумы пикселей при условии, что края являются длинными линиями.

Преобразование Хафа – вычислительный алгоритм, применяемый для параметрической идентификации геометрических элементов растрового изображения.

Функции [*HoughLines*](https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a) и [*HoughLinesP*](https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga8618180a5948286384e3b7ca02f6feeb) библиотеки *OpenCV*– преобразование Хафа. Предназначены для обнаружения линий на изображении. Чтобы применить преобразование, сначала желательна предварительная обработка края. Для преобразований Хафа линии выражаются в полярной системе координат. В общем случае линию можно обнаружить, определив количество пересечений между кривыми. Чем больше пересекающихся кривых, тем больше точек на линии, представленной этим пересечением. Общий случай даёт возможность определить порог минимального количества пересечений, необходимых для обнаружения линии.

Метод Хафа отслеживает пересечение кривых каждой точки изображения. Если количество пересечений превышает некоторый порог, то он объявляет это как строку с параметрами точки пересечения.

Существуют стандартное и вероятностное преобразование линии Хафа. OpenCV реализует два типа преобразований. На рисунках 2.1 и 2.2 отображена разница между стандартным преобразованием и вероятностным преобразованием соответственно.



Рисунок 2.1 – Результат стандартного преобразования Хафа

Стандартное преобразование Хафа реализовано функцией HoughLines в OpenCV. Вероятностное преобразование линии Хафа является более эффективной реализацией преобразования линии. Он дает в качестве вывода экстремумы обнаруженных линий. В OpenCV данный тип преобразования реализован функцией HoughLinesP.

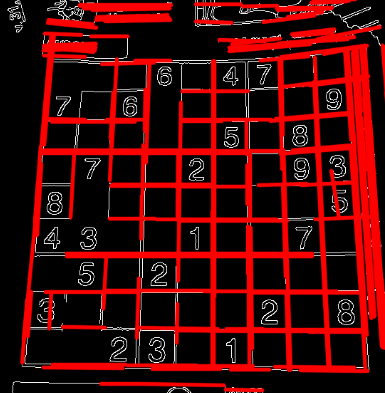


Рисунок 2.2 – Результат вероятностного преобразования Хафа

* 1. **Нейронные сети для обработки изображений**

В настоящее время проводится огромное количество исследований посвящённых созданию новых моделей нейронных сетей, для решения задач семантической сегментации в том числе.

***1.3.1*** Модели нейронных сетей, созданных для сегментации, разделяют на кодировщик и декодер, где кодеры обычно модифицируются непосредственно из сетей классификации, а декодеры состоят из окончательных сверток и повышающей дискретизации [5].

Дописать про кодер, декодер, почему так и как

Использованная для решения задачи модель, относится к данной категории и состоит из кодировщика и декодера.

### Кодировщик выполняет итерацию по входному предложению по одному токену (например, слову) за раз, на каждом временном шаге выводя вектор «вывода» и вектор «скрытого состояния». Затем скрытый вектор состояния передается на следующий временной шаг, а выходной вектор записывается. Кодер преобразует контекст, который он видел в каждой точке последовательности, в набор точек в многомерном пространстве, которые декодер будет использовать для генерации значимого вывода для данной задачи.

### Декодер генерирует ответное предложение поэтапно. Он использует векторы контекста кодировщика и внутренние скрытые состояния для генерации следующего слова в последовательности. Он продолжает генерировать слова, пока не *выведет EOS\_token* , представляющий конец предложения. В декодер можно встроить [механизм внимания](https://arxiv.org/abs/1409.0473), чтобы помочь ему «обратить внимание» на определенные части ввода при генерации вывода.

При выборе конкретного кодера и декодера был проведён анализ и сравнены общие характеристики моделей. Из числа моделей, показывающих хорошие результаты в качестве сегментации, можно выделить кодировщик *ResNet*(50/101) в связке с декодером *PPM\_deepsup.* Хорошие характеристики скорости распознавания имеет кодировщик *MobileNetV2dilated* в связке с декодером *C1\_deepsup.*

На изображении 2.6 представлено сравнение характеристик выбранных архитектур кодировщика и декодера.

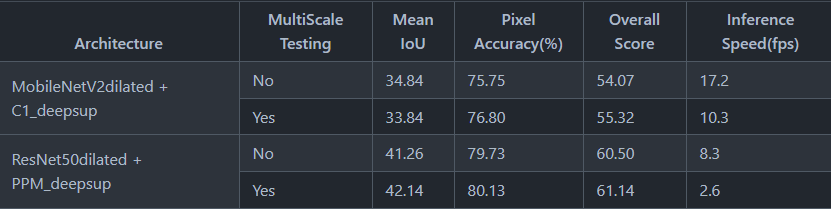


Рисунок 2.6 – Характеристики *ResNet* и *MobileNet*

Подробнее описать происходящее на таблице

Задача распознавания стен относится к задачам *Indoor Object Segmentation* и *Scene Segmentation.*

### Ниже описаны три современные модели для сегментации сцен.

***PSPNet* –** это сеть разбора сцены, которая объединяет глобальное представление с модулем *Pyramid Pooling Module* (*PPM*).

***UPerNet*** – это модель, основанная на *Feature Pyramid Network* (*FPN*) и *Pyramid Pooling Module* (*PPM*). Ему не нужна расширенная свертка, оператор, требующий много времени и памяти. Без наворотов он сравним или даже лучше по сравнению с *PSPNet*, но требует гораздо меньшего времени обучения и меньшего объема памяти графического процессора.

***HRNet* –** это недавно предложенная модель, которая сохраняет представления с высоким разрешением по всей модели без традиционного узкого места.

***1.3.2*** Данные, на которых была обучена сеть имеют важную роль. Количество хороших наборов данных ограничено сложностью нормализации данных. А в случае с задачей сегментации, также и созданием дополнительной разметки на изображениях.

*COCO* (*Common Objects in Context*) – это крупномасштабный набор данных для задач обнаружения и сегментации [6].

*COCO* имеет несколько функций:

– сегментация объекта;

– признание в контексте;

– сегментация пиксельного материала;

– 330 000 изображений (помечено > 200 000);

– 1,5 миллиона экземпляров объектов;

– 80 категорий объектов.

На изображении 2.9 представлен пример размеченных изображений из набора данных *COCO.*



Рисунок 2.9 – Пример изображений набора данных *COCO*

[*ADE20K*](http://groups.csail.mit.edu/vision/datasets/ADE20K/index.html) – это крупнейший набор данных с открытым исходным кодом для семантической сегментации и анализа сцены, выпущенный командой *MIT Computer Vision*[7].

# *ADE20K* состоит из изображений из баз данных *SUN* и *Places*. Изображения полностью аннотированы объектами. Многие изображения также содержат части объектов и под-части объектов. Изображения анонимны – лица и номерные знаки размываются.

Всего для оценки включено 150 семантических категорий, которые включают такие вещи, как небо, дорога, трава, и дискретные объекты, такие как человек, машина, кровать. На изображениях есть неравномерное распределение объектов, имитирующих более естественное появление объектов в повседневной сцене.

Последняя версия набора данных содержит:

– 27 574 изображения (25 574 для обучения и 2000 для тестирования), охватывающих 365 различных сцен;

– 707 868 уникальных объектов из 3 688 категорий, а также их определение и иерархия *WordNet*;

– 193 238 аннотированных частей и под-частей объектов;

– Аннотации многоугольников с атрибутами, временем аннотации, порядком глубины.

На рисунке 2.10 отражено несколько примеров, показывающих изображения, сегментирования объектов:

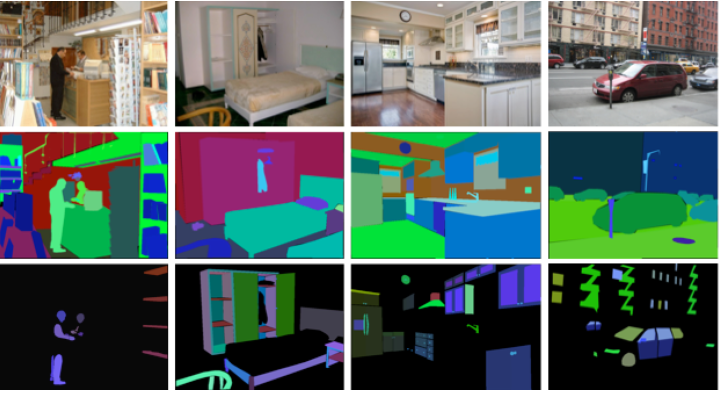


Рисунок 2.10 – Пример изображений набора данных [*ADE20K*](http://groups.csail.mit.edu/vision/datasets/ADE20K/index.html)

Добавить сравнение библиотек ?

Для мобильного устройства была выбрана сеть с моделью кодировщика *MobileNetV2dilated,* моделью декодировщика *C1\_deepsup,* обученная на данных из набора [*ADE20K*](http://groups.csail.mit.edu/vision/datasets/ADE20K/index.html) .

Для использования на персональном компьютере была выбрана сеть с моделью кодировщика *Resnet50dilated,* моделью декодировщика *ppm\_deepsup,* обученная на данных из набора [*ADE20K*](http://groups.csail.mit.edu/vision/datasets/ADE20K/index.html).