Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация

КАФЕДРА Компьютерные системы автоматизации производства

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

"Мониторинг человекопотока на основе анализа видео"

Студент _____ _____ Айбушев Т.К.
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____ Урусов А.В.
(Подпись, дата) (И.О.Фамилия)

Москва, 2015

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой РК9-101
(Индекс)
_____ Гаврюшин С.С.
(И.О.Фамилия)
« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Проектирование РК и СУ РК

Мониторинг человекопотока на основе анализа видео
(Тема курсового проекта)

Студент Айбушев Т.К. РК9-101
(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

1. Техническое задание

Разработать систему мониторинга человекопотока на основе анализа видео

2. Оформление курсового проекта

2.1. Расчетно-пояснительная записка на 25 листах формата А4.

2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)

Презентация 30 слайдов

Дата выдачи задания « 7 » февраля 2015 г.

Руководитель курсового проекта _____ Урусов А.В.
(Подпись, дата) (И.О.Фамилия)

Студент _____ Айбушев Т.К.
(Подпись, дата) (И.О.Фамилия)

Примечание:
1. Задание оформляется в двух экземплярах; один выдаётся студенту, второй хранится на кафедре.

Содержание

Введение

Развитие информационных технологий ведет к формированию новых возможностей в применении вычислительных мощностей компьютера для решения разного рода задач. Повсеместно распространяются новые технологии, такие как: умный дом, интернет вещей, техническое зрение и мное другое. И все эти новые технологии становятся неотъемлемой частью нашей жизни и иногда кажется неразумный, когда простейшие операции выполняются людьми, а могли бы выполнятся компьютерами или роботами. В связи с чем, у пытливых умов, возникают разного рода идеи по рационализации деятельности человека.

Одним из бурно развивающихся направлений является техническое зрение. Сейчас можно сказать, что большая часть фундаментальных основ для данной технологии уже заложено и появляющиеся новые языки программирования и специализированные библиотеки делаю это занятием весьма доступными для человека нового в данной области.

Изучая в университете курс имитационного моделирования я проникся идеей имитации поведения реального объекта в виртуальном мире. Многообразие существующих методов моделирования позволяет создавать модели практически любого объекта. Буть то молекулярная модель невидимого, без дополнительного оборудования, для для человеческого глаза объекта или вполне заметная модель крупного завода. Так мне досталась одна такая задача из объектов окружаемой нас действительности, а именно одна из станций московского метро. Во время создания модели станции передомной стояла задача моделирования пассажиропотоков на станции, как несложно догадаться, задачу подсчета количества пассажиров выполнялась в виде наблюдения и фиксирования их количества. Занятие весьма и весьма монотонное и малоинтересное.

В связи с чем возникла идея попробовать автоматизировать эту задачу. Немного поразмыслив пришла идея анализа видео с камер видеонаблюдения, которые установлены практически в любом месте скопления людей.

1 Предпроектное исследование

1.1 Основы компьютерного зрения

Как научная дисциплина, компьютерное зрение относится к теории и технологии создания искусственных систем, которые получают информацию из изображений. Видеоданные могут быть представлены множеством форм, таких как видеопоследовательность, изображения с различных камер или трехмерными данными, например с устройства Kinect или медицинского сканера. Как технологическая дисциплина, компьютерное зрение стремится применить теории и модели компьютерного зрения к созданию систем компьютерного зрения. Примерами применения таких систем могут быть:

- Системы управления процессами (промышленные роботы, автономные транспортные средства)
- Системы видеонаблюдения
- Системы организации информации (например, для индексации баз данных изображений)
- Системы моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование)
- Системы взаимодействия (например, устройства ввода для системы человеко-машинного взаимодействия)
- Системы дополненной реальности
- Вычислительная фотография, например для мобильных устройств с камерами

Компьютерное зрение также может быть описано как дополнение (но не обязательно противоположность) биологическому зрению. В биологии изучается зрительное восприятие человека и различных животных, в результате чего создаются модели работы таких систем в терминах физиологических процессов.

Компьютерное зрение, с другой стороны, изучает и описывает системы компьютерного зрения, которые выполнены аппаратно или программно. Междисциплинарный обмен между биологическим и компьютерным зрением оказался весьма продуктивным для обеих научных областей. Подразделы компьютерного зрения включают воспроизведение действий, обнаружение событий, слежение, распознавание образов, восстановление изображений и некоторые другие.[1]

Реализация систем компьютерного зрения сильно зависит от области их применения, аппаратной платформы и требований по производительности. Некоторые системы являются автономными и решают специфические проблемы детектирования и измерения, тогда как другие системы составляют под-системы более крупных систем, которые уже могут содержать подсистемы контроля механических манипуляторов (роботы), информационные базы данных (поиск похожих изображений), интерфейсы человек-машина и (компьютерные игры) т.д. Однако, существуют функции, типичные для многих систем компьютерного зрения.[2]

Получение изображений: цифровые изображения получаются от одного или нескольких датчиков изображения, и которые помимо различных типов светочувствительных камер включают датчики расстояния, радары, ультразвуковые камеры и т. д. В зависимости от типа датчика, получающиеся данные могут быть обычным 2D изображением, 3D изображением или последовательностью изображений. Значения пикселей обычно соответствуют интенсивности света в одной или нескольких спектральных полосах (цветные или изображения в оттенках серого), но могут быть связаны с различными физическими измерениями, такими как глубина, поглощение или отражение звуковых или электромагнитных волн, или ядерным магнитным резонансом.[3]

Предварительная обработка: перед тем, как методы компьютерного зрения могут быть применены к видеоданным с тем, чтобы извлечь определённую долю информации, необходимо обработать видеоданные, с тем чтобы они удовлетворяли некоторым условиям, в зависимости от используемого метода. Примерами являются:

- Повторная выборка с тем, чтобы убедиться, что координатная система изображения верна
- Удаление шума с тем, чтобы удалить искажения, вносимые датчиком
- Улучшение контрастности, для того, чтобы нужная информация могла быть обнаружена
- Масштабирование для лучшего различения структур на изображении

Выделение деталей: детали изображения различного уровня сложности выделяются из видеоданных. Типичными примерами таких деталей являются:

- Линии, границы и кромки
- Локализованные точки интереса, такие как углы, капли или точки: более сложные детали могут относиться к структуре, форме или движению.

Детектирование/Сегментация: на определённом этапе обработки принимается решение о том, какие точки или участки изображения являются важными для дальнейшей обработки. Примерами являются:

- Выделение определённого набора интересующих точек
- Сегментация одного или нескольких участков изображения, которые содержат характерный объект

Высокоуровневая обработка: на этом шаге входные данные обычно представляют небольшой набор данных, например набор точек или участок изображения, в котором предположительно находится определённый объект. Примерами являются:

- Проверка того, что данные удовлетворяют условиям, зависящим от метода и применения
- Оценка характерных параметров, таких как положение или размер объекта

- Классификация обнаруженного объекта по различным категориям

1.2 Основные положения библиотеки компьютерного зрения OpenCV

OpenCV (англ. 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.

Применение:

- Для утверждения общего стандартного интерфейса компьютерного зрения для приложений в этой области. Для способствования росту числа таких приложений и создания новых моделей использования PC
- Сделать платформы Intel привлекательными для разработчиков таких приложений за счёт дополнительного ускорения OpenCV с помощью Intel® Performance Libraries (Сейчас включают IPP (низкоуровневые библиотеки для обработки сигналов, изображений, а также медиа-кодеки) и MKL (специальная версия LAPACK и FFTPack)). OpenCV способна автоматически обнаруживать присутствие IPP и MKL и использовать их для ускорения обработки.

В версии 2.2 библиотека была реорганизована. Вместо универсальных модулей cxcore, cvaux, highGUI и других было создано несколько компактных модулей с более узкой специализацией.

1.3 Поколения систем сбора статистики

Еще до того как я определился с методом решения задачи подсчета пассажиров я попытался проанализировать существующие методики. И так поискав материалы из разного рода источников я сформировал вот такую таблицу.

	Используемые технологии	Точность	Дополнительно
I	Прерывания светового луча	60-70%	Ограниченное мест применения
II	Обработка тепловых источников	90-95%	Высокая стоимость внедрения
III	Обработка видео	90-95%	Минимальные затраты
IV	Обработка видео + WiFi индификация	90-95%	Сложность технического исполнения

На первых парах решения данной задачи активно использовались существующие методики по прерыванию светового луча. Так например в московском метро, но для цели предотвращения входа без билеты. По такому же принципу подсчитывали количество пассажиров. Другим направлением является обра-

Рис. 1 – Прерывания светового луча

ботка тепловых источников, а именно датчики срабатывающее на приближение объекта с определенной температурой, а в дальнейшем и камеры способные анализировать тепловые источники. Этот вариант самый подходящий для нас, но чрезмерно дорогостоящий.

3-й подход это анализ видео с простые камер видеонаблюдения. Среди всех перечисленных эта технология имеет наименьшую стоимость и может применяться везде, где установлены системы видео наблюдения.

А вершиной развития этой идеи является сочетание анализа виде + индификация объектов по используемым mac адресам устройств

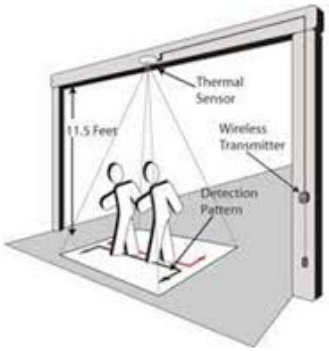

Рис. 2 – Обработка тепловых источников

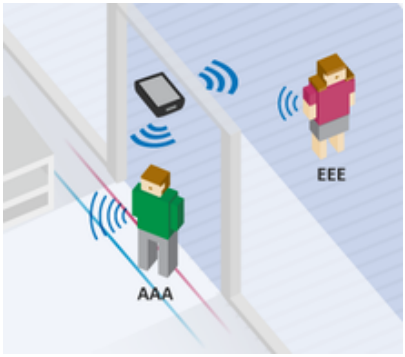

Рис. 3 – Обработка видео + WiFi индификация

1.4 Основные подходы в анализе видео

До начала работы над проектом мной были изучены несколько популярных методов обработки видео реализованных в OpenCV.

- Непрерывная адаптация среднего сдвига (Camshift)
- Оптический поток (Optical Flow)
- Вычитание фона (Background Subtraction)
- Машинное обучение (Machine Learning)
- Обнаружение объекта (Object Detection)

После анализа всем методов было принято решение использовать алгоритм вычитания фона.

2 Техническое задание

2.1 Введение

Во время выполнения курсового проекта по теме "Моделирование транспортного узла станции метро Третьяковская"я столкнулся с задачей подсчета количества пассажиров выходящих из вагона метро. Подсчет количества пассажиров занимает много времени и тяжелого малоинтеллектуального труда. В связи с этим возникла идея автоматизировать подсчеты.

2.2 Общие сведения

Основание для разработки: задание на курсовой проект.

Заказчик: Кафедра «Компьютерные системы автоматизации производства» МГТУ им. Н.Э.Баумана

Разработчик: студент кафедры «Компьютерные системы автоматизации производства» Айбушев Т.К.

Наименование темы разработки: «Мониторинг человеко-потока на основе анализа видео»

2.3 Назначение разработки

Автоматизация мониторинга человеко-потока

2.4 Требования к системе

2.4.1 Требования к реализации

- Система должна быть реализованна в виде программы.
- Параметры должны быть вынесены в отдельный файл конфигураций реализованный на JSON (англ. JavaScript Object Notation)
- Система должна быть реализованна на языке программирования Python со следованием основным постулатам объектно-ориентированного программирования

- Для обработки потокового видео должна быть использована OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом)

2.4.2 Требования к формату ведения разработки

Вся разработка должна вестись с использованием распределённой системы управления версиями файлов Git. Все релевантные материалы должны быть размещены на самом крупном веб-сервис для хостинга IT-проектов и их совместной разработки GitHub.

2.4.3 Требования к точности результатов подсчетов

Точность считается приемлимой если процент погрешности не превышает 15-20% (На отладочном видео). Для проведения верификации должен быть использован видеофрагмент длительностью около 2 минут и с минимум 30 пешеходами, а так же другие видео файлы.

2.4.4 Требования к составу и параметрам технических средств

Модель должна работать на компьютерах со следующими характеристиками:

- Объем ОЗУ не менее 2048 Мб;
- 500MB свободного дискового пространства;
- Современный процессор для хорошей производительности.;

Для безболезненного использования рекомендуется установить(операционная система Ubuntu):

```
$ sudo apt-get install python-opencv
$ sudo apt-get install libgeos-dev
$ sudo pip install shapely
$ sudo pip install json-config-parser
```

2.4.5 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются

2.4.6 Требования к составу и параметрам технических средств

Система должна работать под управлением следующих ОС: Ubuntu

2.4.7 Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

2.4.8 Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

2.5 Стадии и этапы разработки

Плановый срок начала разработки – 10 января 2015г. Плановый срок окончания разработки – 25 апреля 2015г. Этапы разработки:

- Концептуальный этап проектирования системы;
- Технический этап проектирования системы;
- Рабочий этап проектирования системы.

2.6 Порядок контроля и приемки

Прием и контроль работоспособности осуществляется научным руководителем, после полного согласия и выполнение всех поставленных задач.

3 Концептуальный этап проектирования системы

3.1 Описание системы мониторинга

Система мониторинга человеко-потока состоит из трех основных компонентов: тренинга объектов, обработки изображений и алгоритма подсчета. Для обработки изображения подходит алгоритм вычитания фона, который вполне адекватно позволяет решить задачу. Он основан на сравнивании двух соседних изображений друг с другом и нахождения разницы между ними.

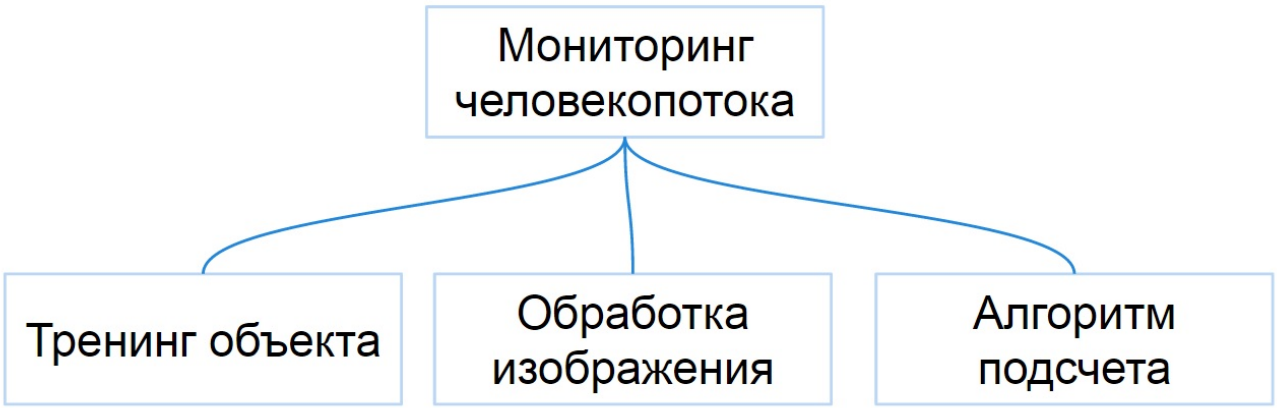

Рис. 4 – Дерево компонентов системы

Трекинг объекта задача, которая имеет множество вариантов решений (см. дерево направлений трекинга)

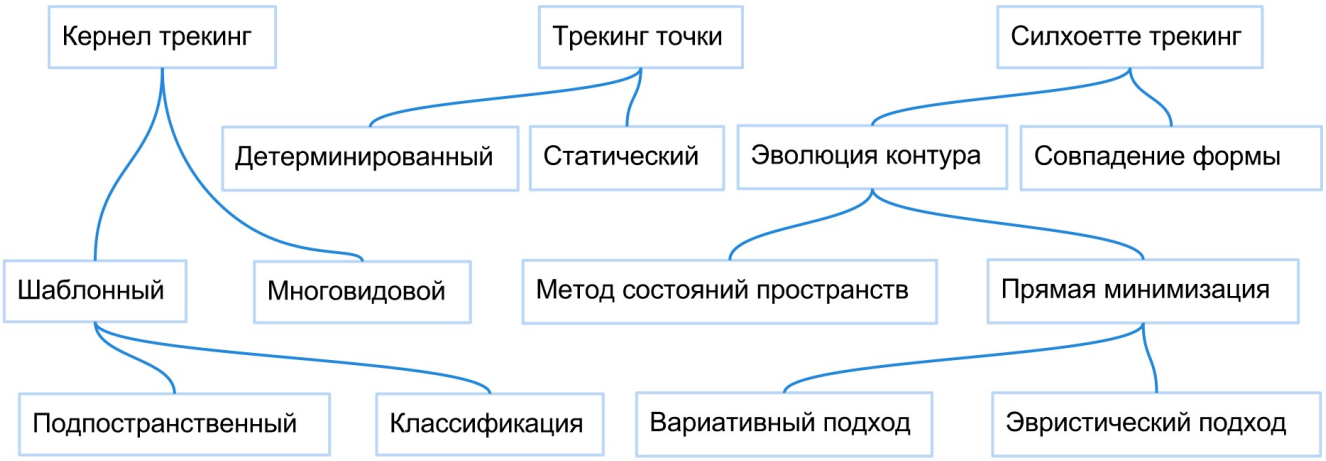

Рис. 5 – Дерево направлений трекинга

Но учитывая используемый алгоритм (вычитание фона), можно получить форму движущегося объекта. Для подсчета нам не важно какой частью тела человек пересекает линию, используемую для подсчета количества людей, поэтому вполне достаточно упростить объект до точки. А из трекинга точки вполне достаточно детерминированного трекинга.

Для подсчета необходимо разработать алгоритм позволяющий подсчитывать количество треков пересекших в одном или другом направлении линию.

4 Техническое проектирование

4.1 Алгоритм формирования трека

Задача формирования трека необходима для корректной работы счетчика.

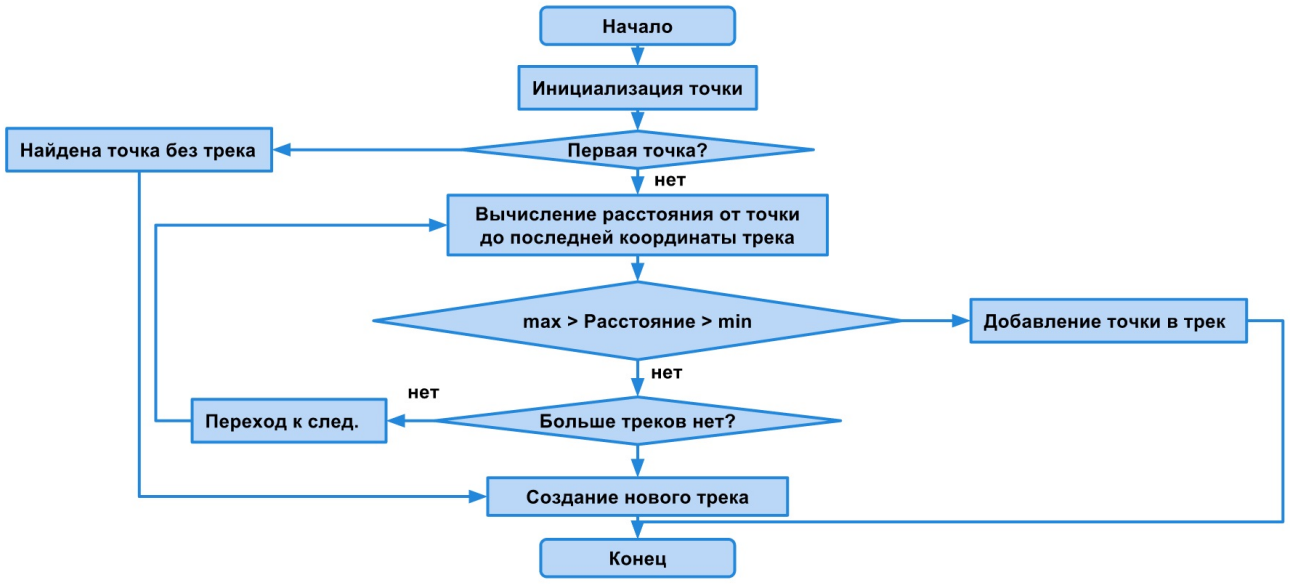

Рис. 6 – Блок-схема алгоритма формирования трека

4.2 Предотвращение многократного срабатывания

Для предотвращения некорректного срабатывания счетчика для описания линии используется эпсилон окрестность. В случае если объект будет двигаться вдоль линии это поможет предотвратить многократное срабатывание. Для избавление от многократного срабатывания на одном и том же треке введены условия, что один и тот же трек может однократно идти вниз и вверх. Так как

на видео мы видим ограниченную область и люды выходят за пределы этой области, то у трека вводится понятие жизненного цикла.

4.3 Алгоритм работы счетчика

Формирование трека по большей части необходимо для корректной работы счетчика. Имея трек точность работы счетчика существенно возрастает за счет избавления от ложных срабатываний.

Рис. 7 – Блок-схема алгоритма работы счетчика

4.4 Описание алгоритма вычитания фона

Алгоритмы вычитания фона широко применяются в задачах видеонаблюдения и активно изучаются в последнее время. Большинство существующих алгоритмов вычитания фона основываются на сравнении цвета пикселей или блоков очередных кадров со своими цветовыми моделями. При этом значительные перепады в освещении обычно приводят к большому числу ошибок, а иногда и к полной деградации работы системы. В методе, предлагаемом в данной работе, весь кадр видео разбивается на блоки, для каждого из которых на основе

бустинга строится отдельный классификатор, определяющий перекрыт данный блок или нет. Благодаря использованию признаков, основанных на сравнении цветов разных пикселей в блоке, достигается большая устойчивость к изменениям освещенности и дрожанию камеры.

- Попиксельные
- Поблочные
- Алгоритмы, основанные на Марковских или Условных случайных полях

Требуя значительного времени при обучении под новую сцену, на этапе выполнения алгоритм демонстрирует высокую скорость, сравнимую со скоростью простейших алгоритмов вычитания фона. Было проведено сравнение точности сегментации видео с помощью данного алгоритма и нескольких существующих аналогов на выложенных в открытый доступ видеороликах. Сравнение показало высокую точность работы предложенного алгоритма.

Большинство методов вычитания фона можно разбить на 3 категории:

Методы первой категории обрабатывают все пиксели очередного кадра независимо. Цвет каждого пикселя на текущем кадре сравнивается с его цветовой моделью. Типичными цветовыми моделями являются нормальное распределение [1], смесь нормальных распределений [1] и непараметрические модели [2]. Метод [1] предполагает, что значение цвета пикселя фона – случайная величина, чье распределение можно аппроксимировать смесью нормальных распределений. Среди недостатков данного метода можно выделить необходимость использования большого числа гауссиан в смеси для моделирования фона, что приводит к существенным вычислительным затратам. Кроме того, данный метод чувствителен к небольшим дрожаниям камеры. Методы второй категории независимо обрабатывают не пиксели, а целые блоки. То есть весь кадр разбивается на блоки, и для каждого из них принимается независимое решение. Несмотря на то, что потенциально такие методы не могут быть идеально точ-

ными, зачастую они дают более приемлемый результат, т.к. используют для принятия решения информацию с целой области. В эту категорию входят, например, метод на основе построения LBP- гистограмм [10] и метод на основе метрики SSD. Последний метод хранит модель фона в виде одного изображения. Результат на каждом кадре производится с помощью сравнения значения метрики близости с порогом. Метрика считается поблочно и равна сумме квадратов разностей между цветами соответствующих пикселей фона и текущего кадра. Предложенный метод также попадает в эту категорию. В последнее время были предложены методы вычитания фона, проводящие сегментацию пикселей изображения, используя модели Марковского и Условного случайного поля для учета пространственных зависимостей между пикселями [4-5]. С помощью данных методов можно дополнительно требовать, чтобы границы областей объектов переднего плана проходили преимущественно по краям на входном изображении. Но, к сожалению, методы данной категории являются более медленными, и почти не используются на практике в системах видеонаблюдения. В работе [3] авторы предлагают алгоритм вычитания фона для использования в видеоконференциях. Поэтому, они используют несколько иную постановку задачи: в качестве переднего плана выступает лишь человек, ближайший к видеокамере. Люди на заднем плане, считаются частью фона. Для решения данной задачи предлагается использовать информацию о наиболее вероятном движении объектов переднего плана. С помощью данного метода можно уменьшить ошибки сегментации из-за движений в фоновом изображении, но с другой стороны необходимо заранее знать наиболее вероятные модели движения для объектов переднего плана, что возможно не в каждом случае. Все алгоритмы вычитания фона условно можно разделить на быстрые, но слабоустойчивые к изменениям освещения и дрожаниям камеры, и качественные, но требующие существенных временных затрат на обработку каждого кадра. Мы предлагаем метод, который может одновременно быстро и качественно решать задачу, основываясь на классификаторах, натренированных для каждого блока. информативные признаки для каждого блока кадра удается выявлять с помощью алгоритма машинного

обучения. Для тренировки алгоритма машинного обучения необходима выборка положительных и отрицательных примеров. В общем случае для ее создания необходимо много данных, размеченных вручную. Эта процедура требует больших временных затрат и не гарантирует получение репрезентативной выборки. Поэтому мы генерируем выборку без объектов переднего плана, имитируя возможные изменения в освещении и настройках видеокамеры, а также выборку, имитирующую возможные перекрытия фона. Для такой генерации достаточно всего одного кадра сцены без объектов переднего плана.

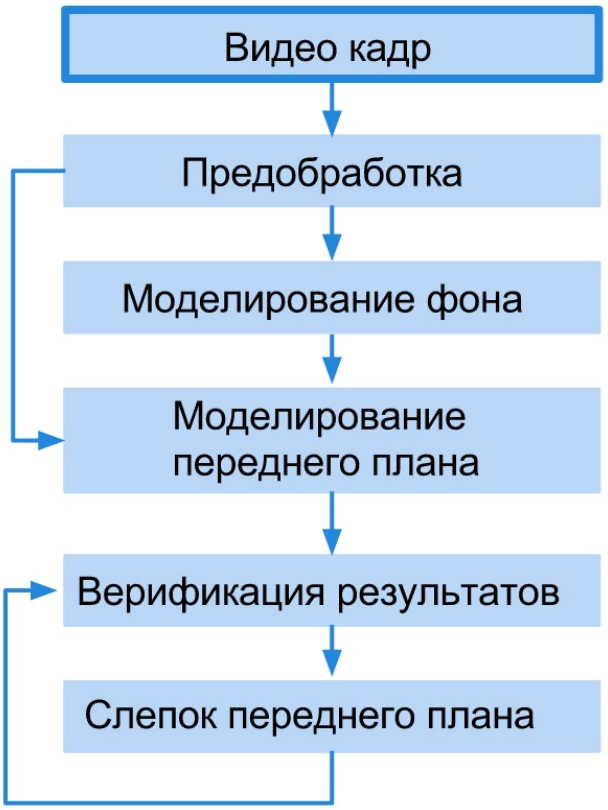

Рис. 8 – Блок-схема алгоритма вычитания фона

5 Рабочий этап проектирования системы

5.1 Визуализация

Во время анализа видео выполняя любые операции связанные с анализом необходимо визуализировать все изменения. Так имея на входе обычное RGB необходимо преобразовать с кажем в серый. Для того чтобы удостоверится в адекватности проделанных манипуляций нужно вузуализировать каждое дей-

ствие. Куда более это критично для трекинга. Проверять правильность формирования трека невозможно или чрезвычайно сложно без визуализации.

Рис. 9 – Пример обработанного видео

А наибольший интерес для нас представляет визуализация подсчета людей, а так же детекция их движения. Так движущиеся объекты описываются красными прямоугольниками, а счетчик выполнен в виде цифр над и под пересекаемой линией.

5.2 Тестирование

Тестирования выявило слабые стороны применения данного метода и наиболее частые из них приведены в диаграмме ложных срабатываний

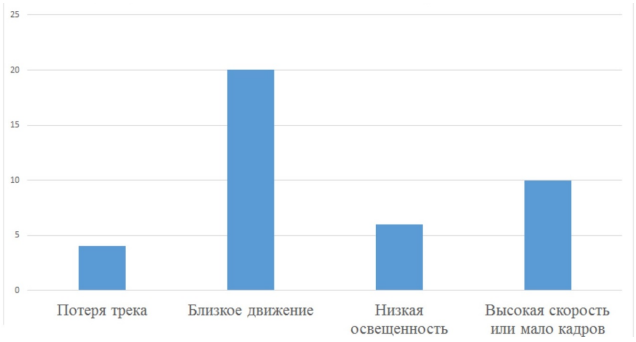

Рис. 10 – Диаграмма ложных срабатываний

5.3 Результаты

В результате работы мы получили систему позволяющая подсчитывать людей при помощи анализа видео. Общие характеристики данной системы приведены ниже(На основе отладочного видео):

- Максимальная точность - 84,5
- Реальное кол-во людей: 32
- Подсчитанное кол-во людей: 27
- Длительность видео: 75 сек.

Система применима для слабопоточных систем с хорошим освещением. К недостаткам следует отнести: некорректное срабатывание при близком движение объектов, необходимость предварительной настройки, работает на любых движущихся объектах и требуется хорошая освещенность.

Рис. 11 – Результаты работы

5.4 Выводы

Характеристики:

- Система имеет высокую точность*
- Быстрая и простая развертка
- Применима для любых движущихся объектах
- Не требовательна к качеству входного видео
- Не подходит для задач с точностью $\tilde{1}00\%$
- Не подходит для сильнопоточных систем

Точность системы не максимальна и может быть повышена. Эта характеристика наиболее критична, что я планирую попробовать сделать в рамкам летней практики. Так планируется добавить возможность подсчета близкоидущих людей.

Список литературы

1) Chris Stauffer, W.E.L. Grimson, Adaptive background mixture models for real-time tracking, CVPR 1999

2) Ahmed Elgammal, David Harwood, Larry Davis, Nonparametric model for background subtraction, ICCV, 2000

3) Jian Sun, Weiwei Zhang, Xiaoou Tang, Heung-Yeung Shum, Background cut, CVPR, 2006

4) A. Criminisi, G. Cross, A. Blake, V. Kolmogorov, Bilayer segmentation of live video, CVPR, 2006

5) Желтов С.Ю. и др. Обработка и анализ изображений в задачах машинного зрения. — М.: Физматкнига, 2010. — 672 с.

6) А.А. Лукьяница ,А.Г. Шишкин Цифровая обработка видеоизображений. — М.: «Ай-Эс-Эс Пресс», 2009. — 518 с.

7) Дэвид Форсайт, Жан Понс Компьютерное зрение. Современный подход = Computer Vision: A Modern Approach. — М.: «Вильямс», 2004. — 928 с.

8) Paul C. Box, Joseph C. Oppenlander (1976), Manual of traffic engineering studies, Institute of Transportation Engineers, p. 17, retrieved December 21, 2010

```
import cv2.cv as cv
from cv2.cv import *
import math
import numpy as np
import json
from shapely.geometry import LineString
with open('config.json', 'r') as f:
    config = json.load(f)

first_point = 0
cur_track_id = 0
cur_count_id = 0
point_with_no_track = ()
list_of_tracks = []
list_of_counters = []


class Statistics:
    def __init__(self):
        self.epsilon = config['epsilon']

    def get_equation_coefficients(self, line):
        f_x = line.xy[0][0]
        f_y = line.xy[0][1]
        s_x = line.xy[1][0]
        s_y = line.xy[1][1]
        A = [[f_x, 1], [f_y, 1]]
        b = [s_x, s_y]
        A = np.array(A)
        b = np.array(b)
        x = list(np.linalg.solve(A, b))
        return x[0], x[1]

    def line_as_a_function(self, x, a, b):
        return a * x + b

    def get_objects_crossing_line_count_up_down(self, counter, track, line):
        prev = track.coordinates[len(track.coordinates) - 2]
        cur = track.coordinates[len(track.coordinates) - 1]
        line_from_track = LineString([prev, cur])
        a_coefficient_line, b_coefficient_line = Statistics.get_equation_coefficients(self,
        line)
        line_function_value_prev = Statistics.line_as_a_function(self, line_from_track.xy[0][
        0], a_coefficient_line, b_coefficient_line)
        line_function_value_cur = Statistics.line_as_a_function(self, line_from_track.xy[1][0
        ], a_coefficient_line, b_coefficient_line)
        track_function_value_prev = prev[1]
        track_function_value_cur = cur[1]
        if line_function_value_cur + self.epsilon > track_function_value_cur >
        line_function_value_cur - self.epsilon and not counter.counter_id in track.
        list_of_counters:
            track.set_epsilon_true(counter.counter_id)
            if track_function_value_prev > line_function_value_prev + self.epsilon and
            track_function_value_prev > line_function_value_prev - self.epsilon:
                counter.add_up_counter()
            elif track_function_value_prev < line_function_value_prev + self.epsilon and
            track_function_value_prev < line_function_value_prev - self.epsilon:
                counter.add_down_counter()
```

```
        return counter.counter_up_value, counter.counter_down_value


class Counter:
    def __init__(self):
        self.counter_up_value = 0
        self.counter_down_value = 0
        self.counter_id = 0

    def add_new_counter_id(self, counter_id):
        self.counter_id = counter_id

    def add_up_counter(self):
        self.counter_up_value += 1

    def add_down_counter(self):
        self.counter_down_value += 1


class Visualization():
    def __init__(self):
        self.line_color = (250, 200, 0)

    def add_line(self, color_image, frame_height, frame_width, font, line_p1_x=0, line_p2_x=
    200, line_p1_y=200,
                 line_p2_y=200, line_color=(100, 200, 0), line_width=10):
        cv.Line(color_image, (line_p1_x, line_p1_y), (line_p2_x, line_p2_y), line_color,
        line_width)

    def add_text(self, color_image, p_x, p_y, font, title="up", text_color=(250, 200, 0)):
        cv.PutText(color_image, title, (p_x, p_y), font, text_color)


class Track:
    def __init__(self):
        self.track_id = ""
        self.coordinates = []
        self.list_of_counters = []
        self.coordinates_time = []
        self.coordinates_speed = []

    def add_point(self, point):
        self.coordinates.append(point)

    def add_point_time(self, time):
        self.coordinates_time.append(time)

    def set_track_id(self, track_id):
        self.track_id = "track %d" % track_id

    def set_epsilon_true(self, counter_id):
        self.list_of_counters.append(counter_id)


class Tracking:
    def __init__(self):
        self.max_distance_between = config['max_distance_between']
        self.min_distance_between = config['min_distance_between']
```

```
        self.list_of_points = []
        self.track = Track()

    def find_nearest_track(self, point, cur_time):
        global first_point, point_with_no_track
        point_with_no_track = ()
        we_found_nearest_track = False
        if first_point == 0:
            point_with_no_track = point
            first_point = 1
        if first_point == 2:
            for track in list_of_tracks:
                if track:
                    last_track_coord = len(track.coordinates) - 1
                    distance_between = math.hypot(track.coordinates[last_track_coord][0] -
                    point[0],
                                                  track.coordinates[last_track_coord][1] -
                                                  point[1])
                    if (distance_between <= self.max_distance_between) and distance_between
                    >= self.min_distance_between:
                        track.add_point(point)
                        we_found_nearest_track = True
            if not we_found_nearest_track:
                point_with_no_track = point
        return first_point, point_with_no_track

    def add_points_to_tracks(self, point, cur_time):
        global first_point, cur_track_id, point_with_no_track
        first_point, point_with_no_track = self.find_nearest_track(point, cur_time)
        if point_with_no_track:
            if first_point == 1:
                self.track.set_track_id(cur_track_id)
                self.track.add_point(point_with_no_track)
                list_of_tracks.append(self.track)
                cur_track_id += 1
                first_point = 2
            else:
                self.track.set_track_id(cur_track_id)
                self.track.add_point(point_with_no_track)
                list_of_tracks.append(self.track)
                cur_track_id += 1


class BorderLine:
    def __init__(self):
        self.line_point_1 = []
        self.line_point_2 = []
        self.line = LineString()

    def set_line_points(self, line_point1, line_point2):
        self.line_point1 = line_point1
        self.line_point2 = line_point2
        self.line = LineString([self.line_point1, self.line_point2])


class Target:
    def __init__(self):
        self.capture = cv.CaptureFromFile(config['video_file'])
```

```
        frame = cv.QueryFrame(self.capture)
        self.frame_size = cv.GetSize(frame)
        self.grey_image = cv.CreateImage(self.frame_size, cv.IPL_DEPTH_8U, 1)
        self.moving_average = cv.CreateImage(self.frame_size, cv.IPL_DEPTH_32F, 3)
        self.min_area = config['min_area']
        self.max_area = config['max_area']
        self.frame_width = self.frame_size[0]
        self.frame_height = self.frame_size[1]
        self.list_of_points = []


    def image_difference(self, first):
        global temp, moving_difference
        color_image = cv.QueryFrame(self.capture)
        if first:
            moving_difference = cv.CloneImage(color_image)
            temp = cv.CloneImage(color_image)
            first = False
        cv.AbsDiff(color_image, temp, moving_difference)
        cv.CvtColor(moving_difference, self.grey_image, cv.CV_RGB2GRAY)
        cv.Threshold(self.grey_image, self.grey_image, 70, 255, cv.CV_THRESH_BINARY)
        cv.Dilate(self.grey_image, self.grey_image, None, 18)
        return color_image, first

    def add_contour_in_storage(self):
        storage = cv.CreateMemStorage(0)
        contour = cv.FindContours(self.grey_image, storage, cv.CV_RETR_CCOMP, cv.
        CV_CHAIN_APPROX_SIMPLE)
        return contour

    @staticmethod
    def get_rectangle_parameters(bound_rect, color_image):
        pt1 = (bound_rect[0], bound_rect[1])
        pt2 = (bound_rect[0] + bound_rect[2], bound_rect[1] + bound_rect[3])
        x_center = abs(pt1[0] - pt2[0]) / 2 + pt1[0]
        y_center = abs(pt1[1] - pt2[1]) / 2 + pt1[1]
        point = (x_center, y_center)
        y_length = abs(pt1[0] - pt2[0])
        x_length = abs(pt1[1] - pt2[1])
        area = x_length * y_length
        return pt1, pt2, point, area

    def get_points_tracking(self, point, area):
        if self.min_area < area < self.max_area:
            self.list_of_points.append(point)

    def run(self):
        line_point_1 = config['line_point_1']
        line_point_2 = config['line_point_2']
        line = LineString([line_point_1, line_point_2])
        counter = Counter()
        statistic = Statistics()
        counter.add_new_counter_id(cur_count_id)
        list_of_counters.append(counter)
        first = True
        writer = cv.CreateVideoWriter("out.avi", cv.CV_FOURCC('D','I','V','X'), 30, self.
        frame_size, True)
        while True:
```

```
            color_image, first = self.image_difference(first)
            contour = self.add_contour_in_storage()
            font = cv.InitFont(cv.CV_FONT_HERSHEY_SIMPLEX, 1, 0.2, 0, 1, 1)
            line_and_text = Visualization()
            line_and_text.add_line(color_image, self.frame_height, self.frame_width, font,
            line_p1_x=line_point_1[0],
                                   line_p1_y=line_point_1[1], line_p2_x=line_point_2[0],
                                   line_p2_y=line_point_2[1],
                                   line_width=4, line_color=(0, 222, 322))
            while contour:
                bound_rect = cv.BoundingRect(list(contour))
                contour = contour.h_next()
                pt1, pt2, point, area = self.get_rectangle_parameters(bound_rect, color_image)
                cv.Rectangle(color_image, pt1, pt2, cv.CV_RGB(255, 0, 0), 1)
                self.list_of_points = []
                self.get_points_tracking(point, area)
                tracking = Tracking()
                cur_time = GetCaptureProperty(self.capture, CV_CAP_PROP_POS_MSEC)
                for point in self.list_of_points:
                    tracking.add_points_to_tracks(point, cur_time)
                for track in list_of_tracks:
                    track.add_point_time(cur_time)
                    if len(track.coordinates_time) > 5*len(track.coordinates):
                        list_of_tracks.remove(track)
                    x_prev = track.coordinates[len(track.coordinates) - 2][0]
                    y_prev = track.coordinates[len(track.coordinates) - 2][1]
                    cur_x = track.coordinates[len(track.coordinates) - 1][0]
                    cur_y = track.coordinates[len(track.coordinates) - 1][1]
                    line_and_text.add_line(color_image, self.frame_height, self.frame_width,
                    font, line_p1_x=x_prev,
                                           line_p1_y=y_prev, line_p2_x=cur_x, line_p2_y=cur_y
                                           , line_width=4,
                                           line_color=(0, 222, 322))
                    line_and_text.add_text(color_image, cur_x, cur_y, font, title=track.
                    track_id)

                    up, down = statistic.get_objects_crossing_line_count_up_down(counter,
                    track, line)
                    line_and_text.add_text(color_image, int(line.xy[0][0]), int(line.xy[1][0
                    ])-10, font, text_color=(100, 200, 50), title="U " + str(up))
                    line_and_text.add_text(color_image, int(line.xy[0][0]), int(line.xy[1][0
                    ])+20, font, text_color=(100, 200, 50), title="D " + str(down))

            cv.WriteFrame(writer, self.grey_image)

            cv.ShowImage("target", color_image)
            cv.ShowImage("target", self.grey_image)
            c = cv.WaitKey(config['wait_key']) % 0x100
            if c == 27:
                break
if __name__ == "__main__":
    t = Target()
    t.run()
```