МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет Механико-математический
Кафедра Программирования

Направление подготовки Математика и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Вопилов Игнат Анатольевич
(Фамилия, Имя, Отчество автора)

Тема работы: Разработка системы оповещения об изменениях состояния игры по изображению с информационного табло

«К защите допущена»

Заведующий кафедрой,
д.ф.-м.н., профессор

Марчук А.Г /_____
(фамилия , И., О.) (подпись, МП)

«…»……20…г.

Научный руководитель

Таранцев И.Г. /_____
(фамилия , И., О.) (подпись, МП)

«…»……20…г.

Дата защиты: «…» ……20…г.

Новосибирск, 2018

Содержание

Введение

Физическая культура и спорт в нашей стране играют очень важную роль в развитии общества. Одним из ключевых факторов развития интереса общества к тем или иным видам спорта является наличие регулярных онлайн-трансляций спортивных состязаний (матчей).

Информационные табло, установленные на стадионах, позволяют зрителю в любой момент узнать состояние игры. Зритель онлайн-трансляций лишен такой возможности, поскольку изображение табло попадает в кадр далеко не всегда. Система титрования может накладывать информацию с табло в удобном для зрителя виде на конечное изображение, показываемое в онлайн-трансляции. Для этого необходимо получать информацию с табло в цифровом виде. В настоящее время далеко не все табло обладают возможностью подключения к стандартным системам титрования. И даже в том случае, если табло имеет возможность подключения, далеко не всегда администрация стадиона дает возможность подключения системы титрования к табло.

Одним из вариантов решения проблемы получения данных с любого информационного табло является создание решения, которое могло бы стандартными средствами захватывать и оцифровывать данные с табло. При этом устройство должно быть быстро устанавливаемым и легко настраиваемым. Идеальным решением этой проблемы является ноутбук с подключенной к нему веб-камерой

Постановка задачи

Целью работы является разработка системы оповещения об изменениях состояния игры по изображению с информационного табло. Имея стандартный персональный компьютер (ноутбук) и стандартную веб-камеру достаточно реализовать программное обеспечение, осуществляющее:

- Захват изображения с камеры.
- Анализ последовательности изображений.
- Распознавание текстовой и цифровой информации на изображении табло.
- Формирование из распознанных символов соответствующих информационных данных.
- Передачу данных в систему титрования.

В связи с тем, что в кадр может попасть оригинальное табло, очень важно, чтобы данные, выводимые системой титрования на экран, не отличались от данных на оригинальном табло. Человек заметит разницу, если задержка системы превысит время реакции человека (порядка одной десятой секунды). Это накладывает не только требование минимизации задержки, но и требование высокой точности распознавания.

Системы титрования адаптированы под работу с определенными протоколами существующих табло. Программное обеспечение должно распознаваться системой титрования так, как будто в сети находится реально подключенное табло.

Так как перед матчем время на установку и настройку оборудования ограничено, очень важно разработать комфортный для пользователя интерфейс, обеспечивающий возможность быстрой настройки параметров системы. Например, положение заново установленной камеры всегда немного отличается от предыдущей установки. Также далеко не всегда имеется возможность во время настройки системы отобразить на табло все возможные символы на всех возможных позициях, чтобы система могла заранее определить и запомнить для каждого символа его графическое представление.

В итоге, разрабатываемое программное обеспечение должно обеспечивать:

- Минимальную погрешность при распознании (кратковременные ошибки допустимы только при ручной коррекции времени в судейской системе).
- Получение времени с минимальной задержкой (с максимальной частотой кадров).
- Передачу информации в локальную сеть по одному из известных протоколов.
- Простоту настройки.

Необходимо протестировать разработанное приложение для того, чтобы выяснить границы его применимости: допустимый уровень шумов во входном изображении, максимальные углы отклонения оси камеры от перпендикуляра к плоскости табло.

1. Алгоритмы распознания символов

Задача распознавания символов является центральным элементом разрабатываемого программного обеспечения. Она является самой ресурсозатратной частью решения, от которой зависят ключевые аспекты подготовки изображения. В связи с этим в первую очередь рассмотрим существующие решения задачи распознавания символов.

Задача распознавания символов является подразделом темы распознавания образов. Для начала коротко о самом распознавании образов. В [1] дается определение “Распознавание образов или теория распознавания образов это раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, процессов, сигналов, ситуаций и т. п. объектов, которые характеризуются конечным набором некоторых свойств и признаков”.

Основными способами для распознавания символов являются:

- Сравнение с заранее созданным шаблоном.
- Распознавание с использованием заданных критериев распознаваемого символа.
- Распознавание при помощи нейронных сетей.

Были рассмотрены несколько решений: сравнение при помощи шаблонов на базе метрики Хэминга, распознавание при помощи нейронных сетей OpenCV и Tesseract.

1.1 Распознавание при помощи метрик

По определению, данному в статье [2] – “Метрика – некоторое условное значение функции, определяющее положение объекта в пространстве.”

Таким образом, если два объекта расположены близко друг от друга, то есть похожи (например, две буквы А написанные разным шрифтом), то метрики для таких объектов будут совпадать или быть предельно похожими.

Наиболее популярной метрикой для распознавания является метрика Хэминга. На её основе (с некоторыми дополнениями) работают некоторые готовые решения для распознавания текста.

Существует несколько реализаций метрики Хэминга:

- По строчкам

  Бинарное изображение центрируется по вертикали, разбивается на строки (Рис. 1) и для каждой строки вычисляется количество черных пикселов.

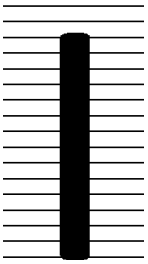

Рис. 1

  Последовательность получившихся чисел сравнивается с шаблоном.

- По строчкам

  Бинарное изображение центрируется по вертикали, разбивается на столбцы и для каждого столбца (Рис.2) вычисляется количество черных пикселов.

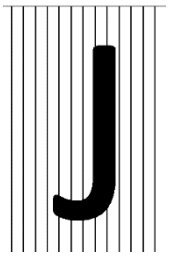

Рис. 2

  Последовательность получившихся чисел сравнивается с шаблоном.

- По сетке

  Бинарное изображение центрируется по вертикали и по горизонтали(Рис 3). Далее каждый пиксел сравнивается с двумерным шаблоном.

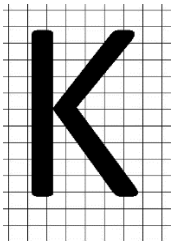

Рис. 3

В каждом из случаев для сравнения используется расстояние Хэмминга – количество позиций, в которых сравниваемые последовательности различны.

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

Однако на практике, реализация любой из этих метрик не дает результата, который удовлетворял бы заданным условиям, так как многие символы по написанию схожие между собой, например «j» «i», что приводит к ошибочному результату сравнения. Аналогичный результат описан в статьях, указанных в обзоре [2]. Единственным вариантом решения данной проблемы является создание новых шаблонов для каждого табло. Это не подходит под условие поставленной задачи (быстрая настройка системы перед игрой).

1.2 Распознавание при помощи критериев

Аналогичным образом показал себя алгоритм сравнения по критериям. Попытки подобрать критерии для успешного распознания цифр разных шрифтов не увенчались успехом. Были рассмотрены разные критерии симметричности и поиска контуров фигур на изображении. Ни один из таких

наборов критериев не показал успешных результатов одновременно на обычных шрифтах и на цифрах семисегментного индикатора. Критерии нужно подбирать для каждого табло индивидуально.

На основе изученных публикаций в интернете [6,7] не удалось найти ни одного удачного решения, которое бы с хорошей вероятностью работало хотя бы в символах одного алфавита. В каждом из рассмотренных языков присутствуют буквы, написание которых очень похоже, например, такие символы как Q и O.

1.3 Распознавание при помощи нейронных сетей

Нейронные сети (искусственные нейронная сеть) — это математическая модель, представляющая систему соединенных и взаимодействующих между собой искусственных нейронов. Искусственный нейрон — функция, преобразующая множество данных входных фактов в один выходной. Изменяя настройку порога возбуждения и весовых коэффициентов разных входных фактов можно изменять “адекватность нейрона”.

Каждый нейрон генерирует свой сигнал на основании своих данных и сигналов, полученных от других нейронов. Искусственные нейроны, соединённые в достаточно обширную сеть с заданным на ней взаимодействием, способны решать сложные задачи. Ключевой особенностью нейронной сети является ее способность к обучению. Подавая на вход нейронной сети набор

данных, и сравнивая ответ сети с правильным ответом, можно корректировать весовые коэффициенты отдельных нейронов.

Современные нейронные сети обладают лучшей точностью распознавания, чем человеческий глаз. На данном этапе развития технологий не существует лучшего решения для поставленной задачи распознавания.

Однако нейронные сети обладают большой вычислительной сложностью, поэтому необходимо убедиться в том, что обычный ноутбук в состоянии анализировать все символы на изображении табло с частотой до 50 изображений в секунду.

Также для нашей задачи очень важно, что бы система работала на любом табло. Это означает, что нейронная сеть должна быть обучена распознавать самые разные цифры, включая символы семисегментного индикатора.

Было принято решение данное программное обеспечение реализовать, с помощью языка C++, поскольку он предоставляет широкие возможности для проектирования программных систем с высоким быстродействием. Для данного языка существует две наиболее известные реализации нейронных сетей для распознавания символов: Tesseract и OpenCV.

Входные изображения символов могут быть искажены как аддитивными шумами, так и линейными и нелинейными искажениями. Также, возможна ситуация, при которой символ, в своем написании будет иметь разрывы.

Для фреймворка OpenCV не было найдено в открытом доступе моделей приемлемого качества для текущей задачи (модели показали низкое качество

работы с аддитивными шумами). При отсутствии предобученных моделей, возникает необходимость самостоятельного обучения нейронной сети, для чего потребовались бы следующие действия: сбор и разметка обучающих примеров, подбор оптимальной архитектуры нейронной сети, а также процедуры обучения. Каждая из этих задач требует большого количества времени, особенно сбор и разметка данных.

В свою очередь Фреймворк Tesseract обладает предобученной нейронной сетью, которая показала приемлемое качество на английских символах и цифрах. Нейронная сеть Tesseract от компании Google уже обучена под огромное количество шрифтов, при этом, быстродействие сети Tesseract оказалось достаточно для требуемой скорости распознавания символов. При тестировании нейронной сети Tesseract на 50 тестовых примерах среднее время распознавания одного символа оказалось менее одной миллисекунды (~0.7мс), что полностью подходит под условие поставленной задачи. Тестирование проводилось на Intel core I5 3317u и 8гб Ram. На ноутбуках, обладающих более низкой производительностью, это время может отличаться.

Нейронная сеть Tesseract была протестирована на изначально заведомо испорченных изображениях символов (Рис 4). Варьировались параметры генерируемых изображений, такие как: изменение ширины или высоты символа, наложение аддитивных шумов, нелинейные искажения.

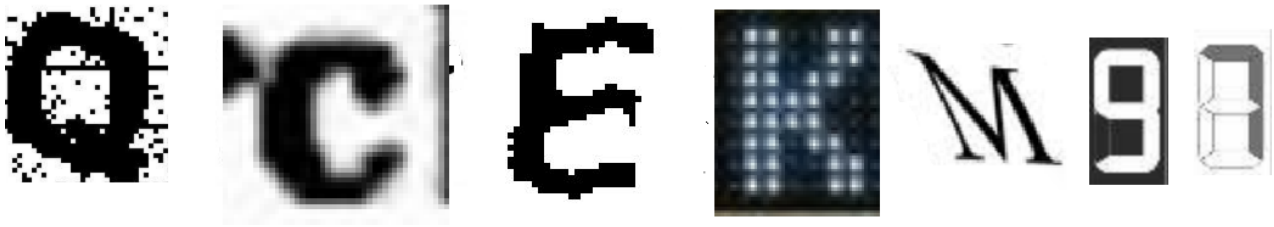

Рис. 4. Примеры верно распознанных символов

На качество распознавания сильное влияние оказало:

- Наложение аддитивных шумов.
- Нелинейные преобразования. (Вызванные стабилизацией камеры)

Слабое влияние на распознание показали такие факторы, как:

- Аффинные преобразования.
- Изменения ширины и высоты символа.
- Обрывы в написании символа.

По результатам тестирования выяснилось, что оптимальным по скорости и качеству распознавания является разрешение символа 20x25 пикселей.

В итоге, из-за невозможности изначально откалибровать шаблоны по существующему табло (то есть прогнать все возможные символы и сгенерировать для них шаблоны) не представляется возможными использование метрики Хэмминга. В дальнейшем будем рассматривать распознавание с помощью нейронной сети Tesseract.

2. Захват изображения

Для захвата изображения, необходимо подключится к вебкамере и получить с неё изображение. Было рассмотрено несколько готовых решений, таких как OpenCV, Linux Media Subsystem, Directshow, МатЛаб.

Почти все рассмотренные решения, кроме OpenCV являются узкоспециализированными и ограничены конкретной операционной системой, поэтому было выбрано решение OpenCV.

3. Предобработка изображения

Захваченное с камеры изображение абсолютно не подходит для непосредственного распознавания нейронной сетью. Кадр имеет большое количество излишней информации, попавшей в объектив камеры. Для успешного распознавания символов нужно избавиться от шумов и других факторов, которые понижают вероятность успешного распознавания. Так же требуется выделить на изображении отдельные символы, которые будут передаваться в нейронную сеть для распознавания. То есть необходимо выполнить предварительную обработку изображения.

Были рассмотрены основные популярные пакеты инструментов, подходящих для решения поставленной проблемы: OpenCV, Linux Media Subsystem, Directshow, МатЛаб.

Linux Media Subsystem и Directshow обладают крайне скудными возможностями в области обработки изображения и не являются кроссплатформенными решениями, тем самым основной акцент делался на остальные пакеты.

МатЛаб – набор инструментов для решения задач технических вычислений. Он обладает очень широким спектром возможностей для обработки изображения, возможностью создания и комбинирования фильтров. Однако скорость обработки изображения была крайне низкой. Реализовать поточную обработку данных с его помощью невозможно.

OpenCV – библиотека содержащая в себе набор алгоритмов компьютерного зрения. Обладает широким инструментом для обработки изображений с открытым исходным кодом. Имеет возможность захвата изображения с камеры. Показала себя с наилучшей стороны. Обладает очень широким спектром фильтров и накладываемых преобразований. При проверке скорости обработки изображения, подошла под критерии поставленной задачи. В итоге, была выбрана библиотека OpenCV.

Как правило, направление камеры не всегда перпендикулярно по отношению к плоскости поверхности табло, поэтому возникает задача перспективного преобразования изображения табло для получения прямоугольного вида самого табло и всех цифр и текста в нем. Это можно выполнить простым матричным преобразованием. Причем матрицу преобразования можно вычислить один раз при настройке системы, а само преобразование выполнять для каждого кадра. Первая задача (поиск матрицы поворота) требует отдельного рассмотрения. Вторая задача решается стандартным фильтром «cvFilter2D» из библиотеки OpenCV.

Данные на каждом конкретном табло расположены в абсолютно разных местах, отсюда возникает задача “вырезания” заданной области из захваченного изображения. Автоматическое выделение областей показало низкую точность, так как на табло, в большинстве случаев присутствует реклама. Выделение области можно выполнить вручную один раз в процессе настройки системы перед игрой. Выделенная может содержать в себе

несколько символов, а так же не всегда совпадает с границами символа, соответственно приходится находить границы символа внутри области, то есть выполнять сегментацию отдельных символов. Также, использование сегментации решит проблему “дрожания” изображения – поскольку камера, как правило, закреплена не очень жестко, а табло находится далеко от камеры, то любое малое перемещение камеры вызывает заметный сдвиг захватываемого изображения относительно выделенной области. Это требует расширения границ выделенной области, и, следовательно, требует сегментации (выделения отдельных символов на относительно большой области изображения). Есть разные решения задачи сегментации, которые будут рассмотрены ниже.

Текст на табло обладает высокой контрастностью, поскольку он должен быть хорошо заметен зрителям в любом случае. При этом информация о цвете не несет никакой полезной информации. Игнорируя информацию о цветности и используя только яркостную компоненту можно не только сократить объем обрабатываемой информации, но и частично избавиться от шумов. Так же необходимо выполнять традиционную обработку изображения для удаления шумов камеры. Это можно выполнить стандартными методами библиотеки OpenCV.

В итоге необходимо рассмотреть решение следующих задач:

1. Поиск матрицы поворота (перспективные преобразования).
2. Сегментация внутри области.
3. Повышение контраста.

3.1 Перспективные преобразования

Для преобразования нам нужно точно знать координаты всех угловых точек пересечения сторон табло. Очевидно, что для нахождения координат этих точек нужно описать стороны найденного контура табло уравнениями прямых линий. В OpenCV есть инструмент, использующий преобразование Хафа [1]. На выходе мы получим набор линий, из которых мы можем выделить самые крайние и самые длинные вертикальные и горизонтальные линии. Далее находим точки пересечения этих линий, тем самым, получив нужные нам углы табло.

Рис 5. Процесс выделения контуров табло на изображении

На Рис. 5 показаны основные шаги, по которым выполняется поиск контура табло на изображении. Сначала выполняется размытие изображение по Гауссу, чтобы избавится от шумов. Затем выполняется бинаризация и поиск самых длинных прямых линий на изображении.

[1] Преобразование Хафа это численный метод, который применяется для извлечения элементов из изображения. Используется в анализе изображений, цифровой обработке изображений и компьютерном зрении. Предназначен для поиска объектов, принадлежащих определённому классу фигур, с использованием процедуры голосования.

Требуется найти пропорции самого табло. На искаженных изображениях наблюдается отклонение центра масс угловых точек найденного контура от точки пересечения диагоналей. При наличии искажений, обязательно будет наблюдаться отклонение. Для определения матрицы поворота, необходимо вычислить угол, под которым расположено табло. (Рис. 6)

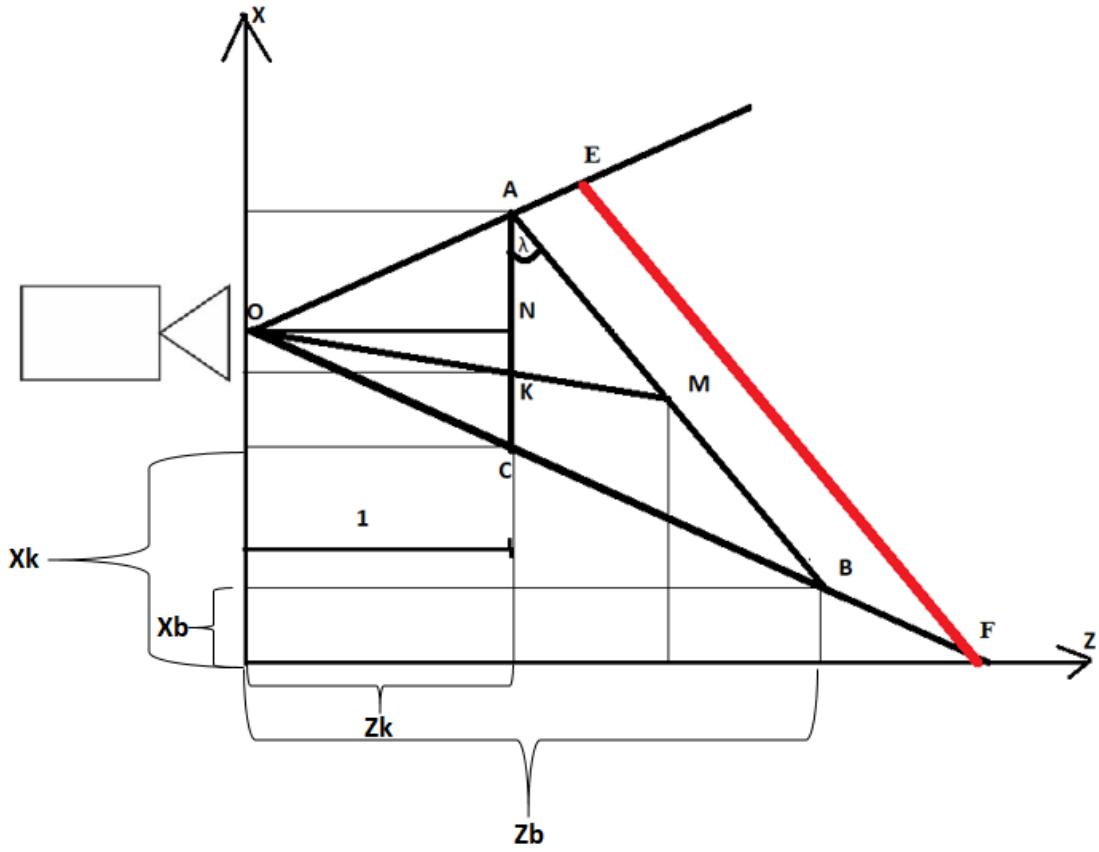

Рис. 6 Проекция табло(красный отрезок) на матрицу камеры . N – точка центра проекции табло на экран, M – проекция точки диагоналей, AC – Проекция табло на матрицу камеры, EF – искомое табло, AB – параллельный перенос EF

Нахождение тангенса угла наклона табло, будем осуществлять с помощью метода координат, рассматривая проекцию на плоскость.

$$Tg(A) = \frac{Zb - Za}{Xb - Xa} = \frac{\frac{Xb}{Xc-1}}{Xb - Xa} = \frac{Xb - Xc}{Xc * (Xb - Xa)}$$

$$\frac{Xm}{Zm} = Xk = \frac{Xb + Xa}{Zb + Za}$$

$$\frac{Xb}{Zb} = Xc$$

$$Xb = Xc * Zb$$

$$Xk = \frac{Xb + Xa}{Zb + 1} = \frac{Xb + Xa}{\frac{Xb}{Xc+1}} = Xc * \frac{Xb * Xa}{Xb + X}$$

$$Xk * (Xb + Xc) = Xc * (Xb + Xa)$$

$$Xk * Xb + Xk * Xc - Xc * Xb - Xc * Xa = 0$$

$$Xb * (Xk - Xc) = Xc * Xa - Xc * Xk = Xc * (Xa - Xk)$$

$$Xb = Xc * \frac{Xa - Xk}{Xk - Xc}$$

$$Tg(A) = \frac{Xb - Xc}{Xc * Xb - Xc * Xa} = Xc * \frac{\frac{Xa-Xk}{Xk-Xc} - Xc}{\frac{Xc^2*(Xa-Xk)}{Xk-Xc}} - Xc * Xa =$$

$$= \frac{Xa - Xk - Xk - Xc}{(Xc * (Xa - Xk)} - Xa * (Xk - Xc)$$

$$Xc * Xa - Xc * Xk - Xa * Xk + Xa * Xc = 2 * Xa * Xc - Xk * (Xa - Xc)$$

$$Tg(A) = \frac{Xa + Xc - 2 * Xk}{2 * Xa * Xc - Xk * (Xa + Xc)} =$$

$$= \frac{Xn - Xk}{Xa * Xc - Xk * Xn} \approx (Xn - Xk)/(Xn * Xc)$$

Получив угол наклона плоскости, составляем матрицу поворота.

3.2 Сегментация

После предобработки и выделения областей происходит сегментация изображения области на строки и символы. Предполагается, что строки текста расположены горизонтально и не создают пересечений друг с другом. Соответственно между каждыми двумя строками будет наблюдаться контрастная полоса низкой яркости. Оператором изначально указывается, сколько строк ожидать в определенной области, тем самым задавая минимальную высоту строки. Соответственно, находя участок, превышающий

пороговую величину минимальной длины на гистограмме средней яркости пиксела по строке, мы находим границы строк.

Затем область с найденной строкой требуется разбить на символы. Построив гистограмму средней яркости по столбцу, мы разбиваем область на участки, в которых превышается пороговая величина яркости.

Для примера рассмотрим гистограмму средней яркости пиксела по строке вдоль вертикальной оси, представленную на рисунке 7. На гистограмме ищем участок более минимальной длины, превышающий пороговую величину яркости. Именно он и будет соответствовать предполагаемой области строки текста. Используя полученную картину, отсекаем верхние и нижние поля.

Рис. 7 Сегментация строк

Далее строим гистограмму средних значений яркости по вертикали вдоль оси X (Рис. 8). Минимумы (или нижние экстремумы) должны соответствовать

интервалам между цифрами. Согласно этому, определяем границы каждой цифры в блоке.

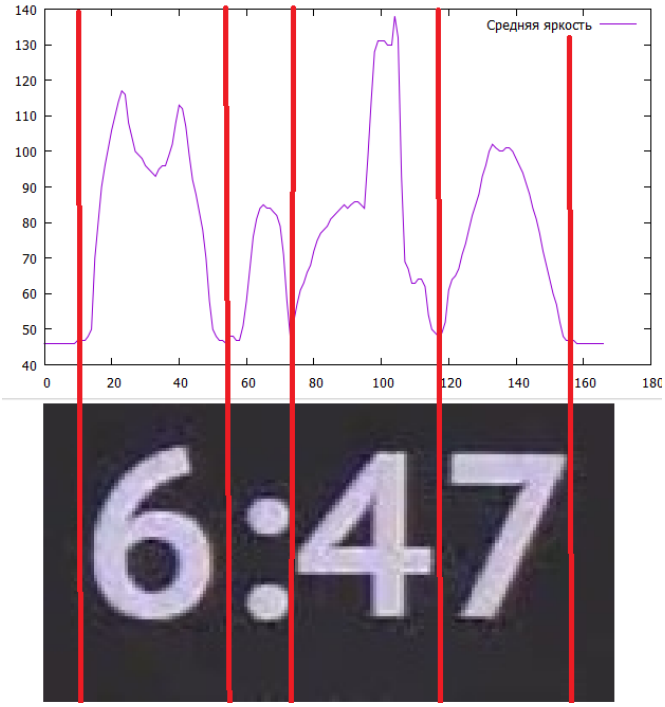

Рис. 8 Сегментация символов

3.3 Повышение контраста

Все промежуточные обработки изображения имеет смысл выполнять только с яркостной компонентой, сохраняя ее значения в интервале от 0 до 255. Это очень популярный формат для обработки изображений. В конце, перед передачей изображения для распознавания, имеет смысл выполнить бинаризацию изображения, повышая контрастность до предела. Это должно повысить точность распознавания. Бинаризация позволяет существенно уменьшить количество информации, с которой приходится работать. Главным параметром бинаризации является значение порога, с которым сравнивается яркость каждого пикселя и в зависимости от результата пикселю присваивается

значение 0 или 1. Неравномерная засветка кадра вызывает проблему определения порога. Существуют различные методы бинаризации, которые можно разделить на две группы – глобальные и локальные. В первом случае величина порога константна во время всего процесса бинаризации. Во втором – изображение разбивается на области, в каждой из которых вычисляется локальный порог.

В случае с глобальными методами порог либо задается вручную, либо считается средняя яркость по всему изображению, однако данный метод показывает хорошие результаты лишь при равномерной освещенности захватываемой области.

Основная проблема бинаризации заключается в нахождении порогового значения, которое отделит символы не только от фона, но и от шума, теней, бликов и прочего информационного мусора. Часто для этого прибегают к методам математической статистики. В методе Брэдли [3] [4] мы рассматриваем со стороны интегральных изображений (Рис. 9):

- Разбиваем изображение на несколько областей со стороной D.
- Берем среднее арифметическое цвета всех пикселей в этой области – C.
- Порог бинаризации для всех пикселов внутри области равен C+t, где t – некоторый коэффициент.

Рис. 9 Разбиение изображения на области

Обычно в алгоритме Бредли берется значения D = 1/8 от ширины изображения и коэффициент t = 15% от среднего значения яркости пикселей внутри области. Однако в процессе тестирования была выявлена проблема корректного выбора параметров D и t при использовании алгоритма для отдельного символа. Размеры изображения символа могут оказаться меньше рекомендованного значения D, а уменьшение параметра D приводит к потере мелких деталей на изображении.

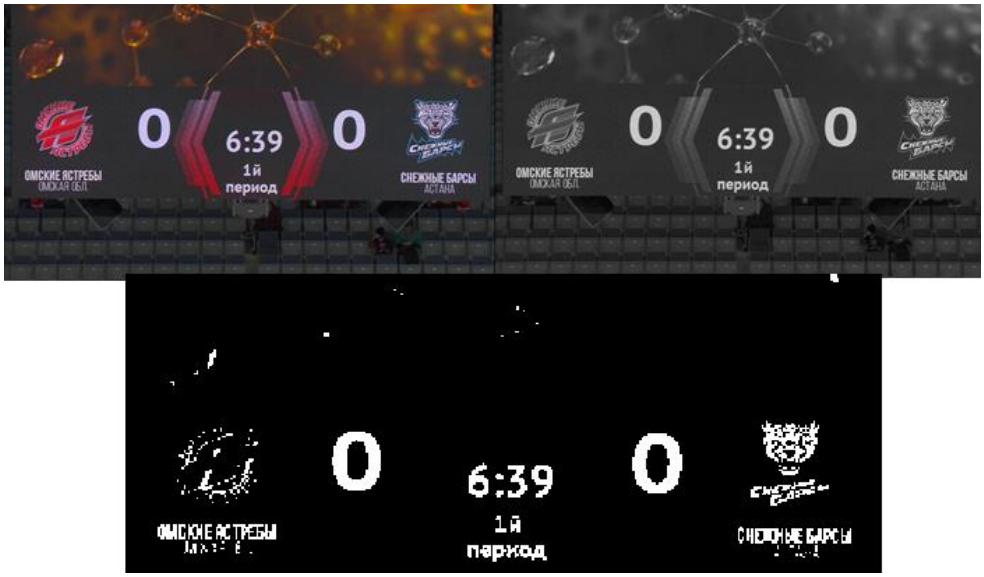

Рис. 10 Результат бинаризации изображения.

В итоге предлагается использовать два способа бинаризации. При поиске табло на изображении мы имеем большую площадь захвата изображения камерой, что вызывает высокую вероятность большой разницы яркости в различных частях изображения, поэтому при поиске матрицы поворота необходимо использовать метод Бредли (Рис 10). В свою очередь при бинаризации перед распознаванием символа необходимо использовать более быстрый алгоритм глобальной бинаризации, так как захватываемая область маленькая, и на ней не будет значительного изменения засветки.

4. Реализация

Для реализации была выбрана платформа Microsoft Visual Studio 2017. Реализация выполнялась на языке C++ стандарта C++ 11 с использованием библиотек OpenCV и Tesseract. Программа является приложением с визуальной оболочкой MFC.

Программа принимает изображение с камеры и отображает в окне распознанный текст в каждой из областей распознавания.

Результаты тестировались на вычислительной платформе: *Intel® Core™ i5-3317U* Processor (3M Cache, up to 2.60 GHz)), 8.00 GB Ram, Windows 10 Pro x64.

Тестирование проводилось на изображениях, с примененными к ним фильтрами перспективных преобразований. На рисунке 11 указаны примеры изображений на которых проводилось тестирование.

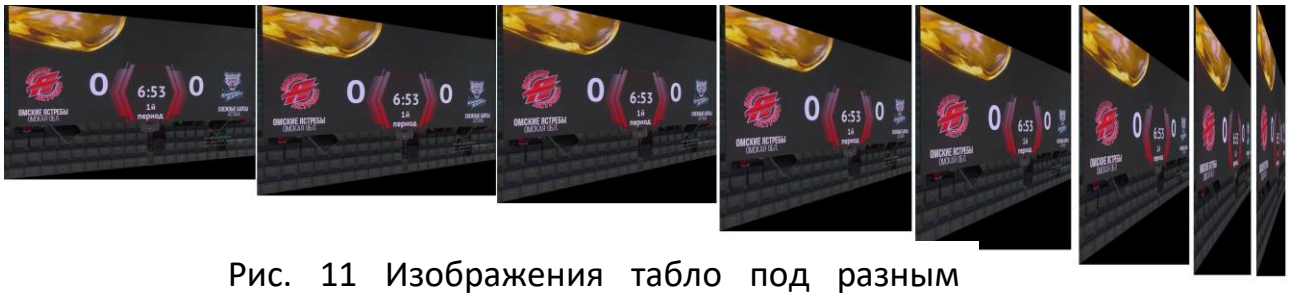

Рис. 11 Изображения табло под разным

По результатам тестирования, удалось выявить угол, при котором начинаются погрешности, при распознавании символов.

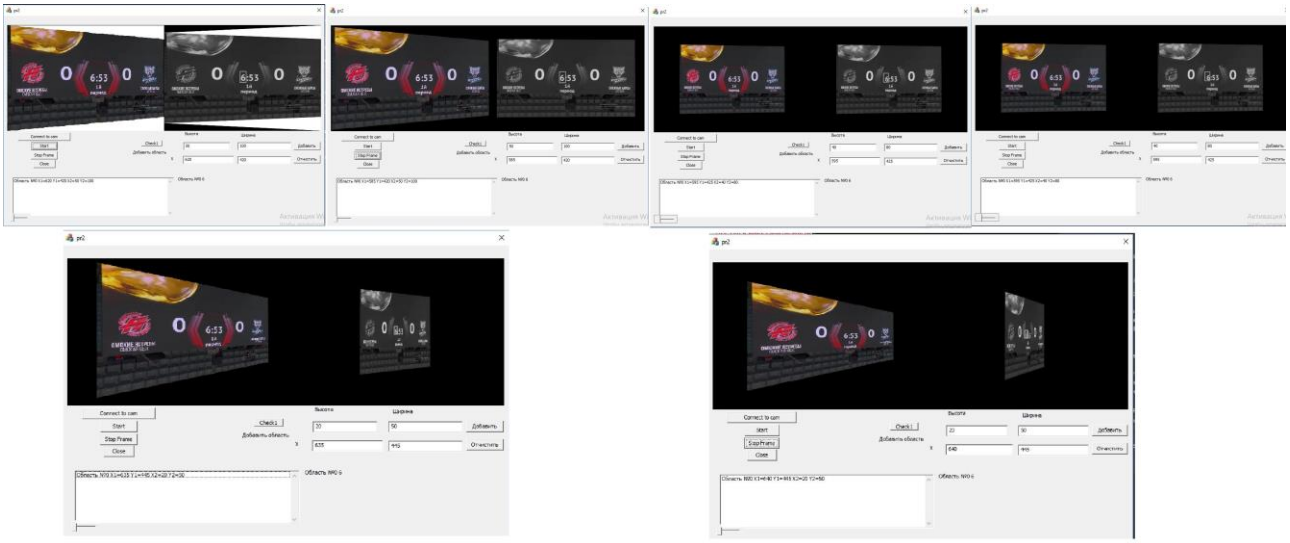

Рис. 12 Результаты тестирования на восстановление табло

На рисунке 12, показаны процесс работы программы, при разных перспективных преобразованиях.

При превышении угла более 60 градусов, начались весомые искажения, при обратном перспективном преобразовании, влияющие на точность распознавания.

Входные данные с добавлением шумов (в разумных пределах) не помешало успешному распознаванию счета и времени.

Так же, программное обеспечение было протестировано на разных видах табло. Рис 13.

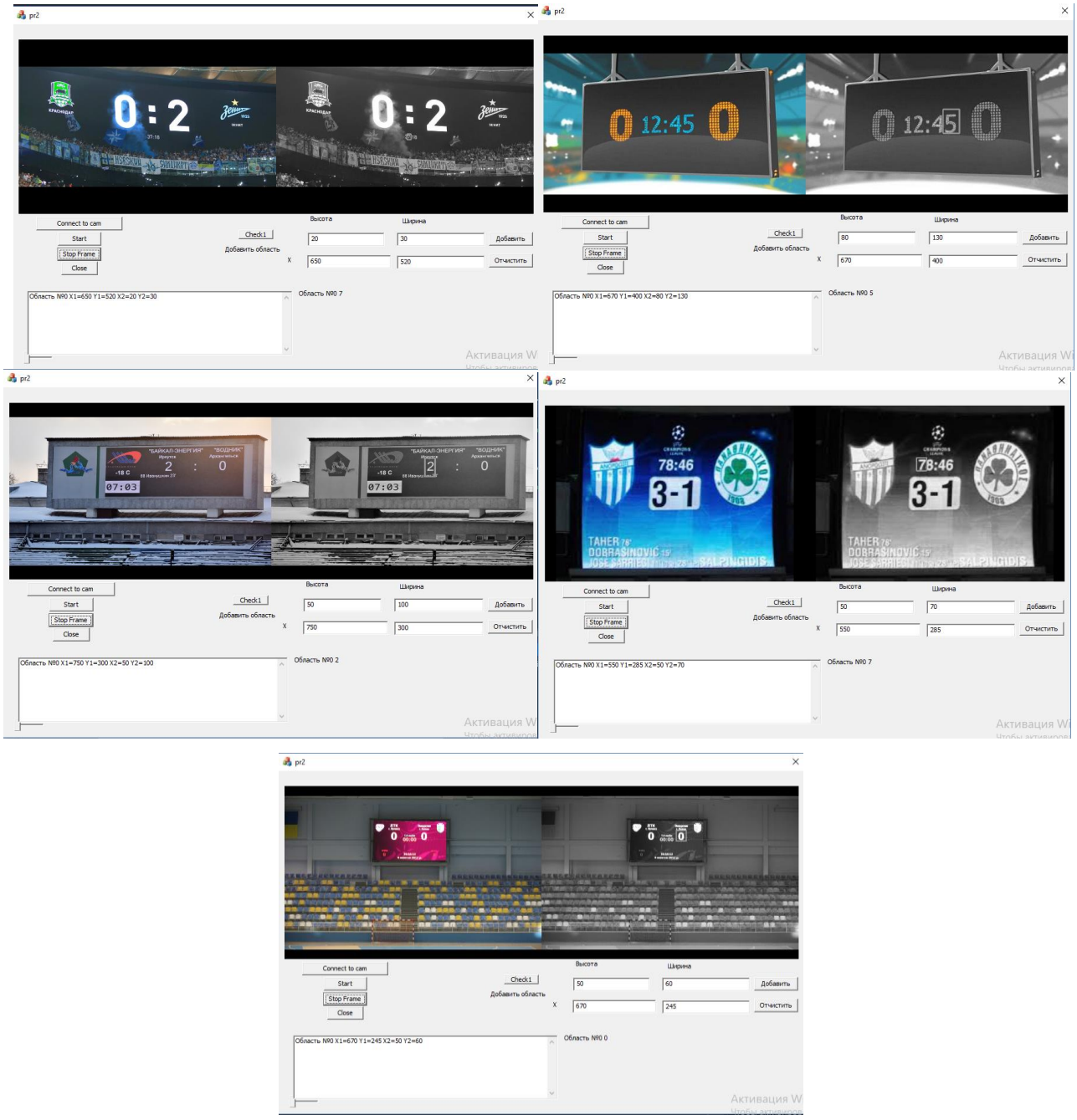

Рис. 13 Примеры распознавания на реальных табло

Заключение

При выполнении работы были получены следующие результаты:

- Проведено исследование и тестирование существующих алгоритмов для распознавания текста; выбран быстрый и точный алгоритм с использованием нейронной сети Tesseract.
- Рассмотрены основные требования к подготовке изображения для распознавания и выбран метод их реализации, удовлетворяющий условиям поставленной задачи.
- Реализовано программное обеспечение, позволяющее захватывать и распознавать значения, отображаемые на информационном табло.
- Реализован интерфейс для быстрой настройки областей распознавания.
- Приложение протестировано на изображениях табло под разными углами наблюдения и с различным уровнем шумов.

Литература

1. Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. — М.: Бином. Лаборатория знаний, 2006. — 752 с. — ISBN 5-947-74384-1.
2. [Электронный ресурс] Методы распознавания текста https://habr.com/post/220077/
3. [Электронный ресурс] А. Федоров Бинаризация черно-белых изображения: состояние и перспективы развития. http://it-claim.ru/Library/Books/ITS/wwwbook/ist4b/its4/fyodorov.htm
4. [Электронный ресурс] Бинаризация изображений: алгоритм Брэдли https://habr.com/post/278435/
5. Арлазаров В.Л., Куратов П.А., Славин О.А. Распознавание строк печатных текстов // Сб. трудов ИСА РАН «Методы и средства работы с документами». — М.: Эдиториал УРСС, 2000. — С. 31-51.
6. [Электронный ресурс] “Распознаем текст, используя расстояние Хэмминга” https://geektimes.com/post/90867/
7. [Электронный ресурс] “Распознавание образов методом потенциальных функций” http://www.delphikingdom.com/asp/viewitem.asp?catalogid=1299