Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Гуданова Варвара Сергеевна

Система распознавания меток игроков в робофутболе

Курсовая работа

Научный руководитель:
ст. преп. А. А. Пименов

Санкт-Петербург
2020

Содержание

Введение

В настоящее время все большую популярность приобретают соревнования, в которых в качестве игроков принимают участия роботы. Одно из самых известных подобных мероприятий – международные соревнования RoboCup [1]. Организаторы позиционируют RoboCup как некоммерческий научный проект по продвижению искусственного интеллекта, робототехники и других связанных областей науки.

В рамках данных соревнований проводятся состязания различных форматов, однако можно выделить самый популярный – робофутбол. В качестве участников робофутбола выступают две команды, у каждой по 6 роботов. Игра проходит по правилам, схожим с правилами обычного футбола [2].

Соревнования по робофутболу в рамках RoboCup проводятся регулярно, однако у данного мероприятия есть свои недостатки:

1. Стоимость роботов очень высока – цена за одного участника может достигать 1000$ и выше. Это делает соревнования недоступными для многих команд, которые хотели бы принять участие.

2. Высокий порог вхождения – в настоящий момент программировать роботов-футболистов может быть сложной задачей для детей младшего школьного возраста. Данную проблему можно решить, например, с помощью интеграции визуального языка программирования.

Существование указанных проблем у RoboCup является мотивацией для создания инженерной инфраструктуры для организации соревнований по робофутболу под руководством студентов и преподавателей СПБГУ с более доступными по цене роботами (далее – 7M роботы) с целью предоставить возможность участия для более широкой аудитории.

1. Описание предметной области

Важной частью робота-футболиста является шапочка с уникальными метками, благодаря которой система распознавания может определить игрока. Один из вариантов шапочек представлен на рис. 1. Цвет центральной метки определяет принадлежность к одной из двух команд (согласно стандарту данная метка может быть желтого или синего цвета), остальные задают номер конкретного робота внутри команды.

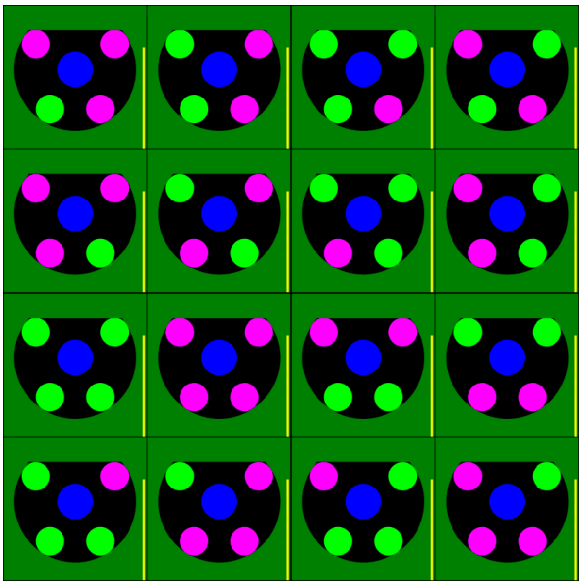


Рис. 1. Шапочки роботов

При рассмотрении игры в реальном мире (без использования симулятора) можно составить следующую схему организации соревнований по робофутболу (рис. 2). Игра проходит на поле зеленого цвета. Камера (или несколько камер) расположена над полем, информация с камеры в режиме реального времени поступает в систему распознавания (`«vision system»`) для обработки. Система распознавания определяет местоположение роботов и мяча, полученные данные передаются в приложение по управлению роботами (`«arbiter»`), которое в свою очередь взаимодействует с программами, написанными участвующими командами (`«team 1»` и `«team 2»`). На основании полученной информации формируются и передаются по сети Wi-Fi новые инструкции для

роботов.

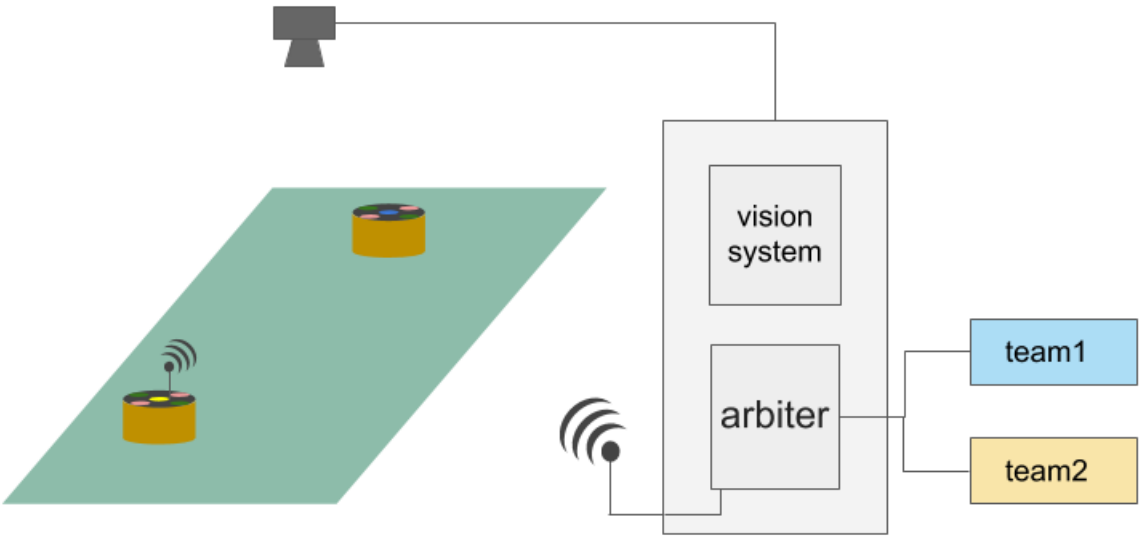


Рис. 2. Организация игры по робофутболу

Аналогичным образом организуется игра с использованием симулятора. Система распознавания принимает видеопоток напрямую из симулятора, сформированные инструкции от алгоритмов участвующих команд передаются обратно.

2. Постановка задачи

Одним из основных модулей инфраструктуры робофутбола является система распознования роботов-игроков, от которой требуется:

1. Распознать метки игроков.
2. Определить местоположение каждого игрока команды, их векторы скорости и местоположение мяча.
3. Отправить данные командам для дальнейшей обработки.
4. Иметь возможность анализировать видео с камеры, из симулятора и работать с уже отснятым материалом.

На основании этого формируется цель данного проекта – интеграция в робофутбол СПБГУ системы распознавания меток роботов-игроков.

Для достижения цели необходимо выполнить следующие задачи:

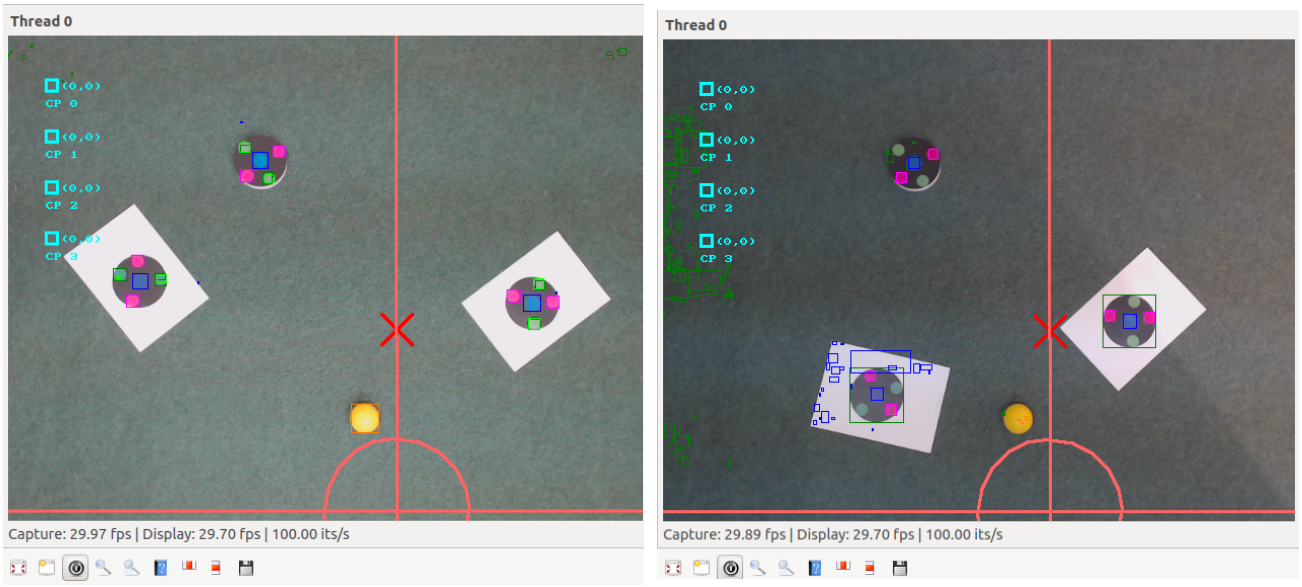
1. Провести анализ уже существующих решений, если такие имеются, оценить качество их работы, проверить на соответствие поставленным требованиям.
2. Разработать свою систему с нуля, либо спроектировать план модификации имеющейся и реализовать его.
3. Провести тестирование добавленных фунциональных возможностей.

3. Обзор

3.1. SSL-vision

В настоящий момент на соревнованиях RoboCup для всех команд предоставлена возможность использовать систему распознавания под названием SSL-vision [3] [4]. Решение о создании единой системы было принято в 2009 году, в настоящий момент SSL-vision разрабатывается добровольцами из участвующих в соревнованиях команд.

Данное решение имеет важную проблему, которая может негативно влиять на проведение соревнований – система зависима от освещения. Этот недостаток удалось обнаружить в процессе апробации. Добавление нового источника света в процессе работы программы негативно отразилось на распознавании меток роботов, результаты стали некорректными (рис. 3).



(а) (б)

Рис. 3. Зависимость корректности распознавания меток от освещения.

На изображении (а) запечатлена первоначальная ситуация – метки распознаются корректно, в соответствии с ожиданием. На изображении (б) показана ситуация уже после добавления нового источника света. Ожидалось увидеть

аналогичные с (а) результаты работы, однако зеленые метки на шапочках роботов и мяч оранжевого цвета не определяются корректно.

Также в ходе тестирования было обнаружено, что даже если одна метка робота распознается некорректно, то в результате данный робот не будет определен.

Кроме того, SSL-vision может работать с камерами указанных разработчиками типов [5] и с готовыми изображениями, но не существует возможности принимать видеопоток напрямую из симулятора для обработки.

3.2. Вывод

Решение, существующее в данный момент, обладает критическими недостатками, а также не удовлетворяет всем требованиям к системе распознавания, сформулированным ранее.

4. Описание решения

На основании апробации, результаты которой приведены в разделе «Обзор», составлен план модификации системы распознавания SSL-vision:

1. Изучить архитектуру решения, выявить модули захвата и анализа видео.
2. Подготовить SSL-vision к работе с роботами 7M.
3. Провести интеграцию модуля захвата с симулятором робофутбола, который разрабатывается на основе симулятора Webots [6].
4. Выбрать способ решения проблем робастности.

4.1. Архитектура SSL-vision

Для возможности модификации SSL-vision было необходимо изучить архитектуру решения.

Работу системы можно представить с помощью рис. 4. Обработка происходит с учетом заранее выбранных пользователем настроек. Задача параметров происходит с помощью пользовательского интерфейса, в котором в том числе можно посмотреть и изменить дерево конфигурации (`Configuration Tree Visualization/Editing`). Имеется возможность задавать размеры поля и роботов, количество камер, определять устройства захвата для каждой камеры, калибровать камеру, задавать таблицу соотвествия цветов (LUT) и некоторые другие параметры.

Обработка изображений происходит в стеке нескольких камер (`Multi-Camera Stack`). Он определяет количество используемых в работе камер и способы обработки и захвата видео (`Global Configuration`).

Работа с каждой камерой происходит в отдельном потоке, что обеспечивает независимость и параллельное выполнение обработки видео с разных

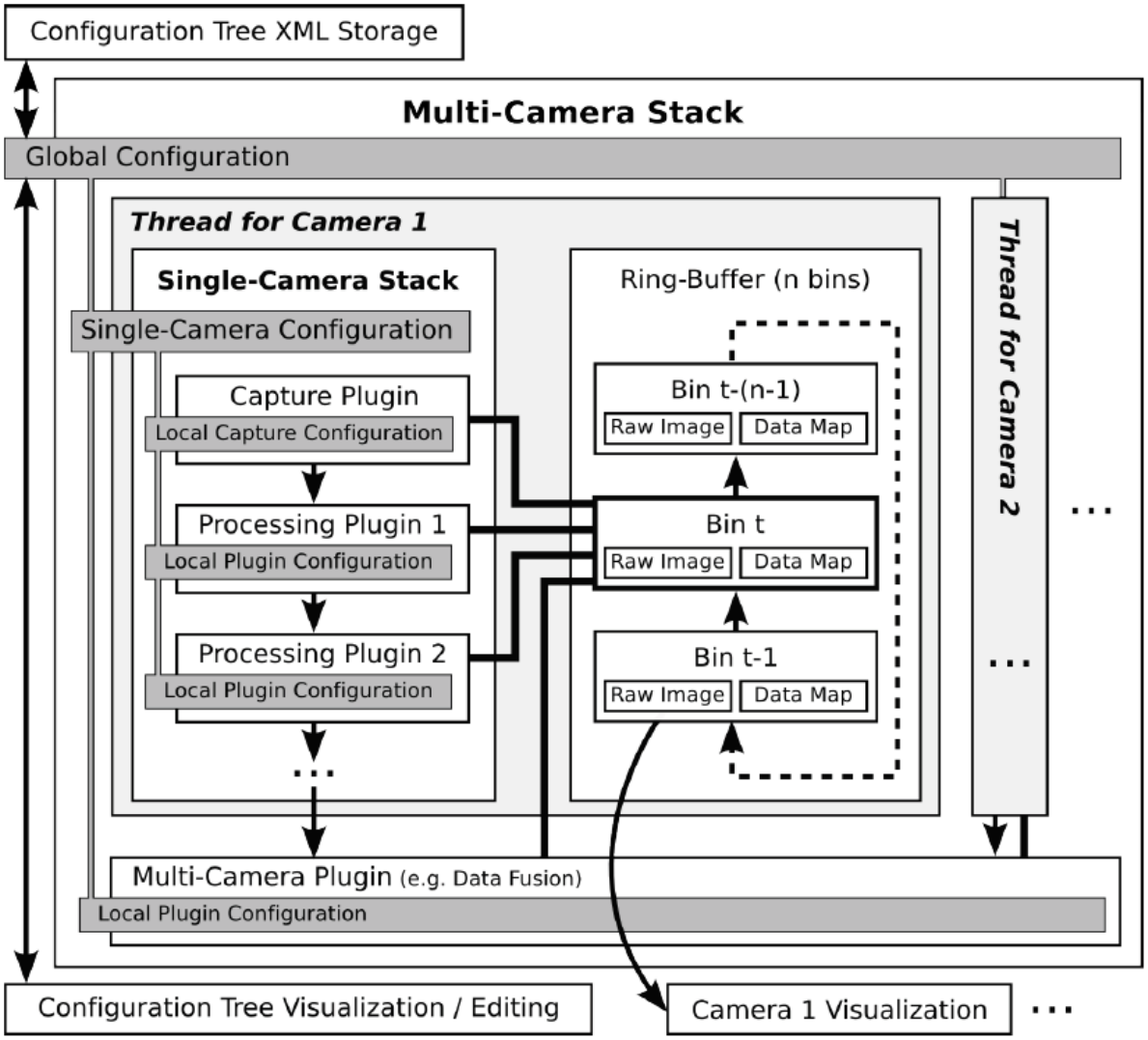


Рис. 4. Архитектура SSL-vision[1]

устройств. Кроме того, такая организация позволяет использовать разные способы захвата изображений для разных потоков без противоречий. Каждому устройству захвата соответствует стек обработки одной камеры (`Single-Camera Stack`). Стек обработки состоит из последовательного выполнения плагинов, каждый из которых выполняет определенную часть работы.

Первым плагином всегда является плагин захвата (`Capture Plugin`). Его задача – получить изображение от выбранного устройства захвата в некотором заранее заданном цветовом пространстве и передать его дальше для обработки.

[1]Изображение взято из статьи SSL-Vision: The Shared Vision System for the RoboCup Small Size League [3]

Для обеспечения дальнейшей работы после получения изображения плагин захвата сохраняет данные в кольцевом буфере (`Ring-Buffer`). Данный буфер представляет из себя связный закольцованный список. При этом каждый плагин из стека обработки имеет доступ к конкретной ячейке данного списка на чтение и запись. Таким образом, каждому плагину доступно изображение и дополнительная информация, которая была записана в буфер до начала его работы. А новая информация будет доступна последующим плагинам. Кроме того, использование такого кольцевого буфера позволяет избежать задержек при обработке. После того как один кадр был обработан, его необходимо визуализировать в приложении (`Camera 1 Visualization`). Однако плагин захвата может занять новую ячейку списка для обработки уже следующего изображения.

Обработку изображения, полученного благодаря плагину захвата, можно разделить на два больших этапа:

1. Сегментация цветов полученного изображения.
2. Определение местоположения мяча и роботов на основе сегментации цветов.

Сегментация цветов происходит с помощью использования библиотеки CMVision [7]. Для выполнения данного этапа используется таблица соответствия (LUT). Каждый пиксель изображения конвертируется из исходного цветового пространства в выходную «цветовую метку» с помощью таблицы соответствия. Такой выходной формат характеризует отношение изначального цвета к требуемому цвету мяча, цвету меток шапочек роботов или другим необходимым цветам.

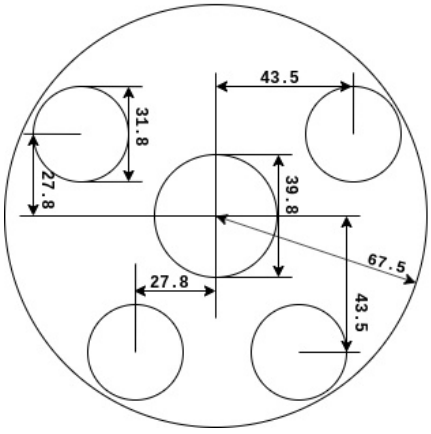
После удачного выполнения предыдущего этапа для обнаружения расположения мяча и роботов, а также ориентации роботов, происходит сравнение полученного изображения с паттерном. Каждая команда, участвующая в соревнования, должна обладать своим паттерном для роботов, в котором описывается соответствие меток номеру робота внутри команды, а также высота

робота в виде желтой линии (пример паттерна приведен на рис. 1). Алгоритм сравнения с паттерном был переиспользован из CMDragon vision system [8].

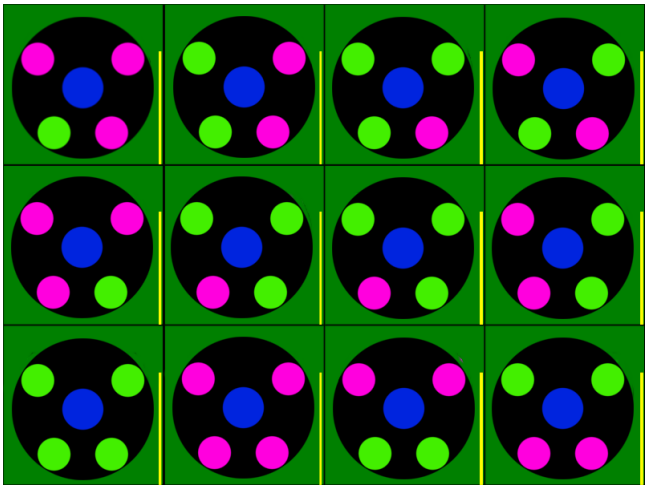
При использовании нескольких камер появляется необходимость синхронизации изображений и результатов их обработки с разных устройств. Этим занимается `Multi-Camera Plugin`. После того как все этапы обработки были выполнены, и полученные данные синхронизированы, результаты работы визуализируются в приложении, а информация о местоположении роботов и мяча публикуется на заранее заданный порт в удобном формате, благодаря использованию Protocol Buffers [9].

4.2. Подготовка SSL-vision к работе с роботами 7M

Так как 7M роботы могут развивать меньшую скорость по сравнению с роботами RoboCup, то было принято решение использовать меньшее по размеру игровое поле. В связи с этим были уменьшены и размеры роботов и меток. Данные характиристики имеют большое значение для корректности работы системы распознавания SSL-vision, поэтому было необходимо вычислить размеры меток на шапочках роботов с сохранением пропорций и создать соответствующий паттерн. Результаты можно увидеть на рис. 5.



(а) Размеры меток на шапочке робота 7M



(б) Паттерн для 7M роботов

Рис. 5. 7M роботы

4.3. Интеграция с симулятором робофутбола

Разобранная выше архитектура SSL-vision позволяет достаточно легко расширять систему, добавляя возможность работы новых устройств захвата. Одной из задач являлась реализация приема видеопотока напрямую из симулятора робофутбола, который разрабатывается на основе симулятора Webots. Так как сам SSL-vision был написан с использованием таких технологий как C++ [10], Qt [11] и OpenCV [12], то именно эти технологии были выбраны для реализации необходимой функциональности.

Для добавления нового способа захвата необходимо:

1. Добавить в дерево конфигурации настройки для принятия видеопотока из Webots, чтобы пользователь имел возможность выбрать данный способ захвата.

2. Реализовать интерфейс `CaptureInterface` (рис. 6).

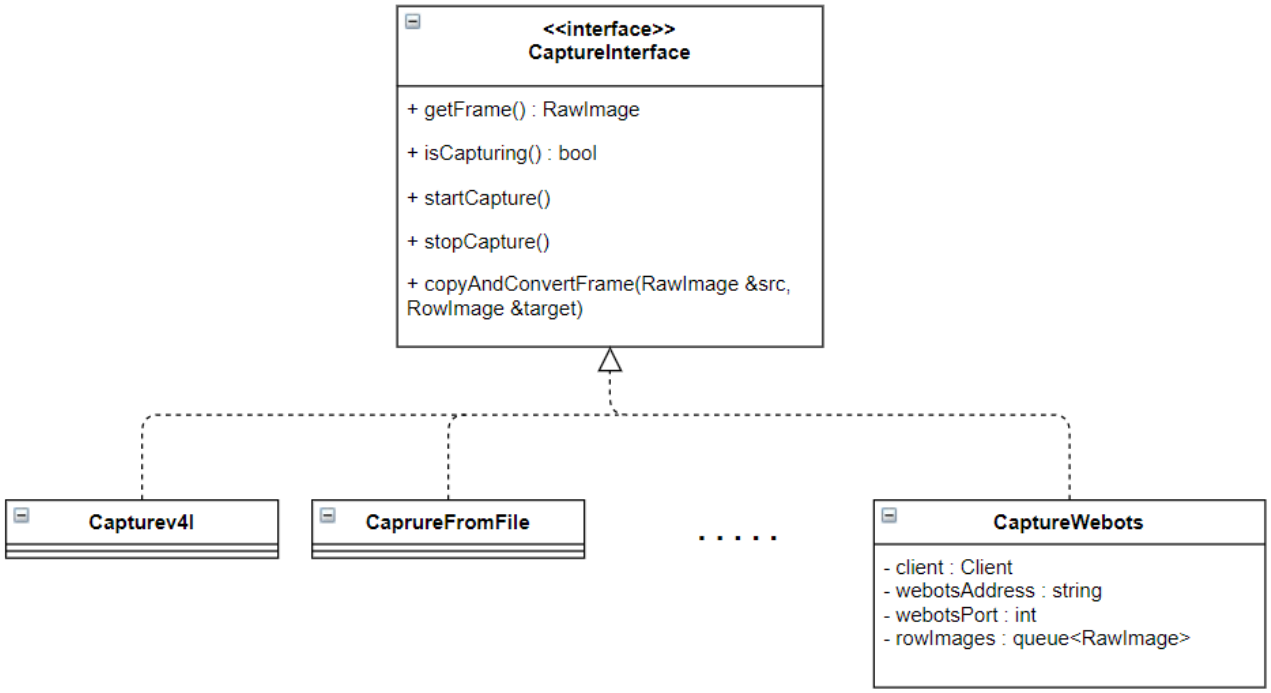


Рис. 6. Реализация интерфейса CaptureInterface

Для каждого способа захвата настройки можно разделить на два типа: `ConversionSettings` и `CaptureSettings`. Первый тип настроек отвечает за формат, в который необходимо конвертировать полученное из устройства захвата изображение. В этом формате кадр будет храниться в ячейке `RingBuffer` (рис. 4). Второй тип настроек отвечает за корректность работы с устройством захвата. Для приема видеопотока из Webots такими параметрами будут `webotsAddres` и `webotsPort`, которые отвечают соответственно за адрес машины и порт, на котором запущен симулятор.

Для корректности реализации интерфейса, необходимо обратить внимание на логику работы потока захвата `CaptureThread`, который инициализируется при нажатии на кнопку Start (рис. 7). При инициализации данного потока вызывается метод `startCapture()` выбранной пользователем реализации интерфейса `CaptureInterface`. Пусть объект данной реализации имеет имя `capture`.

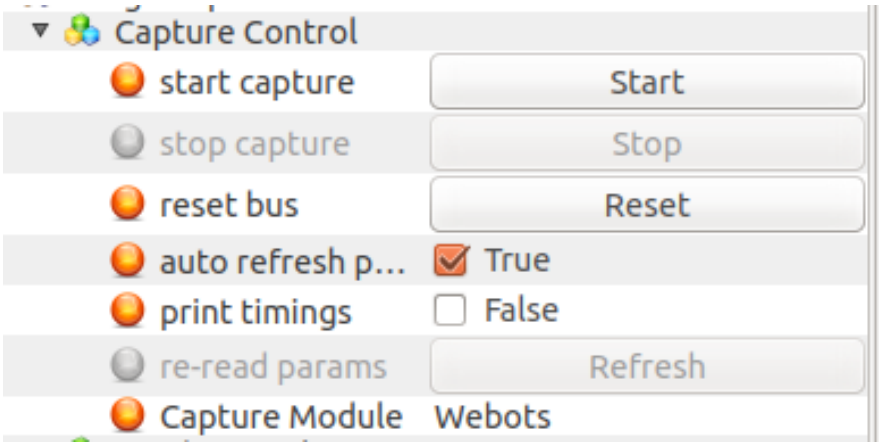


Рис. 7. Запуск захвата видео

После инициализации происходит запуск потока `CaptureThread`. Упрощенно логику его работы можно представить следующим образом:

1. `while true`

    1.1 `currentElement := RingBuffer[currentIndex]`

    1.2 `if capture.isCapturing() then`

1.2.1 `currentImage := capture.getFrame()`
(Берем текущее изображение.)

1.2.2 `success := capture.copyAndConvertFrame(currentImage, currentElement.video)`
(Копируем полученное изображение в предназначенное для этого место в ячейке RingBuffer, при этом, в зависимости от изначальных пользовательских настроек, возможна конвертация в необходимый формат. Возвращает true, если копирование и конвертация прошли удачно, false иначе.)

1.2.3 `if success then <запись дополнительной информации в currentElement>`

1.3 `if stop then return`
(Если процесс захвата был остановлен, выходим.)

Таким образом параллельно работают `CaptureThread` и метод `capture.startCapture()`. При этом в каждом из параллельно выполняемых алгоритмов происходит обращение к очереди `rawImages`, состоящей из полученных из устройства захвата изображений (`CaptureThread` берет элементы из очереди, используя `capture.getFrame()`, а `capture.startCapture()` переодически добавляет элементы в очередь). Для безопасности доступа к разделяемым ресурсам используется мьютекс.

Для принятия видеопотока из симулятора используется сетевой клиент. При вызове метода `capture.startCapture()`, клиент пытается подключиться к указанным пользователем в дереве конфигурации адресу и порту. Если попытка подключения оказалась неудачной, сообщается об ошибке и ее причине. Если клиенту удалось подключиться, то при появлении новых данных от сервера, будет вызываться метод `slotReadyRead()`. В нем происходит основное взаимодействие с сервером, то есть с симулятором робофутбола.

Изображение от симулятора передается в цветовой модели BGRA. В связи с этим был выбран следующий протокол общения с симулятором:

`<ширина изображения : int><высота изображения : int><само изображение в виде сырого потока байт>`.

Таким образом, клиент ожидает два числа типа int и вычисляет ожидаемый размер изображения по формуле:

`size = <высота изображения> * <ширина изображения> * 4`.

Данная формула отвечает реальному количеству передаваемых байт, так как цветовая модель BGRA подразумевает следующее представление изображения: каждый пиксель кодируется чередующимися четырьмя каналами – B (синий), G (зеленый), R (красный), α (альфа канал, прозрачность) (рис. 8).

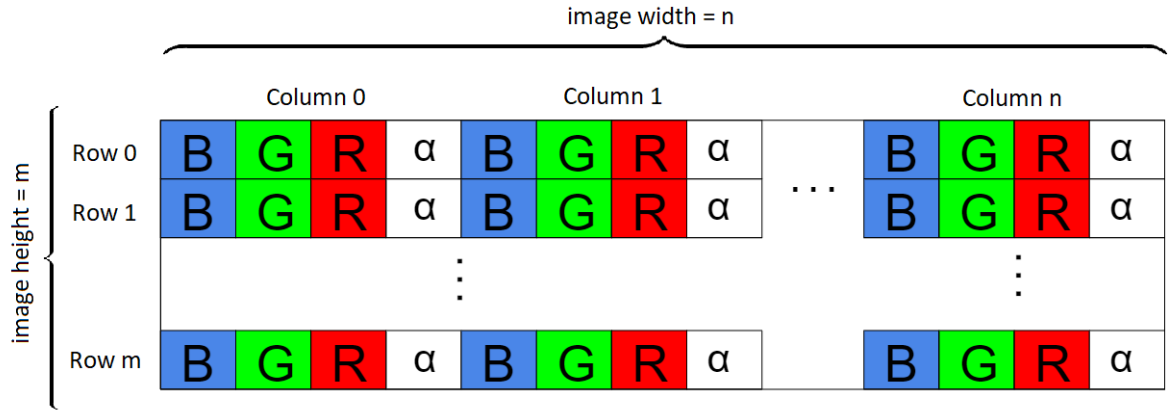


Рис. 8. Цветовая модель BGRA

При этом для дальнейшей корректной работы SSL-vision требуется обернуть изображение в созданный разработчиками контейнер `RawImage` с использованием цветовой модели RGB8. OpenCV предоставляет удобные инструменты для работы с изображениями, поэтому обработку полученного из утройства захвата массива байт можно представить в виде рис. 9.

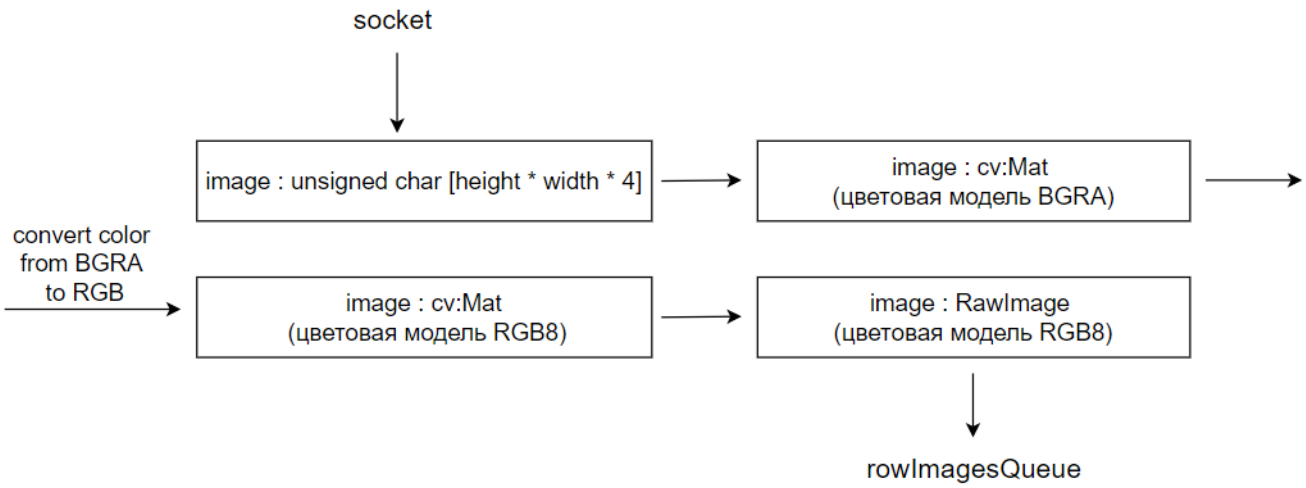


Рис. 9. Преобразование полученного изображения

В результате такой обработки формируется очередь изображений, обернутых в контейнер `RawImage`. Из этой очереди `CaptureThread` по мере возможности достает данные, путем вызова метода `capture.getFrame()`, и складывает в `RingBuffer` для выполнения следующих этапов распознавания.

4.4. Про проблему робастности

Проблемы некорректного распознавания меток роботов появлялись только при динамическом и достаточно сильном изменении света, как добавление нового источника в процессе работы программы. При таком условии неверного распознавания можно избежать путем более тщательно организованного пространства для игры: исключить возможность попадания в помещение естественного освещения, использовать контрастные цвета для меток. На данном этапе было принято решение использовать именно такую тактику, алгоритмически проблема робастности решена не была.

5. Тестирование

При тестировании интеграции с симулятором робофутбола, внимание необходимо обратить на:

- Корректную обработку запуска захвата видео из симулятора: вывод сообщения об ошибке и тип ошибки, если подключиться к симулятору не удалось, или вывод сообщения об успешном подключении.
- Корректная обработка остановки захвата видео из симулятора.
- Отсутствие визуально ощутимой задержки в процессе захвата видео.
- Корректную обработку ошибок, возникших во время захвата: вывод сообщения об ошибке.

Тестирование проводилось вручную, SSL-vision и Webots были запущены на одной машине. Все сценарии были обработаны в соответствии с ожиданием, визуально задержки обнаружено не было. На рис. 10 изображен процесс захвата видео из симулятора робофутбола. Сверху находится окно с запущенным симулятором, снизу SSL-vision, принимающий изображения в режиме реального времени.

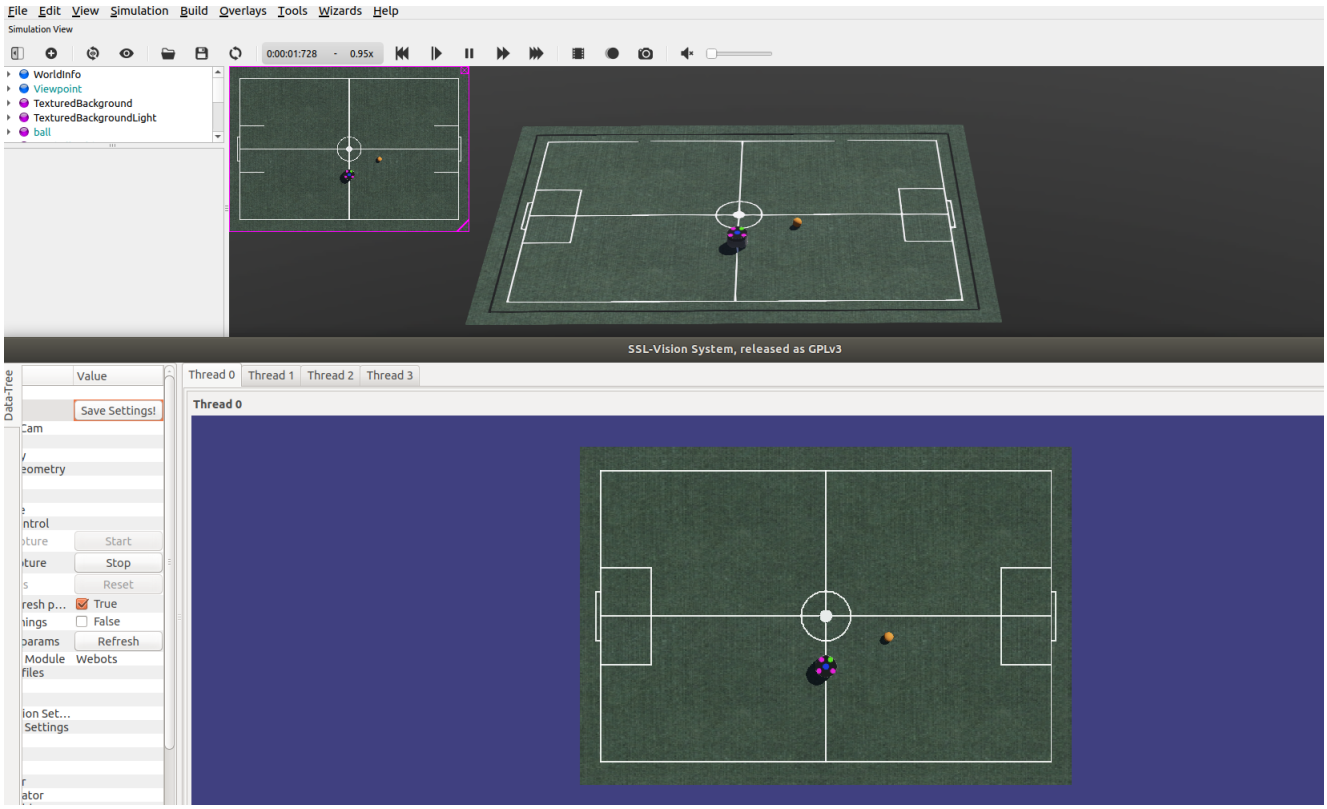


Рис. 10. Процесс захвата видео из симулятора

Заключение

К концу работы были получены следующие результаты:

1. Был проведен анализ SSL-vision, выявлены несоответствия поставленным требованиям.

2. Был разработан и реализован план модификации системы, а именно:

   2.1 Была изучена архитектура решения, выявлены модули захвата и анализа видео.

   2.2 Система SSL-vision подготовлена к работе с роботами 7M.

   2.3 Проведена интеграция модуля захвата с симулятором робофутбола, разработанным на основе симулятора Webots.

   2.4 Алгоритмически проблема робастности на данный момент не решена, для получения корректных результатов при работе с SSL-vision было принято решение тщательно организовать пространство для игры, исключить возможность попадания естественного света на поле.

3. Проведено тестирование добавленной функциональности.

Список используемой литературы

[1] RoboCup Federation official website. — URL: https://www.robocup.org (дата обращения: 06.12.2019)

[2] Rules of the RoboCup Small Size League. — URL: https://robocup-ssl.github.io/ssl-rules/sslrules.html (дата обращения: 18.12.2019)

[3] Zickler S., Laue T., Birbach O., Wongphati M., Veloso M. SSL-Vision: The Shared Vision System for the RoboCup Small Size League. — URL: https://link.springer.com/content/pdf/10.1007%2F978-3-642-11876-0_37.pdf (дата обращения: 18.12.2019)

[4] SSL-vision WiKi. — URL: https://github.com/RoboCup-SSL/ssl-vision/wiki (дата обращения: 05.12.2019)

[5] SSL-vision code. — URL: https://github.com/RoboCup-SSL/ssl-vision (дата обращения: 06.12.2019)

[6] Webots Documentation. — URL: https://cyberbotics.com/doc/guide/foreword

[7] CMVision and Color Segmentation presentation — URL: https://www.kinjo-u.ac.jp/ushida/tekkotsu/10_Vision/pdf/lec002_CMVision.pdf

[8] J. Bruce and M. Veloso, "Fast and accurate vision-based pattern detection and identification,"2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422) (2003 год).

[9] Protocol Buffer Documentation. — URL: https://developers.google.com/protocol-buffers

[10] Stanley Lippman, Josée Lajoie, and Barbara E. Moo, "C++ Primer (5 edition)"(2012 год).

[11] Qt Documentation. — URL: https://doc.qt.io/

[12] OpenCV Webcite. — URL: https://opencv.org/