Здравствуйте, меня зовут Смирнов Игорь, я студент 15-В-1. Сегодня, я хочу представить вашему вниманию ВКР: программная система распознавания дорожной разметки.

2й слайд

Собственно, целью работы и было разработать программную систему для распознавания линий дорожной разметки. Для этого мне необходимо

- Выбрать алгоритм предварительной обработки данных
- Выбрать средства основной обработки данных
- Определить метод принятия решения о найденных линиях

3й слайд

Актуальность этой темы обусловлена ростом популярности программно-аппаратных систем, встраиваемых в автомобиль для помощи водителю и, соответственно, большим интересом со стороны автопроизводителей к данным системам. Ни для кого не секрет, что уже существуют машины с автопилотом, однако все компании вольны сами выбирать подход к решению данной задачи и вести свои исследования в этом направлении. Хотя система с автоматическим пилотированием — это целый комплекс средств, от программной части до технической реализации, компьютерному зрению есть место и в более простых системах, таких как уведомление водителя о съезде с полосы или удержание машины в полосе движения.

4й слайд

Итак, алгоритм работы программы можно разделить на три больших этапа это подготовка изображения и нахождение на ней прямых линий, выбор из найденных линий линии, относящиеся к дорожной разметке и обозначение найденных линий на исходном изображении. Для решения этой задачи я использую библиотеку OpenCV для работы с компьютерным зрением.

5й слайд

Для начала мы получаем очередной кадр изображения с, к примеру, видеорегистратора. Прежде всего нам необходимо перейти от RGB пространства изображения к HSV.

Чаще всего для хранения цифровых изображений используется цветовое пространство RGB. К сожалению, RGB не всегда хорошо подходит для анализа информации. Эксперименты показывают, что геометрическая близость цветов достаточно далека от того, как человек воспринимает близость тех или иных цветов друг к другу.

По этому, при обработки данных в компьютерном зрении используют пространство HSV (Hue, Saturation, Value). В нем присутствует ось Value, обозначающая количество света. Hue представляется в виде угла и отвечает за основной тон. От значения Saturation (расстояние от центра к краю) зависит насыщенность цвета.

Перейдём от RGB модели цвета к HSV, преобразование поможет в дальнейшем искать определённые цвета на изображении независимо от их насыщенности и яркости.

6й слайд

Следующим шагом, необходимо сделать из изображения бинарную маску с теми цветами, которые нам нужны, а нужны нам цвета линий дорожной разметки, т.е. белый и оттенки серого. В данном случае бинарная маска накладывается не на изображение в оттенках серого, а на исходное изображение в данном случае это полученное после смены цветового пространства, чтобы помимо белого цвета, можно было обнаружить на изображении жёлтый цвет и его оттенки. Бинарная маска накладывается на изображение простым путём. Мы имеем верхнюю границу цвета во всех каналах и нижнюю. Функция проходит по всем пикселям изображения и проверяет попадает пиксель в диапазон или нет. Если да – значит пиксель станет белым в результате, нет – черным.

$$res(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0 \wedge\ lowerb(I)_1 \leq src(I)_1 \leq upperb(I)_1$$

Cv2.inRange(image, low, high)

В результате мы получаем бинарную маску, коэффициенты к которой подбираются опытным путём. В данном случае логично, что виден большой кусок неба белого цвета и линии на дороге.

7й слайд

Перейдём к векторизации изображения, здесь мы применим два преобразования, которые лежат в основе компьютерного зрения.

Применим алгоритм Кэнни (анг. Canny) – коротко, этот алгоритм находит границы объектов. Кроме особенных частных случаев трудно найти детектор, который бы работал существенно лучше, чем детектор Канни.
Алгоритм работает в несколько этапов: - Перевод изображения в градации серого - Сглаживание - Поиск градиентов - Подавление не-максимумов - Определение сильных и слабых границ - Поиск верных слабых границ - Очистка от оставшихся границ

Для подавления шума, воспользуемся размытием изображения фильтром Гаусса.
Необходимо выбрать параметры фильтра, обеспечивающие наилучшее подавление шума.
Влияние пикселей друг на друга при гауссовой фильтрации обратно пропорционально квадрату расстояния между ними: коэффициент пропорциональности, а, следовательно, и степень размытия, определяются параметром **σ**. Чрезмерное повышение коэффициента приведёт к усилению усреднения вплоть до равномерно чёрного цвета всех пикселей.

Оператор Собеля часто применяют в алгоритмах выделения границ. По сути, это дискретный дифференциальный оператор, вычисляющий приближенное значение градиента яркости изображения. Результатом применения оператора Собеля в каждой точке изображения является либо вектор градиента яркости в этой точке, либо его норма. Оператор Собеля основан на свёртке изображения небольшими целочисленными фильтрами в вертикальном и горизонтальном направлениях

Полученное изображение имеет толстые края**. 8й слайд**
В идеале конечное изображение должно иметь тонкие края. Значит, необходимо выполнить подавление не-максимумов, чтобы уменьшить толщину полученных ребер
Подавление работает путем нахождения пикселя с максимальным значением в ребре. На изображении это происходит, когда пиксель q имеет интенсивность, которая больше, чем оба значения p и r, где пиксели p и r являются пикселями в направлении градиента q. Если это условие истинно, то мы сохраняем пиксель, в противном случае мы устанавливаем пиксель в ноль (делаем его черным пикселем).
В результате получается изображение с разной яркостью ребер. Установив некоторое пороговое значение яркости пикселей, классифицируем каждое ребро как сильное или слабое (сильное – яркая линия, слабое – темная).
Теперь, определив сильные и слабые границы, необходимо выяснить какие слабые рёбра действительно являются рёбрами. Для этого необходимо проследить какие слабые ребра являются связанными с сильными. Такие рёбра приравниваются к сильным. Слабые рёбра удаляются.
9й слайд: в результате применимо к нашему примеру имеем такое изображение.
Теперь, имея изображение, на котором можно увидеть только очертания объектов, используем преобразование Хафа - это метод для поиска линий, кругов и других простых форм на изображении.
10й слайд
Преобразование Хафа основывается на представлении искомого объекта в виде параметрического уравнения. Параметры этого уравнения представляют фазовое пространство (т.н. аккумуляторный массив/пространство, пространство Хафа).
Затем, берётся двоичное изображение (например, результат работы детектора границ Кенни). Перебираются все точки границ и делается предположение, что точка принадлежит линии искомого объекта — т.о. для каждой точки изображения рассчитывается нужное уравнение и получаются необходимые параметры, которые сохраняются в пространстве Хафа.
Финальным шагом является обход пространства Хафа и выбор максимальных значений, за которые «проголосовало» больше всего пикселей картинки, что и даёт нам параметры для уравнений искомого объекта.
Применив данное преобразование найдём все прямые линии на изображении.
На данный момент наши данные это набор из всех линий на изображении, которые не отсеялись двоичной маской.

11й слайд

Осталось выявить какие из линий подходят нам и являются частью разметки.

Для этого нам нужно ввести некоторые признаки полосы дорожной разметки:

- линия кандидат должна иметь некоторый небольшой наклон относительно вертикали, так как изображение с камеры находится в перспективе; (однако этот пункт должен отдельно проверяться в момент, когда линия находится по центру кадра, к примеру, при перестроении на другую полосу)
- линия кандидат должна быть ниже горизонта;
- линия кандидат должна быть достаточно близко в координатах кадра к тем, что были несколькими кадрами ранее;
- уклон у линии кандидата не должен отличаться сильно от уклона линии уже определённой алгоритмом дорожной разметки в предыдущих кадрах.

К последним двум пунктам я привёл изображения для примера. Если изображение 1 будет первым кадром, а вторым кадром будет изображение 2, то алгоритм не должен распознать дорожную разметку на втором кадре, так как она находится слишком далеко.

Для этого создан небольшой буфер с 4 последними кадрами до текущего и если линия не совпадает ни с одним из них, то она ложная и не относится к разметке.

(при первом запуске алгоритм просто «обучается» не используя этот буфер и записывая в него свои результаты на основании только первых двух признаков). Если алгоритм не может соотнести линии дорожной разметки с теми что находятся в буфере, то он пропускает такой кадр, и через 4 кадра буфер опустошается и алгоритм заполняет его вновь, на деле это происходит менее чем за 1 секунду.

12й слайд

Последним этапом необходимо просто выделить найденные линии на исходном изображении и таким образом видим поверх кадра видео полосы, найденные алгоритмом.

13й слайд

Тестирование системы показало приблизительно 80% точность распознавания дорожной разметки алгоритмом. Сложность тестирования состоит в том, что для точного тестирования системы, необходимо наличие размеченных данных, с которыми можно сверять работу алгоритма. Разметить огромное количество видеоданных для тестирования не представляется возможным, поэтому в качестве тестирования алгоритм запускался на нескольких изображениях и результат его работы оценивался «на глаз».

Теперь, вашему вниманию предлагаю небольшое видео с работой алгоритма.

14й слайд

Небольшая похвала самому себе за лучший доклад на конференции ИСТ-2019 в своей секции.

15й слайд. Спасибо за внимание, ваши вопросы

Функция Гаусса для двумерного случая:

$$G\_gen(x,y,\sigma) := \frac{1}{2\pi\cdot\sigma^2}\cdot e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} A, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & +2 & +1 \end{bmatrix} A$$

Изображение: Лена$|G| = \sqrt{G_x^2 + G_y^2}$

$$\angle G = \arctan(\frac{G_y}{G_x})$$