Санкт-Петербургский государственный университет

Математико-механический факультет

Кафедра системного программирования

Распознавание характеристик объектов в робофутболе

Курсовая работа студента 344 группы

Черняева Арсения Витальевича

Научный руководитель: А.А. Пименов

Санкт-Петербург

2015

Оглавление

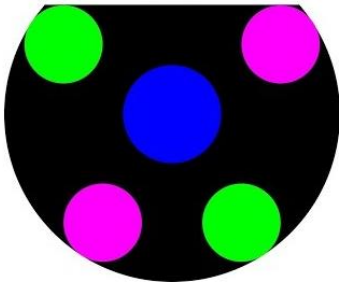
Введение

Робофутбол – это соревнование по футболу между двумя командами автономных роботов. Данная область популярна, т.к. предоставляет исследователям возможность реализовать искусственный интеллект, позволяющий набору независимых (от человека или даже друг от друга) объектов вместе решать поставленную задачу.

Существует большое количество различных лиг роботов-футболистов. Они могут отличаться размерами поля и моделей, типом управления (централизованное или децентрализованное), размерами команд. В каждом случае для каждого объекта, реализующего управление, возникает целый набор задач, которые необходимо решить: снятие данных о состоянии системы, их обработка, получение положений других объектов, принятие решения и отправление команд.

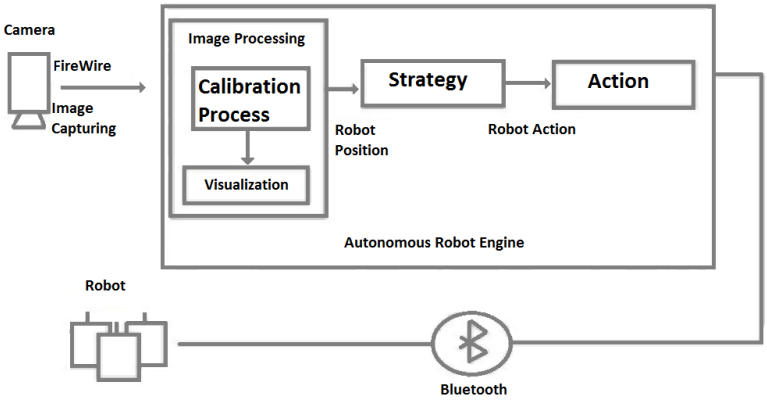
Одним из вариантов игры является Футбол Малой Лиги роботов – соревнование на поле небольшого размера команд футболистов, не превосходящих 18 сантиметров в диаметре и 15 сантиметров по высоте. Каждый робот имеет нанесенное на верхнюю поверхность изображение (см. Изображение 1). Центральный круг своим цветом (желтым или синим) кодирует номер команды, вокруг него на равном расстоянии расположены боковые круги, их цвета (красные или зеленые) позволяют идентифицировать номер робота.

Расположение кругов выбрано таким, чтобы угол, образуемый передними элементами и центральным кругом, был больше остальных. Это необходимо, чтобы на изображении, полученном с верхней камеры, было понятно, куда смотрит футболист.



Изображение 1

Над полем расположены две камеры, собирающие информацию о всех объектах и затем передающие ее на сервер. Сервер обрабатывает полученную информацию, получает характеристики объектов – координаты, направление движения, скорость и пр. – и передает их дальше каждой команде для выработки стратегии поведения. Схема процесса предоставлена на изображении 2.



Изображение 2

Процедура преобразования изображения в данные об объектах облагается множеством требований, в их числе:

- Скорость – данные должны обрабатываться достаточно быстро, чтобы обеспечивать своевременность всех действий роботов
- Точность – важно передавать точный набор данных, чтобы команды, выполняемые футболистами, были обоснованы
- Устойчивость – небольшое (возможно, случайное) изменение параметров окружения (например, изменение освещения) не должно влиять на корректность работы алгоритма

Постановка задачи

Целью данной курсовой работы является написание алгоритма распознавания характеристик объектов (положения и направления взгляда роботов). Процесс реализации должен включать в себя:

- Анализ существующего решения, выявление его преимуществ и недостатков
- Написание собственного алгоритма
- Его анализ, проверка на предмет требований, написание модификаций, улучшающих его характеристики
- Сравнение полученного и существующего решений

Все тестирования будут проводиться на компьютере с процессором Intel Core i3-2310M.

Анализ существующего решения

Кафедрой теоретической кибернетики Математико-механического факультета ранее была реализована система компьютерного зрения, позволявшая получать серию изображений футбольного поля из различных источников, обрабатывать их и показывать пользователю результат.

Основные этапы работы распознающего алгоритма следующие:

1. Производится размытие изображения с использованием ядра:

$$K = {}^{1}/_{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

2. Изображение переводится в черно-белое и затем бинаризуется с порогом бинаризации, который определяется на этапе калибровки

3. В полученном изображении выделяются кластеры, из их числа выделяются регионы, близкие к эллипсу.

   Выделение регионов производится за проход по изображению. Сперва выделяется линия – набор подряд идущих белых пикселей в одной строке. При анализе каждой линии рассматривается линия на предыдущей строке, смежная с данной. Регион рассматриваемой линии считается равным региону смежной. Если смежных линий несколько, то они все объединяются в один регион.

4. Подсчитывается среднее значение цвета в каждом из регионов, определяется настоящий цвет полученной точки.

5. Из полученного списка регионов выбираются регионы цвета, соответствующего цвету центрального круга. Вокруг них методом простого перебора ищутся 4 региона цвета, соответствующего цвету боковых кругов шапки робота.

6. Если получилось найти такой набор объектов, то полученное множество считается роботом. Направление взгляда робота определяется как средняя точка линии, соединяющей два самых удаленных друг от друга смежных боковых кругов. Номер команды и номер робота в команде однозначно определяются по цветам регионов.

Отдельно стоит рассмотреть следующие фазы обработки:

1. Калибровка. Она необходима для определения порога бинаризации, а также для составления списков (облаков) точек цветового пространства RGB, которые будут считаться принадлежащими к одному из основных цветов. Эти списки позволят в дальнейшем в любом кадре для каждого пикселя определять принадлежность его одному из вышеуказанных цветов.

Изображение 3

Для калибровки составляется специальная сцена из заранее известных объектов, расположенных в определенном порядке (см Изображение 3). Далее выбирается такой порог бинаризации, чтобы после него оставались все необходимые регионы, близкие к эллипсам, и никаких посторонних. Поскольку в каждой четверти находится ровно по одному роботу, то возможно по координатам центра региона определить, какому футболисту он принадлежит. Остается только взять цвета регионов и внести в соответствующие списки.

2. Определение настоящего цвета RGB-точки.

Пусть необходимо узнать, верно ли, что настоящий цвет точки – цвет X. После калибровки известен список точек RGB-пространства цвета X. Для ответа на поставленный вопрос составляется матрица ковариаций A (она характеризует разброс значений по каждой из осей RGB-пространства). Ее собственные вектора и собственные значения позволяют ограничить облако точек эллипсоидом (собственные вектора задают направления осей эллипсоида, а собственные значения – это длины полуосей). Отдельно вычисляется центр эллипсоида C – среднее значение точек из списка. Принадлежность цветовой точки R такому эллипсоиду можно вычислить по следующей формулы:

$$(R - C)^T A^{-1} (R - C) \leq const$$

Подробнее об обоснованности приведенных вычислений можно прочесть в [4].

Матрицы ковариации для каждого из цветов считаются при каждой калибровке, что позволяет экономить время при каждой попытке выполнить определение цвета.

Плюсы алгоритма:

- Высокая скорость обработки (~20 кадров в секунду)
- Стабильная работа и хорошая точность при неизменяемых параметрах среды
- Низкий шанс срабатывания на посторонние объекты – все лишние объекты с очень большой вероятностью не пройдут один из этапов алгоритма

Минусы алгоритма:

- Дефект региона (например, блик на круге шапки робота) ведет к его потере и, как следствие, потере всего робота
- Система неустойчива к изменениям среды (например, изменению освещения). В подобных случаях необходимо произвести повторную калибровку

Вывод:

Главная проблема алгоритма состоит в том, что он не удовлетворяет требованиям устойчивости. Это необходимо учесть при написании решения.

Описание алгоритма

В ходе исследовательской работы был реализован алгоритм, базирующийся на идее голосования за центр робота каждой точки, близкой по цвету к цвету кругов шапки робота. Этапы работы алгоритма следующие:

1. Производится размытие изображения с ядром $K = {}^{1}/_{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$

2. Определяется настоящий цвет каждой точки (или определяется, что цветовая точка не принадлежит никакому цвету).

3. Создается структура данных, позволяющая хранить результаты голосования за каждый пиксель как за центр робота.

4. Каждый пиксель изображения, цвет которого определен, как один из настоящих цветов, голосует за все пиксели, находящиеся от него не дальше определенного расстояния D (данный подход родственен алгоритму Хафа). Величина D определяются следующим образом:

   - Для пикселей, настоящий цвет которых соответствует цвету бокового круга шапки робота (зеленый или красный) D представляет собой величину, равную расстоянию от центра робота до дальней точки каждого бокового круга, измеренную в пикселях и вычисляется по следующей формуле:

     $$D = C_1/h$$

     Здесь $C_1 = (\frac{f}{c})(R + r_б)$, где R – расстояние от центра центрального круга до центров боковых кругов, $r_б$ – радиус бокового круга, f – фокусное расстояние камеры, h – высота камеры над полем, c – размер пикселя. Величина C_1 в условиях поставленной задачи считается постоянной.

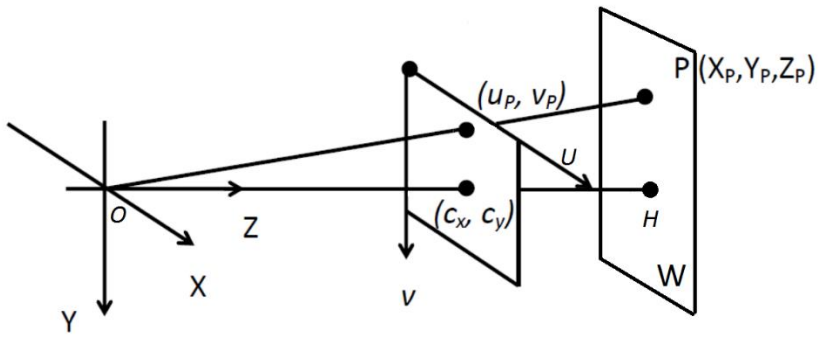
   - Для пикселей, настоящий цвет которых соответствует цвету центрального круга шапки робота (синий или желтый) или цвету мяча (оранжевый) D представляет собой величину, равную расстоянию от центра робота до границы центрального круга, измеренную в пикселях, и вычисляется по следующей формуле:

     $$D = C_2/h$$

     Здесь $C_2 = (\frac{f}{c})r_ц$, $r_ц$ – радиус центрального круга. Величина C_2 в условиях поставленной задачи считается постоянной.

В обоих случаях D подбирается таким, чтобы в область голосования каждой точки попадал центр робота.

Эти формулы следуют из простых признаков подобия. На Изображении 4 представлена проективная модель камеры.



Изображение 4

Сцена W должна быть отображена на проективную плоскость, образованную векторами U и V.

$P(X_p, Y_p, Z_p)$ – точка в пространстве, (U_p, V_p) – ее образ на изображении. $R = |PH|$, $r = |(U_p - C_x, V_p - C_y)|$, $h = |OH|$, фокусное расстояние $f = |OC|$.

В таком случае, если известно R, то искомая величина $r = f\frac{R}{h}$

5. Находится максимальное значение, хранящееся в структуре для голосования. Далее любой пиксель, значение структуры для которого больше определенного процента T от найденной величины (выбрано значение $T = 80\%$) выделяется, из данных точек составляются кластеры, центры которых принимаются за координаты центров роботов.

6. Чтобы найти угол поворота робота, ищутся все точки цвета, соответствующего цвету бокового круга, находящиеся на расстоянии, не превосходящем величины $R_2 = (\frac{f}{c})R/h$, каждая точка голосует за угол центра бокового круга которому она принадлежит, далее на интервале $[0; 2\pi]$ выбираются 4 области, в которых достигается максимальное значение. Центры этих областей будут считаться углами, образуемыми вертикалью и центром бокового круга. В качестве угла поворота робота выбирается центр максимальной по длине дуги окружности между центрами этих областей.

   Величина R_2 выбирается таким образом, чтобы было учтено достаточно много точек из боковых кругов, и при этом исключалось влияние точек, принадлежащих шапкам других роботов.

7. Чтобы найти номер команды робота, рассматривается цвет его центра.

8. Чтобы найти номер робота внутри команды, необходимо определить, какие цвета окружают центральный круг (при этом важен порядок). Для того, чтобы определить это, рассматриваются все точки на расстоянии от центра робота, не превосходящем величины R_2. Выделяются 4 группы цветов, по группе в каждой из четвертей окружности робота, начиная от направления его движения. Значение цветов точек в этих группах усредняется, тем самым определяются цвета всех кругов вокруг центрального фрагмента, с учетом их порядка.

Калибровка: заключается в указании списков представителей настоящих цветов. Для этого была реализована функциональность, позволяющая на этапе калибровки указывать точки на изображении, цвета которых затем заносились в соответствующие списки.

Цветовая метрика: в качестве начальной модели, использовалась стандартная метрика в пространстве, RGB-точка считается точкой определенного цвета, если ее расстояние до одной из точек облака этого цвета меньше некоторого порога, определяющегося заранее.

В ходе тестирования были выявлены следующие преимущества и недостатки:

Преимущества:

- Более гибкая калибровка, нет необходимости в составлении специальной сцены
- Небольшие дефекты на изображении не влияют на работу алгоритма

  Возможно определение положения объекта в ситуациях, когда один из его регионов перестал определяться (или ввиду различных явлений перестал быть эллиптическим). Однако возможны проблемы на этапе определения угла поворота, а также номера робота в команде.

Недостатки:

- Скорость работы

  Скорость алгоритма оказалась слишком низкой (~1 кадр в секунду), в первую очередь из-за большого количества дорогостоящих операций определения цвета, которые линейно зависят от количества элементов в облаке точек

- Определение угла поворота робота может определяться неправильно из-за небольшого количества шумов, добавляющих новые деления областей между боковыми кругами.

  Это важная проблема, поскольку неправильное определение угла поворота влечет неправильное определение робота в команде.

Было принято решение проводить модификации алгоритма в целях, в первую очередь, увеличения скорости работы, чтобы затем вводить дополнительную функциональность, решающую остальные проблемы.

Модификации

Параллелизация

Первый подход, позволяющий ускорить обработку изображений, заключается в распределении нагрузки между несколькими потоками. Поскольку в ходе алгоритма часто происходит независимая (или почти независимая) обработка пикселей, то было принято решение использовать openMP для параллелизации вычислений в циклах.

При голосовании точек такой грубый подход уже не подходит, т.к. при параллельной обработке нескольких пикселей возможен одновременный доступ к одним и тем же данным для голосования. Для ускорения работы в таком случае изображение разделяется на полосы шириной D (берется максимальное из двух D). Одновременное голосование за пиксель двух точек из двух разных полос, номера которых отличающихся на число, кратное 3, невозможно, в таком случае расстояние между этими двумя полосами было бы не больше 2D. Следовательно работа распределяется следующим образом:

1. Последовательно обрабатываются наборы полос, имеющих одинаковый остаток при делении на 3.
2. Обработка таких полос распараллеливается
3. Внутри полосы пиксели обрабатываются последовательно

Данный подход позволяет довольно сильно увеличить скорость работы алгоритма. Однако стоит учитывать, что результаты будут сильно зависеть от машины, на которой происходит тестирование.

Изменение метрики

Как уже говорилось ранее, изначально выбранная метрика обладала существенными недостатками. Подсчет расстояния в трехмерном пространстве занимает довольно много времени, а работу усложняет то, что облако точек (или иначе список представителей цвета) может быть сколь угодно большим, и его увеличение влечет увеличение времени определения цвета.

Первый подход в этой области заключается в изменении метрики на метрику, ранее предложенную в алгоритме кафедры теоретической кибернетики. Суть ее заключается в том, что облако точек окружается эллипсоидом, и утверждается, что точка является точкой данного цвета, если ее цветовые компоненты лежат внутри эллипсоида. Проверка принадлежности точки эллипсоиду описана в главе «Анализ существующего решения».

Данный подход был реализован, для вычисления матриц ковариации использовались алгоритмы библиотеки openCV. Однако тестирование показало, что метрика плохо подходит в случаях частых обращений к операции определения принадлежности эллипсоиду. Основная проблема в таком случае – частое обращение к функциям из внешней библиотеки. Было принято решение перейти к следующему решению.

Второй подход заключается в ограничении облака точек параллелепипедом. Самое простое решение будет включать 6 проверок вхождения в интервал, по две на каждую цветовую составляющую. Однако этот результат можно улучшить.

Новый алгоритм основывается на статье [1], его основная идея заключается в следующем: для каждого цвета для каждой оси заводится массив логических переменных (обозначим их inColorR, inColorG, inColorB, в конкретном случае из 256 элементов. Значение в ячейке x, равное True, означает, что по данной оси значение цвета удовлетворяет вхождению в параллелепипед.

Пример заполнения массивов (если каждая компонента цвета состоит из 10 значений):

Пусть точки (1, 4, 2) и (3, 3, 6) составляют цветовое облако. В таком случае параллелепипед со сторонами 1-3, 3-4 и 2-6 ограничивает наше облако. Тогда массивы выглядят следующим образом:

inColorR[10] = {0, 1, 1, 1, 0, 0, 0, 0, 0, 0};

inColorG[10] = {0, 0, 0, 1, 1, 0, 0, 0, 0, 0};

inColorB[10] = {0, 0, 1, 1, 1, 1, 1, 0, 0, 0};

Теперь, чтобы проверить, что точка (r,g,b) лежит внутри параллелепипеда, достаточно проверить истинность следующего выражения:

inColorR[r] & inColorG[g] & inColorB[b]

Преимущество данного подхода заключается в том, что для проверки принадлежности точки цвету необходимо выполнить лишь 2 логических операции (в эллиптической метрике для этого необходимо выполнить более 20 операций сложения и умножения).

Недостаток заключается в том, что параллелепипед будет включать в себя довольно большую область пространства RGB, поэтому следует внимательнее относиться к выбору точек для списка цветов.

После проведения первых двух модификаций общая скорость алгоритма возросла до минимальных требований, которые ставились перед ним (скорость, соизмеримая со скоростью старого решения). В дополнение к этому была разработана еще одна модификация, добавляющая важную функциональность.

Поиск объекта в области его предыдущего местоположения

Если предположить, что перемещение робота является плавным, а кадры считываются и обрабатываются довольно часто, то смещение робота на соседних кадрах будет достаточно небольшим (при максимальной скорости объекта 20 см/с и скорости обработки 20 кадров в секунду получается, что смещение робота за кадр составляет не более одного сантиметра, что почти не заметно на изображении). Это дает основание предполагать, что следующее положение робота следует искать в небольшом регионе вокруг предыдущей точки его расположения.

В ходе исследовательской работы был реализован алгоритм, позволяющий по набору положений роботов определять и пересчитывать их координаты. Его суть проста: для каждого робота рассматривается область размерами превышающая его размеры на небольшую величину. К этой области применяется ранее описанный алгоритм, с несколькими отличиями: поскольку уже известно, что робот в области будет один (больше одного не поместится), и уже известен его номер, то необходимо искать лишь новое положение и направление лицевой части, что ускоряет работу.

На базе этой модификации можно добавить еще несколько, влияющих на устойчивость: ввести для каждой области свой список цветов, а также пересчитывать его в каждом кадре в случае изменения среднего значения цвета (что позволяет не прекращать работу из-за потери роботов в случае изменения освещения одной из частей поля). Исследовательская работа в этой области еще ведется.

Преимущества (помимо преимуществ, перечисленных в базовом алгоритме):

- Высокая скорость работы (с учетом предыдущих модификаций достигается значение 25-30 кадров в секунду)
- Нет необходимости в вычислении номера робота.

  Таким образом уменьшена зависимость между этапами работы алгоритма, уменьшаются последствия ошибок определения угла (система все равно получит корректные координаты и идентификационные данные робота)
- Возможность более гибкой настройки для каждого робота.

  Данное преимущество заключается в том, что для каждого робота можно хранить свой список цветов, который можно будет пересчитывать в случае смещения цветового пространства (например, при постепенном затемнении одной из частей поля). Таким образом уменьшается вероятность необходимости повторной калибровки

Недостатки:

- Усложненный этап калибровки, необходимо указывать начальные положения роботов
- Система работает, пока движения роботов непрерывны. После снятия робота с поля, например, по техническим причинам, нужна повторная калибровка. Также стоит отметить, что данный алгоритм не допускает появления новых объектов на поле без перекалибровки

Чтобы уменьшить влияние первого недостатка, была проведена последняя модификация.

Встраивание изначального алгоритма в этап калибровки

В целях упрощения этапа калибровки для метода локальных областей было принято решение автоматизировать определение центров областей, где необходимо искать роботов.

После определения цветов запускается базовый алгоритм. Его задача – находить все центры и отображать на экран. Если принято решение, что данный набор центров подходит для дальнейшего этапа, запускается модифицированная версия.

Таким образом, этап калибровки упрощается. Однако второй недостаток предыдущей модификации сохранен, система требует повторной калибровки при удалении или добавлении робота.

Заключение

Был реализован алгоритм, позволяющий определять по изображению футбольного поля координаты и углы поворотов объектов, находящихся на нем, а также проведены модификации к нему. Было проведено сравнение с предыдущим решением, которое показало:

- Скорость:

  В сравнении с изначальным алгоритмом решение обладает большей скоростью работы (~25 кадров в секунду против ~20 кадров в секунду у предыдущего решения).

- Устойчивость:

  Алгоритм позволяет определять характеристики роботов даже в случае дефекта одного из боковых регионов. Однако цветовые шумы вблизи футболиста могут негативно сказываться на определении угла поворота

- Точность:

  Для сравнения точности алгоритмов был проведен ряд измерений. Каждый раз вычислялась разница между центром робота и полученным результатом, измеренная в координатах пиксельной сетки. Результаты отображены на графиках (График 1 для старого алгоритма, График 2 для нового алгоритма).

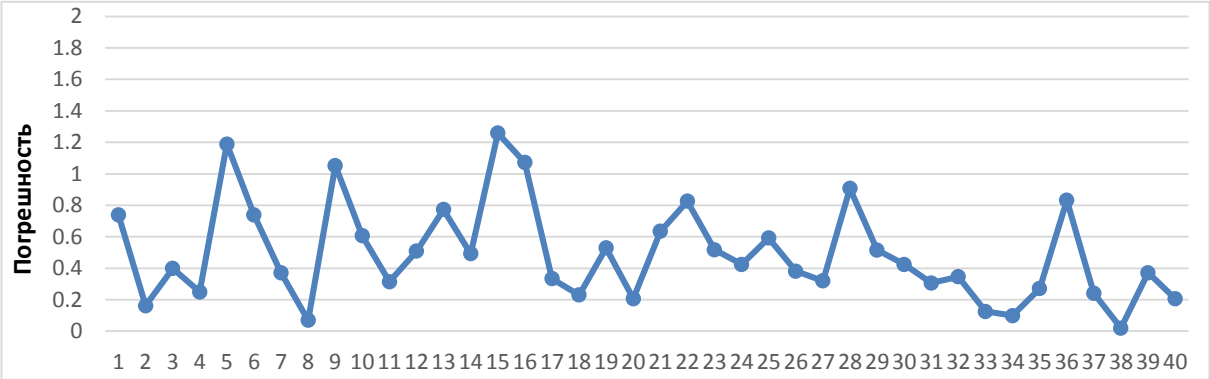


График 1

Математическое ожидание ошибки в ряде измерений старого алгоритма – 0.49, среднеквадратичное отклонение – 0.31

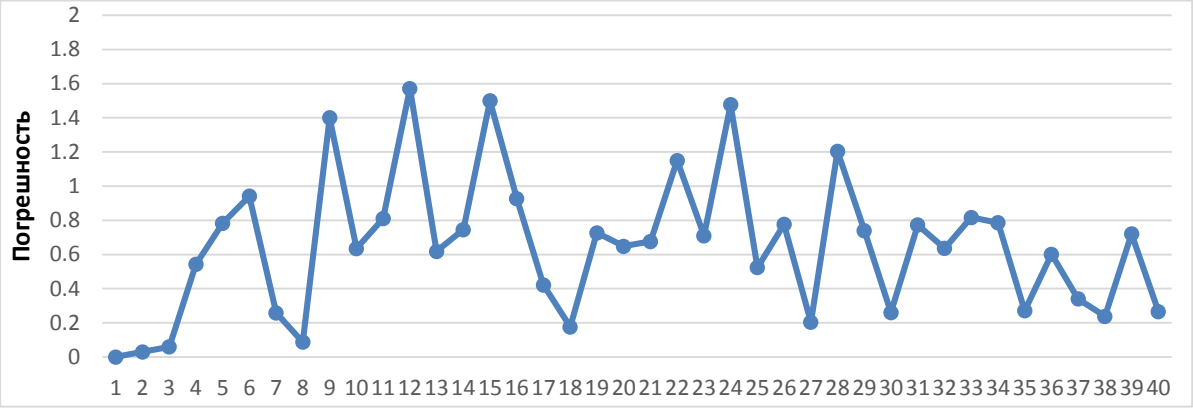


График 2

Математическое ожидание ошибки в ряде измерений нового алгоритма – 0.61, среднеквадратичное отклонение – 0.38.

Из этого можно сделать вывод, что точности обоих алгоритмов довольно высоки, средняя погрешность составляет не более 1 пикселя. Средняя ошибка старого решения чуть ниже, что объясняется отсутствием влияния боковых кругов на определение координат робота.

Однако определение угла, как уже говорилось выше, требует последующих модификаций, основное направление которых – избавление от ложных точек, относящихся к определенному цвету.

- Калибровка:

  Изменена, позволяет не начинать работу с калибровочной сцены каждый раз при изменении свойств среды

Дальнейшие направления работы

- Улучшение алгоритмов определения угла поворота робота

  Первый шаг в этом направлении - убирание случайных небольших цветовых помех (точек, которые случайно были определены как принадлежащие зеленому или красному цвету)

- Модернизация алгоритма локальных окрестностей, позволяющая не проводить повторную калибровку при добавлении нового робота

  Идея заключается в том, чтобы каждый раз запускать базовый алгоритм, определяющий положения всех объектов. Если обнаружены новые объекты, то надо добавить их в список рассматриваемых областей

- Модернизация алгоритма локальных окрестностей, позволяющая не проводить повторную калибровку при удалении робота

  Идея заключается в том, чтобы, если в области слишком мало точек проголосовало за центр робота, то следует больше не рассматривать эту область

Список литературы

[1] Bruce J., Balch T., and Veloso M. Fast and Inexpensive Color Image Segmentation for Interactive Robots // Intelligent Robots and Systems, 2000.

[2] Gerber R. Getting Started with OpenMP // Intel Developer Zone, 2012, URL: https://software.intel.com/en-us/articles/getting-started-with-openmp (дата обращения: 2 Апреля 2015).

[3] OpenCV Documentation, URL: http://docs.opencv.org/ (дата обращения: 10 Февраля 2015).

[4] Jolliffe I.T. Principal Component Analysis, Second ed. // Springer, 2002.