УДК 517.977.56

СИСТЕМА АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ НАСТОЛЬНОЙ ИГРОЙ "ЛАБИРИНТ"

Пронькин В.Г.
Донецкий национальный технический университет
кафедра Компьютерной инженерии
E-mail: emberkorvin@rambler.ru

Аннотация

Пронькин В.Г. Система автоматического управления настольной игрой "Лабиринт". *Разработана и оптимизирована система автоматического контроля на основе игры-лабиринта "Kugellabyrinth". Разработаны алгоритмы анализа и распознания получаемых от веб-камеры изображений поверхности лабиринта, алгоритмы нахождения кратчайшего пути для последующего движения шарика.*

Введение

Одной из важнейших теорий при проектировании систем управления является теория автоматического контроля. Контролируемыми объектами могут быть различные устройства: от простых автоматических бытовых приборов до сложных военных разработок. Многие университеты занимаются изучением теорий контроля. Одним из таких университетов является университет Штутгарта, где автор статьи проходил практику.

Было предложено разработать и улучшить алгоритмы имеющейся в лаборатории модели игры-лабиринта, которая является прообразом более технологической системы, обеспечивающей стабильность положения платформы мобильной установки. Кроме этого данная разработка должна дать и улучшить знания по теории автоматического контроля, по получению изображений с веб-камеры, быстрой их обработке, распознаванию и анализу, а также нахождению кратчайшего пути между точками, управлению сервоприводами через сервоконтроллер, по работе с USB- и COM-портами.

1. Изучение конструкции и принципов работы лабиринта

В основе конструкции лежит настольная игра – лабиринт ("Kugellabyrinth"). Игроку предлагается при помощи двух ручек управлять наклоном поверхности лабиринта с целью привести шарик из точки старта к финишу. При этом шарик не должен упасть в яму. Даже имея достаточный опыт игры, не каждый сможет быстро и без ошибок пройти лабиринт. Поэтому предложен вариант модификации лабиринта для автоматического контроля шара компьютерной системой.

Для этого к плоскости лабиринта присоединяются «руки» компьютера – два сервомотора, которые позволяют отклонять ее в разные стороны на заданные углы. Управление моторами осуществляется через сервоконтроллер, который преобразует полученные через USB-порт команды от компьютера в различные уровни тока/напряжения для двух сервомоторов. Возможности сервоконтроллера позволяют присоединить до 6 сервомоторов, но для управления лабиринтом достаточно лишь двух, которые отклоняют поверхность в двух перпендикулярных друг другу плоскостях.

Для слежения за текущим положением шара, анализом конфигурации лабиринта к конструкции добавляется веб-камера – «глаза» компьютера. Она устанавливается над поверхностью устройства на штанге. Веб-камера позволяет с частотой 30 Гц (т.е. 30 раз в секунду, один раз за 33мс) получать изображение лабиринта в компьютер (рис. 1, 2).

Рисунок 1 – Автоматическая система лабиринта

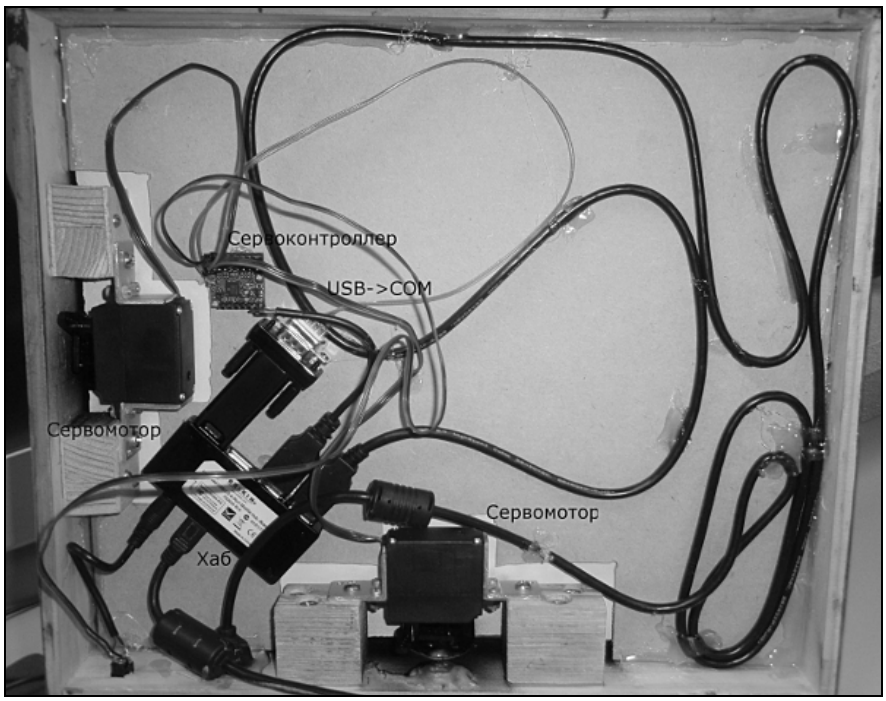

Рисунок 2 – Нижняя часть лабиринта

Таким образом, реализуется цикл системы управления:

1. Получение изображения (веб-камера).
2. Анализ изображения (компьютер).
3. Реагирование на ситуацию (компьютер).
4. Управление плоскостью лабиринта (сервомоторы и контроллер).

5. Перемещение шарика (физические процессы на объекте) и переход к п.1.

Игра заканчивается, когда шарик попадает к заданной конечной точке. Получение изображения от веб-камеры осуществляется в цикле с частотой примерно 30 Гц при помощи соответствующих библиотек языка программирования Java.

2. Анализ получаемого изображения лабиринта от веб-камеры

Стены лабиринта имеют средние по яркости значения – обычно они не такие яркие, как шар, и не такие темные как отверстия или свободное пространство. Отверстия (ямы) являются самыми темными объектами на получаемом от веб-камеры изображении. Таким образом, правильное отсечение слишком темных отверстий и слишком ярких стен и шарика дает изображение свободного пространства, где и будет проходить поиск маршрута к цели. Изображение лабиринта представлено на рис. 3.

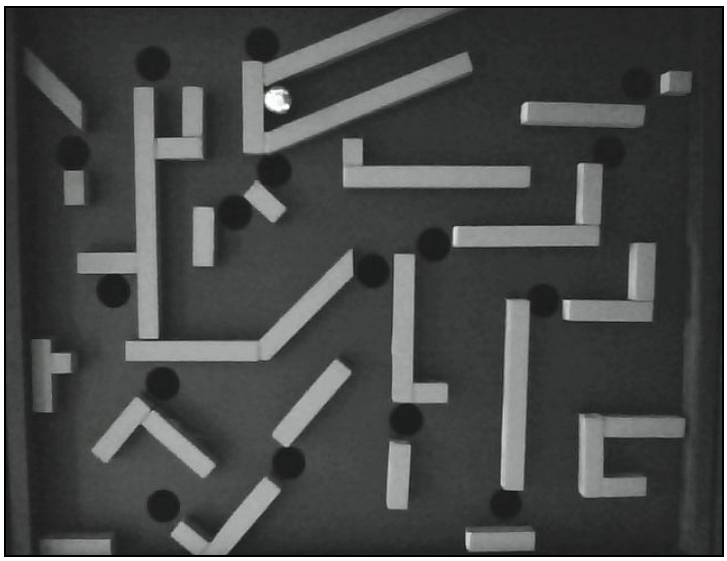

Рисунок 3 – Изображение лабиринта от веб-камеры

Для разграничения уровней яркости необходимо построить гистограмму яркостей изображения (рис. 4) – т.е. подсчет количества пикселей каждого уровня яркости. В анализе гистограммы не участвуют крайние «нулевые» уровни яркости, т.е. уровни, которые не присутствуют на изображении.

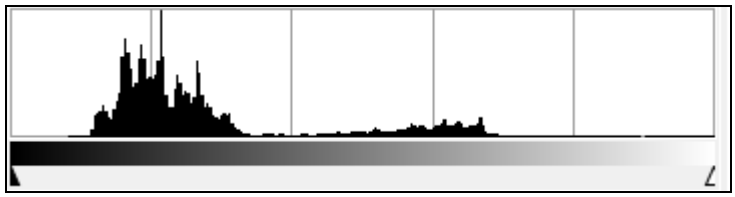

Рисунок 4 – Гистограмма уровней яркости изображения

Далее предлагается искать максимум в первой половине гистограммы, т.е. максимально «темный» уровень. Этот уровень, а также прилегающие к нему – это свободное пространство лабиринта и отверстия. Более яркие – это стены и шар. Т.е. первое разграничение берется после максимально темного уровня, когда значение на гистограмме падает почти до нуля. Второе разграничение должно отделять свободную поверхность от отверстий. Оно будет проходить после первого локального максимума на гистограмме уровней. Таким образом, если на изображении взять все уровни серого, лежащие между двумя ограничителями, то получаем свободное пространство лабиринта, в котором далее можно искать путь для движения шарика (рис. 5).

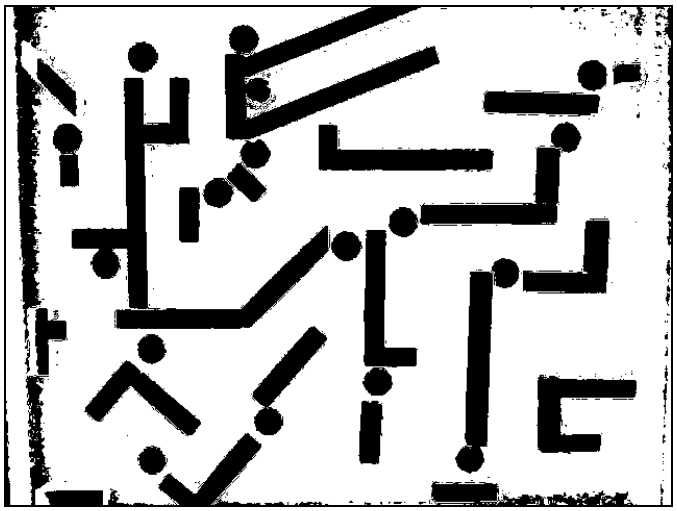

Рисунок 5 – Изображение после применения границ

3. Нахождение кратчайшего пути между начальной и конечной отметками

При нахождении пути необходимо учитывать, что шарик имеет определенный диаметр, а потому не может пройти сквозь узкое место. Изображение при анализе подвергается масштабированию. Т.е. массив, по которому ищут путь, значительно меньше самого изображения. Изображение разбивается на квадраты, и каждый квадрат пикселей представляет собой один элемент конечного массива. Если в квадрате больше занятых пикселей, то и элемент считается занятым. Таким образом затраты ресурсов на вычисление пути будут снижены без значительной потери качества, так как шаг масштабирования меньше радиуса шарика. Выбран шаг «8 точек».

Далее идет поиск пути по массиву. Поиск осуществляется волновым алгоритмом. Алгоритм заключается в следующем:

1. Из точки старта начинает распространяться волна чисел:
    a. Начальная точка добавляется в текущий фронт волны.
    b. Для всех точек текущего фронта идет поиск прилегающих свободных не проанализированных ранее точек (элементов массива).
    c. Если найдена конечная точка, то п.2a.
    d. Иначе каждая найденная точка добавляется в новый фронт волны.
    e. Также каждой точке присваивается порядковый номер волны (итерации).
    f. Если во фронте новой волны есть элементы, то их переписывают во фронт текущей волны, счетчик итераций увеличивают на один и переходят в п. 1b.
    g. В случае отсутствия элементов во фронте считается, что финиш не найден, то есть путь не существует.
2. На данном этапе находится непосредственно кратчайший путь:
    a. Текущей точкой устанавливается точка финиша.
    b. Вокруг текущей точки ищется точка с номером на единицу меньшим текущей.
    c. Эта точка устанавливается текущей и запоминается в массив пути.
    d. Проверяется, достигнута ли точка старта. Если да, то выход из алгоритма.
    e. Иначе идет переход к пункту 2b.

В результате будет сформирован массив пути. Так как желательно уменьшить число точек пути, а также увеличить длину пробега шарика между точками, то при обратном проходе (п.2) алгоритм в первую очередь анализирует точку по ходу движения, т.е. с координатами противоположными предыдущей точке. В массив пути заносятся только точки, где направление движения было изменено. Результаты работы изображены на рис. 6.

Рисунок 6 – Результаты поиска пути

На рисунке можно видеть квадраты после масштабирования, различными уровнями яркости обозначены номера фронтов волн при поиске пути. Сам путь отмечен "средним" серым цветом и квадратами с номерами порядкового номера элемента пути. Черным цветом показаны преграды, включающие в себя также отверстия. Итак, путь через лабиринт найден.

4. Управление движением шарика по маршруту

В каждой итерации цикла управления автоматическим движением шарика система получает текущее положение шарика и определяет, в какой точке пути на текущий момент он находится. Таким образом, программа всегда знает номера текущей и номера следующей позиции шарика, т.е. позиции, куда необходимо "доставить" шар.

Управление моторами реализовано при помощи соответствующих библиотек Java. Реализованы такие команды:

– начальный сброс – приведение плоскости в начальное состояние;
– поворот 1го мотора на заданный угол;
– поворот 2го мотора на заданный угол.

Заключение

Итак, в результате была разработана и оптимизирована система автоматического управления лабиринтом. Разработаны алгоритмы анализа и распознания получаемых от веб-камеры изображений поверхности лабиринта, алгоритмы нахождения кратчайшего пути и движения шарика.

Список литературы

1. Хорстман, Корнелл. Java 2. Библиотека профессионала, том 1. Основы, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 896 с.: ил. – Парал. тит. англ.

2. Зайцев Г.Ф. Теория автоматического управления и регулирования.– 2-е изд., перераб. и доп.– К.: Выща шк. Головное изд-во, 1989.– 431 с.