

# Курсовая работа по теме «Система антитеррористической безопасности»

Николай Макаров

Май 2023

## Содержание

<b>1 Введение</b>	<b>3</b>
1.1 Описание проблемы . . . . .	3
1.2 Описание набора данных . . . . .	3
1.2.1 Снимки пустой кабины лифта . . . . .	3
1.2.2 Снимки со стикерами на стенках лифта и т.д . . . . .	4
1.2.3 Снимки, содержащие оставленные предметы . . . . .	5
1.3 Постановка задачи . . . . .	5
<b>2 Основная часть</b>	<b>6</b>
2.1 Выделение рабочей поверхности . . . . .	6
2.2 Получения оценок на шум . . . . .	7
2.3 Метрики оценки работы . . . . .	7
2.4 Сравнение алгоритмов . . . . .	8
2.4.1 Стратегия «max» . . . . .	8
2.4.2 Стратегия «mean» . . . . .	9
2.4.3 Стратегия «median» . . . . .	10
2.5 Сравнение стратегий . . . . .	11
<b>3 Выводы</b>	<b>12</b>
<b>4 Приложения</b>	<b>13</b>
4.1 Приложение 1: матрицы ошибок . . . . .	13
4.2 Приложение 2: детектор границы Кэнни . . . . .	14
4.3 Приложение 3: выравнивание гистограммы для BGR изображений . . . . .	14
4.4 Приложение 4: ссылка на репозиторий . . . . .	15

## Перечень терминов и определений

1. Confusion matrix/Матрица ошибок - это таблица, в которой с реальными ответами сопоставлены прогнозы алгоритма. Структура матрицы ошибок представлена на рисунке 1.

		Predicted 0	Predicted 1
Actual 0	TN	FP	
	FN	TP	
Actual 1			

Рис. 1: Структура матрицы ошибок.

2. Precision - определяется формулой  $\frac{TP}{TP+FP}$ . В рамках нашей задачи, интерпретируется как доля действительно оставленных предметов среди всех участков изображения, которые алгоритм выделил, как инородные. Например, если на снимке на полу кабины лежит 3 ручки, а алгоритм выделил 2 из них, да ещё и добавил 5 рамок для совершенно пустых мест, то precision будет равен  $\frac{2}{7}$ .
3. Recall - определяется формулой  $\frac{TP}{TP+FN}$ . В рамках нашей задачи интерпретируется как отношение тех забытых предметов, что алгоритм смог найти к количеству всех оставленных в кабине лифта объектов. В примере выше recall составит  $\frac{2}{3}$ .
4.  $F_\beta$  - определяется формулой  $\frac{(1+\beta^2)precision*recall}{\beta^2*precision+recall}$  - взвешенное гармоническое среднее recall и precision. Если  $\beta > 1$ , то больший вес имеет recall. Мы будем измерять  $F_2$ .
5.  $MaxNeighbour_x$  - фильтр, который текущему пикселю присваивает наиболее частое значение среди всех его пикселей-соседей, находящихся от него на расстоянии не более, чем  $x$ .
6.  $M_x$  - сокращение для  $MaxNeighbour_x$ .
7.  $Dilate_x$  - фильтр, присваивающий текущему пикселю максимальное значение среди всех его пикселей-соседей, находящихся от него на расстоянии не более, чем  $x$ .
8.  $D_x$  - сокращение для  $Dilate_x$ .
9. Валидационный набор данных - набор данных, на котором мы тестируем алгоритм и подбираем его параметры.
10. validation - сокращение для валидационного набора данных.
11. Тестовый набор данных - набор данных, на котором мы тестируем качество работы алгоритма. Представляет собой данные, которые алгоритм никогда не видел, то есть, ни на одном этапе обучения они не присутствовали.
12. test - сокращение для тестового набора данных.
13. color - сокращение для цветного изображения.
14. gray - сокращение для чёрно-белого изображения.

# 1 Введение

## 1.1 Описание проблемы

Моей задачей было придумать и реализовать алгоритм поиска оставленных в кабине лифта вещей. Это действительно значимая проблема, потому что забытые в лифте предметы могут нести реальную угрозы жизни и здоровью людей.

## 1.2 Описание набора данных

Набор данных состоит из 151 снимка кабины лифта размера 1600x1200, полученных с помощью камеры, расположенной непосредственно внутри кабины. Эту выборку мы сразу разделил на обучающую, валидационную и тестовую. Из обучающей выборки (которая содержит только фотографии пустой кабины лифта) мы будем извлекать оценки на шум, речь о которых пойдёт ниже, на валидационной подбирать параметры алгоритма, а на тестовой выборке будем проверять корректность работы алгоритма. С точки зрения получения какой-либо информации, наш алгоритм никогда не будет видеть снимке, лежащие в тестовом наборе данных.

Изображения, содержащиеся в нашем наборе данных, можно разделить на 3 категории:

### 1.2.1 Снимки пустой кабины лифта

Мы начинаем со снимка пустой кабины лифта.



Рис. 2: Кадр в начальный момент времени

Во время движения лифта кабина шатается из стороны в сторону, что приводит в движение и камеру, которая находится внутри кабины. Из-за этого все последующие снимки получаются несколько сдвинутыми относительно начального положения. Помимо этого, освещённость кабины также не является постоянной, от кадра к кадру она может довольно сильно варьироваться.



Рис. 3: Кадр с изменённой освещённостью. Видно, что помещение стало более ярким.

Глазами это заметить в статике не получится, но второй снимок смешён относительно первого. Увидеть это возможно, взглянув на оптический поток между двумя этими изображениями.

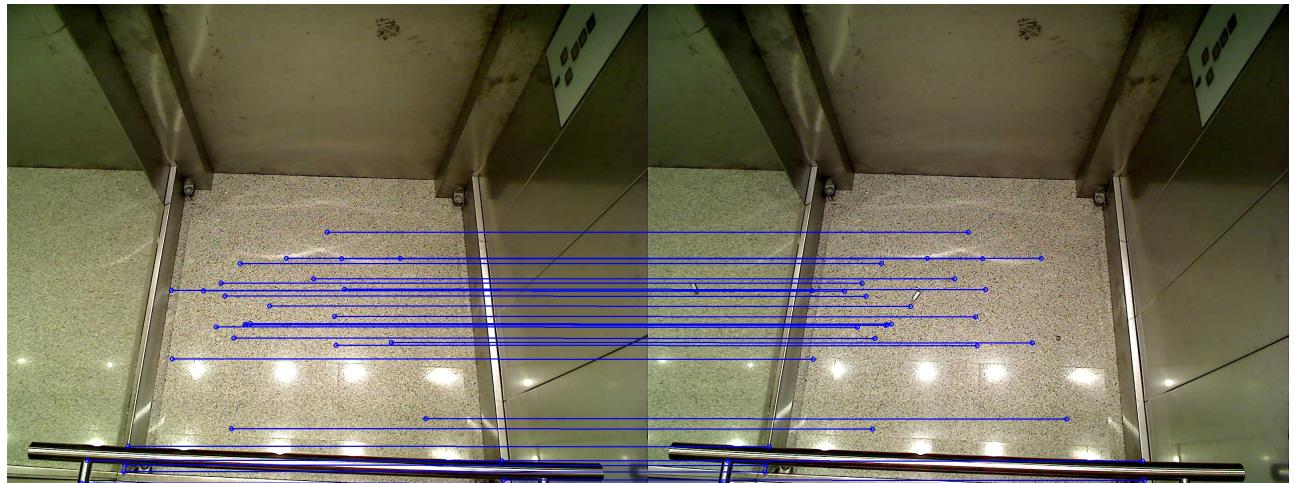


Рис. 4: Оптический поток между двумя изображениями

В статье это заметить едва ли получится, но не все из этих отрезков имеют одинаковую длину. Более того, Они также чуть-чуть отличаются по направлению.  
Ещё одна проблема оптического потока - его довольно долго считать. На одноплатном компьютере то может стать узким местом алгоритма.

### 1.2.2 Снимки со стикерами на стенках лифта и т.д

Часто люди клеят на стенки лифта объявления, стикеры и т.д. Едва ли эти вещи несут в себе какую-то угрозы, поэтому их мы хотим игнорировать.



Рис. 5: На зеркало приклеен черный стикер.

Также на левой стенке кабины расположено зеркало, в котором могут отражаться лежащие внизу предметы. Отражения мы также хотим игнорировать.  
Помимо этого, дверь лифта может открываться и закрываться, что, разумеется, отличает такой кадр от оригинального. Но пол кабины при этом остаётся неизменным, значит, на такие изменения мы также хотим закрывать глаза.

### 1.2.3 Снимки, содержащие оставленные предметы

Наконец мы дошли до самой многочисленной категории изображений - это снимки, содержащие оставленные в кабине лифта предметы.



Рис. 6: На полу кабины лежат 3 предмета.

На снимке выше в кабине лифта лежат 3 предмета, а также в зеркале отражается один из них. Мы хотим выделить все 3 объекта в красную рамочку, а отражения проигнорировать.

## 1.3 Постановка задачи

Итак, мы хотим создать и реализовать алгоритм, способный выделять на снимке предметы, который там изначально не было и обводить их в красную рамку (т.е., делать детекцию). При этом, этот алгоритм должен игнорировать предметы, появляющиеся на стенах кабины лифта и не реагировать на отражение в зеркале. Также он должен учитывать изменчивую освещённость помещения и небольшой сдвиг снимков друг относительно друга, вызванный движением кабины.

## 2 Основная часть

С помощью оптического потока сдвиг кабины мы компенсировать не сможем, поэтому я сформулировал следующую рабочую гипотезу: **кабина в действительности устойчива, но каждый раз, когда мы делаем снимок, к изображению добавляется некий случайный шум, который портит изображение.**

Если мы имеем дело с шумом, то можно попробовать извлечь его из запущенных изображений и оценить. В рамках нашей гипотезы оценка шума с его последующим удалением, и, возможно, постобработка результата позволят нам выделить инородные объекты на изображении.

Учитывая всё вышесказанное, решение поставленной задачи можно разбить на 3 последовательных этапа:

1. Выделение на стартовом изображении рабочей поверхности - пола кабина лифта. Как результат работы на этом шаге - бинарная маска, выделяющая рабочую зону;
2. Получение из набора изображений, не содержащих оставленных внутри кабины предметов, случайного шума. Аппроксимация этого шума с его последующим удалением, а также, возможно, небольшая постобработка позволяют нам качественно отделить поверхность пола кабины от инородных объектов. Как результат работы на этом шаге - бинаризованное изображение, где 0 означает, что пиксель принадлежит рабочей зоне, а 255 - что пиксель является частью забытого в лифте предмета;
3. Выделение предмета, лежащего на полу кабины в красную рамку. Как результат работы на этом шаге - оригинальное изображение, на котором все найденные инородные объекты выделены в рамку;

Теперь перейдём к более подробному описанию каждого из этих шагов.

### 2.1 Выделение рабочей поверхности

Для выделения рабочей поверхности я использовал детектор границ Кэнни и небольшой постпроцессинг, но обо всё по порядку.

Для начала, я уменьшил начальное изображение, которое вы можете видеть на рисунке 1, в 4 раза, это здорово улучшило работу алгоритма, и выполнил выравнивание гистограммы. Затем я пропустил результат через фильтр Кэнни. На выходе получил следующее изображение:

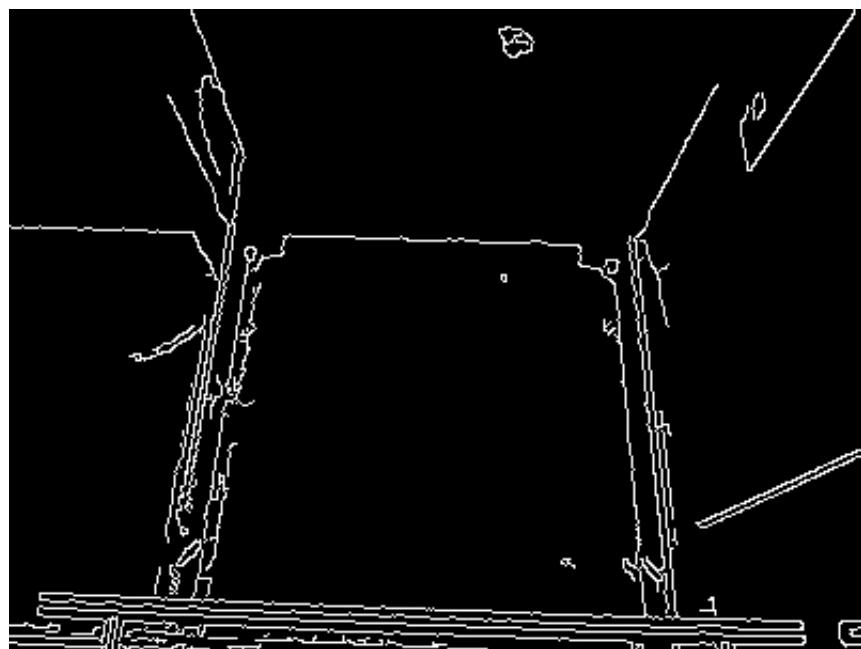


Рис. 7: Границы, выделенные фильтром Кэнни

Как вы можете заметить, алгоритм отработал не идеально: он обнаружил несуществующую границу в нижнем левом углу кабины и в целом выделил границы достаточно неровно. Однако, мы можем легко это исправить с помощью небольшой пост-обработки - для начала, добьёмся непрерывного контура границы, сделать это можно, например, применением фильтра dilation (возможно, последовательно несколько раз). После этого все пиксели, лежащие внутри этого контура, которые алгоритм посчитал внутренними, пометим, как пиксели рабочей поверхности. У нас всё ещё остаются неровные края и внутренние прорехи,

как на рисунке 6. От них мы можем легко избавиться, например, так: если снизу и сверху от пикселя есть пиксели рабочей зоны, то он сам также является пикселем рабочей зоны. Это крайне просто, но довольно эффективный метод улучшения результата работы фильтра Кэнни. В результате получаем следующую маску;



(a) Вычисленная маска.

(b) Идеальная маска.

Рис. 8: Сравнение вычисленной по алгоритму и идеальной маски.

Результат не идеальный, но его можно попытаться улучшить, например, используя иной алгоритм обнаружения границ или более сложную постобработку изображения. Но в целом, мы получили достаточно корректную маску.

## 2.2 Получения оценок на шум

При получении оценок на шум мы следовали вполне просто логике. Давайте выберем одно любое изображения, не содержащее инородных объектов и отличное от начального изображения, и вычтем его из начального изображения. Всё, что получилось в их разности - это шум. Если мы возьмём максимальное значение пикселя этой разности и минимальное, то получим верхнюю и нижнюю оценку на шум соответственно. Если же мы поступим аналогичным образом для каждого из «пустых» изображений, то получим 2 массива: один содержит верхние оценки, другой - нижние. Затем из этих наборов мы можем выбрать одну верхнюю и одну нижнюю оценку для шума. Я рассматривал оценки, полученные как среднее всех оценок, медиану всех оценок и наибольшее из значений (наименьшее для оценки снизу).

Также у алгоритма вычисления границ на шум есть 2 параметра - это количество каналов изображений, с которыми он работает и применение выравнивания гистограммы для изображений. Здесь может возникнуть вопрос, а как алгоритм применим к цветным картинкам, они ведь содержат 3 канала? При работе с BGR изображениями, значение всех трёх каналов пикселя суммируются, а затем происходит вычитание. Учитывая это, получаем следующую таблицу оценок.

Стратегия	Цветные изображения		Чёрно-белые изображения	
	С выравниванием	Без выравнивания	С выравниванием	Без выравнивания
mean	-523.875, 425.5	-390.875, 308.125	-174.5, 147.5	-127.875, 106.375
median	-491.5, 329	-381.5, 212	-162.5, 112	-126, 73
max	-701, 650	-533, 515	-235, 225	-175, 179

Таблица 1: Нижняя и верхняя оценки шума для различных комбинаций параметров алгоритма оценки.

Оценки мы получили, теперь самое время их применить и посмотреть, насколько удобно будет работать с изображениями, их которых мы с помощью такой аппроксимации убрали шум.

## 2.3 Метрики оценки работы

Если мы хотим понять, например, какой подход к выбору оценок лучше, или с чем лучше работать - с цветными или чёрно-белыми картинками, нужно ввести метрики для оценивания работы алгоритма. Как это принято в задаче детекции, будем считать две оценки - precision и recall.

Помимо этого, иногда полезна иметь перед глазами непосредственно матрицу ошибок. Для каждого из представленных ниже алгоритмов матрицу ошибок можно найти в Приложении 1.

## 2.4 Сравнение алгоритмов

Мы будем замерять целевые метрики как на тестовой, так и на обучающей выборке. В итоговой таблице, которую вы сможете увидеть в разделе 2.4.4 мы результаты работы алгоритма с различными наборами параметров, а пока исследуем, какая постобработка наиболее оптимальная для различных стратегий.

### 2.4.1 Стратегия «max»

При данной стратегии шум покрывается самым сильным образом, мы можем ожидать, что алгоритм даст достаточно консервативные результаты. В силу этого, мы выбрали 2 варианта постобработки: с применением фильтра MaxNeighbour и тремя последовательными применениями фильтра Dilate, и сразу с применением фильтров Dilate, без MaxNeighbour.

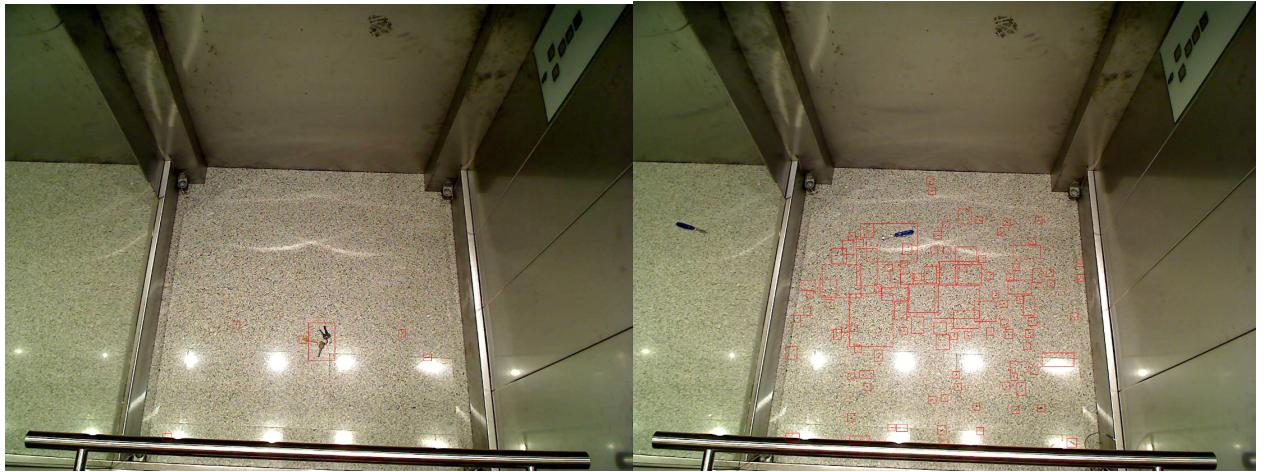
с применением трёх фильтров Dilate, и с более жесткий вариант, в котором перед фильтрами Dilate идёт фильтр MaxNeighbour.

Для первого варианта постобработки итоговая таблица выглядит следующим образом:

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	0.065	0.812	0.246	0.450	0.818
Gray, без выравнивания	<b>0.084</b>	<b>0.906</b>	<b>0.306</b>	0.400	<b>0.909</b>
Color, с выравниванием	0.063	0.821	0.241	<b>0.563</b>	0.818
Color, без выравнивания	0.055	0.872	0.220	0.370	<b>0.909</b>

Таблица 2: Результаты с применением фильтров D<sub>7</sub> × 3

Как вы можете заметить, у этого алгоритма, независимо от параметров, довольно низкий precision, но высокий recall. То есть, алгоритм верно находит большинство оставленных предметов, но вместе с этим, он выделяет и довольно много (примерно 1 к 10) лишних областей. Однако, не всё так плохо - зачастую, алгоритм выделяет всего лишь 2-3 лишних области, но иногда случается так, что лишних рамок становится порядка 100, именно эти редкие случаи и портят общую картину.



(a) Не самый плохой случай: всего 3 лишних рамки.

(b) Ужасный случай.

Рис. 9: Ошибки бывают очень разные.

Фильтр MaxNeighbour должен довольно мягко (мягче, чем Erode), очищать изображение от шума, поэтому количество ошибок FP должно стать сильно меньше, а значит, precision должен вырасти.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	0.565	0.222	0.253	<b>1.000</b>	0.272
Gray, без выравнивания	0.589	<b>0.368</b>	<b>0.398</b>	<b>1.000</b>	0.364
Color, с выравниванием	<b>0.672</b>	0.334	0.371	<b>1.000</b>	0.364
Color, без выравнивания	0.489	0.248	0.275	<b>1.000</b>	<b>0.455</b>

Таблица 3: Результаты с применением фильтров M, D<sub>7</sub> × 3

Как мы можем видеть, precision действительно стал намного лучше - он вырос в 8-9 раз. К несчастью, recall стал заметно ниже - фильтр MaxNeighbour счёл часть изображений шумом, поэтому метрика упала в 2-3 раза.

В целом, налицо trade-off: либо хороший recall, но крайне низкий precision, либо неплохой precision, но заметно более низкий recall.

В целом, от этой стратегии мы ожидали довольно консервативных оценок. Она даёт верхнюю оценку на шум, то есть, призвана максимально его нивелировать, конечно, часть предметов из-за этого будут не замечены. В противовес этому, мы ожидаем, что ложных рамок практически не будет - как мы видим выше, желаемого мы добиться не смогли.

Вынесем в итоговую таблицу этого раздела лучшие варианты алгоритма. Здесь важно заметить, что мы можем сравнивать алгоритмы только по результатам их работы на валидационном датасете, ведь тестовая выборка символизирует данные, которые алгоритм будет получать в процессе своей работы, то есть, обучаться он них мы не можем.

Параметры и постобработка	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, без выравнивания; D <sub>7</sub> × 3	0.084	<b>0.906</b>	0.306	0.400	<b>0.909</b>
Gray, без выравнивания; M <sub>3</sub> , D <sub>7</sub> × 3	0.589	0.368	<b>0.398</b>	<b>1.000</b>	0.364
Color, с выравниванием; M <sub>3</sub> , D <sub>7</sub> × 3	<b>0.672</b>	0.334	0.371	<b>1.000</b>	0.364

Таблица 4: Лучшие наборы параметров и фильтров для постобработки при стратегии «max»

То есть, мы либо находим 90% всех инородных объектов в кабине лифта, но на каждую верную рамку будет, в среднем, 11 некорректных, либо находим всего 33% от оставленных предметов, но число некорректных рамок будет на порядок ниже, чем в первом случае.

#### 2.4.2 Стратегия «mean»

От этой стратегии выбора оценки на шум мы ожидаем более высоких показателей precision и recall. Оценки на шум будут более мягкими, из-за этого на этапе постобработки можно будет проводить более тонкую настройку. Здесь мы выбрали следующие наборы фильтров: M<sub>3</sub> × 3, D<sub>7</sub> × 3; M<sub>3</sub> × 4, D<sub>7</sub> × 3; M<sub>3</sub> × 5, D<sub>7</sub> × 3. Посмотрим, какие параметры наиболее лучшим образом согласуются с различными вариантами постобработки.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	0.654	0.761	0.737	0.909	0.909
Gray, без выравнивания	<b>0.658</b>	0.872	<b>0.819</b>	<b>1.000</b>	<b>1.000</b>
Color, с выравниванием	0.647	0.752	0.728	<b>1.000</b>	0.909
Color, без выравнивания	0.628	<b>0.880</b>	0.815	<b>1.000</b>	<b>1.000</b>

Таблица 5: Результаты с применением фильтров M<sub>3</sub> × 3, D<sub>7</sub> × 3

Здесь мы видим намного более существенные результаты: и precision, и recall довольно высокие, более того, такой подход уже превосходит результат, который нам гарантирует стратегия «max». Но, возможно, мы сможем добиться и более весомых результатов.

Теперь посмотрим, что будет, если увеличить количество фильтров MaxNeighbour. Ожидается, что precision вырастит, а recall несколько упадёт.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	<b>0.804</b>	0.735	0.748	<b>1.000</b>	0.909
Gray, без выравнивания	0.714	0.812	0.790	0.909	0.909
Color, с выравниванием	0.790	0.769	0.773	<b>1.000</b>	0.818
Color, без выравнивания	0.741	<b>0.855</b>	<b>0.830</b>	<b>1.000</b>	<b>1.000</b>

Таблица 6: Результаты с применением фильтров M<sub>3</sub> × 4, D<sub>7</sub> × 3

Результаты даже лучше, чем мы ожидали - precision вырос довольно сильно - на 20-30 %. В то же время, recall упал всего лишь на 5-10 %.

Если же мы добавим ещё один MaxNeighbour фильтр, то precision должен подрасти ещё, а recall упасть, возможно, довольно сильно.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	0.853	0.692	0.719	<b>1.000</b>	0.727
Gray, без выравнивания	0.808	0.786	0.790	<b>1.000</b>	<b>0.909</b>
Color, с выравниванием	<b>0.856</b>	0.710	0.735	<b>1.000</b>	0.727
Color, без выравнивания	0.794	<b>0.821</b>	<b>0.816</b>	0.909	<b>0.909</b>

Таблица 7: Результаты с применением фильтров  $M_3 \times 5, D_7 \times 3$

Старый trade-off имеет место и здесь - либо выше recall, либо выше precision, но разница теперь намного менее сильно выражена.

Лучшие алгоритмы, полученные с помощью такой стратегии, представлены в таблице ниже:

Параметры и постобработка	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, без выравнивания; $M_3 \times 3, D_7 \times 3$	0.658	0.872	0.819	<b>1.000</b>	<b>1.000</b>
Color, без выравнивания; $M_3 \times 3, D_7 \times 3$	0.628	<b>0.880</b>	0.815	<b>1.000</b>	<b>1.000</b>
Gray, с выравниванием; $M_3 \times 4, D_7 \times 3$	0.804	0.735	0.748	<b>1.000</b>	0.909
Color, без выравнивания; $M_3 \times 4, D_7 \times 3$	0.741	0.855	<b>0.830</b>	<b>1.000</b>	<b>1.000</b>
Color, с выравниванием; $M_3 \times 5, D_7 \times 3$	<b>0.856</b>	0.710	0.735	<b>1.000</b>	0.727
Color, без выравнивания; $M_3 \times 5, D_7 \times 3$	0.794	0.821	0.816	0.909	0.909

Таблица 8: Лучшие наборы параметров и фильтров для постобработки при стратегии «mean»

Лучшее, что мы имеем на этом этапе: либо мы верно находим 90% оставленных предметов, но 40% всех рамок будут выделять пустоту, либо всего 15% от числа всех рамок будут некорректными, но мы обнаружим лишь 70% оставленных вещей.

#### 2.4.3 Стратегия «median»

Стратегия «median» ещё более «мягкая», чем «mean». Мы ожидаем, что именно она даст наилучшие результаты. Применим и в этом случае те же методы постобработки, что и для стратегии «mean».

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	0.339	0.923	0.687	1.000	0.909
Gray, без выравнивания	0.222	<b>0.949</b>	0.573	0.846	<b>1.000</b>
Color, с выравниванием	<b>0.349</b>	0.932	<b>0.699</b>	<b>1.000</b>	0.909
Color, без выравнивания	0.193	<b>0.949</b>	0.532	<b>1.000</b>	<b>0.909</b>

Таблица 9: Результаты с применением фильтров  $M_3 \times 3, D_7 \times 3$

Сразу же мы видим отличное значение параметра recall - 95% всех оставленных предметов найдены верно. Но и число ложных срабатываний алгоритма велико - на 1 верный прогноз мы наблюдаем, в среднем, 4 неверных.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	<b>0.554</b>	0.880	0.787	<b>1.000</b>	0.909
Gray, без выравнивания	0.408	0.923	0.737	0.917	<b>1.000</b>
Color, с выравниванием	0.541	0.906	<b>0.798</b>	<b>1.000</b>	0.909
Color, без выравнивания	0.369	<b>0.949</b>	0.722	<b>1.000</b>	<b>1.000</b>

Таблица 10: Результаты с применением фильтров  $M_3 \times 4, D_7 \times 3$

Ещё один фильтр MaxNeighbour смог неплохо сократить число ложных срабатываний алгоритма, но оно всё ещё довольно велико.

Параметры	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, с выравниванием	<b>0.716</b>	0.863	<b>0.829</b>	<b>1.000</b>	<b>0.909</b>
Gray, без выравнивания	0.570	0.906	0.811	<b>1.000</b>	<b>0.909</b>
Color, с выравниванием	0.683	0.846	0.808	<b>1.000</b>	<b>0.909</b>
Color, без выравнивания	0.568	<b>0.923</b>	0.821	<b>1.000</b>	<b>0.909</b>

Таблица 11: Результаты с применением фильтров  $M_3 \times 5, D_7 \times 3$

В целом, стратегия «median» позволяет добиться, в среднем, более высоких значений целевой метрики. Как и прежде, выделим лучшие варианты алгоритма.

Параметры и постобработка	Validation			Test	
	Precision	Recall	F <sub>2</sub>	Precision	Recall
Gray, без выравнивания; $M_3 \times 3, D_7 \times 3$	0.222	<b>0.949</b>	0.573	0.846	<b>1.000</b>
Color, с выравниванием; $M_3 \times 3, D_7 \times 3$	0.349	0.932	0.699	<b>1.000</b>	0.909
Gray, с выравниванием; $M_3 \times 4, D_7 \times 3$	0.554	0.880	0.787	<b>1.000</b>	0.909
Color, без выравнивания; $M_3 \times 4, D_7 \times 3$	0.369	<b>0.949</b>	0.722	<b>1.000</b>	<b>1.000</b>
Gray, с выравниванием; $M_3 \times 5, D_7 \times 3$	<b>0.716</b>	0.863	<b>0.829</b>	<b>1.000</b>	0.909
Color, без выравнивания; $M_3 \times 5, D_7 \times 3$	0.568	0.923	0.821	<b>1.000</b>	0.909

Таблица 12: Лучшие наборы параметров и фильтров для постобработки при стратегии «median»

В очередной раз, trade-off примерно следующий: либо мы верно находим 95% оставленных вещей, но 2 из 3 предсказаний будут неверными, либо находим только 86% объектов, но доля неверных прогнозов опустится до 30%.

## 2.5 Сравнение стратегий

Теперь составим финальную таблицу, в которой сравним наилучшие алгоритмы, которые нам удалось выработать для каждой из трёх стратегий.

№	Стратегия, параметры и постобработка	Validation			Test	
		Precision	Recall	F <sub>2</sub>	Precision	Recall
1	Max, Gray, без выравнивания; $D_7 \times 3$	0.084	0.906	0.306	0.400	0.909
2	Max, Gray, без выравнивания; $M_3, D_7 \times 3$	0.589	0.368	0.398	<b>1.000</b>	0.364
3	Max, Color, с выравниванием; $M_3, D_7 \times 3$	0.672	0.334	0.371	<b>1.000</b>	0.364
4	Mean, Color, без выравнивания; $M_3 \times 3, D_7 \times 3$	0.628	0.880	0.815	<b>1.000</b>	<b>1.000</b>
5	Mean, Color, без выравнивания; $M_3 \times 4, D_7 \times 3$	0.741	0.855	<b>0.830</b>	<b>1.000</b>	<b>1.000</b>
6	Mean, Color, с выравниванием; $M_3 \times 5, D_7 \times 3$	<b>0.856</b>	0.710	0.735	<b>1.000</b>	0.727
7	Median, Color, без выравнивания; $M_3 \times 4, D_7 \times 3$	0.369	<b>0.949</b>	0.722	<b>1.000</b>	<b>1.000</b>
8	Median, Gray, с выравниванием; $M_3 \times 5, D_7 \times 3$	0.716	0.863	0.829	<b>1.000</b>	0.909

Таблица 13: Сравнение лучших алгоритмов, подобранных для каждой из трёх стратегий.

Можем выделить 3 наиболее успешных варианта нашего алгоритма: 5, 6 и 7.

Алгоритм 6 верно находит только лишь 70% от всего числа оставленных в кабине лифта предметов, но также он достаточно редко срабатывает ложно - всего в 1 случае из 7. В то же время, Алгоритм 7 способен обнаружить уже 95% инородных объектов, но вероятность ложного срабатывания сильно повышается - в среднем, до 2 случаев из 3. Алгоритм 5 обладает наиболее оптимальным соотношение recall и precision - он найдёт, в среднем, 86% предметов и ошибётся в 1 случае из 4.

### **3 Выводы**

Основываясь на проведённом нами исследовании, можно сделать следующие вывод:

- Лучше работать с цветными изображениями, чем с чёрно-белыми;
- Выравнивание гистограммы - едва ли узкое место нашего алгоритма, но об однозначной пользе или вреде мы сказать не можем;
- Чем пытаться максимально подавить шум с помощью его оценивания, лучше взять довольно мягкую оценку и подобрать хорошие фильтры для постобработки;

## 4 Приложения

### 4.1 Приложение 1: матрицы ошибок

Для трёх алгоритмов, которые мы признали наиболее оптимальными по тому или иному критерию, нами были составлены матрицы ошибок для валидационной и тестовой выборки.

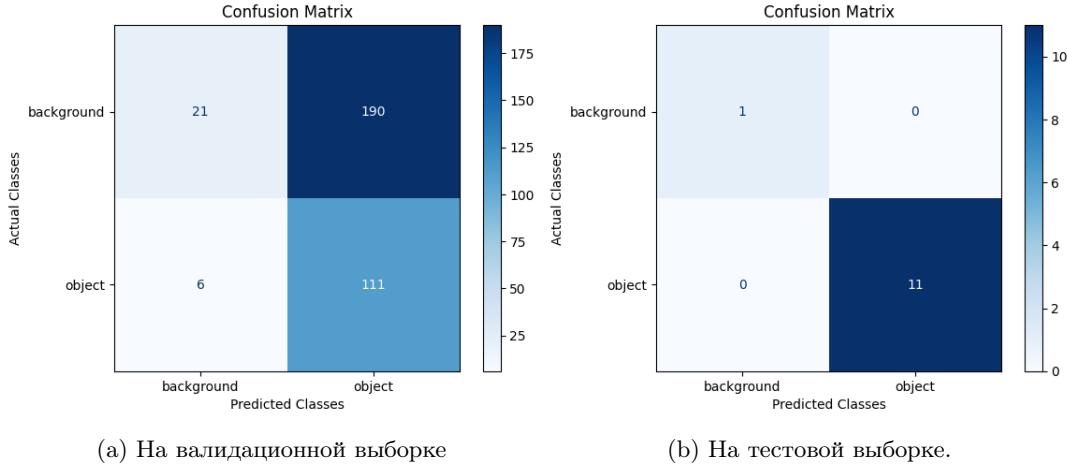


Рис. 10: Матрицы ошибок алгоритма 5 на разных выборках.

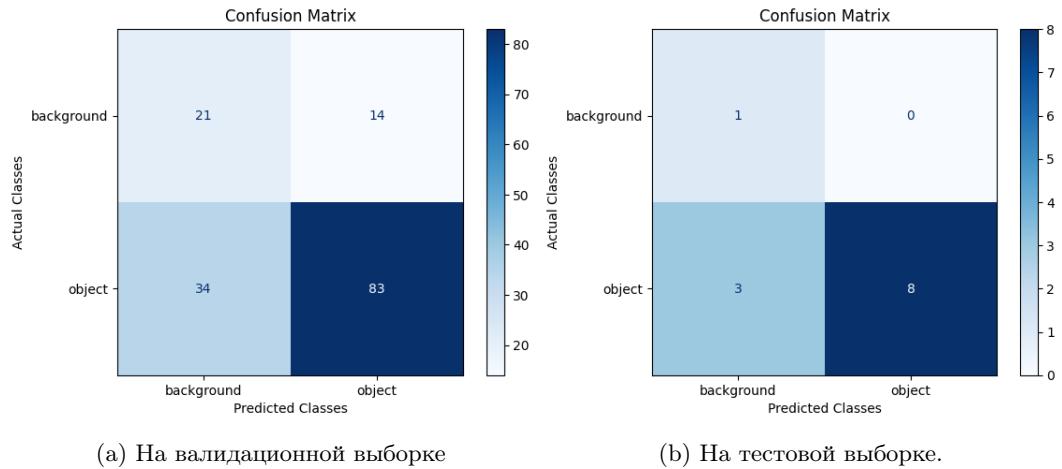


Рис. 11: Матрицы ошибок алгоритма 6 на разных выборках.

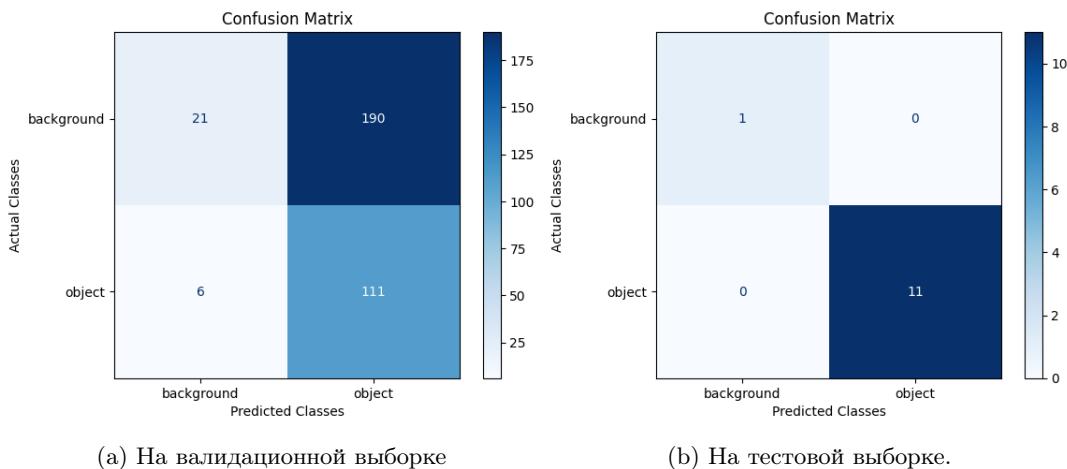


Рис. 12: Матрицы ошибок алгоритма 7 на разных выборках.

## 4.2 Приложение 2: детектор границы Кэнни

Детектор границ Кэнни работает следующим образом:

1. Применяем к изображению Гауссовский шум;
2. Для каждого пикселя считаем амплитуду градиента и его направление. Амплитуда градиента вычисляется по формуле  $\sqrt{G_x^2 + G_y^2}$ , направление -  $\arctan(\frac{G_y}{G_x})$ .  $G_y, G_x$  - градиент вдоль у и х соответственно, полученные с помощью свёртки с ядром Собеля. Направление градиента округлено так, чтобы совпадать с вертикальным, горизонтальным или одним из двух диагональных направлений;
3. Выявляются локальные максимумы амплитуды градиента: пиксель сравнивается с соседними пикселями в направлении градиента. Если текущий пиксель - локальный максимум, то он считается пикселием границы.
4. Мы определяем 2 значения threshold1 и threshold2. Все пиксели, которые были признаны пикселями границы на предыдущем шаге, сравниваются с этими числами. Если значение амплитуды превышает threshold2, то является пикселием границы. Если лежит между threshold1 и threshold2, то является пикселием границы, только если он связан с пикселием, который уже лежит в границе. Если меньше threshold1, то этот пиксель принадлежит внутренности.

## 4.3 Приложение 3: выравнивание гистограммы для BGR изображений

Для того, чтобы выровнять гистограмму BGR изображений, используйте следующий алгоритм:

1. Перейдите от BGR к пространству YCrCb. В этом пространстве: Y - компонента яркости, Cr, Cb - красная и синяя компонента соответственно;
2. Выполните выравнивание гистограммы для канала Y;
3. Перейдите обратно в пространство BGR;

#### 4.4 Приложение 4: ссылка на репозиторий

В этом репозитории вы можете найти весь исходный код и данные с которыми я работал. Также реализован cli, так что вы можете наглядно убедиться, что всё работает.



Рис. 13: QR-код на репозиторий с проектом.