Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Южный федеральный университет»

Факультет математики, механики и компьютерных наук

Направление подготовки 010400 — «Информационные технологии»

Задачи поиска и анализа шаблонных конфигураций объектов в видеопотоке

Выпускная квалификационная работа
на степень бакалавра
студента
Д. С. Людовских

Научный руководитель:
старший преподаватель
М. В. Пучкин

Ростов-на-Дону
2014

Содержание:

1. Введение……..….3

2. Постановка задачи и её характеристики…….……..4

3. Общий ход работы……...8

4. Применение сторонних библиотек……..17

5. Аспекты реализации……..19

6. Заключение…….25

7. Список литературы……26

1. Введение

В данной работе рассматривается задача обнаружения и распознавания на видеозаписи автомобилей в процессе торможения. По современным техническим требованиям каждое транспортное средство должно быть оборудовано стоп-сигналами – фонарями красного цвета, включающимися при торможении. На легковых автомобилях такие фонари расположены в форме равнобедренного треугольника, с основанием, параллельным поверхности дороги.

Таким образом, была поставлена задача рассмотреть алгоритмы поиска и анализа шаблонных конфигураций на изображениях, и реализовать приложение, позволяющее в реальном режиме времени обнаруживать и распознавать автомобили в процессе торможения.

Задача обнаружения объектов заключается в установлении наличия на изображении объекта, обладающего некоторыми определенными характеристиками. В данном случае под объектом понимается определенная конфигурация из трех компактных (по сравнению с линейными размерами изображения) областей красного цвета. Дополнительные сложности связаны как с наличием нескольких объектов на изображении (например, двух тормозящих рядом автомобилей), так и с наличием других источников света – например, запрещающих сигналов светофоров, которые довольно сложно отличить от стоп-сигналов автомобилей.

В качестве результата работы предполагается приложение, обнаруживающее в режиме реального времени на видеозаписи автомобили в процессе торможения и выделяющее их некоторым образом – например, ограничивающим прямоугольником.

2. Постановка задачи

В рамках выпускной квалификационной работы на степень бакалавра необходимо решить следующие задачи:

1. Проанализировать особенности задачи поиска и выделения в видеопотоке автомобилей в процессе торможения с точки зрения задач распознавания образов.
2. Разработать поэтапный алгоритм решения как композицию стандартных методов, выполнить подбор параметров.
3. Реализовать приложение, иллюстрирующее возможности решения поставленной задачи.

Данная задача может иметь актуальность при работе в режиме «онлайн», что накладывает определенные ограничения. Прежде всего, отметим, что подразумевается работа с видеопотоком – видеозаписью в реальном времени, представляющей из себя последовательность кадров с одинаковыми линейными размерами, разрешением и глубиной цвета, а также сходными параметрами искажения цвета (некоторые устройства довольно сильно искажают цвета при получении видеозаписи, однако это искажение цветов будет характерным для всей последовательности кадров).

В основной своей массе мобильные устройства, которые могут быть использованы в автомобилях для получения и обработки видеопотока, отличаются как довольно посредственными характеристиками видеокамеры, так и относительно небольшой вычислительной мощностью. Наиболее распространенными устройствами, подходящими для решения данной задачи, являются смартфоны и планшетные компьютеры, что и будет учитываться в данной работе.

Итак, предполагается работа с видеопотоком, полученным с мобильного устройства, что налагает следующие ограничения (связанные с типичными характеристиками таких устройств):

- Ширина: 640 пикселей
- Высота: 480 пикселей
- Горизонтальное разрешение: 96 точек на дюйм
- Разрешение по вертикали: 96 точек на дюйм
- Глубина цвета: 24 бит на пиксел

Видеопоток получен с помощью устройства, установленного внутри автомобиля. Изначально рассматриваем базовую задачу – обнаружение объектов на одном кадре. Предполагается, что именно эта задача будет являться наиболее сложной. Обработка последовательности кадров в некоторых случаях никакого принципиального отличия не имеет – момент торможения должен быть установлен довольно быстро (не более 1-2 секунд), что практически не позволяет использовать последовательность кадров для верификации предположений об обнаруженных автомобилях.

Задача обнаружения:

Изображение проверяется на наличие автомобилей в процессе торможения, если на изображении присутствуют останавливающиеся автомобили, то они выделяются.

Автомобиль будем считать останавливающимся, если у него включены стоп-сигналы. Подразумевается, что стоп-сигналы каждого автомобиля – три фонаря красного цвета, включающиеся при нажатии водителем на педаль тормоза (случаи неисправности одного или нескольких фонарей не рассматриваем). Мощность излучения стоп-сигнала выше, чем у габаритных огней. В соответствии с техническими требованиями на легковых автомобилях необходима установка двух стоп-сигналов по обе стороны ав-

томобиля, а также центрального стоп-сигнала, расположенного выше линии правого и левого стоп-сигналов. Таким образом, предполагаем, что искомые объекты – три довольно компактных области красного цвета, образующих равнобедренный треугольник, основание которого параллельно проезжей части. Из этого предположения можно получить критерий для двух нижних стоп-сигналов: модуль разности их абсцисс должен быть в несколько раз больше модуля разности их ординат:

$$|x_1 - x_2| > k|y_1 - y_2|$$

где (x_1, y_1) и (x_2, y_2) – координаты геометрических центров масс левой и правой нижних областей, соответствующих нижним стоп-сигналам. Константу k необходимо подбирать экспериментальным путем, разумным представляется выбор значения в диапазоне 10-20.

Пример работы:

Изображение на входе (рис. 2.1):

Рис. 2.1 Автомобиль останавливается перед кольцом (изображение на входе, алгоритм должен выделить автомобиль).

Изображение на выходе (рис. 2.2):

Рис. 2.2.Автомобиль останавливается перед кольцом (изображение на выходе, правильная работа алгоритма).

3. Общий ход работы

Работа программы разбивается на 2 этапа:

1. Обработка изображения
2. Поиск шаблонов

В свою очередь этап «Обработки изображения» разбивается на:

1. Обрезание верхней части
2. Применение фильтров обработки
3. Выделение областей

На вход этапа «Обработки изображения» подается исходное изображение, на выходе получаем множество областей.

Этап «Поиск шаблонов» разбивается на:

1. Анализ областей
2. Поиск подходящих под описание шаблонов
3. Выделение подходящих шаблонов

На вход этапа «Поиск шаблонов» подается исходное изображение и множество областей полученных на этапе «Обработки изображения», на выходе получаем итоговое изображение.

Рассмотрим этапы подробней.

3.1. Обработка изображения

Одним из наиболее простых и естественных способов обнаружения объекта (или объектов) является выделение его по цвету. Например, все запрещающие знаки на дороге красного цвета. Идея подхода состоит в нахождении порога, разделяющего изображение на светлый объект и темный фон.

3.1.1. Обрезание исходного изображения

Так как известно, что в верхней трети изображения автомобилей точно быть не может, то её можно обрезать для уменьшения площади обработки изображения.

3.1.2. Применение фильтров обработки

Поиск автомобилей в процессе торможения ведется по стоп-сигналам, которые загораются ярко-красным цветом. Таким образом, на изображении нужно выделить области ярко-красного цвета. И по этим областям и будет вестись поиск шаблонов.

Этот этап является одним из ключевых. Чем лучше на этом этапе будут выделяться области, тем легче будет на следующих этапах. Для выделения областей обычно используются фильтры. Фильтр — это инструмент, позволяющий редактировать изображение. Фильтры применяются для обработки зашумленного видеопотока, для повышения качества видео, либо для улучшения и облегчения последующей обработки.

Для выделения ярко-красных областей нужно применить последовательность фильтров: пороговый фильтр цвета, фильтр переводящий цветное изображение в изображение в оттенках серого (полутоновое изображение), фильтр переводящий полутоновое изображение в черно-белое изображение (бинарное изображение).

Пороговый фильтр получает на входе изображение и 3 диапазона, соответствующие цветам в RGB. Этот фильтр для каждого пикселя входного изображения делает следующее преобразование: если каждая компонента цвета пикселя входит в соответствующий диапазон, то цвет компонент пикселя не меняется, если же какой-то из цветов не входит в диапазон, то этот пиксель перекрещивается в черный (#000000).

Еще одна причина, почему стоит вырезать верхнюю часть — это присутствие на ней большого количества областей с высокой компонентой красного цвета (светофоры, яркие вывески, небо и т. д.), затрудняющих поиск.

Важно, чтоб порог был подобран верно. Если диапазоны цветовых компонент будут широкими, то на следующих этапах придется обрабатывать очень много информации, и возможны будут ложные срабатывания. С другой стороны если диапазоны будет слишком маленькими, то возможны ситуации, когда будут пропадать правильные варианты.

Далее изображение, полученное пороговым фильтром, переводится в изображение в оттенках серого с помощью соответствующего фильтра. Для каждого пикселя по формуле $Y' = 0.2126R' + 0.7152G' + 0.0722B'$ высчитывается новый цвет (Y′, Y′, Y′).

Последний фильтр, применяется для бинаризации изображения, в основе фильтра лежит метод Оцу.

Метод Оцу – это алгоритм вычисления порога бинаризации для полутонового изображения, используемый в области распознавания образов и обработки изображений.

Алгоритм позволяет разделить пиксели двух классов («полезный» и «фоновые»), рассчитывая такой порог, чтобы внутриклассовая дисперсия была минимальной.

Метод Оцу ищет порог, уменьшающий дисперсию внутри класса, которая определяется как взвешенная сумма дисперсий двух классов:

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t),$$

где веса ω_i — это вероятности двух классов, разделенных порогом t, а σ^2_i — дисперсия этих классов.

Оцу показал, что минимизация дисперсии внутри класса равносильна максимизации дисперсии между классами:

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_1(t)\omega_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2,$$

которая выражается в терминах вероятности ω_i и среднего арифметического класса μ_i, которое, в свою очередь, может обновляться итеративно. Эта идея привела к эффективному алгоритму.[1]

Алгоритм

1. Вычислить гистограмму и вероятность для каждого уровня интенсивности.
2. Вычислить начальные значения для $\omega_i(0)$ и $\mu_i(0)$.
3. Для каждого значения порога от t = 1 .. до максимальной интенсивности:
    1. Обновляем ω_i и μ_i
    2. Вычисляем $\sigma^2_b(t)$.
    3. Если $\sigma_b(t)$ больше, чем имеющееся, то запоминаем σ_b и значение порога t.
4. Искомый порог соответствует максимуму $\sigma^2_b(t)$.

По полученному бинарному изображению можно выделять области.

3.1.3. Выделение областей

После получения бинарного изображения нужно выделить из него области и занести их в специальную структуру данных BLOB.

BLOB (Binary Large Object — двоичный большой объект) — специальный тип данных, предназначенный, в первую очередь, для хранения изображений, аудио и видео, а также компилированного программного кода.

С помощью массива BLOB’ов на следующем этапе происходит поиск шаблонов автомобилей в процессе торможения.

3.2. Поиск шаблонов

Существует множество важных задач распознавания объектов, которые включают поиск на изображении окон, имеющих простую форму и стилизованное содержимое. Например, лицо анфас имеет вид овального окна, причем (при глубоком) масштабе все лица выглядят приблизительно одинаково. Другой пример: для камеры, установленной в передней части машины, все красные знаки светофоров выглядят однотипно.

Исходя из приведенных соображений, напрашивается такой подход к распознаванию объектов: определить все окна изображений, имеющие определенную форму, и проверить их на предмет наличия значимого объекта. Если о размерах объекта ничего не известно, поиск можно проводить при разных масштабах. В общем случае этот подход называется сравнением с шаблоном. Действительно, имеются определенный объекты, для эффективного поиска которых можно использовать схему сравнения с шаблоном.

Кроме того, хотя с помощью простой схемы сравнения с шаблоном многие объекты найти трудно (подобным образом трудно, например, найти конкретного человека, поскольку набор возможных окон, представляющих лицо, необъятен), очевидно, что для нахождения объектов можно эффективно использовать соображения относительно связи различных типов шаблонов.[2]

3.2.1. Анализ областей

На этом этапе происходят попытки отбросить неверные области.

Отсекаются очень малые по площади BLOB’ы. Тут нужно правильно подобрать размер площади, с которых происходит отбрасывание неверных областей, чтобы не потерять правильные варианты.

Попытки отсекать области по параметру заполненности не привели к хорошим результатам, не было найдено взаимосвязи между нужными и не-

нужными областями, поэтому было решено не отбрасывать по этому параметру.

3.2.2. Поиск подходящих под описание шаблонов

В полученном массиве ищутся тройки областей, геометрические центры масс которых образуют равнобедренный треугольник, основание которого параллельно проезжей части (рис. 3.1). Из этого предположения получаем критерий для двух нижних стоп-сигналов: модуль разности их абсцисс должен быть в несколько раз больше модуля разности их ординат:

$$|x_1 - x_2| > k|y_1 - y_2|,$$

где (x_1, y_1) и (x_2, y_2) – координаты геометрических центров масс левой и правой нижних областей, соответствующих нижним стоп-сигналам. Константа k подбирается экспериментальным путем.

Центральный же стоп-сигнал должен быть расположен примерно посередине между левым и правым стоп-сигналами на оси X:

$$k_1|x_1 - x_2| < max(|x_3 - x_1|, |x_3 - x_2|) < k_2|x_1 - x_2|,$$

где x_1 и x_2 – абсциссы геометрических центров масс левой и правой нижних областей, соответствующих нижним стоп-сигналам,

а x_3 – абсцисса геометрического центра масс области посередине, соответствующей центральному стоп-сигналу.

Константы k_1 и k_2 подбираются экспериментальным путем.

А по оси Y область центрального стоп-сигнал должна образовывать треугольник с левой и правой нижними областями так, чтобы его высота составляла примерно 0.5 от длины основания (диапазон значений высоты необходимо подобрать в процессе экспериментов):

$$k'_1|x_1 - x_2| < max(|y_3 - y_1|, |y_3 - y_2|) < k'_2|x_1 - x_2|,$$

где (x_1, y_1) и (x_2, y_2) – координаты геометрических центров масс левой и правой нижних областей, соответствующих нижним стоп-сигналам,

а y_3 – абсцисса геометрического центра масс области посередине, соответствующей центральному стоп-сигналу.

Константы k'_1 и k'_2 подбираются экспериментальным путем.

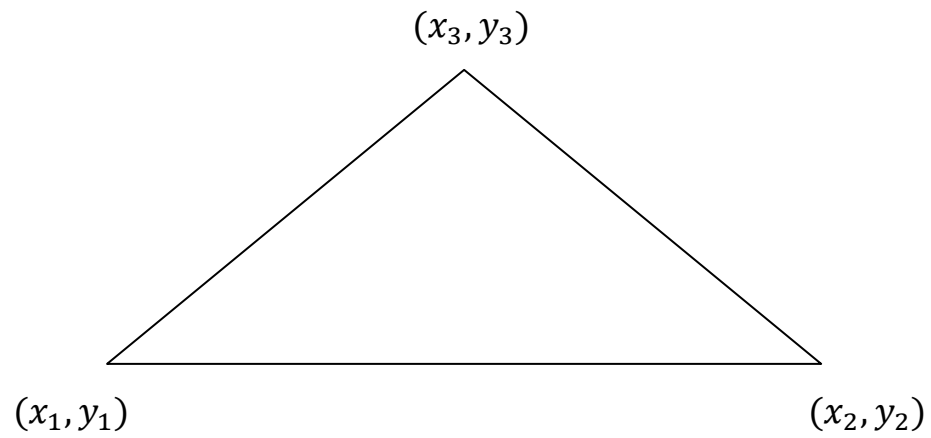

Рисунок 3.1

Критерии для стоп-сигналов (рис 3.1):

1. $|x_1 - x_2| > k|y_1 - y_2|$
2. $k_1|x_1 - x_2| < max(|x_3 - x_1|, |x_3 - x_2|) < k_2|x_1 - x_2|$
3. $k'_1|x_1 - x_2| < max(|y_3 - y_1|, |y_3 - y_2|) < k'_2|x_1 - x_2|$

Где (x_1, y_1) и (x_2, y_2) – координаты геометрических центров масс левой и правой нижних областей, соответствующих нижним стоп-сигналам. А (x_3, y_3) – координата геометрического центра масс области посередине, соответствующей центральному стоп-сигналу.

Все константы необходимо подобрать в процессе экспериментов.

Расположение стоп-сигналов (рис. 3.2):

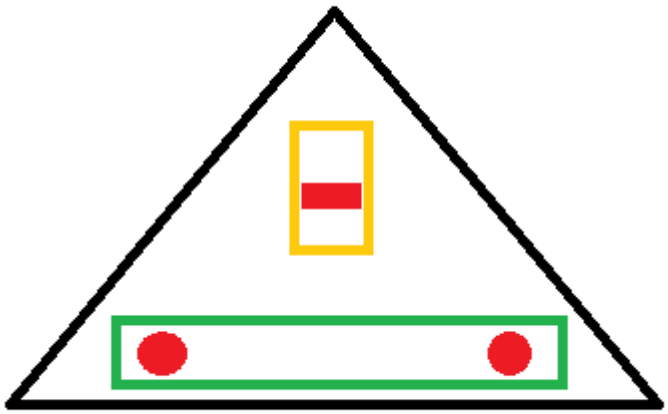

Рисунок 3.2 Схема (шаблон) расположения стоп-сигналов.

3.2.3. Выделение подходящих шаблонов

После того как мы нашли тройки областей соединяем их точки в треугольник и переносим на исходное изображение.

4. Применение сторонних библиотек

Для написания программы обнаружения и распознавания автомобилей в процессе торможения было решено использовать стороннюю библиотеку.

Было решено выбрать одну из следующих библиотек:

- **AForge.NET** — библиотека алгоритмов компьютерного зрения и алгоритмов искусственного интеллекта, разработанная Андреем Кирилловым под .NET Framework. Содержит библиотеки по обработке изображений и видео, работы с нейронными сетями, неклассическими логиками и генетическими алгоритмами. Исходный код доступен в условиях лицензии GPL (GNU General Public License).[4]

- **OpenCV** (*Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии BSD.[5]

- VXL ('*Vision-something-Library*') — это набор бесплатных библиотек с открытым исходным кодом на языке C++ для работ с компьютерным зрением. Идея заключается в том, чтобы заменить X одной из многих букв, то есть G (*Vision Geometry Library*) представляет собой библиотеку для работы с точками, кривыми и т. д. , N (VNL,

Vision Numerics *Library*) представляет собой библиотеку для работы с числами, I (*Vision* Imaging *Library*) является библиотекой обработки изображений и т.д. Эти библиотеки могут быть использованы для общих научных вычислений, а также для работы с компьютерным зрением.[6]

- VIGRA (Vision with Generic Algorithms) — это бесплатная библиотека компьютерного зрения с открытым исходным кодом, которая специализируется на настраиваемых алгоритмах и структурах данных. VIGRA может быть легко адаптирована к конкретным потребностям целевого приложения без ущерба для скорости выполнения, с использованием методов шаблонов, аналогичных STL C++.[7]

Было решено выбрать связку язык программирования C# + библиотека AForge.NET , так как она содержит больше средств, необходимых для решения поставленной задачи, а также поскольку она удобна и легка в применении, к ней написана подробная документация и есть большое количество примеров приложений, которые демонстрируют использование встроенных алгоритмов. Эту библиотеку отличает обширное множество различных встроенных фильтров, необходимых для обработки изображений.

5. Аспекты реализации

Приложение выполнено на языке программирования C# с использованием сторонней библиотеки AForge.NET.

5.1. Обработка изображения

Исходное изображение с помощью композиции фильтров преобразуется в бинарное изображение.

5.1.1. Обрезание исходного изображения

Обрезание верхней трети производится с помощью стандартных средств языка C#.

Пример (рис. 5.1 и рис. 5.2):

Рисунок 5.2. До обрезания

Рисунок 5.3. После обрезания

5.1.2. Применение цветового фильтра

Выделение областей с ярко-красным цветом выполняется с помощью фильтра ColorFiltering из библиотеки AForge.NET.

Перевод в полутоновое изображение с помощью фильтра Grayscale.

А бинаризация изображения с помощью фильтра OtsuThresholding.

Подбор порога осуществлялся на основе тестов для выборки из 25 изображений.

Пример:

Первоначальное изображение (рис. 5.3):

Рисунок 5.4. Перед применением цветового фильтра

После применения фильтра с выбранными диапазонами (рис. 5.4):

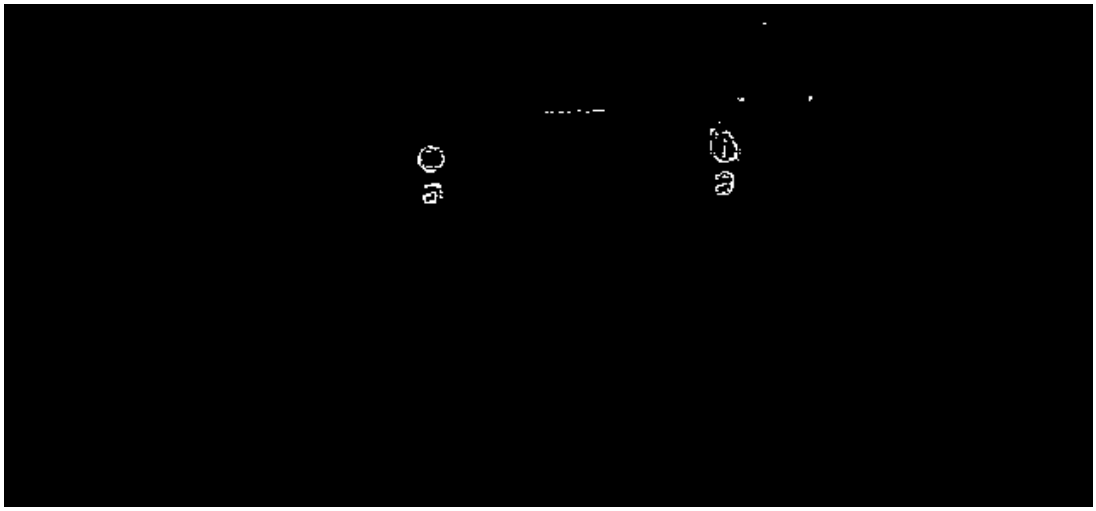

Рисунок 5.5. Приемлемое качество, хотя и есть лишние области

После применения фильтра с широкими диапазонами (рис. 5.5):

Рисунок 4.5. Слишком много лишних областей

После применения фильтра с узкими диапазонами (рис. 5.6):

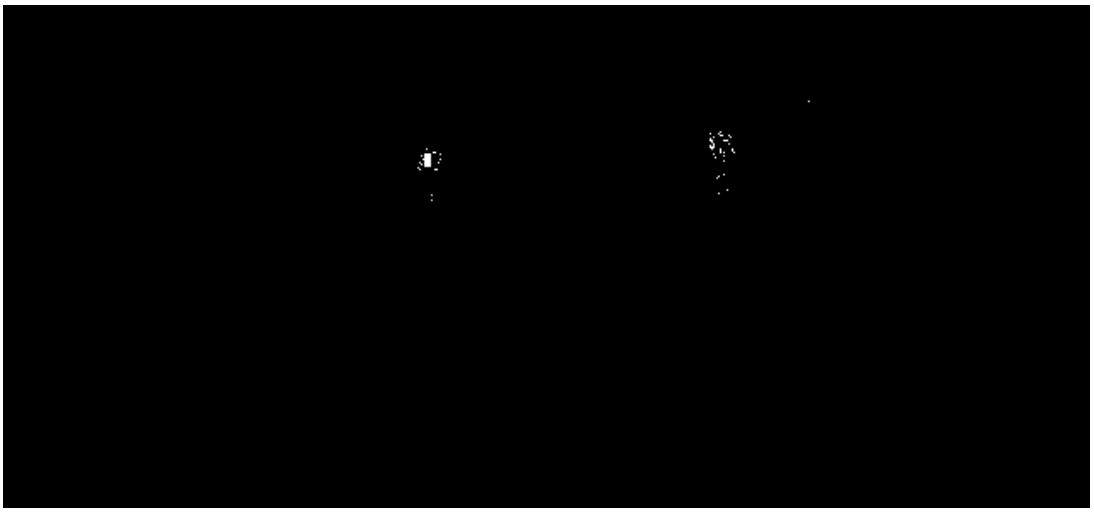

Рисунок 5.6. Исчезла область центрального стоп-сигнала, машина не будет найдена

5.1.3. Выделение областей

Выделение областей происходит с помощью специального класса BlobCounter, который по изображению выделяет области на черном фоне и заносит их в массив Blob’ов.

Класс Blob содержит много полезных свойств:

- Area — площадь области
- CenterOfGravity — центр масс
- Fullness — заполнненость [0,1]
- ID
- Image — изображение области
- Rectangle — прямоугольник внутри которого содержится область
- и другие

С помощью массива Blob’ов на следующем этапе происходит поиск останавливающиеся автомобили.

5.2. Поиск шаблонов

На этом этапе на основе полученного ранее массива Blob’ов выделяются останавливающиеся автомобили (если они есть).

5.2.1. Анализ областей

Малые по площади области отбрасываются по свойству Area класса Blob, но серия тестов показала, что много шумов остаётся, так что можно отбросить только очень малые площади, иначе не смогут определяться машины вдалеке.

Пример (рис 5.7):

Рисунок 5.7 Автомобиль вдалеке, останавливающийся перед кольцом

После применения фильтра (рис 5.8):

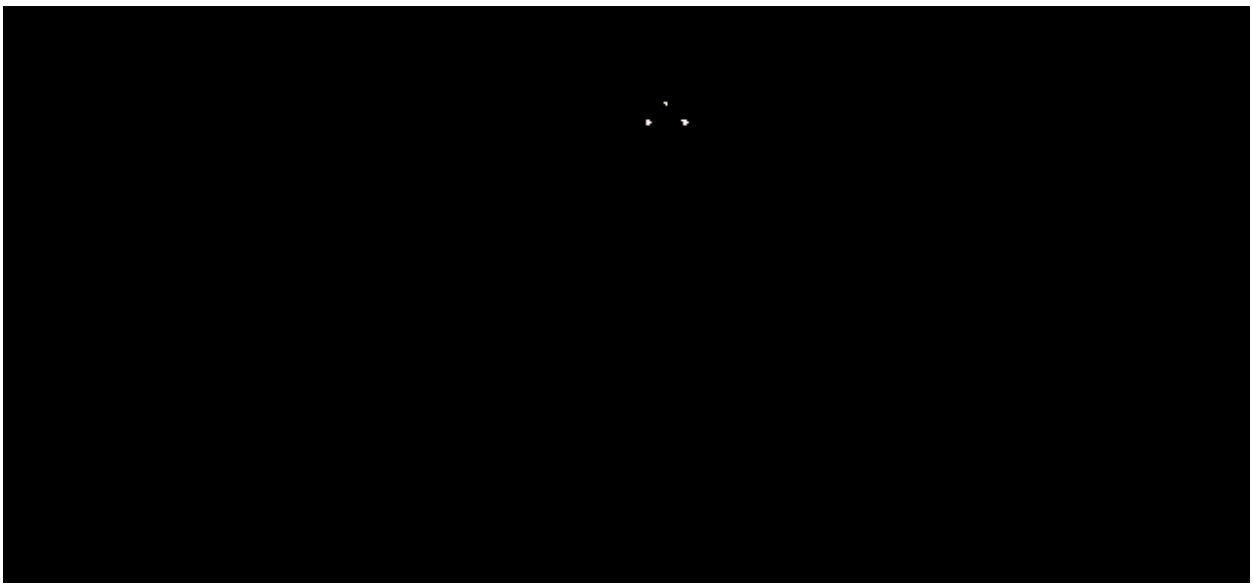

Рисунок 5.8 Автомобиль вдалеке, останавливающийся перед кольцом (после применения фильтра)

Значит, что можно отбрасывать только области с меньшей площадью, чем площадь центрального стоп-сигнала на рис. 5.8 .

5.2.2. Поиск подходящих под описание шаблонов

Тройки областей ищутся по параметру CenterOfGravity класса Blob, которые как бы создают треугольник

5.2.3. Выделение подходящих шаблонов

После того как мы нашли тройки областей соединяем их точки CenterOfGravity в треугольник и переносим на исходное изображение.

6. Заключение

В данной работе была рассмотрена задача обнаружения и распознавания на видеозаписи автомобилей в процессе торможения.

Были изучены различные алгоритмы распознавания образов. Итогом стал проект обнаружения автомобилей в процессе торможения на языке программирования C# с использованием сторонней библиотеки AForge.NET.

В рамках проекта были реализованы следующие алгоритмы:

- Алгоритм выделения областей по цветовой компоненте, опирающийся на пороговый фильтр, фильтр перевода цветного изображения в полутоновое и на метод Оцу.
- Алгоритм поиска шаблонов.

7. Список литературы

1. N. Otsu (1979). «A threshold selection method from gray-level histograms». *IEEE Trans. Sys., Man., Cyber.* **9**: 62-66
2. Форсайт, Понс, Жан. Компьютерное зрение. Современный подход. Пер. с англ. — М.: Издательский дом «Вильямс», 2004. — 928 с.
3. Л. Шапиро, Дж. Стокман. Компьютерное зрение. Пер. с англ. — М.: БИНОМ. Лаборатория знаний. 2006. — 752 с.
4. Описание библиотеки AForge.Net.
   www.aforgenet.com/
5. Описание библиотеки OpenCV.
   www.opencv.org/
6. Описание библиотеки VXL.
   www.vxl.sourceforge.net/
7. Описание библиотеки VIGRA.
   www.ukoethe.github.io/vigra/
8. Дистрибутив библиотеки AForge.Net.
   www.aforgenet.com/framework/downloads.html