Санкт-Петербургский государственный университет

Математико-механический факультет
Кафедра системного программирования

Чусовитин Денис Андреевич

Распознавание линии на базе контроллера ТРИК

Курсовая работа

Научный руководитель:
к. ф.-м. н., доцент Вахитов А. Т.

Санкт-Петербург
2016

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Алгоритм **5**

2.1. Гомография 5

2.2. Размытие 6

2.3. Бинаризация 7

2.4. Выделение контуров 7

2.5. Фильтр контуров 8

2.6. Выделение ломаных линий 8

3. Реализация **10**

Заключение **11**

Список литературы **12**

Введение

Автономное решение робота по линии - одна из популярных задач робототехники, которая является частой дисциплиной на соревнованиях. Обычно они проводятся на поле белого цвета с черной линией, которая может быть как непрерывной, так и прерывной. Целью участников является проехать роботом по линии, причем сделать это как можно быстрей и аккуратней.

Традиционно эту задачу решают с помощью камеры, направленной строго вниз. Робот едет вперед и обрабатывает данные с камеры. Если линия смещается с центра полученного изображения, то робот поворачивает так, чтобы снова двигаться по центру линии. Иногда вместо камеры используют специальный датчик линий, но принцип его работы похож на описанный выше способ. При использовании данного решения приходится ограничивать скорость передвижения робота, так как при большой скорости он может не успеть повернуть вовремя, что влечет за собой потерю линии.

Таким образом было предложено поместить камеру так, чтобы она смотрела вперед. Тогда робот будет собирать информацию о линии, находящейся впереди него и сможет использовать её. Например, разгонятся по прямому участку, тормозить перед поворотами или срезать их, если это разрешено правилами. При таком положении камеры появляется необходимость распознавания линии впереди робота, так как существующие решения направлены на работу с линией, расположенной непосредственно под камерой. Данная работа направлена на решение описанной проблемы.

1. Постановка задачи

В рамках курсовой работы были выделены следующие задачи:

- Собрать необходимые видеоматериалы с камеры
- На их основе придумать и реализовать алгоритм распознавания линии
- Переписать алгоритм с учетом работы на платформе ТРИК и перенести его на контроллер[4]

2. Алгоритм

Для работы алгоритма необходимо заранее посчитать матрицу гомографии. Например, её можно получить с помощью шахматной доски. Получив изображение с камеры, нужно выбрать четыре точки на изображении, относящихся к доске, и сопоставить им подобранные точки в координатах поля. Для упрощения подбора точек можно расположить доску по нижней границе изображения. Далее, из полученных пар четырех точек можно легко получить матрицу гомографии.

Также необходимо выбрать порог бинаризации, который зависит от освещения поля.

2.1. Гомография

Рис. 1: Изображение с камеры робота

На вход алгоритма поступает поток изображений (рис. 1). В первую очередь мы переводим изображение из плоскости камеры в плоскость поля. На этом этапе мы избавляемся от части внешних посторонних объектов. Кроме того, теперь появляется возможность работать в координатах поля.

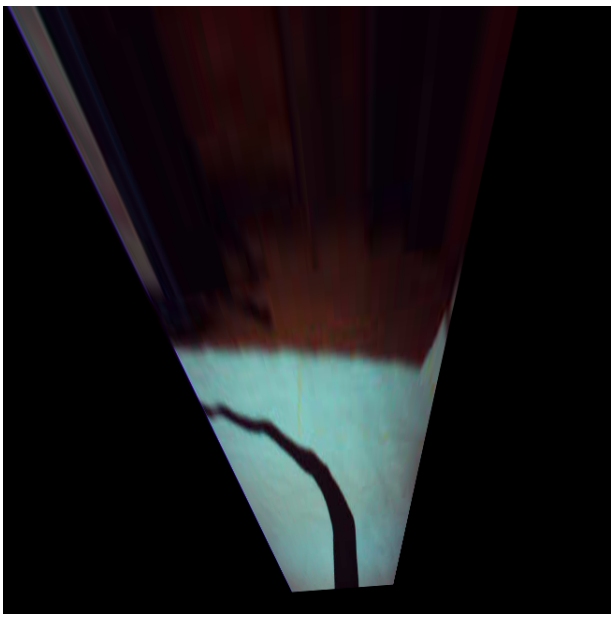

Рис. 2: После применения гомографии

2.2. Размытие

Изображение с камеры имеет небольшое зашумление из-за качества камеры. Для его подавления мы воспользуемся фильтром Гаусса[3]. Выбор этого фильтра был обусловлен тем, что он учитывает расстояния до соседних пикселей и при этом довольно эффективно подавляет шум. В результате его применения мы получаем рис. 3

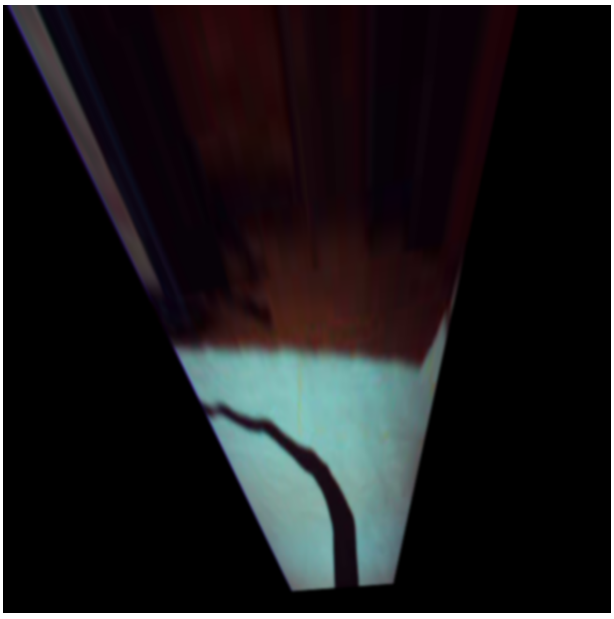

Рис. 3: Размытое фильтром Гаусса изображение

2.3. Бинаризация

На этапе бинаризации мы выделяем линию. Для этого переводим изображение в формат HSV и используем заданный порог (рис. 4). В ходе экспериментов выяснилось, что данный способ бинаризации обладает невысокой мерой полноты (recall), но зато имеет высокую меру точности (precision). Это является важным фактором, так как дополнительная фильтрация влечет лишние вычислительные затраты, что критично для выполнения на платформе ТРИК.

Таким образом, можно предполагать, что белыми пикселями отмечена либо линия на поле, либо внешние объекты.

Рис. 4: Бинаризованное изображение

2.4. Выделение контуров

Далее мы ищем контуры следующим образом: ищем объект, затем идем по его границе, заполняя контур. Пройденные пиксели отмечаем серым цветом, чтобы не искать контур у того же объекта. Необходимость в выделении конутров появилось ввиду того, что линия может казаться прерывной из-за неровностей поля. Кроме того, как было сказано выше, сама линия бывает прерывистой.

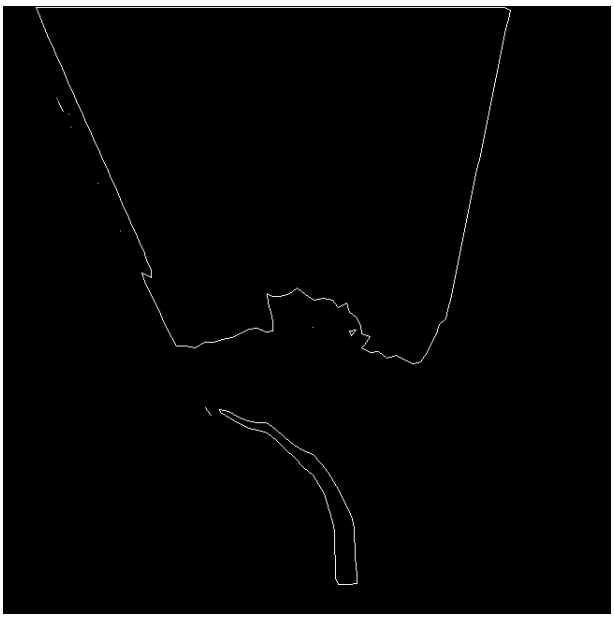

Рис. 5: Найденные контуры

2.5. Фильтр контуров

После нахождения контуров ищем минимальные расстояния между ними. С помощью заданного параметра, обозначающего минимальное расстояние линии до границы поля, мы составляем матрицу смежности контуров. Далее отмечаем те контуры, которые находятся близко к верхней границе изображения. От них, с помощью поиска в глубину, по составленной матрице смежности убираем все оставшиеся "плохие" контуры. Потом удаляем слишком маленькие контуры. Оставшиеся считаем искомыми. (рис. 6)

Из предположения о том, что после бинаризации на изображении в области поля находятся только пиксели линии, мы получаем, что оставшиеся контуры - искомые. Исключения могут составить случайные контуры за пределами поля, но расстояния до них будут слишком велики.

2.6. Выделение ломаных линий

Сначала мы сужаем контуры до ломаной. Для этого идем по контуру с двух параллельных сторон и дополняем ломаную точками, находящимися посередине прямой. Полученные ломаные имеют много зве-

ньев, поэтому с помощью алгоритма Рамера — Дугласа — Пекера[2] аппроксимируем её до ломаной с меньшим числом точек (рис. 7).

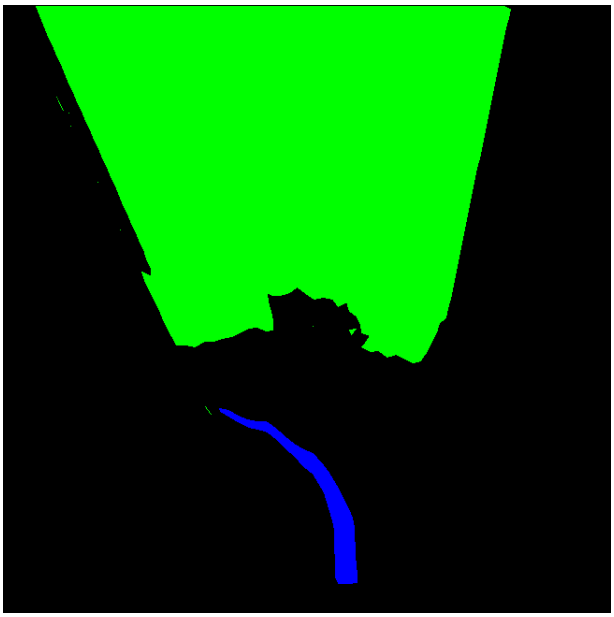

Рис. 6: Контуры после фильтрации. Синим отмечены контуры, относящиеся к линии, зеленым - отброшенные.

Данный этап необходим, так как по ломаным можно будет легко построить путь, а также они легко интерпретируются в команды для робота.

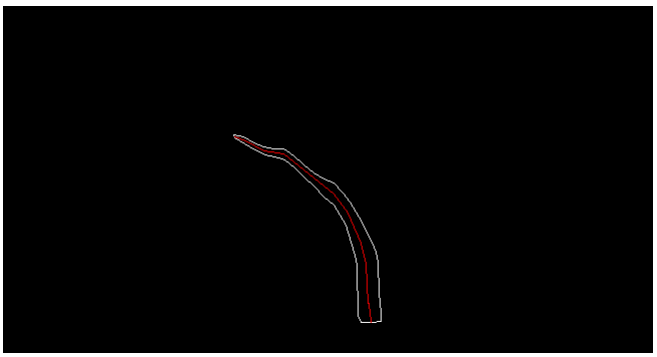

Рис. 7: Ломаная линия по центру контура.

3. Реализация

Первоначально алгоритм написан с помощью библиотеки компьютерного зрения OpenCV[1]. Она хорошо подходит для составления и тестирования алгоритма, так как в ней содержатся множество реализованных функций.

К сожалению, использование этой библиотеке в ТРИК для данной задачи не является возможным, поэтому следующем этапом работы стало переписывание выбранных функций без OpenCV.

Финальным этапом было внедрение в ТРИК. Типичной архитектурой для обработки видео является передача видеопотока на процессор DSP, специализированный для обработки цифровых сигналов. затем обработанные результаты передаются на ARM, процессор общего назначения. Так как гомография и фильтрация являются самыми вычислительно сложными задачами, было принято решение реализовать первые три этапа алгоритма на DSP, а остальные на ARM. Перед внедрением я решил оценить скорость алгоритма, воспользовавшись готовым проектом. Выяснилось, что DSP в лучшем случае обрабатывал бы не более двух кадров в секунду. Такая скорость неприемлема для робота, который двигается быстро, так как он может пропустит часть линии. В итоге я принял решение завершить работу на данном этапе.

Заключение

В рамках курсовой работы был придуман и реализован алгоритм распознавания линии. Тестирование проводилось по ходу составления алгоритма на основе собранных данных. Несмотря на допущения и упрощения, использованные в ходе работы, требуется дополнительные оптимизации по скорости работы алгоритма.

Список литературы

[1] OpenCV. Collection of algorithms for image processing. — URL: http://docs.opencv.org/3.0.0/.

[2] Ramer Urs. An iterative procedure for the polygonal approximation of plane curves. — 1972.

[3] Szeliski Richard. Computer Vision: Algorithms and Applications. — 2010.

[4] ТРИК Робототехнический контроллер. — URL: http://www.trikset.com/ (дата обращения: 23.05.2016).