Министерство образования Российской Федерации
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра прикладной информатики

УДК 681.03

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, д.т.н., проф.
_____ С.П. Сущенко
«___»_____ 2006 г.

Бусыгин Леонид Александрович

ЛОКАЛИЗАЦИЯ АВТОМОБИЛЬНОГО НОМЕРА В ПОТОКЕ ВИДЕОДАННЫХ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Дипломная работа

Научный руководитель,
д.т.н. А.В. Скворцов
Исполнитель,
студ. гр. 1412 Л.А. Бусыгин

Электронная версия дипломной работы помещена
в электронную библиотеку. Файл
Администратор

Томск – 2006

Реферат

Дипломная работа 33 с., 20 рис., 1 табл., 5 источников.

РАСПОЗНАВАНИЕ ОБРАЗОВ, СЕГМЕНТАЦИЯ ДВИЖУЩИХСЯ ОБЛАСТЕЙ, ОБНАРУЖЕНИЕ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ, RGB, MOTION DETECTION, COMPUTER VISION, DIRECT SHOW.

Объект исследования – системы компьютерного зрения.

Цель работы – разработка алгоритмов и написание программы, осуществляющей обнаружение и локализацию автомобильной номерной пластины в потоке видеоданных.

Рассмотрены основные методы решения и их сравнительные характеристики. Разработан и реализован эффективный алгоритм локализации номера.

Содержание

Введение

В современном мире автоматизации, люди пытаются заменить человеческий труд машинным везде, где это возможно. И наблюдательная способность человека не исключение. Ее призваны заменить видеодетекторы. Конечно, следует заметить, что видеодетекторы не способны заменить человека полностью, но помочь ему, они объективно могут. Поэтому во всем мире большое внимание уделяют системам компьютерного зрения.

Данная дипломная работа посвящена локализации автомобильных номерных пластин в видео потоке. Проблема, несомненно, актуальна сегодня, когда автомобиль стал неотъемлемой частью жизни каждого человека. Контроль над транспортными средствами в нашей стране оставляет желать лучшего. И видеодетекторы могут помочь в этом. С их помощью можно контролировать, например, скоростной режим или выезд за двойную сплошную линию. Эта практика с успехом используется в некоторых странах западной Европы. Но, пожалуй, главной остается проблема идентификации транспортного средства. Эта задача решается сейчас многими разработчиками, и некоторые достигли значительного успеха. Но все это коммерческие проекты, и, как правило, очень дорогие.

Задачу распознавания номерной пластины можно условно разделить на две подзадачи: обнаружение и локализация номера автомобиля и собственно распознавание символов. Эта работа, как уже упоминалось ранее, посвящена первой подзадаче, т.е. обнаружению и локализации номерной пластины. Решение задачи должно соответствовать двум основным требованиям: обнаружение с высокой степенью точности и низкой трудоемкостью алгоритма, позволяющей производить вычисления в реальном режиме времени.

Задача локализации заключается в указании размеров и точных координат пластины на изображении. Локализация происходит в несколько этапов. Первый этап: выделение движущегося объекта. Второй этап: исследование этого объекта на предмет возможных мест нахождения пластины. Третий этап: проверка подозрительных областей на принадлежность к классу номерных пластин и уточнение координат.

1 Существующие решения

В мире около тридцати компаний предлагают аналогичные решения по распознаванию автомобильных номеров. Как правило, все они имеет общую архитектуру, как аппаратную, так и программную. Многие системы уже имеют опыт внедрения. Продукты постоянно совершенствуются, появляются новые алгоритмы, и новые аппаратные решения, но базовая архитектура, как правило, принципиально не изменяется.

1.1 Общая архитектура

Архитектура систем машинного зрения состоит из трех базовых элементов: видеокамера, формирующая видеоинформацию, плата, преобразующая аналоговый сигнал в цифровой (АЦП), и программное обеспечение, выполняющее анализ видеоданных (рисунок 1).

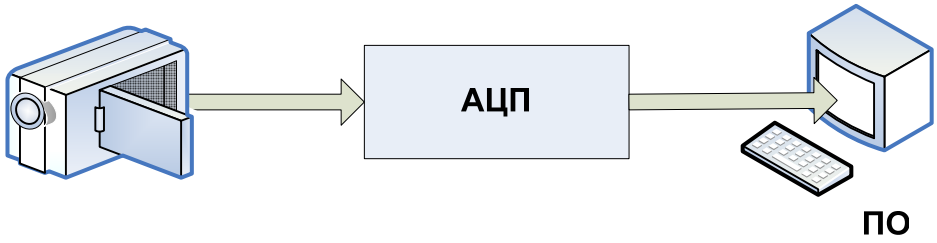


Рисунок 1 – Стандартная система машинного зрения

Последний элемент, в свою очередь состоит из нескольких модулей. Набор модулей зависит от задачи, в случае систем считывания автомобильных номеров он следующий: модуль выделения движения, модуль поиска номерной пластины на объекте и модуль распознавания символов (рисунок 2). Иногда производят предобработку изображения, для улучшения качества работы алгоритмов.

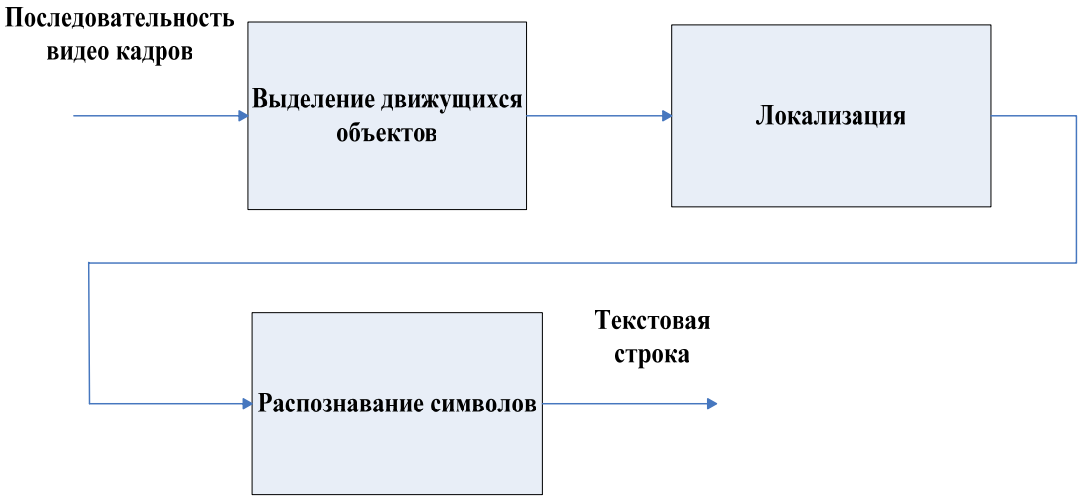


Рисунок 2 – Схема модуля считывания автомобильного номера

Наиболее трудоемкими (с точки зрения затрат времени центрального процессора) являются первые два этапа. Некоторые решения работают без первого модуля (выделение

движущихся объектов), и в некоторых случаях эти решения выигрывают в скорости вычислений. Это зависит от сложности сцены, так если на заднем плане будет только дорожное полотно (или еще что-либо однообразное и монотонное), то имеет смысл отключить детектор движения, или использовать его только частично (есть движение или нет, без выделения контуров). В таком случае, простой задний план будет анализироваться гораздо быстрее, по сравнению со временем выделения контура движущегося объекта. Но в случае, когда задний план очень сложен, выигрыш от использования детектора движения будет значительным, так как программе не придется анализировать покоящиеся объекты на заднем плане.

Скорость обработки данных очень важна в системах реального времени. Это отражается, в первую очередь на экономических показателях системы. Так, при грамотно написанных алгоритмах, потребуется в несколько раз меньше вычислительных ресурсов, и, следовательно, стоить система будет гораздо дешевле. С другой стороны быстродействие системы определяет максимально возможную скорость автомобиля. Его значение зависит частоты кадров камеры и АЦП, и количеством кадров, которые система может проанализировать в секунду. Меньшая из этих величин определяет время обнаружения пластины. Таким образом, максимальная скорость автомобиля будет вычисляться по простой формуле:

$$V = S / T,$$

где S – длина дорожного полотна, попадающего в обзор камеры, T – время обнаружения и распознавания. Среди прочих характеристик систем важными являются также минимальный размер символов, минимально допустимая освещенность, максимально возможный наклон номерной пластины, требования к расположению камеры.

1.2 Программные продукты

На данный момент на рынке России присутствует достаточно много готовых программных и программно-аппаратных продуктов, решающих проблему распознавания автомобильных номеров. Среди российских компаний можно выделить:

«Автоинспектор» (компания ISS),

«Авто-Интеллект» (компания ITV),

«Дигнум-Авто» (компании ЗАО РУСИСТ Безопасность),

«Система считывания регистрационных номеров автомобилей» (ИИТ Системы компьютерного зрения),

«CarFlow» (компании ООО МегаПиксел)

и другие.

К сожалению, многие компании предоставляют очень мало информации о продаваемых ими продуктах. Таким образом, вполне вероятно, что некоторая часть компаний продает не полностью свое решение. Многие, например, покупают готовый SDK(набор программных компонент) и соединяют его со своим аппаратным решением. В таком случае круг «чистых» разработчиков не так велик, как кажется на первый взгляд.

Все перечисленные компании в один голос заявляют о очень высоких показателях своих систем. У большинства точность распознавания (по данным их интернет-сайтов) составляет 95% - 98%. Однако, проверить эту информацию, к сожалению, не представляется возможным. Как правило, демо-версии предлагаемых решений не несут в себе ни какой информации о помехоустойчивости.

Цены на эти продукты варьируются от $1500-3000 на один видео канал. При масштабном внедрении системы получится значительная сумма.

1.3 Резюме

В настоящее время эта тема очень актуальна в России. Большое количество разработчиков трудится над этими задачами. Многие достигли значительных успехов. Однако, масштабного внедрения ни одна система не получила до сих пор. Это свидетельствует либо об излишней дороговизне продуктов, либо об их низкой точности. Таким образом, перспективы у данной работы, несомненно, есть.

2 Методы сегментации движущихся объектов

Задачу выделения движущихся объектов представляет собой сумму двух подзадач – выделение пикселей переднего плана и объединение их в объекты. Для первой подзадачи существует несколько алгоритмов. Здесь проблема выбора подходящего алгоритма решается очень просто, так как эффективность алгоритма пропорциональна его сложности. В данной работе был реализован один из самых эффективных алгоритмов – алгоритм, основанный на вероятностных моделях, для цветного изображения формата RGB.

Вторая же подзадача была решена специфически для автомобильного потока, и более того, исключительно для локализации номера. То есть алгоритмы, примененные для ее решения, не являются общими, и не могут использоваться для решения других задач. Главным принципом было: лучше выделить область больше реального объекта, чем оставить номерную пластину за областью выделения. При этом нужно сохранять и требования к трудоемкости. Таким образом, алгоритм не является точным (с точки зрения выделения границ объекта), что в данной работе не требуется, но является надежным (с точки зрения вероятности попадания номерной пластины в область выделения) и быстродействующим.

2.1 Простейший алгоритм выделения движущихся областей

Метод наложения кадров с успехом используется в большинстве систем обнаружения движения. Суть метода заключается в выделении заднего плана и наложении на него каждого нового кадра. Области, отличающиеся от заднего плана, интерпретируются как передний план. Простейший алгоритм:

1. Сохраняем первое изображение видеопоследовательности и принимаем его за задний план, назовем его B.
2. Каждый следующий кадр C сравниваем с задним планом, и пиксели, не удовлетворяющие неравенству $|B[i][j]-C[i][j]|<\delta$, рассматриваются как передний план.

Выделим основные проблемы, связанные с работой алгоритма: шум камеры, который принимается за передний план, динамический задний план (например, движение листьев на дереве), изменение заднего плана (например, автомобиль, остановившийся на стоянку), изменение освещения и тени от движущихся объектов. Кроме того, алгоритм предусматривает указание пользователем заднего плана (первый кадр), что неприемлемо.

2.2 Алгоритм, основанный на вероятностных моделях

Почти все проблемы решает алгоритм, основанный на вероятностных моделях. Алгоритм создает модель сцены, используя смесь нормальных распределений, и с поступлением каждого нового кадра обновляет модель и классифицирует каждый пиксель как принадлежащий к заднему или к переднему плану. Таким образом

$$S \approx \sum_{p=1}^{k} w_p * N(x, \mu_p, {\delta_p}^2),$$

где

$$N(x, \mu_p, \delta_p{}^2) = \frac{1}{\sqrt{2\pi\delta^2}} * e^{-\frac{(x-\mu)^2}{2\delta^2}}.$$

Каждому слагаемому в сумме соответствует процесс в пикселе сцены, который характеризуется параметрами нормального распределения (мат. ожиданием и дисперсией) и коэффициентом *w*, который называется весом и является показателем того, насколько часто данный процесс в данном пикселе попадал в поле зрения камеры. Параметр *k* (максимальное кол-во процессов) выбирается в соответствии с ресурсами компьютера, обычно берут значения от 3 до 5.

Алгоритм:

1) На первом кадре видеопоследовательности происходит инициализация модели. В каждом пикселе создается один процесс со следующими параметрами: *w = 0*, $\mu = c$, $\delta^2 = \delta^2{}_{std}$, где *c* - текущее значение в данном пикселе, а $\delta^2{}_{std}$ - дисперсия по умолчанию (выбирается вручную). Все пиксели сегментируются в задний план.

2) Для каждого следующего кадра и для каждого пикселя:

1. Поиск процесса, которому удовлетворяет значение данного пикселя *c*. Для каждого процесса в модели применяется порог

$$\frac{|\mu - c|}{\delta} \le \varepsilon.$$

Если текущее значение с для некоторого процесса удовлетворяет порогу, то данный процесс помечается как текущий и переходим к пункту 3. Если порог не выполнился ни для одного процесса, то переходим к пункту 2.

2. Создание нового процесса. Оценки мат. ожидания и дисперсии выбираются следующим образом: $\mu = c$, $\delta^2 = \delta^2{}_{std}$. Если количество процессов в модели уже равно *k*, то ищется процесс с наименьшим весом, его вес не меняется, а остальные параметры приравниваются к параметрам нового процесса, данный процесс помечается как текущий и переходим к пункту 5. Если количество процессов в модели еще не достигло максимума то новый процесс добавляется к списку процессов, его вес приравнивается 0, этот процесс помечается как текущий и переходим к пункту 4.

3. Обновление статистики текущего процесса. Оценки мат. ожидания и дисперсии обновляются с помощью низкочастотного фильтра рекурсивного сглаживания. Обозначим за μ_{t-1} и $\delta^2{}_{t-1}$ оценки параметров текущего процесса на предыдущем шаге, а за μ_t и $\delta^2{}_t$ оценки на текущем шаге, тогда

$$\mu_t = (1 - a_1) * \mu_{t-1} + a_1 * c,\ \delta^2 = (1 - a_2) * \delta^2_{t-1} + a_2 (c - \mu_t)^2,$$

где a_1, a_2 - параметры фильтра позволяющие регулировать скорость обучения.

4. Обновление весов процессов. Обозначим за $w_{i,t-1}$ вес *i*-того процесса на предыдущем шаге, а за $w_{i,t}$ вес *i* -того процесса на текущем шаге, тогда

$$w_{i,t} = (1 - a_3) * w_{i,t-1} + a_3 * M(i),$$

где a_3 - параметр, отвечающий за скорость изменение веса, а *M(I)*– функция равная 1 при *i* равном индексу текущего процесса и равная 0 при всех остальных значениях *i*.

5. Классификация пикселя. Алгоритм классификации очень прост: применение порога к весу текущего процесса. Если значение веса текущего процесса больше чем значение порога, то пиксель классифицируется как пиксель заднего плана, иначе как пиксель переднего плана. Порог следует выбирать из интервала (0,1).

Метод, используемый для классификации пикселей (пункт 2.5) очевидно допускает принадлежность к заднему плану сразу нескольких процессов. Именно по этому при наличии динамического заднего плана алгоритм через некоторое время адаптируется к его повторяющимся движениям. Более того, алгоритм способен адаптироваться к таким движениям сцены как дождь или снег. Также алгоритм способен адаптироваться к изменениям заднего плана т.к. через некоторое время после изменения вес созданного при изменении процесса превысит порог, и он начнет сегментироваться в задний план. К медленным изменениям освещения алгоритм способен адаптироваться благодаря системе оценки параметров, т.к. по мере плавного изменения цвета пикселей заднего плана алгоритм будет переобучаться на новые значения. К быстрым изменениям освещения алгоритм адаптируется так же, как он адаптируется к изменениям заднего плана.

Для подавления теней используется тот факт, что при попадании на некоторую точку тени ее цветовой тон не изменяется, а изменяется лишь яркость. Т.е., если пиксель имел цвет *(R,G,B)*, то при попадании на него тени, цвет изменится так – *a*(R,G,B),* где коэффициент *a* определяет, насколько упала освещенность в точке.

Алгоритм был опубликован научно-популярным online-журналом «Графика и мультимедиа», авторы Виктор Гаганов и Антон Конушин.

В некоторых системах используется чуть более простой и быстрый алгоритм выделения движения, использующий только полутоновое изображение. Но эффективность этого алгоритма резко снижается. Возможности подавления теней резко ограничиваются, и даже вообще пропадают. Качество сильно снижается и при выделении самого объекта. На рисунке 3 видно как слабо отличается от фона полутоновое изображение автомобиля, и как сильно отличается на цветном RGB изображении.

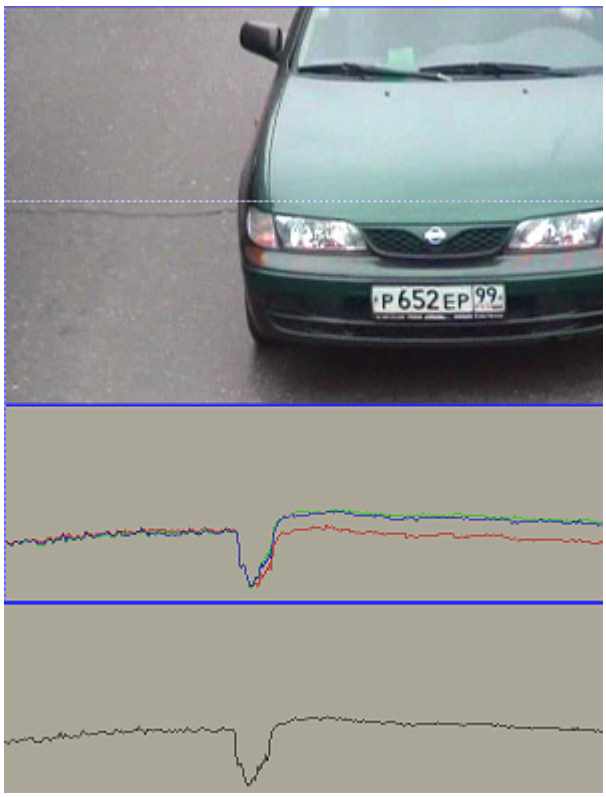


Рисунок 3 – Эффект от использования цветного изображения

Описанный алгоритм позволяет с достаточной точностью выделить пиксели, принадлежащие переднему плану. Огромное количество параметров, которое влияет на работу алгоритма, является одновременно и его универсальностью и сложностью его настройки. Так, например, при указании коэффициента *a* (используемого для подавления теней), возникает вопрос: как найти оптимальную границу между тенью и объектами, имеющими цвет фона, но другой яркости. Все эти проблемы были решены экспериментальным путем. При реализации алгоритма для потока видеоданных с разрешением 720x576, он требовал слишком много ресурсов процессора, поэтому было принято решение проверять не каждый пиксель, а например каждый восьмой. Таким образом, трудоемкость вычислений уменьшалась в восемь раз, а качество практически не терялось.

2.3 Поиск объектов

Перед поиском объектов имеет смысл произвести фильтрацию бинарного изображения, чтобы удалить случайные пиксели, выделенные по ошибке, и усилить области, выделяющие реальный объект. Как правило, для такой задачи применяется медианный фильтр, работающий следующим образом: изображение делится на блоки, в каждом блоке подсчитывается количество пикселей заднего и переднего планов, вычисляется их отношение и если оно превышает порог, то весь блок закрашивается пикселями заднего плана, иначе переднего. Применение этого фильтра в данной задаче показало плохие результаты, так как очень часто автомобиль выделялся не сплошной областью, а тонким контуром. В результате, фильтр удалял контур, и автомобиль не был выделен, либо разделялся на два объекта.

Решением этой проблемы стала разработка специфического алгоритма. Суть заключается примерно в следующем: нужно замкнуть контур автомобиля, закрасить все внутренние области и выпрямить сильно вогнутые части контура.

Алгоритм для замыкания контура. Идея заключается в удалении разрывов на линиях переднего плана. Алгоритм выполняется для каждой горизонтальной и вертикальной линии сцены. Ищутся такие последовательности пикселей сцены, в которых соотношение пикселей переднего и заднего плана больше заданного порога. При этом в них не может присутствовать серия пикселей заднего плана больше определенной длины, которая задает минимально возможное расстояние между машинами. Все эти последовательности заполняются пикселями переднего плана. Таким образом, алгоритм скрепляет разделенные части объекта, что позволяет при последующей обработке выделить объект целиком. Пример работы алгоритма показан на рисунке 4.

Рисунок 4 – Фильтрация сцены

Алгоритм выпрямления сильно вогнутых частей контура и закраски внутренних областей требует четырех проходов по сцене. На первом проходе закрашиваются области вогнутые влево, на втором вправо, на третьем вверх и на четвертом вниз. На каждом проходе происходит поиск соответствующей области, например область вогнутая вверх выглядит так:

111…111

100…001,

где 1 это пиксель объекта, а 0 пиксель сцены. Если такая область найдена, то вся нижняя линия закрашивается пикселями объекта. Пример работы алгоритма показан на рисунке 4.

Рисунок 5 – Результат работы алгоритма закраски

После обработки данных этими фильтрами, можно легко выделить отдельные объекты, которые впоследствии передать на дальнейшую обработку. Пример выделенного объекта показан на рисунке 6.

Рисунок 6 – Выделенный движущийся объект

2.4 Резюме

Рассмотренные в этой главе алгоритмы позволяют решить поставленную задачу выделения движущихся объектов. Алгоритм, основанный на вероятностных моделях, решает, почти все проблемы связанные с шумом камеры и динамическим задним планом, а использование цветного изображение позволяет эффективно отличать тени от объектов, и резко повышает качество работы алгоритмов. Также алгоритм позволяет регулировать трудоемкость вычислений путем изменения интервалов накладываемой сетки. Поиск объектов также эффективно выполняет поставленную задачу, алгоритм работает с линейной трудоемкостью.

3 Математические методы локализации объектов

Ключевым моментом в распознавании образов является отбор признаков характеризующих образ. Проблема отбора и упорядочивания признаков трудно формализуема, и зависит от конкретной задачи. Качество всей системы в первую очередь зависит от подобранных признаков, насколько точно они характеризуют объект и насколько устойчивы они к различным шумам. Задача формирования признаков очень сложна, и является в некотором смысле творческой задачей. Современное состояние этой области знаний не дает четких методов, позволяющих решать такие задачи. Поэтому до настоящего времени подбор признаков остается процедурой эвристической.

3.1 Методы, основанные на анализе контура

Выделим основные признаки автомобильной номерной пластины: первое, что бросается в глаза – прямоугольная форма, белый фон и черные символы (для обычных номеров). Для поиска прямоугольных форм на изображении можно применить алгоритм выделения контуров. Задача выделения контуров состоит в построении бинарного изображения, содержащего очертания объектов. Наиболее часто используемый подход к решению задачи обнаружения перепадов на одноцветном изображении схематически показан на рис. 7.

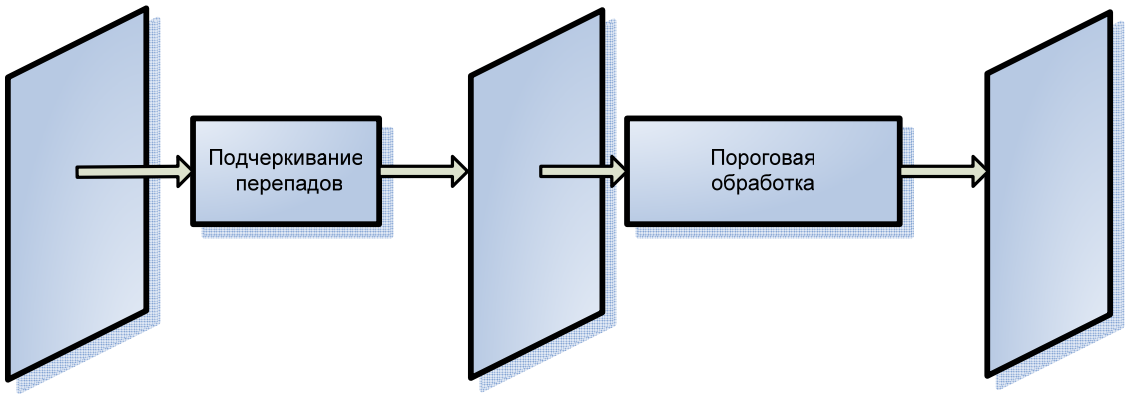


Рисунок 7 – Общий вид процедуры выделения контуров

Одним из наиболее очевидных и простых способов обнаружения границ является дифференцирование яркости, рассматриваемой как функция пространственных координат. Эффект дифференцирования видно на простом «одномерном» примере. До дифференцирования сигнал имеет вид представленный на рисунке 8,*а*. После дифференцирования – вид на рис. 8,*б*, и теперь контур легко выделяется пороговой обработкой.

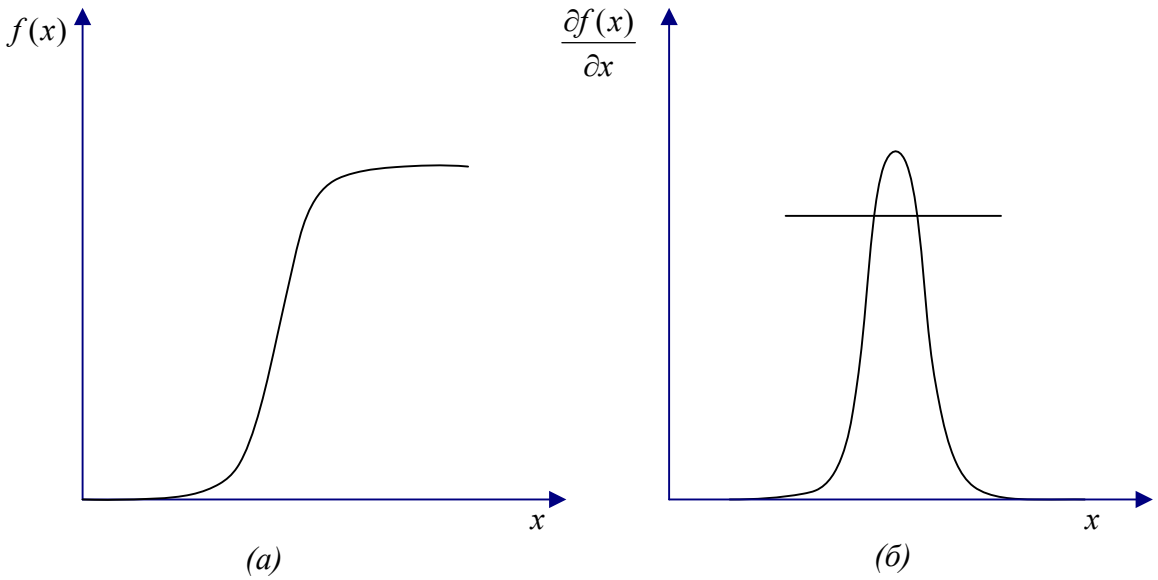


Рисунок 8 – Дифференциальный метод выделения контура

Очевидно в двумерном случае, если мы имеем изображение со значениями яркости $f(x_1, x_2)$, то обнаружение контуров, перпендикулярных оси x_1, обеспечивает взятие частной производной $\partial f / \partial x_1$, а перпендикулярных оси x_2 – частной производной $\partial f / \partial x_2$. Эти производные характеризуют скорости изменения яркости в направлениях x_1 и x_2 соответственно (рисунок 9).

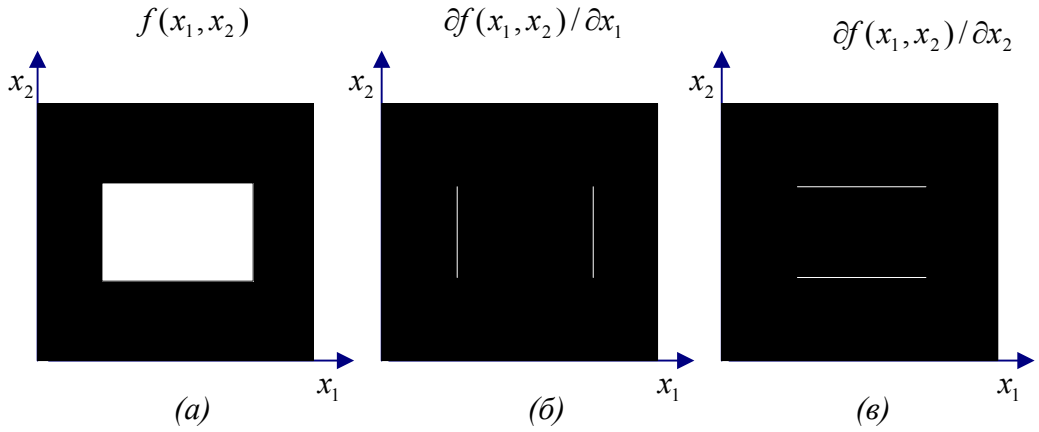


Рисунок 9 – Дифференциальный метод выделения контура на изображении

Нам, однако, необходимо найти характеристику, позволяющую обнаружить контур независимо от его ориентации. В качестве такой характеристики, являющейся признаком наличия контура в локальной области, можно использовать градиент яркости:

$$\text{grad } f(x_1, x_2) = \nabla f(x_1, x_2).$$

Градиент - это вектор (в нашем случае в двумерном пространстве), ориентированный по направлению наиболее быстрого возрастания функции $f(x_1, x_2)$ и имеющий длину, пропорциональную этой максимальной скорости (максимальному значению частной производной по направлению). Так как направление нас не интересует, ограничимся рассмотрением модуля градиента (длины вектора):

$$|\nabla f(x_1, x_2)| = \sqrt{\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2}\ .$$

В случае цифровых изображений, представленных матрицей отсчетов, вместо производных берутся дискретные разности:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \approx s_1(n_1, n_2) = f(n_1, n_2) - f(n_1 - 1, n_2),$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} \approx s_2(n_1, n_2) = f(n_1, n_2) - f(n_1, n_2 - 1).$$

При реализации процедуры детектирования контуров стараются избегать трудоемких операций типа умножения и извлечения квадратного корня. Поэтому используют выражения, вычисляемые проще, «аппроксимирующие» дискретный градиент. Чаще всего модуль градиента заменяют выражением:

$$\hat{g}(n_1, n_2) = |s_1(n_1, n_2)| + |s_2(n_1, n_2)|.$$

Для вычисления величин s_1 и s_2 можно также применить оператор Собела, используя линейную обработку масками 3x3:

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \text{ и } \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}.$$

Существуют и другие приближения градиента. Следует отметить, что применение любых градиентных операторов дает обычно сходные результаты. Различия наблюдаются только в их устойчивости к шуму. Пример работы алгоритма на реальном изображении показан на рисунке 10.

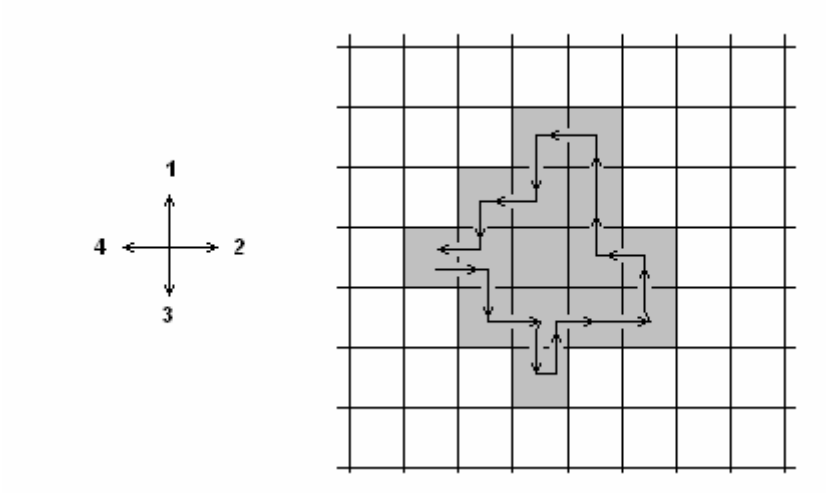
Рисунок 10 – Результат работы алгоритма выделения контуров

После применения алгоритма выделения контуров есть несколько вариантов анализа. Например, можно выделить прямые линии и найти среди них те, которые вместе составляют прямоугольник похожий на номер. Так как реальное изображение, как привило, зашумленное, то многие линии разрываются, имеются небольшие бугры, скачки и т.д. Поэтому полученные линии пока не пригодны для анализа. Устранить мелкие разрывы можно путем объединения линий, которые являются продолжением друг друга. После этого можно анализировать выделенные прямые линии. Поиск прямоугольника заключается в выделении пар параллельных прямых с пропорциями соответствующими автомобильной номерной пластине.

Второй вариант представление контуров в виде цепных кодов. Метод цепного кодирования был предложен Фриманом. Он заключается в том, чтобы границу объекта расположенного на дискретной сетке, представить в виде набора элементарных отрезков. Тогда полной характеристикой границы объекта в каждой точке является направление требуемого отрезка (рисунок 11). В данном случае предполагается, что точки на границе являются только 4-х связными (отрезок накладывается лишь в четырех направлениях). Иногда применяют модификацию данного метода, использующую 8-связную модель.

Несомненным достоинством представления границы изображаемого объекта цепным кодом является простота реализации алгоритма его описания, простота получения на основе этого описания некоторых других геометрических характеристик объекта (например: периметр, площадь, линейные размеры по вертикали и горизонтали), возможность

достижения инвариантности к преобразованиям подобия – масштабированию изображения, его переносу и повороту. Основным недостатком является высокая неустойчивость получаемых описаний к искажениям в изображениях.



Цепной код: 2323122141143434

Рисунок 11 – Пример построения цепного кода

Методы, основанные на анализе контуров, позволяют находить номер различного размера (проблема масштабируемости) и под различным наклоном. Однако, у них есть очень важный минус – трудоемкость вычислений, даже на небольшой картинке время обнаружения может достигать нескольких секунд, что неприемлемо. Проход по изображению скользящим окном (например, оператор Собела) сам по себе уже трудоемкая операция, а последующий анализ контуров, которых может быть очень много, добавляет к этому значительную нагрузку.

3.2 Метод корреляции

Рассмотрим, как можно сравнить две последовательности, состоящие из значений, одновременно выбираемых из двух соответствующих сигналов. Если два сигнала похоже меняются при переходе от точки к точке, то меру их корреляции можно вычислить, взяв сумму произведений соответствующих пар точек. Данное предложение становится более аргументированным, если рассмотреть две независимые и случайные последовательности данных. В этом случае сумма произведений стремиться к исчезающе малому случайному числу по мере увеличения пар точек. В то же время, если сумма конечна, это указывает на наличие корреляции. Отрицательная сумма указывает на отрицательную корреляцию, т.е. увеличение одной переменной связанно с уменьшением другой. Таким образом, взаимную корреляцию $r_{12}(n)$ двух последовательностей данных $x_1(n)$ и $x_2(n)$, содержащих по N элементов, можно записать как

$$r_{12}(n) = \frac{1}{N}\sum_{n=0}^{N-1} x_1(n)x_2(n).$$

В некоторых случаях корреляция, определенная указанным выше способом, может быть нулевой, хотя две последовательности коррелируют на 100%. Это может произойти,

например, когда два сигнала идут на в фазе (как часто и бывает). Данная ситуация иллюстрируется сигналами на рисунке 12.

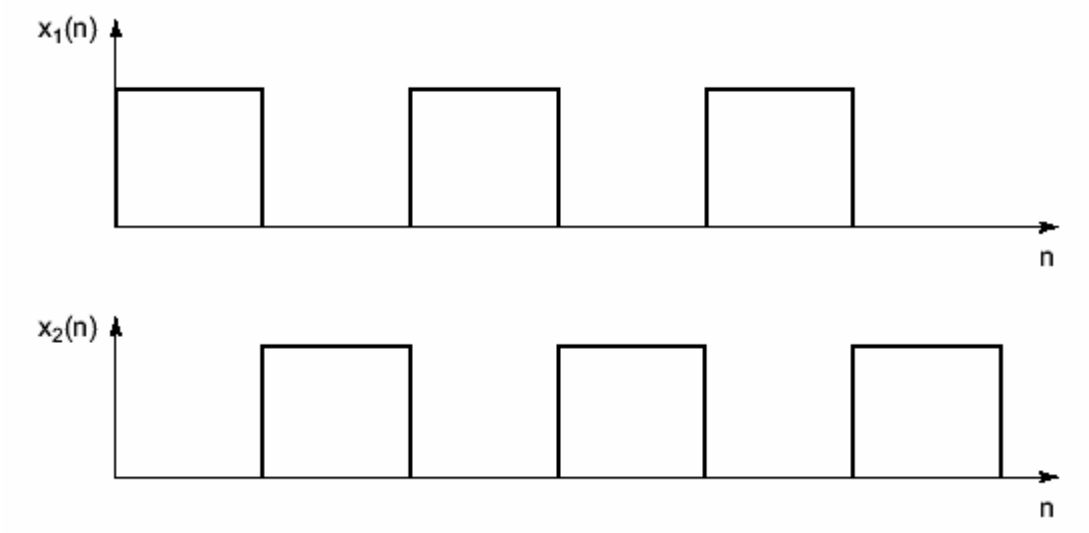


Рисунок 12 – Сигналы со 100% корреляцией, идущие не в фазе

Таким образом, требуется следующая формула взаимной корреляции:

$$r_{12}(j) = \frac{1}{N}\sum_{n=0}^{N-1} x_1(n)x_2(n+j)\,.$$

На практике, когда два сигнала коррелируют, их фазовая связь, скорее всего, не известна (а в нашем случае ее как раз и требуется найти), так что корреляцию нужно находить для всех возможных задержек, чтобы установить наибольшее значение корреляции, которое затем считается истинным. Для вычисления корреляции существует эффективный алгоритм, использующий преобразование Фурье – алгоритм быстрой свертки. Так как свертка и корреляция вычисляются очень похоже, то алгоритм быстрой свертки можно применить и для вычисления корреляции. Сравнительная трудоемкость вычислений показана в следующей таблице.

Таблица 1 – Число действительных умножений, требуемых для выполнения свертки двух последовательностей

Кол-во точек	**Прямой метод**	**Быстрая свертка**	**Соотношение**
8	64	448	7
16	256	1 088	4,25
32	1 024	2 560	2,5
64	4 096	5 888	1,4375
128	16 384	13 312	0,8125
256	65 536	29 696	0,4531
512	262 144	65 536	0,250
1024	1 048 576	143 360	0,1367
2048	4 194 304	311 296	0,0742

Двумерная корреляция вычисляется аналогично, только для функций двух переменных. Алгоритм быстрой свертки позволяет анализировать достаточно большие изображения, за приемлемое время, но здесь возникают другие проблемы. Задача локализации автомобильных номеров осложнена тем, что мы не имеем точного эталона. Ведь на номерной пластине могут быть любые цифры. А искать все возможные комбинации не представляется возможным. В принципе можно искать не всю номерную пластину, а только одну цифру или букву. В таком случае придется несколько раз осуществлять поиск, пока не будет найден некоторый символ, и после этого анализировать область вокруг него. Кроме того, встает проблема масштабируемости, искажений изображения (из-за наклона камеры) и наклона самого номера. К таким изменениям алгоритм очень чувствителен. В таком случае придется либо осуществлять поиск по нескольким эталонам одного и того же символа, либо преобразовывать изображение к эталонному виду. В сумме получается очень трудоемкая операция.

3.3 Специальный метод

У номерных пластин есть и другой отличительный признак – сильные скачки яркости от черного к белому и от белого к черному. Сделаем горизонтальное сечение изображения и построим график зависимости амплитуды яркости от координаты X. Область номерной пластины на графике отразится характерными скачками (рисунок 13, линия сечения показана пунктиром).

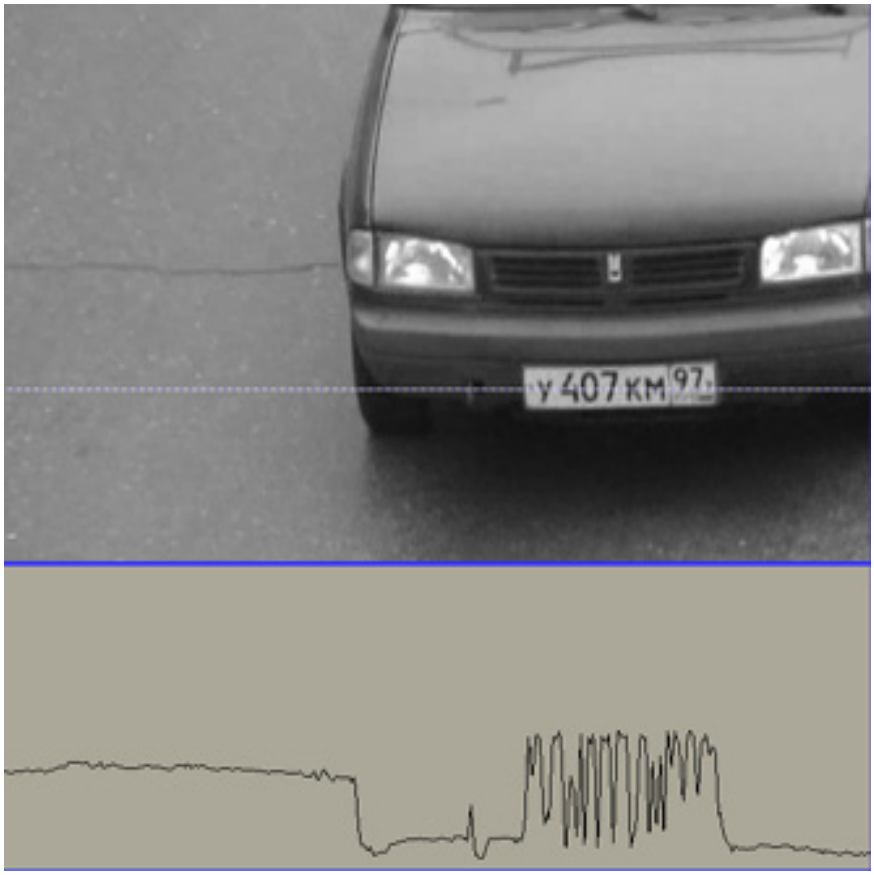


Рисунок 13 – График амплитуды яркости

Это свойство и послужило основой для следующего алгоритма. Сканируем изображение построчно и строим функцию следующего вида:

$$f(x) = \sum_{i=0}^{x-2} | I(i+1) - I(i) |,$$

где *I(i)* значение яркости в соответствующем пикселе. Функция *f(x)* в области номерной пластины начнет быстро возрастать (рисунок 14). После этого сглаживаем функцию *f(x)* и находим ее производную (рисунок 15). Места с высокими значениями производной и есть подозрительные области. Вычислив вторую производную можно определить горизонтальные края пластины (рисунок 16).

Этот метод позволяет с высокой скоростью определить предполагаемые области расположения номерной пластины. Кроме того, необязательно сканировать каждую строчку исходного изображения. Можно задать минимально возможную ширину пластины и проверять сечения сделанные с интервалом в половину минимальной ширины. Это ускорит, и без того быстрый, процесс поиска в несколько раз. Метод абсолютно устойчив к различным шумам, каплям грязи и многих других дефектов, сильно мешавших другим методам. Минусом данного алгоритма является его слабая устойчивость к наклонам (номер перекошен). В данной задаче этим можно пренебречь, так как сильно перекошенные номера (угол более 20 градусов) встречаются крайне редко.

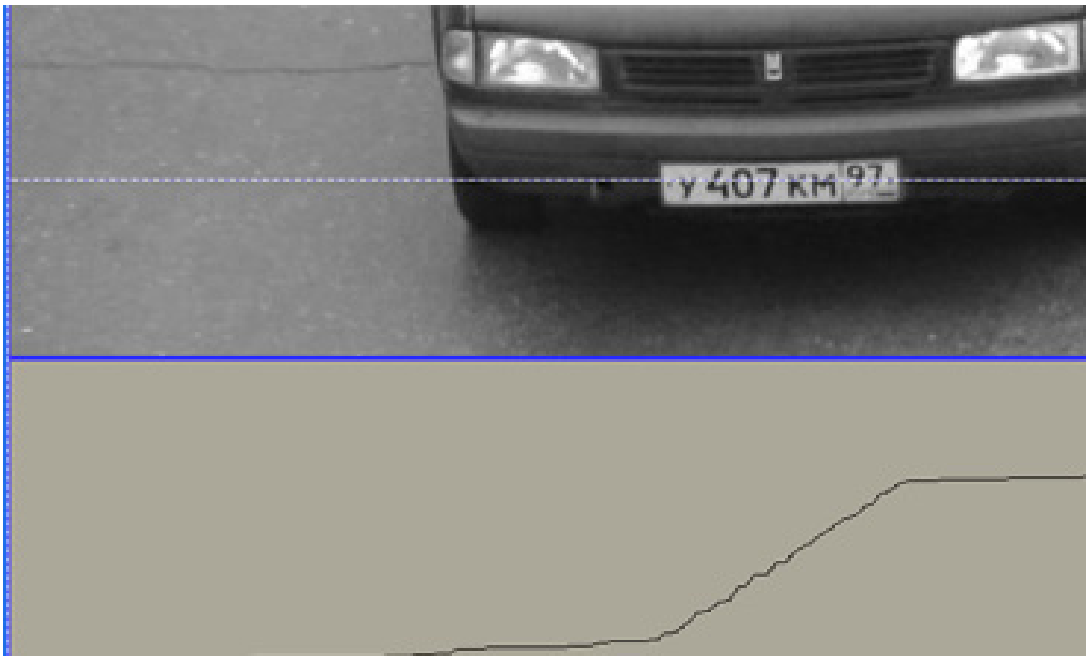


Рисунок 14 – Вид функции *f(x)*

Рисунок 15 – Производная функции *f(x)*

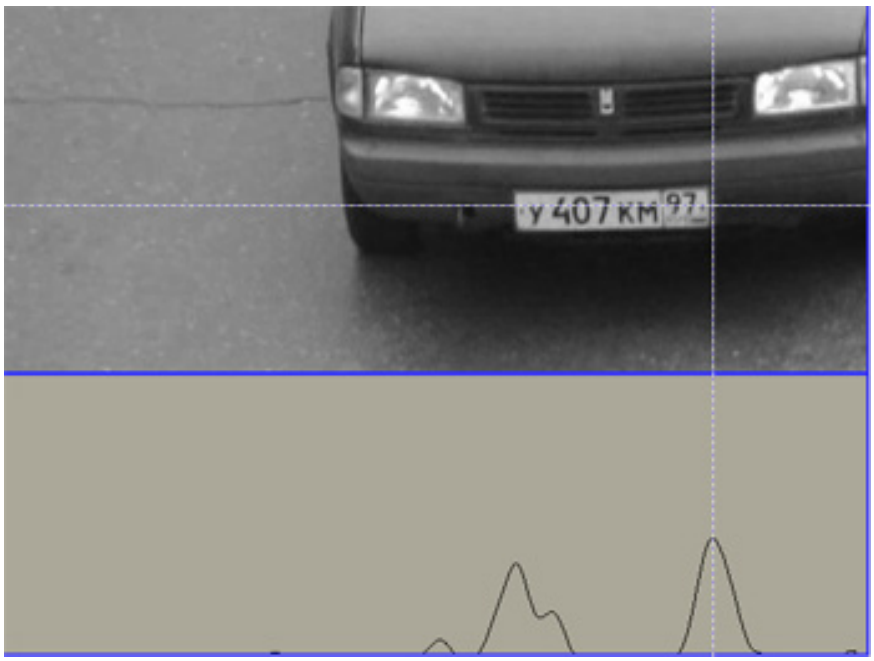


Рисунок 16 – Вторая производная функции *f(x)*

3.4 Резюме

Рассмотренные алгоритмы используются в с успехом используются для решения различных задач распознавания образов. Для решения же поставленной задачи ни один из известных, опубликованных методов не подходил по каким-либо требованиям. Поэтому был разработан специальный метод, опирающийся на специфику поставленной задачи. Этот метод решал все проблемы локализации номера. Он идеально быстр, быстрее в принципе не возможно разработать алгоритм, так как этот метод анализирует только малую часть изображения, не говоря уже о нескольких проходах по всему изображению, которые используются в других алгоритмах. Любой другой алгоритм, анализирующий меньшее количество информации, будет иметь вероятность ошибки (пропустить номерной знак), так как номер сможет поместиться в не анализируемой области. Кроме того метод уникально помехоустойчив. Он находит почти любой номер, который сможет прочитать человек, (исключения составляют лишь номерные знаки с очень малой контрастностью < 5%). Таким образом, метод на голову выше всех остальных конкурентов.

4 Программная реализация

В программе были реализованы, методы, выделенные раннее в соответствующих главах, т.е. алгоритм выделения движения, основанный на вероятностных моделях и специальный метод локализации автомобильного номера. Для того чтобы алгоритмы заработали на практике, были добавлены небольшие изменения, но суть алгоритмов не изменилась. Программа в целом имеет красивую архитектуру. При реализации был использован язык C++ и технология DirectShow.

4.1 Общая архитектура

Программа разделена на несколько частей. Каждая часть выполняет свою функцию и передает обработанную информацию далее. Проект состоит из трех больших модулей:

- Модуль выделения движущихся объектов.
- Модуль локализации номерных пластин.
- Модуль проверки объектов (номеров).

В свою очередь модуль выделения движущихся объектов также состоит из двух подмодулей:

- Модуль выделения движущихся областей.
- Модуль поиска объектов.

Архитектура программы показана на рисунке 17.

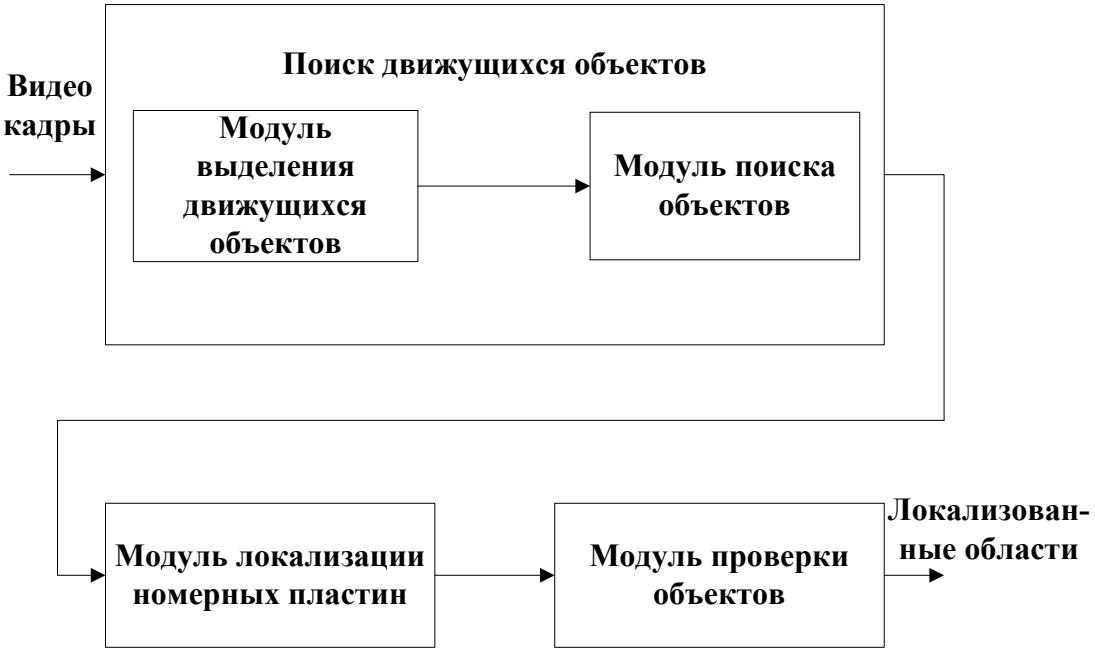


Рисунок 17 – Общая архитектура программы

Программа представляет собой фильтр DirectShow, который на вход принимает поток видео кадров в формате RGB24, а на выходе выдает также поток видео кадров, с очерченными контурами автомобилей и их номеров. Формат RGB был выбран ввиду его распространенности и простоты в использовании. Фильтр можно встраивать в различные графы, как с «живым» источником видео (с видео камеры), так и с закодированным (из

файла). Пример графа анализирующего видео информацию из файла показан на рисунке 18.

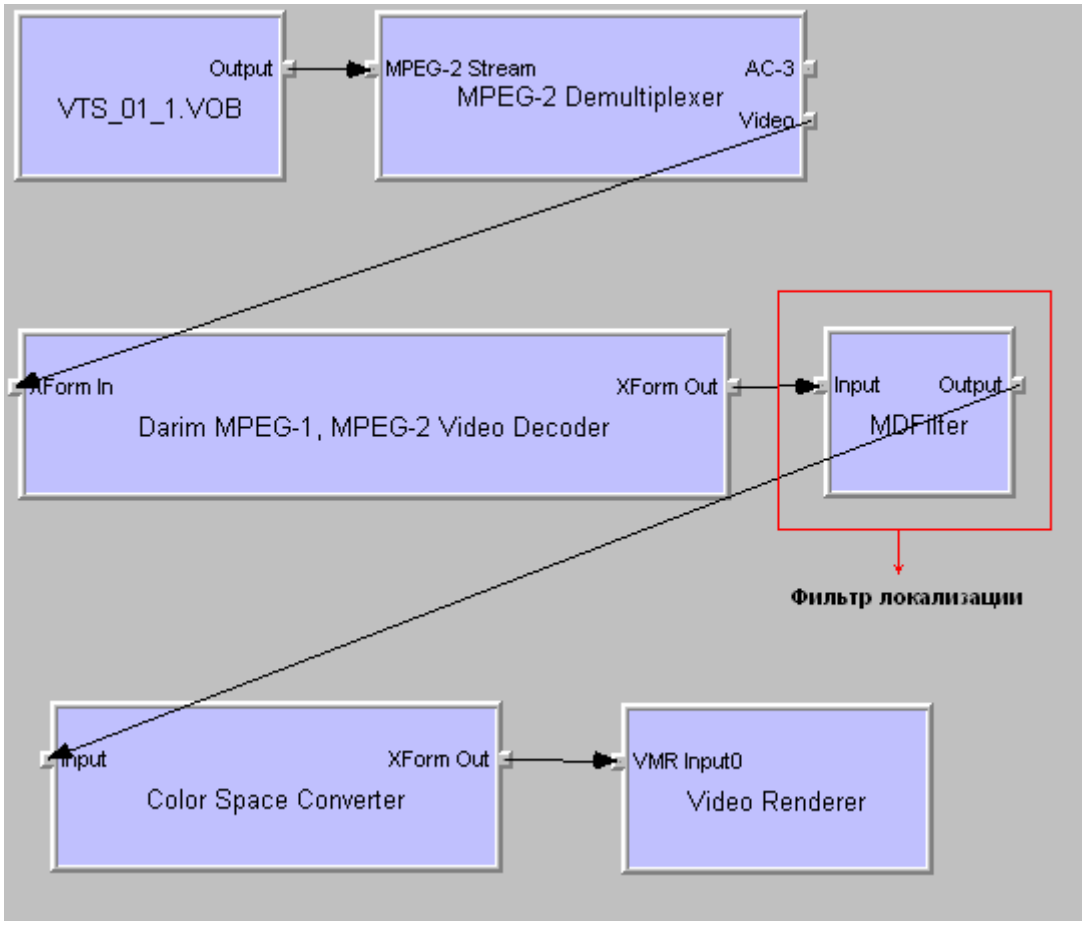


Рисунок 18 – Пример графа фильтров DirectShow со встроенным фильтром локализации номеров

4.2 Технология DirectShow

DirectShow — это API, позволяющий Windows-приложениям управлять широким спектром устройств ввода аудио и видеоинформации, в частности цифровыми видеокамерами, веб-камерами, DVD-приводами и платами ТВ-тюнеров. Он обеспечивает программную поддержку множества форматов — от WAV и AVI до Windows Media. Кроме того, DirectShow является расширяемой технологией и позволяет сторонним разработчикам поддерживать собственные специализированные устройства, форматы и компоненты обработки.

Технология DirectShow основана на технологии COM. Сутью данной технологии является то, что программы строятся из компонент, то есть из исполняемых файлов, которые никак не надо "связывать" со своим проектом - их достаточно зарегистрировать в операционной системе, и они будут доступны любой программе исполняющейся на данной машине. Т.е. их использование в своей программе производится без операций сборки модуля.

Кирпичики технологии DirectShow это COM объекты называемые **фильтр,** каждый из которых выполняет некоторую операцию над мультимедиа потоком. Например, такие операции как:

- чтение из файла;

- получение данных из устройств ввода аудио и видеоинформации;
- декодирование потока данных, некоторого формата (например, MPEG-4);
- передача данных на графическую или аудио карту.

Фильтры получают на вход данные и передают обработанные данные на выход. Например, фильтр, который декодирует MPEG-4, получает на вход видео поток в формате MPEG-4, а на выходе отдает не компрессированное видео в формате RGB, пригодное для передачи в видео карту для отображения на экране монитора.

Для выполнения определенной задачи, приложение соединяет несколько фильтров в цепочку таким образом, что выход одного фильтра является входом другого. Цепочка соединенных фильтров называется граф (filter graph).

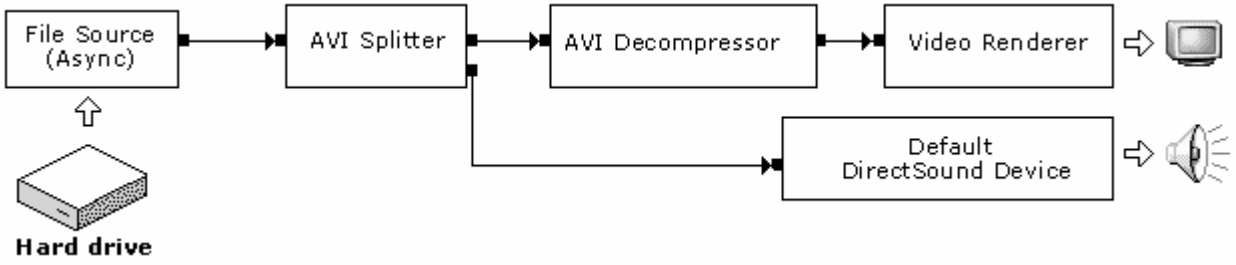


Рисунок 19 – Пример графа фильтров DirectShow

Объект, который управляет графом, называется Filter Graph Manager. Типичное DirectShow приложение должно обеспечить три основных шага, которые показаны на рисунке 6.

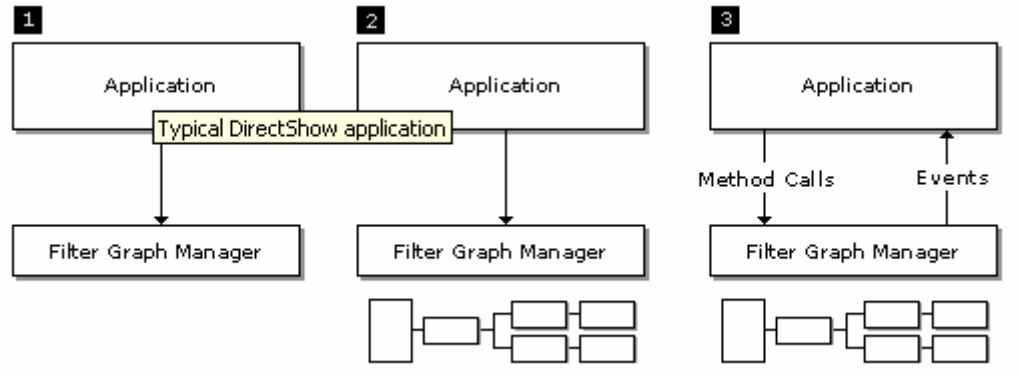


Рисунок 20 – Необходимые шаги для построения DirectShow приложения

- Создание объекта Filter Graph Manager.
- Используя Filter Graph Manager, строится граф фильтров.
- Контроль графа и реагирование на события.

Резюме

Реализация программы была выполнена довольно качественно, и в принципе пригодна для использования. Однако программа имеет большей ресурс оптимизации кода, некоторых участков алгоритмов. Например, в некоторых местах, можно заменить операцию вычисления квадратного корня, операцией возведения в квадрат. Таким образом, скорость вычислений можно значительно увеличить. Но общая концепция программы, и ее основных алгоритмов, несомненно лучшие для решения поставленной задачи.

Заключение

По результатам данной дипломной работы можно сделать следующие основные выводы:

1. Исследовано состояние проблемы, проведен анализ существующих систем локализации номеров.
2. Изучены основные математические методы обнаружения и локализации объектов и их сравнительные характеристики.
3. Разработан быстрый и эффективный алгоритм локализации номера, основанный на специфике поставленной задачи.
4. Реализован модуль, с использованием технологии DirectShow, осуществляющий обнаружение и локализацию номерных пластин.

В настоящее время производится объединение с модулем распознавания символов. Полноценное приложение планируется использовать в системах видео-наблюдения компании Darim Vision Ltd.

Список использованных источников

1. Гашников М.В., Глумов Н.И., Ильясова Н.И. Методы компьютерной обработки изображений. – М.: ФИЗМАТЛИТ, 2003.–784 с.
2. Гаганов В., Конушин А. Сегментация движущихся объектов в видео потоке // Научно-популярный online-журнал Графика и Мультимедиа [Электронный ресурс]. – 2004. – Режим доступа к журн.: http://cgm.graphicon.ru, свободный.
3. Хемминг Р.В. Цифровые фильтры. – М.: Советское радио,1980. – 224 с.
4. Хорн Б.К.П. Зрение Роботов. – М.: МИР, 1989. – 487 с.
5. ГОСТ 50577-93. Знаки государственные регистрационные транспортных средств типы и основные размеры. Технические требования. Введен 01.01.94. – М.: Издательство стандартов. – 5 с.

Приложение А. Руководство программиста

Описание классов представлены в виде заголовочных файлов (*.h) написанных на языке C++. Для использования класса необходимо включить в проект егог заголовочный класс.

Структура, содержащая информацию о процессе (элементе заднего фона) для одной компоненты цвета.

```
struct SProcess
{
    double m;     // математическое ожидание процесса
    double d;     // дисперсия процесса
};
```

Структура, содержащая информацию о процессе (элементе заднего фона) для трех цветов.

```
struct SProcess3D
{
    SProcess prc0;      // компонента R
    SProcess prc1;      // компонента G
    SProcess prc2;      // компонента B
    double w;           // вес процесса
    BYTE state;         // состояние процесса (0 - не инициализирован, 1 - текущий, 2 –
                        // другой)
};
```

Горизонтальная линия.

```
struct SHorline
{
    SHorline();                 // конструктор
    SHorline(int _x1,int _x2);  // конструктор
    int x1;                     // начало линии
    int x2;                     // конец линии
};
```

Линия.

```
struct SLine
{
    bool operator != (SLine l); // оператор неравенства
```

```
        SLine(int _x1,int _y1,int _x2,int _y2);       // конструктор
        double GetLength();          // функция возвращает длину линии
        int x1; // координата x начала линии
        int y1; // координата y начала линии
        int x2; // координата x конца линии
        int y2; // координата y конца линии
};
```

Структура, содержащая информацию об экстремумах первой производной функции *f(x).*

```
struct SPeak
{
        SPeak(int _x,int _y,int _v); // конструктор
        SPeak();       // конструктор
        int x;  // координата x
        int y;  // координата y
        int v;  // значение производной
};
```

Класс **CSample** используется для работы с видео кадром в формате **RGB24**

```
class CSample
{
        // методы класса
        void PaintLine(SLine l,COLORREF c);   // рисует линию в кадре
        void PaintRect(RECT rc,COLORREF c);          // рисует прямоугольник в кадре
        void exch(int *a, int *b);     // меняет местами два значения
        BYTE GPixel(int x, int y, int nbyte);        // возвращает значение одной из ком
                                                     // понентов цвета в указанном пикселе
        BYTE GPixelLum(int x, int y);// возвращает значение яркости в указанном пикселе
        BYTE SPixel(int x, int y,BYTE r,BYTE g,BYTE b); // устанавливает цвет пикселя
        BYTE SPixel(int x, int y,COLORREF c); // устанавливает цвет пикселя

        // переменные класса
        BYTE *m_pSample; // указатель на видео кадр
        BYTE *m_pTemp; // временный указатель на видео кадр
        int m_iHeight; // высота кадра
        int m_iWidth;// ширина кадра
```

```
};
```

Класс **CMotionDetector** используется для обнаружения движущихся объектов.

```
class CMotionDetector
{
    // методы класса
    CMotionDetector(int iWidth,int iHeight);          // конструктор класса
    virtual ~CMotionDetector();          // деструктор класса
    int Detect(IMediaSample *pMediaSample);       // функция поиска объектов
    int SceneFilter();      // закрашивает области внутри контура
    int SceneMedianFilter(int BlockSize);      // медианный фильтр
    int AnalysisScene(BYTE *p);        // выполняет поиск объектов на сцене
    BYTE GScene(int i, int j);   // возвращает значение пикселя сцены (задний или
                                 //передний план)
    BYTE SScene(int i, int j,BYTE a);// устанавливает значение пикселя сцены (задний
                                     // или передний план)
    int SetCurState(int i, int j, int k);    // устанавливает состояние процесса
    int GetCurProcess(BYTE *p,int i, int j);  // возвращает текущий процесс
    SProcess3D *GetProcess(int i, int j, int num);     // возвращает процесс

    // переменные класса
    SProcess3D *m_pFreq; // массив процессов
    BYTE *m_pScene; // массив пикселей сцены
    SHorline *m_pObjects; // структура содержащая выделенные объекты
    int m_iObjectsSize; // количество выделенных объектов
    CLocalizator *m_pLocal; // указатель на класс CLocalizator
    CSample *m_pCurSample; // указатель на переменную класса CSample
    int m_iFramesScaned; // количество проанализированных кадров
    int m_iMacroBlockSize;      // указывает на размер дискретизации при выделении
                                // движения
    int m_ik;              // количество процессов
    double m_id;           // дисперсия по умолчанию
    double m_am;           // параметр скорости обучения для МО
    double m_ad;           // параметр скорости обучения для дисперсии
    double m_aw;           // параметр скорости обучения для веса
};
```

Класс **CLocalizator** используется для локализации номеров.

```
class CLocalizator
{
public:
    int FindHorizontalEdges(int *Deriv2,int len, int *x0,int *x1);   // поиск
                                                    // горизонтальных границ номера
    int FindEdges(int x,int y,RECT *rc); // поиск границ номера
    void SavePeaks(int *func, int len,int x,int y); // осуществляет поиск пиков
    void DeletePeaks(int *func, int len); // удаляет тонкие пики
    void GetDerivative1(int *func,int *deriv, int len); // считает первую производную
    void GetSmooth(int *func,int *Smooth, int len); // сглаживает функцию
    void GetDerivative2(int *func,int *deriv, int len); // считает вторую производную
    void GetFunction(int *Func,int line, int x0, int x1); //  возвращает функцию f(x)
    int Find(CSample *pSample,SHorline *pObjects,int ObjectsSize);
                                        // осуществляет поиск номеров
    void QuickSort(int l,int r); // осуществляет быструю сортировку пиков
    CSample *m_pCurSample; // указатель на переменную класса CSample
};
```

Приложение Б. Акт внедрения