Департамент образования муниципия Кишинев.

Лицей имени Н. М. Спэтару

Сектор информатики.

**Тема :**

**“Speed Supervisor”**

**программа для повышения безопасности в школах.**

Ученик 9Б

Мельник Роман

Преподаватель:

Майкова В.Э.

Кишинев, 2018

Содержание

1. Цели проекта.
2. Задачи проекта.
3. Принцип работы.
   1. Этапы.
   2. Детектирование.
      1. Метод Виолы-Джонса.
      2. Интегральное представление изображения.
      3. Гистограмма ориентированных градиентов.
   3. Трекинг.
      1. Фильтр Кальмана.
      2. Венгерский алгоритм.
   4. Вычисление скорости.
   5. Обработка результатов
4. Описание реализации проекта и способов применения.
5. Дальнейшие планы.
6. Библиография.

Цели и задачи.

# Цели

Целью моего проекта является повышение безопасности и дисциплины в школах. Не секрет, что во время перемен младшеклассники развлекаются, занимаются активным отдыхом, что подразумевает под собой бег по помещению. Преподавателям не всегда удается это проконтроллировать, потому процент столкновений и несчастных случаев, являющихся результатом бега учеников довольно высок. Для коррекции этого показателя мною была придумана идея специальной программы, которая использовала бы установленные в коридорах школы камеры видеонаблюдения для получения изображения, его обработки и фиксировании превышения скорости уеником в реальном времени. Таким образом администрация школы всегда сможет отследить нарушителя и провести с ним воспитательную беседу.

# Задачи

Для решение вышеописанной задачи и реализации программы необхоидмо было выполнить следующие шаги :

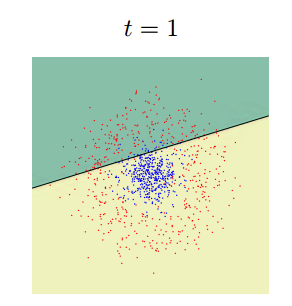
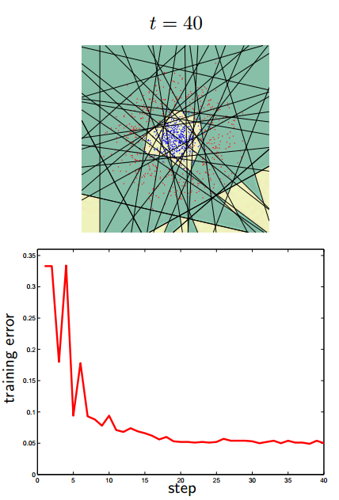
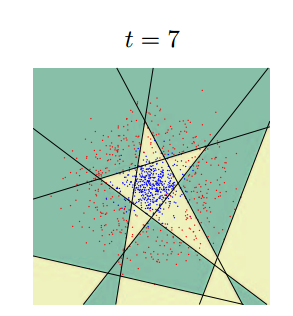
1. Изучить основы компьютерного зрения и глубокого обучения.
2. Изучить библиотеку OpenCV.
3. Изучить методы поиска объекта на изображении и приобрести навыки по их реализации.
4. Изучить методы трекинга объекта и приобрести навыки по их реализации.
5. Совместить полученные знания в готовом проекте.

В итоге, спустя несколько месяцев работы программа была готова. Ниже представлен принцип её работы.

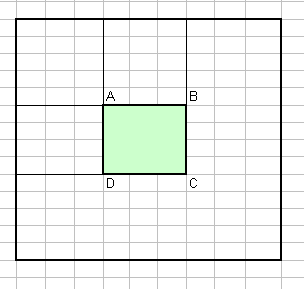
Принцип работы

Работу программы можно разделить на три части : поиск координат человека на кадре (детектирование), попытка установления соответствия найденных координат определенного человека с его же координатами на прошлом кадре(трекинг), расчёт условной скорости движения. Обнаружение объекта реализовано двумя способами : алгоритмом Виолы-Джонса и гистограммой ориентированных градиентов.

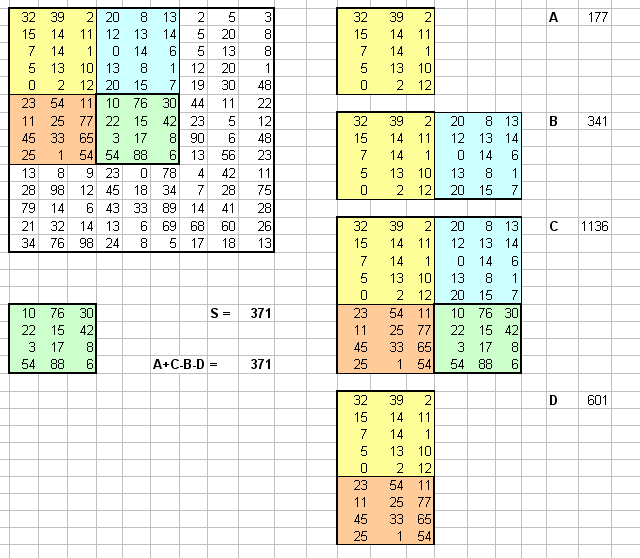
# Метод Виолы-Джонса

Метод Виолы-Джонса заключается в построении каскада слабых классификаторов на основе алгоритма AdaBoosting. Во время обучения из множества слабых классификаторов составляется один сильный. На картинках ниже представленно то, как увеличивается точность алгоритма со временем обучения на примере двумерного пространства.

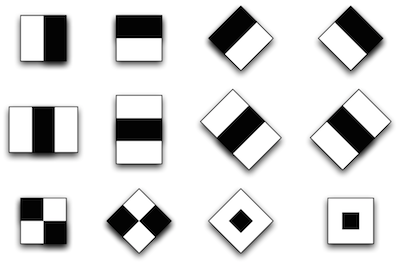
Для определения принадлежности к классу в каскаде находится сумма значений слабых классификаторов этого каскада. Каждый слабый классификатор выдает два значения в зависимости от того больше или меньше заданного порога значение признака, принадлежащего этому классификатору. В конце сумма значений слабых классификаторов сравнивается с порогом каскада и выносится решения найден объект или нет данным каскадом. Классом, в нашем случае, является человек. Для построения каскада используется обучающая выборка положительных примеров изображений, где объект присутствует вместе с его координатами, и отрицательными, где объект отсутствует. Каждое изображение конвертируется в интегральное представление - матрицу, размерность которой совпадает с размерностью исходного изображения. Каждый элемент матрицы II(x,y) содержит в себе сумму пикселей изображения в прямоугольнике от (0,0) до (x,y). Одной из важнейших особенностей интегрального представления является возможность очень быстро вычислить сумму пикселей произвольного прямоугольника (или любой другой фигуры, которую можно аппроксимировать несколькими прямоугольниками).Например, интересующий нас прямоугольник – ABCD.



Тогда сумма пикселей внутри этого прямоугольника будет равняться II(A)+II(C)-II(D)-II(B). Это легко можно проверить в таблице Excel’я.



Данное свойство очень полезно, так как признаками для классификаторов служат примитивы Хаара, которые состоят из смежных прямоугольных областей. Они позиционируются на изображении, далее суммируются интенсивности пикселей в областях (благодаря интегральному представлению сделать это не составляет труда), после чего вычисляется разность между суммами закрашенных и незакрашенных областей. Эта разность и будет значением определенного признака, определенного размера, определенным образом спозиционированного на изображении.



Примеры примитивов Хаара.

Для каждого признака строится слабый классификатор, который дает наименьшую ошибку на всех тренировочных примерах.

Выходом алгоритма обучения является база классификаторов в формате XML, которая позже будет использоваться для алгоритма обнаружения. На его вход поступает изображение, которое затем конвертируется в интегральное представление, а результатом работы является массив прямоугольников R(x,y,w,h), определяющих положение найденных объектов на изображении. Алгоритм работает по принципу "скользящего окна". То есть рамка, размером, меньшим, чем исходное изображение, двигается с некоторым шагом по изображению, и с помощью каскада слабых классификаторов определяет, есть ли в рассматриваемом окне искомый предмет. Чем меньше рамка и шаг, тем точнее и медленнее детектирование.

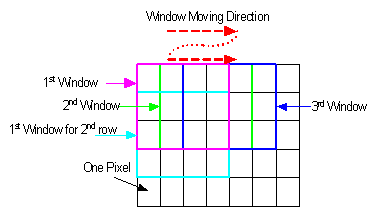
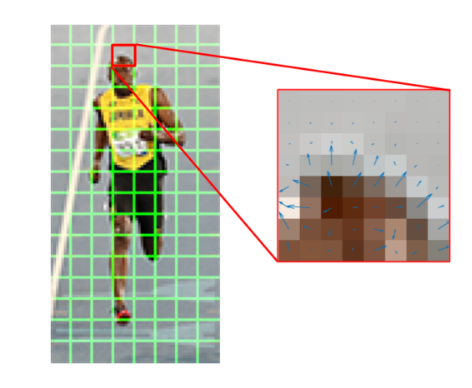


Схема движения скользящего окна

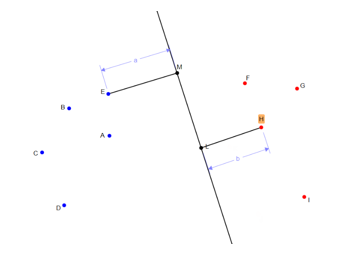
Плюсами метода Виолы-Джонса является очень высокая скорость распознавания при достаточно маленьком количестве ошибок, что дает возможность использовать его для поиска множества предметов на изображений в режиме реального времени.

# Гистограмма ориентированных градиентов (HOG)

Вторым методом для обнаружения людей на изображении является гистограмма ориентированных градиентов(HOG). Это простая процедура выделения признаков состоит в постройке для каждого пикселя градиента яркости, сравнивая его с соседними пикселями, которая показывает направление от светлых пикселей к более темным. Далее изображение разбивается на ячейки 8x8, внутри которых градиенты каждого пикселя группируются и ячейке присваивается стрелочка направленная в ту сторону, куда и большинство пикселей внутри неё. Таким образом, мы рассматриваем только направление изменения яркости на изображении.



Пример разбивания изображения на ячейки



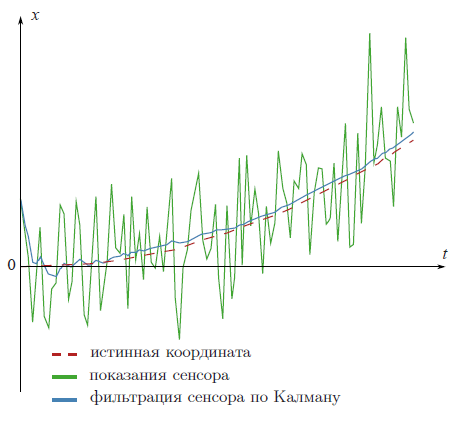
Разделение двух классов SVM

Следующим шагом является построение классификатора на основе гистограмм из обучающей выборки. В данном проекте в качестве линейного классификатора для HOG дескриптора используется метод опорных векторов(SVM), который проводит между двумя классами оптимальную разделяющую гиперплоскость, максимально удаленную от двух крайних элементов каждого класса. После чего можно использовать алгоритм обнаружения, который работает почти так же, как и описанный выше.

HOG позволяет получать более корректные и правдивые результаты, по сравнению с каскадами Хаара, однако и скорость работы этого алгоритма медленнее. Выбор метода детектирования зависит от конкретного случая использования программы.

# Фильтр Калмана

Теперь, когда мы имеем координаты людей, необходимо отследить конкретного человека на всех кадрах, чтобы можно было вычислить его скорость. Для этого в программе используется фильтр Кальмана и алгоритм Куна решения задачи о назначениях(Венгерский). Фильтр Кальмана это рекурсивный фильтр, который оценивает состояние динамической системы. Любой измерительный прибор обладает некоторой погрешностью, на него может оказывать влияние большое количество внешних и внутренних воздействий, что приводит к тому, что информация с него оказывается зашумленной. Фильтрация позволяет избавиться от шумов и получить более точные измерения. К примеру, у нас есть значения с сенсора, который измеряет какую-либо величину. Основываясь на характере измеряемого процесса фильтр так же выдает свое значение этой величины. Истинное значение должно находиться где-то между тем, что выдал сенсор и тем, что предположил фильтр. Сравнивая эти значение, фильтр пытается определить максимально приближенное к истине, на каждом шаге работы корректируя некоторый коэффициент который используется для его вычисления.



В нашем случае, динамической системой является траектория пути человека на изображении и фильтруем мы её координаты. К каждому найденному объекту применяется фильтр, который корректирует его расположение. Но, когда мы получаем информацию о найденных людях на изображении, мы не знаем, какому человеку какой фильтр соответствует. Для установления этого соответствия используется Венгерский алгоритм.

# Венгерский алгоритм

Объясним его на примере. Допустим, у нас есть 4 работника и 4 рабочих места. На каждом рабочем месте определенный работник получает какую-то заработную плату. Это легко можно представить в виде матрицы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Работник/Работа | 1 | 2 | 3 | 4 |
| 1 | **1** | **7** | **1** | **3** |
| 2 | **1** | **6** | **4** | **6** |
| 3 | **17** | **1** | **5** | **1** |
| 4 | **1** | **6** | **10** | **4** |

Мы, как работодатель заинтересованы в том, чтобы каждый работник получал минимальную заработную плату, но при этом, чтобы каждому работнику соответствовала одна работа, а каждой работе один работник. Это можно было бы сделать перебором, но уже при 10 типах работ и работников возникнут трудности, так как он будет происходить за факториальное время. Для решения подобного рода задачи нам и нужен Венгерский алгоритм. Он состоит из следующих шагов :

**1.** Необходимо вычесть из каждого элемента строки минимальный, таким образом, чтобы в каждой строке и в каждом столбце присутствовал ноль. Если же этого не произошло, необходимо повторить эту же операцию со столбцами. В нашем случае получилась вот такая матрица.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Работник/Работа | 1 | 2 | 3 | 4 |
| 1 | **0** | **6** | **0** | **2** |
| 2 | **0** | **5** | **3** | **5** |
| 3 | **16** | **0** | **4** | **0** |
| 4 | **0** | **5** | **9** | **3** |

**2.** Далее, исходя из расположения нулей в матрице строится двудольный граф, где для каждого нуля назначается ребро.

**X**

**Y**

**X**

**Y**

Исходный граф

Наибольшее паросочетание

**3.** На построенном графе необходимо найти наибольшее паросочетание. То есть построить еще один двудольный граф, где степени вершин будут равняться единице. Для этого строятся все комбинации максимальных паросочетаний и ищется то, где количество ребер больше. Делается это с помощью чередующихся цепей.

**4.** Конечной целью является нахождения совершенного паросочетания, когда каждая вершина имеет степень единицы. Если найденное наибольшее паросочетания не является совершенном, то есть в нем присутствуют вершины с нулевой степенью, как на рисунке выше, то необходимо для первой области наибольшего паросочетания X составить множество элементов, которые в него не входят - Xm. Теперь, из этого множества нужно составить еще два - X' и Y'. В них войдут вершины, по которым нужно пройти из какого-либо элемента Xm в область Y и вернуться обратно в область X согласно принципу чередующихся цепей.

**X**

**Y**

**xm = { X₄ }**

**X’ = { X₄, X₂ }**

**Y’ = { Y₁ }**

Теперь, используя эти два множества и преобразованную матрицу мы должны составить еще одну. Во множестве X' у нас есть строки, которые будут участвовать в преобразовании, а во множестве Y' столбцы, которые не будут. Теперь, среди элементов(они отмечены желтым), которые находятся на пересечении нужных нам столбцом и строк мы ищем минимум.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Работник/Работа | 1 | 2 | 3 | 4 |
| 1 | **0** | **6** | **0** | **2** |
| 2 | **0** | **5** | **3** | **5** |
| 3 | **16** | **0** | **4** | **0** |
| 4 | **0** | **5** | **9** | **3** |

Этот минимум вычитается из строчных выделенных элементов и прибавляется ко всем элементам незадействованных столбцом.

Мы прибавляем всем элементам, которые помечены зеленым минимум желтых элементов – 3, а из всех желтых его вычитаем. Темно-зеленые элементы относятся и к тому и к тому множеству, потому мы их не трогаем.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Работник/Работа | 1 | 2 | 3 | 4 |
| 1 | **3** | **6** | **0** | **2** |
| 2 | **0** | **2** | **0** | **2** |
| 3 | **19** | **0** | **4** | **0** |
| 4 | **0** | **2** | **6** | **0** |

Таким образом, мы получаем еще одну преобразованную матрицу и теперь повторяем выше перечисленные действия до тех пор, пока наше наибольшее паросочетание не окажется совершенным.

Построим двудольный граф для полученной матрицы и найдём его наибольшее паросочетание.

**X**

**Y**

**X**

**Y**

Новый граф

Наибольшее паросочетание

Как мы видим, у нас получилось совершенное паросочетание. Каждая вершина находиться в 1 степени.

Теперь, отметим на исходной матрице элементы, которые получились в совершенном паросочетании.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Работник/Работа | 1 | 2 | 3 | 4 |
| 1 | **1** | **7** | **1** | **3** |
| 2 | **1** | **6** | **4** | **6** |
| 3 | **17** | **1** | **5** | **1** |
| 4 | **1** | **6** | **10** | **4** |

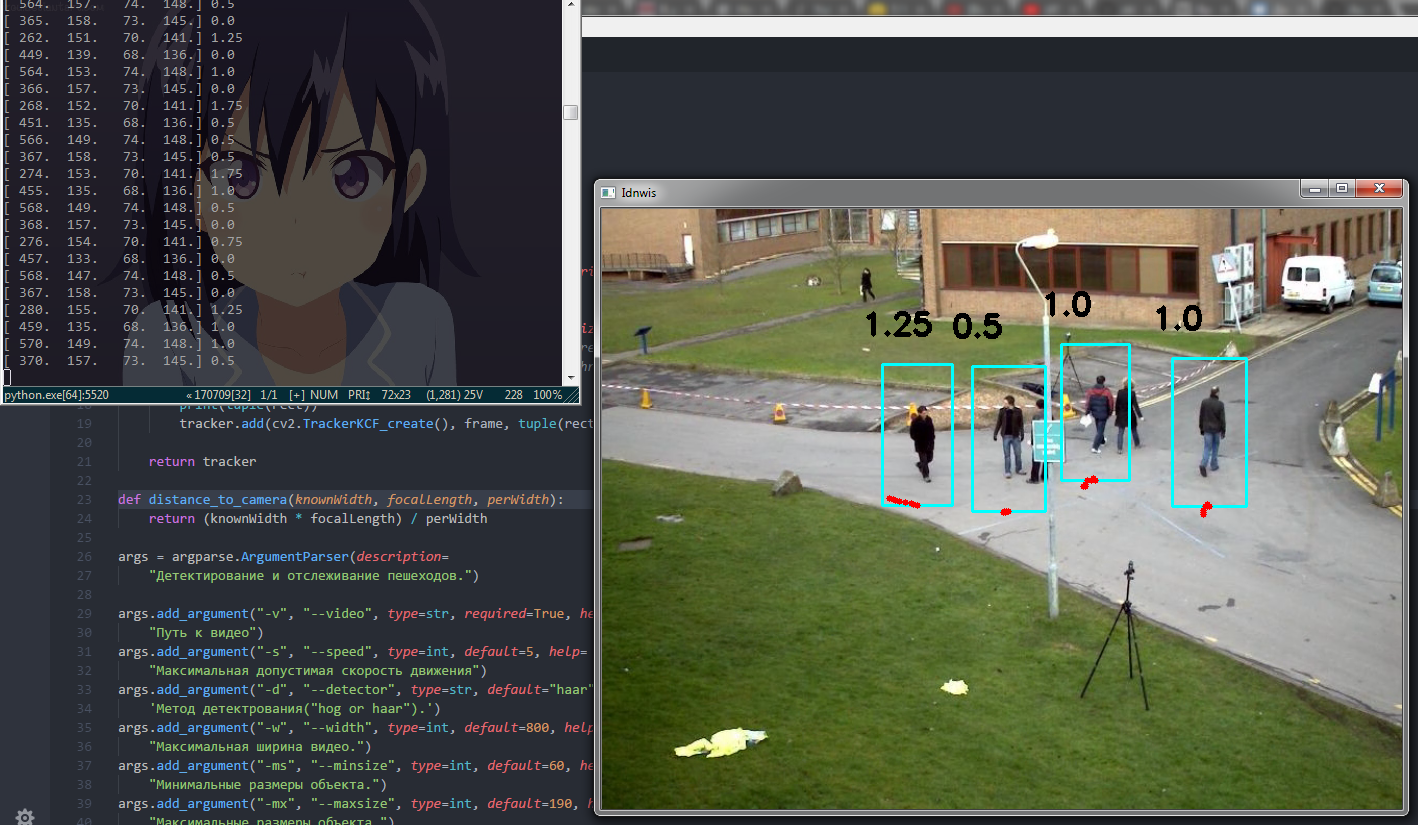
Это и будет решением задачи.

# Вычисление скорости и обработка результатов.

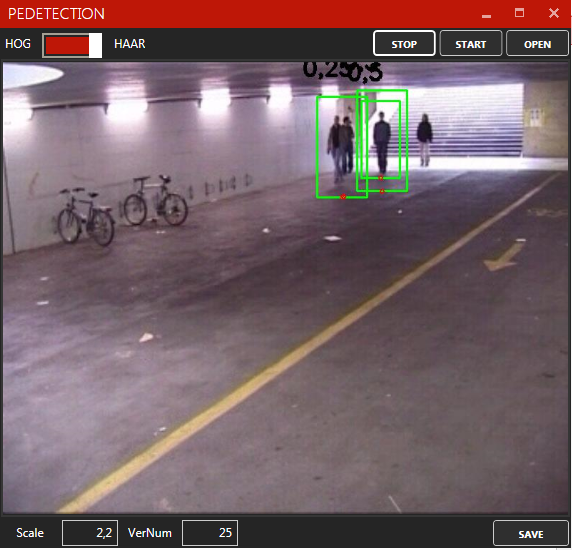
В программе этот алгоритм используется для установления соответствия координат, которые выдает фильтр и координат, которые выдают методы детекции. Если их число не совпадает, следовательно, из поле зрения камеры либо ушел человек, либо пришел новый. В первом случае мы ждем некоторое время, прежде чем удалить не использующийся фильтр, так как человек возможно просто перекрыт каким-то другим или же он ненадолго вышел из кадра. Если по истечению времени фильтру не было найдено соответствие то он удаляется. Во втором случае мы просто создаем новый фильтр для нового человека. Комбинация фильтра Кальмана и алгоритма Куна дает хорошие результаты в задаче трекинга множества объектов и не является ресурсоемкой, что так же позволяет использовать её в реальном времени. Теперь, имея координаты людей на протяжении некоторой секвенции последовательно идущих изображении можно вычислить условную величину, отражающую в себе скорость движения объекта основываясь на скорости изменения его координат. Для преобразования этого значения в общепринятые единицы измерения необходимо знать фокусное расстояние камеры. Далее, опытным путем устанавливается порог этой величины, по пересечению которого можно считать, что человек бежит. Вычисляется он опытным путем для каждой камеры отдельно. Когда порог будет достигнут, программа зафиксирует объект, который его превысил, чтобы в дальнейшем возможно было опознать нарушителя.

Реализация проекта

Проект реализован в двух версиях. Первая написана на Python’e, с помощью открытой библиотеки компьютерного зрения OpenCV. Для её использования необходимо установить Python 3 версии и выше а так же все зависимости, которые указаны в коде. Это первая версия без пользовательского интерфейса, все настройки указываются в коде и аргументах командной строки.



Вторая версия реализована на C#, в ней используется фреймворк OpenCV - EmguCV. Эта версия обладает таким же функционалом как и первая, но уже с пользовательским интерфейсом, однако она менее гибкая.



Дальнейшие планы и библиография.

В будущем в программу будет добавлена возможность отправки изображений с зафиксированными нарушениями на какой-либо интернет-ресурс или же электронную почту для более удобной возможности просмотра. Так же планируется создание веб-сервиса на основе программы, который будет принимать на себя все сложные и ресурсоемкие вычисления. Тем самым, для эффективной работы программы не нужен будет мощный компьютер. Достаточно лишь хорошего интернет соединения.

# Библиография.

<https://opencv.org/> - официальный сайт OpenCV.

<http://www.emgu.com/> - официальный сайт EmguCV.

<https://oxozle.com/2015/04/11/metod-raspoznavaniya-lic-violy-dzhonsa-viola-jones> - метод Виолы-Джонса.

<https://www.learnopencv.com/histogram-of-oriented-gradients/> - описание работы HOG дескриптора.

<https://habrahabr.ru/post/166693/> - описание работы фильтра Калмана.

<http://studentdavestutorials.weebly.com/> - принципы мультитрекинга.

<https://www.youtube.com/watch?v=RmTecuCqnU0> – описание Венгерского алгоритма.