|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Специального машиностроения**

КАФЕДРА **СМ11 «Подводные роботы и аппараты»**

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Андреев Евгений Викторович

*фамилия, имя, отчество*

Группа СМ11-11М

Тип практики **Научно-исследовательская работа**

Название предприятия **НУК СМ МГТУ им. Н.Э. Баумана**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Андреев Е. В.

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Макашов А. А. *подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра **«Подводные роботы и аппараты»** **(СМ11)**

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

**(производственной практики)**

на предприятии **НУК СМ МГТУ им. Н.Э. Баумана**

Студент Андреев Евгений Викторович, СМ11-11М

(фамилия, имя, отчество; индекс группы)

**Тема научно-исследовательской работы:**

Использование каскадного детектора для построения системы позиционирования подводного аппарата

**Дата выдачи задания «» февраля 2020 г.**

**Руководитель НИР**   **/** Макашов А. А.

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /** Андреев Е. В.

(подпись, дата) (Фамилия И.О.)

РЕФЕРАТ

Отчёт на стр., ч., рис., источников, таблиц.

ИСПОЛЬЗОВАНИЕ КАСКАДНОГО ДЕТЕКТОРА ДЛЯ ПОСТРОЕНИЯ СИСТЕМЫ ПОЗИЦИОНИРОВАНИЯ ПОДВОДНОГО АППАРАТА

Перечень ключевых слов: АНПА, опорный маркер специального вида, алгоритм детектирования, каскадный детектор Хаара, донная зарядная станция, видеокамера.

Целью данной работы является исследование возможности использования каскадного классификатора Хаара для позиционирования АНПА у донной зарядной станции с помощью опорных маркеров специального вида.

В процессе работы был проведён сбор и систематизация информации по подготовке данных и обучению каскадного классификатора Хаара методом Виолы-Джонса. Были спроектированы опорные маркеры специального вида двух типов и изготовлен макет донной зарядной станции. Обучены несколько каскадных детекторов различных конфигураций для обнаружения каждого типа маркеров. Проведена калибровка камеры, получера матрица камеры и вектор коэффициентов дисторсии, разработан пример продвинутой фильтрации информации, полученной на выходе каскадного детектора. Исследованы величины задержек и потоков IP-камеры.

В результате исследования было проведено сравнение производительности различных каскадов, проанализированы скорость и точность работы алгоритмов позиционирования на базе каскадов Хаара и LBP, а также влияние разрешения видеопотока и количества позитивных и негативных примеров, использованных при обучении.

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc41920271)

[ВВЕДЕНИЕ 6](#_Toc41920272)

[1 Каскадный классификатор Хаара 7](#_Toc41920273)

[2. Используемые маркеры и конфигурация донной станции 8](#_Toc41920274)

[2.1 Разработка маркеров специального вида 8](#_Toc41920275)

[2.2 Описание макета донной зарядной станции 9](#_Toc41920276)

[3 Определение положения камеры 10](#_Toc41920277)

[4. Обучение каскадного детектора 11](#_Toc41920278)

[4.1 Этап подготовки данных 11](#_Toc41920279)

[4.2 Этап обучения каскада 14](#_Toc41920280)

[5 Применение каскадного детектора для определения координат 19](#_Toc41920281)

[5.1 Калиброква камеры 19](#_Toc41920282)

[5.2 Определение масштабного коэффициента 20](#_Toc41920283)

[5.3 Описание алгоритма фильтрации 22](#_Toc41920284)

[1.3 Сравнение каскдов Хаара и LBP 25](#_Toc41920285)

[2.4 Определение относительных смещений 27](#_Toc41920286)

[2.5 Определение поворота относительно нормали 27](#_Toc41920287)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc41920288)

[Приложение А. Сценарий для администрирования большого количества изображений в процессе обучения каскадного классификатора 31](#_Toc41920289)

[Приложение Б. Сценарий для наполнения файла фоновых изображений 32](#_Toc41920290)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете о НИР применяются следующие сокращения и обозначения:

АНПА – Автономный необитаемый подводный аппарат;

ВК – Видеокамера;

ДЗС – Донная зарядная станция;

СК – система координат;

LBP – Least binary patterns.

# ВВЕДЕНИЕ

В настоящее время всё большее применение находят автономные необитаемые подводные аппараты (АНПА). К видам технических работ, осуществляемых аппаратами, можно отнести обследование трубопроводов и кабелей, проверку точности карт, фото- и видеосъёмку, в том числе маршрутную, осмотр опор эстакад и платформ и много другое.

В этой связи актуальной задачей является разработка методов позиционирования подводного аппарата по данным видеосистемы. Для наведения предлагается использовать маркеры специального вида, исследование проводилось применительно к задаче стыковки АНПА с донной зарядной станцией.

Предполагается, что аппарат оснащён всеми необходимыми измерителями параметров движения, вопросы маневрирования в данной работе не рассматриваются.

Цель работы – исследование каскадного детектора для построения системы видеопозиционирования АНПА.

Задачи:

* сбор и систематизация информации по подготовке данных и обучению каскадного классификатора Хаара;
* разработка маркеров специального вида для построения системы видеопозиционирования;
* сравнение производительности различных каскадов;
* исследование готовых программных решений в библиотеке OpenCV для определения положения камеры относительно объекта.

# 1 Каскадный классификатор Хаара

Каскадный классификатор Хаара представляет собой специальный детектор границ, построенный на каскадах решающих деревьев, каждое из которых содержит в своих листьях один из примитивов Хаара (см. рисунок Рисунок 1) и два пороговых значения соответствующие ему.

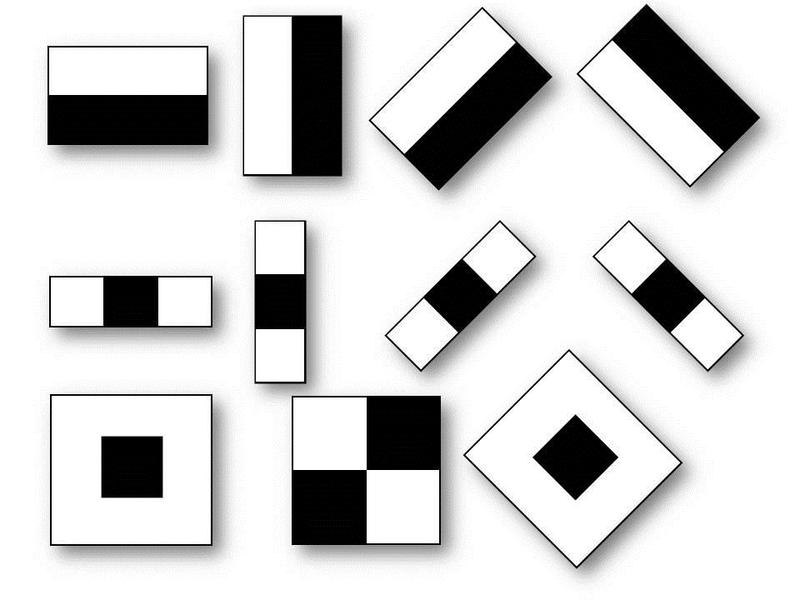


Рисунок – Некоторые примитивы Хаара

Обучение каскада методом Виолы-Джонса [1, 2] построено на однократном вычислении интегрального изображения и последующим последовательном переборе примитивов и расчёте значений признаков как разницы между суммарной яркостью пикселей, покрытых былой областью текущего примитива и суммарной яркости пикселей чёрной области. Сложность вычисления признака так же как и получения значения пикселя остается O(1): значение каждой подобласти можно вычислить скомбинировав 4 значения интегрального представления. Для определения наличия или отсутствия объекта на изображении в каждом каскаде находится сумма значений слабых классификаторов этого каскада. Каждый слабый классификатор выдает два значения в зависимости от того больше или меньше заданного порога значение признака, принадлежащего этому классификатору. В конце сумма значений слабых классификаторов сравнивается с порогом каскада и выносится решения найден объект или нет данным каскадом. Авторам статьи [1] удалось создать быстрый алгоритм поиска объектов, который пользуется успехом уже больше десятилетия.

# 2. Используемые маркеры и конфигурация донной станции

Использлование каскада Хаара позволит задействовать несколько небольших специально спроектированных опорных меркеров, разместив их на грани донной зарядной станции в отдалении друг от друга. Тем самым удаётся обойти ограничения, связанные с габаритами ARuCo-маркеров, для которых имеется готовая реализация алгоритмов видеопозиционирования в OpenCV.

## 2.1 Разработка маркеров специального вида

Для решения задачи навигации были разработаны 2 типа маркеров специального вида, показанных на рисунке Рисунок 2. Все они имеют размеры 30х30 мм.

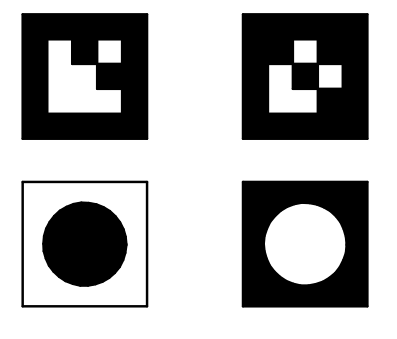


Рисунок 2 – Различные варианты опорных меток

Первый ряд представляет собой ARuCo-подобные маркеры с укрупнёнными внутренними структурными элементами. Стандартный модуль “aruco”, входящий в состав библиотеки OpenCV предлагает пользователю на выбор предопределённые словари меток размерами 4 х 4, 5 х 5 или 6 х 6 клеток. Больший размер внутренних элементов предполагает более устойчивое детектирование или же возможность уменьшить габариты маркера при неизменном расстоянии обнаружения. Однако же нет никакой необходимости равняться на алгоритмы, заложенные в известную библиотеку, тем более что поставлена задача использовать машинное обучение.

Маркеры второго ряда состоят из ещё более простых геометрических фигур и их комбинаций, именно на эти маркеры пал окончательный выбор, так как их преимущество над ARuCo и им подобными заключается в наличии круга в центре опорной метки. В дальнейшем планируется применять преобразование Хаффа для продвинутой фильтрации объектов, распознанных каскадом, и уточнения центров маркеров.

## 2.2 Описание макета донной зарядной станции

Для удобства тестирования и отладки разрабатываемых алгоритмов был собран небольшой макет донной зарядной станции, показанный на рисунке Рисунок 3. Пропорции станции приблизительно соответвуют пропорциям реального объекта: усечённая пирамида со стороной нижнего основания в 250 мм, верхнего – в 125 мм и высотой 125 мм.

Введём систему координат (СК), связанную с донной зарядной станцией. Её центр расположим в точке, где предполагается наличие стыковочного порта, ось абсцисс разместим в плоскости грани и направим горизонтально направо, ось ординат – в плоскости грани вверх. Ось OZ смотрит наружу и перпендикулярна грани.

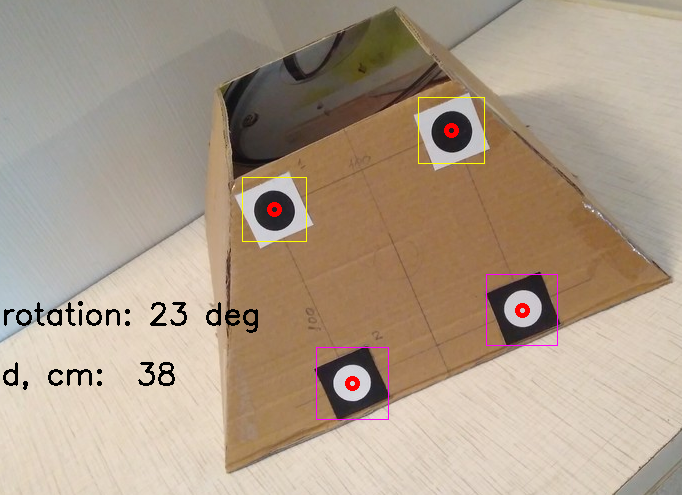


Рисунок 3 – Макет донной зарядной станции

Выбранные маркеры размерами 30 х 30 мм расположим в углах кварата со сторой 100 мм и геометрическим центром в начале введённой СК.

Каждый каскад (первый - для верхних, второй - для нижних) возвращает массив прямоугольников, внутри которых с определённой вероятностью на изображении находится искомый объект. Как видно из рисунка выше, границы прямоугольников не всегда соответствуют границам объекта, к тому же, объект может иметь сложную форму или быть искажённым в зависимости от угла зрения. Красные кружочки на рисунке Рисунок 3 обозначают центры соответствующих прямоугольников, а не центры маркеров.

# 3 Определение положения камеры

Задача определения положения камеры относительно объекта, для которого известны экранные и трёхмерные координаты минимум 4-х точек, исследована достаточно хорошо [3, 4] и в англоязычной литературе называется Perspective-n-Point Camera Pose Estimation.

# 4. Обучение каскадного детектора

Значительную часть исследования занял сбор информации об обучении каскадного классификатора и формулирование эффективных подходов к работе с большим объёмом обучающих данных.

## 4.1 Этап подготовки данных

Приведём последовательность действий для подготовки изображений и их разметки

1. Для тренировки каскада потребуется два набора данных: с объектом и без него. Допускается наличие в кадре одновременно нескольких объектов. Удобнее всего не снимать последовательно несколько сотен кадров, заснять два видео. Подобный подход позволит существенно сэкономить время, а также одновременно учесть все негативные эффекты, такие как размытия, расфокусировка, возникающие во время движения реального робота.
2. Полученные видеофайлы разбиваются на кадры. Например, это можно сделать с помощью бесплатной утилиты “Free Video to Jpeg Converter”. Наибольшее количество *уникальных* кадров определяется по формуле:

F \* T,

где

F – частота, кадров/с;

T - длительность видеоролика, с.

1. Для удобства предлагается переименовать изображения, оставив короткий псевдоним из латинских символов и порядковый номер (далее будет обоснована важность этого шага). Например, IMG\_20200509\_144931.jpg превращается в with0001.jpg. Используется бесплатное ПО FastStone Image Viewer, имеющее инструменты пакетной обработки фотографий.
2. Для тренировки алгоритма, как правило, используются изображения небольшого разрешения (1280 х 720 пикселей, 640 х 480 пикселей или меньше). В противном случае обучение займёт чрезвычайно много времени. В этом же ПО после переименования можно отмасштабировать изображения с уменьшением разрешения до указанных выше значений.
3. С помощью утилиты opencv\_createsamples.exe (входит в пакет поставки фреймворка OpenCV) на позитивных изображениях необходимо вручную указать объект(-ы). Запук утилиты производится командой:

opencv\_annotation.exe --annotations=good\_2.dat

--images=./extracted\_images/with\_original

Ключ «annotations» с параметром «good\_2.dat» задаёт выходной файл, в который будет сохранена информация о разметке, ключ «images» определяет папку с изображениями, содержащими требуемый объект. На рисунке Рисунок 4 показан процесс аннотации изображения, содержащего макет донной зарядной станции с нанесёнными на неё маркерами специалльного вида. Предполагается осуществлять навигацию автономного необитаемого подводного аппарата по системе технического зрения в процессе его стыковки с зарядной станцией.

О том, как взаимодействовать с программой разметки [5], после запуска напоминает подсказка в командной строке:

\* mark rectangles with the left mouse button,

\* press 'c' to accept a selection,

\* press 'd' to delete the latest selection,

\* press 'n' to proceed with next image,

\* press 'esc' to stop.

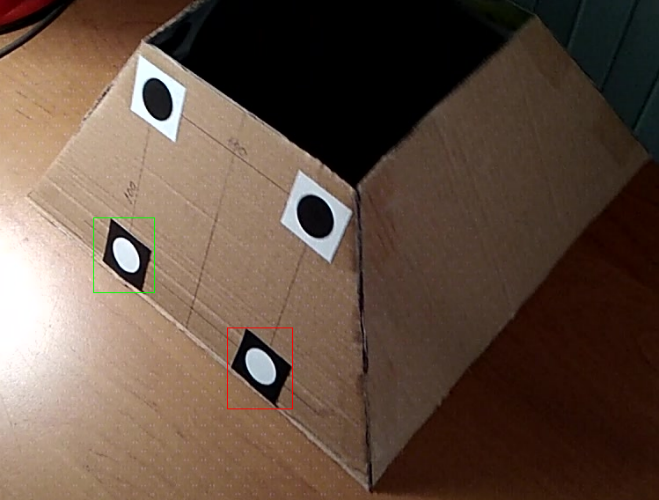


Рисунок – Процесс разметки изображений. Подлежащие детектированию объекты выделены рамками. Слева – подтверждённая аннотация (после нажатия клавиши «с»), справа – не подтверждённая

Программа не будет откликаться на действия пользователя, если выбрана отличная от английской раскладка клавиатуры. По нажатии клавиши «Escape» приложение завершит работу и по указанному пути появится файл с содержимым, показанным на рисунке Рисунок 5.

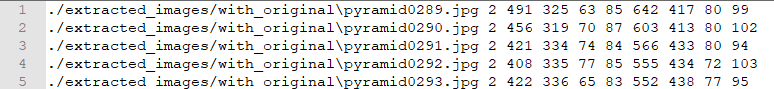


Рисунок – Вид содержимого файла аннотаций изображений с объектом

Каждая строчка соответствует отдельной аннотации, сначала указан путь к файлу (относительный или абсолютный, с зависимости от указанных при запуске программы разметки аргументов); следующий параметр – количество объектов на изображении, в данном примере на каждом кадре по два маркера одного вида; оставшиеся цифры обозначают координаты прямоугольников: левый верхний угол, затем правый нижний. Порядок описания прямоугольников соответствует порядку разметки пользователем.

Вообще говоря, файл негативных примеров может содержать любые изображения, включая и те, которые роботу не доведётся встретить в процессе выполнения задачи. Однако детектор будет функционировать намного лучше, если в качестве негативных примеров использовать изображения обстановки, в которой предстоит вести обнаружение. На рисунке Рисунок 6 показан фрагмент такого файла. Он не содержит никакой лишней информации, кроме относительного пути к изображениям.

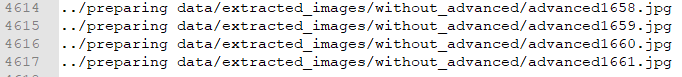


Рисунок – Вид содержимого файла с описанием фоновых изображений

## 4.2 Этап обучения каскада

Для обучения каскада [5, 6] требуется передать последнему файл-вектор, представляющий собой бинарный набор сжатых изображений объекта. Рассмотрим подробнее команду создания вектора:

opencv\_createsamples.exe -info good\_1\_fixed.dat -vec samples.vec -num 1274 -w 24 -h 24 -show

Данное действие выполняет программа opencv\_createsamples.exe, также входящая в пакет поставки фреймворка компьютерного зрения. С ключём «info» указывается путь к файлу с аннотациями, полученному ранее. Ключ «vec» обозначает имя выходного вектор-файла, «num» - это количество позитивных изображений, а параметры «w» и «h» специфицируют ширину и выстоу впикселях. Чем больше размеры объекта, тем дольше будет проходить обучение, но тем точнее будет детектирование. Важно, чтобы отношение указанных величин соответствовало пропорциям реального объекта: к примеру, для поиска на изображении конфеты или болта стоит указать -w 40 -h 10, а для обнаружения маркера как на рисунке 2 лучше оставить квадратные пропорции.

Запуск обучения каскада выполняется командой

opencv\_traincascade.exe -data haar\_output -vec samples.vec -bg bad.dat -numStages 16 -numThreads 12 -w 24 -h 24 -numPos 1100 -numNeg 2955 -mode ALL,

где

data – путь к каталогу куда будут сохранены результаты;

vec – входной вектор-файл;

bg – файл фоновых изображений (background);

numStages – количество итераций (каскадов);

numThreads – количество задействованных потоков процессора;

w, h – ширина и высота изображений вектора в пикселях;

numPos – количество позитивных примеров, необходимых для обучения каскада. Это число обязательно должно быть меньше числа изображений в vec-файле. Причина будет объяснена ниже;

numNeg – количество фоновых изображений, чем больше, тем лучше;

mode – режим на стандартном наборе примитивов Хаара или же на расширенном (включая повёрнутые на 90º).

Программа отобразит следующий вывод:

PARAMETERS:

cascadeDirName: haar\_output

vecFileName: samples.vec

bgFileName: bad\_navigation.dat

numPos: 1100

numNeg: 2955

numStages: 16

precalcValBufSize[Mb] : 1024

precalcIdxBufSize[Mb] : 1024

acceptanceRatioBreakValue : -1

stageType: BOOST

featureType: HAAR

sampleWidth: 24

sampleHeight: 24

boostType: GAB

minHitRate: 0.995

maxFalseAlarmRate: 0.5

weightTrimRate: 0.95

maxDepth: 1

maxWeakCount: 100

mode: ALL

Number of unique features given windowSize [24,24] : 261600

Здесь важными параметрами являются minHitRate и maxFalseAlarmRate. Первый обозначает точность определения, которую должен достичь каскад на каждой стадии. Значение 0,995 означает, что из 1000 изображений 5 будут ложноположительными срабатываниями. Второй – какое количество изображений, не содержащих объект, будет отсеяно на каждой стадии. К примеру, для обученного каскада из 8 стадий будет отсеиваться 1 – 0,58 = 0,996 = 99,6% всех негативных изображений.

По завершении обучения каждой стадии программа будет выдавать подобные сообщения:

===== TRAINING 12-stage =====

<BEGIN

POS count : consumed 1100 : 1195

NEG count : acceptanceRatio 2955 : 2.89746e-05

Precalculation time: 4.876

+----+---------+---------+

| N | HR | FA |

+----+---------+---------+

| 1| 1| 1|

+----+---------+---------+

| 2| 1| 1|

+----+---------+---------+

| 3| 1| 1|

+----+---------+---------+

| 4| 1| 1|

+----+---------+---------+

| 5| 0.999333| 0.836887|

+----+---------+---------+

| 6| 0.999333| 0.843316|

+----+---------+---------+

| 7| 0.998667| 0.649746|

+----+---------+---------+

| 8| 0.996667| 0.581726|

+----+---------+---------+

| 9| 0.996667| 0.461591|

+----+---------+---------+

END>

Training until now has taken 0 days 1 hours 23 minutes 3 seconds.

Третий столбец – значение FalseAlarmRate, переход к следующей стадии происходит, когда оно становится менее 0,5. Фраза «Required leaf false alarm rate achieved. Branch training terminated.» сообщит об успешном завершении обучения классификатора [4]. Работа обученного детектора по распознаванию маркеров специального вида паказана на рисунке Рисунок 7.



Рисунок – Работа каскадного классификатора Хаара. Слева видно одно ложноположительное срабатывание

На каждом этапе каскад совершает ошибки, принимая некоторые фоновые изображения за изображения с объектом. Для тренировки каждой последующей стадии берутётся количество изображений, соответвтующих параметру numPos плюс все ошибочные с предыдущей стадии. В примере вывода выше для 12-й стадии обучения присутствует строчка POS count : consumed 1100 : 1195,сообщающая о том, что для тренировки нынешней стадии было взято 1195 изображений, 1100 из которых изначально переданы в vec-файле. Следовательно, на прерырущем этапе ошибочно были распознаны 95 изображений.

Приблизительно количество позитивных изображений, необходимых для создания vec-файла, можнно оценить по формуле:

V ≥ numPos + (numStages - 1) \* (1 - minHitRate) \* numPos + S,

где

V – число изображений, использованных для создания вектора;

S – суммарное количество нераспознанных изображений на всех стадиях.

По приведённой выше формуле также можно оценить необходимое число объектов, передаваемых opencv\_traincascade.exe в параметре -numPos 1100. При меньшем числе пользователь получит ошибку «Bad argument (Can not get new positive sample. The most possible reason is insufficient count of samples in given vec-file).»

# 5 Применение каскадного детектора для определения координат

## 5.1 Калиброква камеры

Для более точного определения реальных координадт объекта необходимо произвести калибровку камеры с целью получения матрицы камеры и устранения различных абераций оптической системы. Матрица камеры используется для проецирования точек трёхмерного пространства на плоскость изображения. Нелинейные параметры внутренней калибровки, такие как коэффициенты дисторсии, также имеют важное значение, хотя и не могут быть включены в линейную модель, описываемую матрицей внутренней калибровки. Большинство современных алгоритмов калибровки камеры определяет их вместе с параметрами линейной части модели. Параметры внутренней калибровки относятся только к камере, но не к сцене, поэтому они изменяются только в том случае, когда меняются соответствующие настройки камеры.

При использовании камеры свет из снимаемой сцены фокусируется и захватывается. Этот процесс уменьшает число измерений у данных, получаемых камерой, с трёх до двух (свет из трёхмерной сцены преобразуется в двухмерное изображение). Поэтому каждый пиксель на полученном изображении соответствует лучу света исходной сцены. Во время калибровки камеры происходит поиск соответствия между трёхмерными точками сцены и пикселями изображения.

В случае идеальной камеры-обскуры для задания такого соответствия достаточно одной матрицы проекции. Однако в случае более сложных камер искажения, вносимые линзами, могут сильно повлиять на результат. Таким образом, функция проецирования принимает более сложный вид и часто записывается как последовательность преобразований.

Для получения матрицы камеры и коэффициентов дисторсии воспользуемся приложением opencv\_interactive-calibration.exe, запустив программу с ключами -t=chessboard -sz=25.125.

В качестве шаблона для калибровки было выбрано изображение шахматной доски с шириной клетки в 25,125 мм. На рисунке Рисунок 8 показан процесс интерактивной калибровки, в левой части отображены захваченные точки на текущем и предыдущих кадрах, а в правой – зоны покрытия кадра тестовым изображением.

|  |  |
| --- | --- |
|  |  |

Рисунок 8 – Процесс интерактивной калибровки

При наличии достаточного количества данных программа выдаст сообщение об успешной калибровке, а вычисленные матрица камеры и коэффициенты дисторсии будут сохранены в файле cameraParameters.xml. Подробное руководство по интерактивной калибровке камеры описано в [].

## 5.2 Определение масштабного коэффициента

Для измерения расстояния по видеопотоку требуется произвести калибровку камеры по расстоянию. С этой целью был изготовлен стенд, повторяющий конфигурацию маркеров ДЗС, и отснята серия изображений, показанных на рисунке Рисунок 9.

|  |  |
| --- | --- |
|  |  |
|  |  |

Рисунок 9 – Изображения для калибровки камеры. Расстояние на фотографиях сверху составляет 500 мм, снизу – 1000 мм.

Плоскость с маркерами расположена строго перпендикулярно лучу, соединяющему центр объектива и центр квадрата, сторона квадрата составляет 100 мм. У детектированных каскадами маркеров находился геометрический центр, представляющий собой пересечение диагоналей прямоугольника. Расстояния между каждой парой вершин были усреднены, результаты измерения приведены в таблице . Исходные изображения имели высокое разрешение (3840 х 2160) для уменьшения погрешностей измерения.

Таблица 2 – Результаты калибровки камеры по расстоянию

|  |  |
| --- | --- |
| Расстояние, мм | Значение, пикселей |
| 500 | 600 |
| 1000 | 300 |

В пересчёте по ширине на разрешение 640 х 480 пикселей имеем 100 px при 500 мм и 50 px при 1000 мм. Итоговый фрагмент кода по определению расстояния показан ниже:

int w = frame.cols;

float scale = float(w) / 640;

double average = (upper + lower) / 2;

double calculated\_distance = 50 \* 100 / (average / scale);

if (calculated\_distance > 0 && calculated\_distance <= 200)

this->dist = calculated\_distance;

Здесь

w – ширина кадра;

sacle – масштабный коэффициент кадра;

average – среднее арифметическое длин верхнего и нижнего ребра.

## 5.3 Описание алгоритма фильтрации

Для получения относительно устойчивого детектирования необходимо было разработать процедуру фильтрации данных, получаемых во время работы каскадных детекторов. Возможны 4 варианта работы классификатора:

1. объект присутствует на изображении и был обнаружен;
2. объект отсутствует и не был обнаружен;
3. объект присутствует, но обнаружен не был;
4. объект отсутствует, но был обнаружен.

Первые 2 случая соответствуют ожидаемой работе детектора, последние два – ложно отрицательные и ложно положительные срабатывания. Поскольку в данном исследовании мы в основном опираемся на работу каскада Хаара, то уменьшить вероятность наступления 3-го случая без изменения параметров каскада не представляется возможным, в то время как для ложно положительных срабатываний можно предложить усовершенствованный алгоритм распознавания.

Блок-схема первой части алгоритма фильтрации представлена на рисунке Рисунок 10.

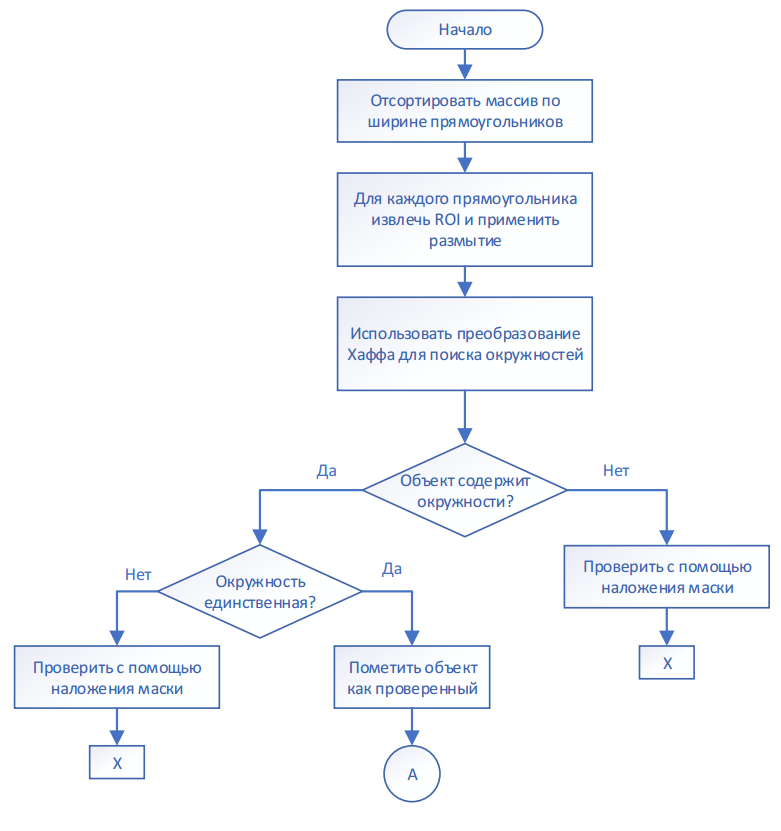


Рисунок 10 – Блок-схема фильтрации данных каскада, часть 1

Как уже было упомянуто в разделе 2.2, каждый каскад возвращает массив прямоугольников, внутри которых с определённой вероятностью на изображении находится искомый объект. В начале данный массив сортируется по ширине прямоугольника по убыванию. С помощью координат прямоугольника из исходного изображения извлекается ROI (Region of Interest) и применяется сглаживание. Внутри каждого ROI производится поиск окружностей с использованием преобразования Хаффа. Если каскад «утверждает». что в данной области находится искомый объект, при этом в ней не обнаружено ни одной окружности или же окружностей больше одной, допустимо произвести проверку с помощью наложения маски.

Маска (см. рисунок 11, правая часть) представляет собой динамически генерируемый шаблон, предназначенный для сравнения с исходным изображением (рис. 11 слева).

|  |  |
| --- | --- |
|  |  |

Рисунок 11 – ROI с распознанным маркером и сгенерированная маска

В случае если зафиксировано совпадедние, объект также был бы помечен как проверенный.

На блок-схеме, представленной на рисунке Рисунок 10, соответствующие ветви помечены знаком «Х». Это означает что в текущем состоянии научно-исследовательской работы эти варианты не реализованы в виду их черезмерной трудоёмкости и необходимости предварительного сбора и статистической обработки большого объёма калибровочных данных. Ниже приведён фрагмент кода, демонстрирующий возможный вариант использования данного подхода.

...

else if (circles.size() > 1) {

Mat t;

if (m\_type == markerType::black\_circle) {

t = Marker::get\_template\_t1(roi.rows, roi.cols);

threshold(roi, roi, 200, 255, 0);

absdiff(roi, t, roi);

int nonZero = countNonZero(roi);

//if (nonZero < 0.1 \* frame\_gray.cols) {

// hough\_valid.push\_back(objects[i]);

//}

}

else {

t = Marker::get\_template\_t2(roi.rows, roi.cols);

threshold(roi, roi, 200, 255, 0);

absdiff(roi, t, roi);

int nonZero = countNonZero(roi);

//if (nonZero < 0.15 \* frame\_gray.cols) {

// hough\_valid.push\_back(objects[i]);

//}

}

} ...

Статичческие методы Marker::get\_template\_t1(...) и Marker::get\_template\_t1(...) возвращают сгенерированный шаблон с указанными параметрами ширины и высоты. В качестве упрощения принято считать, что окружность радиуса 0,35 от ширины прямоугольнка находится в центре маркера, проективные искажения не учитываем. Далее ищется матрица абсолютных разностей бинаризованного чёрно=белого изображения объекта и его двухтоновой маски. По количеству ненулевых элементов можно делать вывод о присутствии на изображении маркера.

## 1.3 Сравнение каскдов Хаара и LBP

Утилита «opencv\_traincascade.exe» позволяет обучать как каскады Хаара, так и LBP (Local Binary Patterns). В открытых источниках [5] упоминается, что 2-й тип каскадного детектора в несколько раз быстрее каскада Хаара и на 15-20% менее точен. Было проведено собственное сравнение, результаты которого представлены в таблице 1.

Таблица 1 - Сравнение производительности детектирования на основе каскадов Хаара и LBP

|  |  |  |
| --- | --- | --- |
| Тип | Источник видеоряда | Скорость, кардов/с |
| LBP | Тестовое видео, 1280 х 720 пикселей | 12,5 |
| HAAR | 11 |
| LBP | Потоковое видео с домашней веб-камеры, 640 х 480, разрешение по умолчанию для OpenCV | 17 |
| HAAR | 15 |
| LBP | Тестовое видео, 1280 х 720 пикселей, детектируется только один тип маркера (1-й тип - чёрный круг на белом фоне) | 17 |
| HAAR | 15 |
| LBP | Потоковое видео с домашней веб-камеры, 640 х 480, детектируется только один тип маркера (1-й тип - чёрный круг на белом фоне) | 20 |
| HAAR | 19 |

По данным таблицы 1 ожидаемой разницы в производительности не наблюдается. Однако в выводе программы «opencv\_traincascade.exe» при равных размерах поискового окна количество уникальных комбинаций примитивов, их размеров и положений сильно различается:

|  |  |
| --- | --- |
| stageType: BOOST  featureType: HAAR  sampleWidth: 24  sampleHeight: 24  boostType: GAB  minHitRate: 0.995  maxFalseAlarmRate: 0.5  weightTrimRate: 0.95  maxDepth: 1  maxWeakCount: 100  mode: ALL  Number of unique features given windowSize [24,24] : **261600** | stageType: BOOST  featureType: LBP  sampleWidth: 24  sampleHeight: 24  boostType: GAB  minHitRate: 0.995  maxFalseAlarmRate: 0.5  weightTrimRate: 0.95  maxDepth: 1  maxWeakCount: 100  Number of unique features given windowSize [24,24] : **8464** |

Сравнете значения, выделенные жирным. Автор связывает подобное поведение с особенностью разработанных маркеров: каскадам необходимо детектировать плоское контрастное изображение из простых геометрических фигур, в результате чего большое количество примитивов отсеивается на ранних стадиях. Таким образом, для детектирования достаточно весьма ограниченного набора примитивов и соответствующих им слабых классификаторов. Однако время полного обучения каскадов различается значительно:

|  |  |
| --- | --- |
| featureType: HAAR | featureType: LBP |
| ===== TRAINING 5-stage =====  <BEGIN  POS count : consumed 1100 : 1113  NEG count : acceptanceRatio 4617 : 3.13382e-05  Precalculation time: 5.091  +----+---------+---------+  | N | HR | FA |  +----+---------+---------+  | 1| 1| 1|  +----+---------+---------+  | 2| 1| 1|  +----+---------+---------+  | 3| 1| 0.565735|  +----+---------+---------+  | 4| 1| 0.565735|  +----+---------+---------+  | 5| 1| 0.286766|  +----+---------+---------+  END>  Training until now has taken 0 days **0 hours 41 minutes 35 seconds**.  ===== TRAINING 6-stage =====  <BEGIN  POS count : consumed 1100 : 1113  NEG count : acceptanceRatio 2 : 1.01725e-05  Required leaf false alarm rate achieved. Branch training terminated. | ===== TRAINING 5-stage =====  <BEGIN  POS count : consumed 1100 : 1107  NEG count : acceptanceRatio 4617 : 8.73515e-05  Precalculation time: 0.33  +----+---------+---------+  | N | HR | FA |  +----+---------+---------+  | 1| 1| 1|  +----+---------+---------+  | 2| 1| 1|  +----+---------+---------+  | 3| 1| 0.347412|  +----+---------+---------+  END>  Training until now has taken 0 days **0 hours 2 minutes 8 seconds**.  ===== TRAINING 6-stage =====  <BEGIN  POS count : consumed 1100 : 1107  NEG count : acceptanceRatio 0 : 0  Required leaf false alarm rate achieved. Branch training terminated. |

## 2.4 Определение относительных смещений

## 2.5 Определение поворота относительно нормали

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Rapid Object Detection using a Boosted Cascade of Simple Features, Paul Viola, Michael Jeffrey Jones. [электронный ресурс]. URL: <https://www.researchgate.net/publication/3940582_Rapid_Object_Detection_using_a_Boosted_Cascade_of_Simple_Features> (дата обращения: 10.05.2020).
2. Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц [электронный ресурс]. URL: <https://habr.com/ru/post/133826/> (дата обращения: 10.05.2020).
3. Страница документации фреймворка OpenCV. Real Time pose estimation of a textured object [электронный ресурс]. URL: <https://docs.opencv.org/master/dc/d2c/tutorial_real_time_pose.html> (дата обращения: 12.05.2020).
4. EPnP: An Accurate O(n) Solution to the PnP Problem. Vincent Lepetit, Francesc Moreno-Noguer, Pascal Fua, [электронный ресурс]. URL: <https://link.springer.com/article/10.1007/s11263-008-0152-6> (дата обращения: 12.05.2020).
5. Страница документации фреймворка OpenCV по обучению каскадного классификатора [электронный ресурс]. URL: <https://docs.opencv.org/4.1.1/dc/d88/tutorial_traincascade.html> (дата обращения: 12.05.2020).
6. Обучение каскада Хаара на примере поиска символов автомобильного номера OpenCV [электронный ресурс]. URL: <https://kostyakulakov.ru/opencv-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BA%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%B0-%D1%85%D0%B0%D0%B0%D1%80%D0%B0/> (дата обращения: 12.05.2020).
7. Страница форума stackoverflow.com [электронный ресурс]. URL: <https://stackoverflow.com/questions/50186866/opencv-cascade-training-fast-fails-required-leaf-false-alarm-rate-achieved-br> (дата обращения: 16.05.2020).
8. Страница форума stackoverflow.com [электронный ресурс]. URL: <https://stackoverflow.com/questions/8791178/haar-cascades-vs-lbp-cascades-in-face-detection> (дата обращения: 16.05.2020).

# Приложение А. Сценарий для администрирования большого количества изображений в процессе обучения каскадного классификатора

abs\_path = 'Your\_absolute\_path\_to\_base\_folder/'

which\_file = 2

if which\_file == 1:

f = open(abs\_path+'good\_1.dat')

f\_fixed = open(abs\_path+'good\_1\_fixed.dat', 'a')

else:

f = open(abs\_path + 'good\_2.dat')

f\_fixed = open(abs\_path + 'good\_2\_fixed.dat', 'a')

data = f.readlines()

print(type(data))

for line in data:

first\_part = line.strip().split()[0]

#print(first\_part)

file\_name = first\_part.split("\\")[-1]

print(file\_name)

if which\_file == 1:

try:

shutil.move(abs\_path+"from\_folder/"+file\_name,

abs\_path+" /to\_folder/"+file\_name)

line = line.strip().split('\\')

new\_line = line[0] + "/to\_folder/" + line[1] + "\n"

print(new\_line, end='')

f\_fixed.write(new\_line)

except FileNotFoundError:

print(file\_name, " does not exist")

input()

else:

try:

shutil.move(abs\_path+"from\_folder\_2/"+file\_name,

abs\_path+"to\_folder\_2/"+file\_name)

line = line.strip().split('\\')

new\_line = line[0] + "/to\_folder\_2/" + line[1] + "\n"

print(new\_line, end='')

f\_fixed.write(new\_line)

except FileNotFoundError:

print(file\_name, " does not exist")

input()

# Приложение Б. Сценарий для наполнения файла фоновых изображений

import os

f = open('bad.dat', 'w')

for i in range(1, 2956):

s = './extracted\_images/without\_800/advanced' + '{:04d}'.format(i) + ".jpg\n"

f.write(s)

# print(s, end = '')