МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

КУРСОВАЯ РАБОТА

СТАТИСТИЧЕСКИЙ АНАЛИЗ ВИДЕОПОТОКОВ ВЕБИНАРОВ

Работу выполнил _____Б.М. Ибрагимов
(подпись)

Направление подготовки 01.03.02 «Прикладная математика и информатика» курс 3

Направленность (профиль) «Системное программирование и компьютерные технологии» (Математическое и программное обеспечение вычислительных машин)

Научный руководитель
канд., физ.-мат. наук _____В.В. Подколзин
(подпись, дата)

Нормоконтролер
ст. преп. _____А.В. Харченко
(подпись, дата)

Краснодар
2019

РЕФЕРАТ

Курсовая работа 39 с., 9 рис., 7 источников.

ВЕБИНАР, OPENCV, ВЕДУЩИЙ, ОБЛАСТЬ КОНФЕРЕНЦИИ, КРИТЕРИИ ОЦЕНКИ ЭФФЕКТИВНОСТИ, АНАЛИЗ

Объектом изучения является вебинары и область конференции в вебинарах.

Цель курсовой работы – частичная автоматизация процесса оценки эффективности вебинаров.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В Сибирском государственном университете телекоммуникаций и информатики, с которым у КубГУ имеется соглашение о сотрудничестве в сфере образования, науки, научной и инновационной деятельности №1 от 29.05.2018 г., стоит задача оценивания качества контактной работы, реализуемой посредством вебинаров, в ходе дистанционного обучения.

Вебинар (от англ. webinar - онлайн-семинар, онлайн-конференция) — это онлайн-трансляция с целью обучения или корпоративной встречи. В дальнейшем будут рассматриваться лишь обучающие вебинары. В вебинаре всегда участвуют две стороны: докладчик-ведущий и слушатели-участники. Обычно участники могут видеть ведущего. Ведущий такой возможности не имеет. В течении вебинара все участники его слушают. Вебинар может проходить как в прямой трансляции, так и быть доступным в записи. К техническим возможностям вебинара можно отнести:

а) демонстрацию презентаций;

б) возможность быстро получать и сразу видеть точные результаты голосований и опросов среди участников вебинара;

в) передача участникам нужных файлов и ссылок;

г) обратная связь – в виде чата, куда можно написать свой вопрос или просто поделиться впечатлениями (если вебинар в прямой трансляции).

Для того, чтобы оценить эффективность вебинаров, используются разнообразные критерии оценки вебинаров. К таким общим критериям относятся:

а) количество зарегистрированных пользователей;

б) распространение информации о предстоящем вебинаре;

в) количество участников вебинара;

г) активность участников.

Количество зарегистрированных пользователей показывает, насколько выбранная тема востребована у целевой аудитории.

Распространение информации о предстоящем вебинаре, например, в социальных сетях, говорит о том, что есть пользователи, которые готовы зарекомендовать этот вебинар и другим людям, что тема вебинара и анонс привлекли аудиторию.

Количество участников на вебинаре — это индикатор интереса слушателей к теме вебинара.

Чем активнее участники вебинара будут взаимодействовать с ведущим, тем больше вероятность того, что они, например, оставят заявку, подпишутся на рассылку, посмотрят запись вебинара [1].

Перечисленные выше критерии являются лишь частью. Всю процедуру оценки эффективности вебинара проводят специальные люди.

В данной курсовой работе рассматриваются вопросы составления методов частичной автоматизации процедуры оценки эффективности вебинаров.

1 Использованный инструментарий

1.1 Общие сведения об OpenCV

OpenCV (Open Source Computer Vision Library) – библиотека, позволяющая работать, проводить разные операции с компьютерным зрением, с открытым исходным кодом. Изначально библиотека написана на языках C и C++ (на данный момент эта библиотека написана и для таких языков программирования как Python, Ruby, Matlab и т.д.) и способна работать на компьютерах с разной ОС (Linux, Windows, MacOS).

Основной целью OpenCV является предоставление пользователям простого в использовании интерфейса, который позволит людям строить сложные приложения, использующие компьютерное зрения. Библиотека OpenCV содержит более 500 функций, которые охватывают многие области компьютерного зрения, такие как: инспекция фабричной продукции, медицина, безопасность, пользовательский интерфейс, калибровка камеры, робототехника. OpenCV была использована во многих приложениях и научно-исследовательских работах: сшивка изображений спутниковых карт, выравнивание отсканированных изображений, уменьшение шума на медицинских снимках, анализ объектов, системы обнаружения вторжения, автоматический мониторинг, системы контроля, калибровка камеры, приложения для военных, беспилотники, наземные и подводные аппараты. Библиотека так же была задействована для распознавания звуков и музыки, при помощи анализа спектрограмм [2].

1.2 Инструментарий работы с видеофайлами

В OpenCV имеется несколько функций для работы с видео. Наиболее распространенными являются функции чтения и записи видео. Как один из вариантов, для включения веб-камеры (в случае записи видео) и вывода кадров

с камеры (или кадров видеофайла) в отдельном окне, создается объект класса VideoCapture. Аргументом метода open () этого объекта, является индекс устройства или имя и путь открываемого для показа и работы видеофайла. Так как камера чаще всего одна, то в качестве индекса устройства ставится 0 (или -1). Если же подключенных камер больше, то указывают соответствующий номер камеры с которой будут получены кадры на обработку и вывод. После того, как был получен доступ к камере, уже имеется возможность в цикле захватить кадр(ы) с камеры и работать с кадром(ми) как с изображением, а затем выводить результат обработки либо как обычное изображение, либо в непрерывном видеопотоке с веб-камеры (имеет смысл, если ведется работа с каждым кадром) [3]. Пример получения кадров с камеры (и открытия видеофайла) предоставлен ниже:

```
VideoCapture capture;
Mat cadr;
capture.open("/home/beslan/Рабочий стол/video3.mp4");
// (Открыть видеофайл)
// capture.open( 0 ); (Получить кадры с камеры)
while(1) {
capture >> cadr; // Захват кадра и помещение его как картинку в cadr
// Работа с кадром и его вывод
if(waitKey() > 0) break;
}
```

В этом примере при нажатии любой клавиши, работа с видеофайлом будет прекращена.

Одно из свойств видео, которое необходимо знать при его анализе, это длительность видео. Так как нет определенного метода получения

длительности видео, то его получают использованием другого метода. Пример кода получения длительности видео, предоставлен ниже:

```
    int frameCount, seconds, minutes, hours;
    fps = capture.get(CV_CAP_PROP_FPS);
    frameCount=
int(capture.get(CV_CAP_PROP_FRAME_COUNT));
    duration = frameCount/fps;
    seconds = (int)duration%60;
    minutes = int(duration/60);
    hours = int (minutes/60);
```

В этом примере используется метод get(). Таким образом, чтобы узнать длительность видео, необходимо и достаточно знать значения двух параметров, а именно количество кадров в секунду (capture.get(CV_CAP_PROP_FPS)) и общее количество кадров (capture.get(CV_CAP_PROP_FRAME_COUNT)).

Чтобы совершать некоторые действия после воспроизведения видео, следует проверять кадры видео на предмет содержания в нем изображения (функция empty()). Если изображение является пустым, то это сигнал к тому, что видео завершилось и следует выйти из цикла воспроизведения видео.

1.3 Инструментарий работы с изображениями

Растровое изображение состоит из пикселей, который отвечает за цвет изображения в его определенной точке. Изображение сводится к матрице, элементы которой соответствуют пикселям, т. е. они могут быть векторами. Ввиду того, что изображений это матрица, над изображениями доступны все стандартные операции, относящиеся к матрицам. Матрицу можно создавать безотносительно изображения, тогда её элементы задают уже не обязательно

цвет. Поэтому, в качестве её элементов можно использовать вектора произвольного размера, но определенного типа. Класс Mat описывает матрицу, которая служит и для хранения изображений. Пиксель, а точнее элемент матрицы, является вектором определенного типа имеющий фиксированный размер. Размер вектора определяется типом изображения, а точнее выбранным цветовым пространством для данного изображения, типом пикселей. Для цветных изображений используются вектора длины 3 (например, соответствующие цветовым компонентам красный, зеленый и синий), а для монохромных (например, градации серого цвета) – вектора длины 1 (например, соответствующие яркости). Для собственных типов размер вектора может быть произвольным.

Для вывода изображения в отдельное окно (или чтобы обновить содержимое окна новым изображением), используется функция imshow(), где в качестве параметров выступают имя окна и название изображения (объекта класса Mat).

Для клонирования (в буквальном смысле) изображения используется метод класса Mat, clone(). В этом случае, копируется в новое изображение все данные клонируемого изображения [3].

1.3.1 Использование пороговых значений в преобразовании изображения

Если значение пикселя больше порогового значения, то ему присваивается одно значение, иначе другое. Для этого имеется функция Threshold. Её синтаксис:

```
Threshold (CvArr* in_image, CvArr* output_image,
                double threshold, double max_value, int threshold_type),
```

где

in_im – входное изображение (в данное работе это объект класса Mat);

out_im – выходное изображение (тоже объект класса Mat);

threshold – пороговое значение;

threshold_type – тип порогового преобразования;

max_value – максимальное значение.

Имеется несколько типов порогового значения. Принцип работы этих типов приведен ниже:

Тип CV_THRESH_BINARY (1):

$$out\_im(x,y) = \begin{cases} max\_value, \text{ если } in\_im(x,y) > threshold \\ 0, \text{ иначе} \end{cases} \quad (1)$$

Тип CV_THRESH_BINARY_INV (2):

$$out\_im(x,y) = \begin{cases} 0, \text{ если } in\_im(x,y) > threshold \\ max\_value, \text{ иначе} \end{cases} \quad (2)$$

Тип CV_THRESH_TRUNC (3):

$$out\_im(x,y) = \begin{cases} threshold, \text{ если } in\_im(x,y) > threshold \\ in\_im(x,y), \text{ иначе} \end{cases} \quad (3)$$

Тип CV_THRESH_TOZERO (4):

$$out\_im(x,y) = \begin{cases} in\_m(x,y), \text{ если } in\_im(x,y) > threshold \\ 0\ , \text{ иначе} \end{cases} \quad (4)$$

Тип CV_THRESH_TOZERO_INV (5):

$$out\_im(x,y) = \begin{cases} 0, \text{ если } in\_im(x,y) > threshold \\ in\_im(x,y), \text{ иначе} \end{cases} \quad (5)$$

Обычное пороговое преобразование Threshold не учитывает то, что части объектов могут иметь различную яркость из-за разности в освещённости. Если использовать адаптивное пороговое преобразование AdaptiveThreshold, которое рассматривает значение не в одном пикселе, а в окрестности пикселя, то эта ситуация исправляется [3].

1.3.2 Изменение цветового пространства изображения

Для изменения цветового пространства изображения используется функция cvtColor(). Её синтаксис приведен ниже:

void cvtColor (InputArray in_image, OutputArray out_image, int code),

где

in_image – входное изображение;

out_image – изображение на выходе;

code – код преобразования цветового пространства.

Функция преобразует входное изображение из одного цветового пространства в другое. В случае преобразования в-из цветового пространства RGB порядок каналов должен быть задан явно (RGB или BGR). Формат цвета по умолчанию в OpenCV это BGR (байты перевернуты).

Имеется несколько основных кодов преобразования цветового пространства:

Из BGR в RGB (COLOR_BGR2RGB). Здесь меняются местами каналы R и B. Наоборот – аналогично.

Из BGR в Gray (COLOR_BGR2Gray) (6):

$$Y \leftarrow R * 0.299 + G * 0.587 + B * 0.114. \quad (6)$$

Из BGR в HSV (англ. Hue, Saturation, Value — тон, насыщенность, значение (яркость)) (COLOR_BGR2HSV) (7):

$$V \leftarrow min(R,G,B)$$
$$S \leftarrow \begin{cases} \frac{V-min(R,G,B)}{V}, & \text{если } V \neq 0 \\ 0, & \text{иначе} \end{cases}$$
$$H \leftarrow \begin{cases} \frac{60*(G-B)}{V-min(R,G,B)}, & \text{если } V = R \\ 120 + \frac{60*(B-R)}{V-min(R,G,B)}, & \text{если } V = G \\ 240 + \frac{60*(R-G)}{V-min(R,G,B)}, & \text{если } V = B \end{cases} \quad (7)$$
$$H \leftarrow \begin{cases} H + 360, & \text{если } H < 0 \\ H, & \text{иначе} \end{cases}.$$

Диапазон значений на выходе (8):

$$0 \leq V \leq 1,\ 0 \leq S \leq 1,\ 0 \leq H \leq 360. \quad (8)$$

Поэтому (9):

$$V \leftarrow k1 * V;\ S \leftarrow k2 * S;\ H \leftarrow k3 * H, \quad (9)$$

где

$$\begin{cases} k1 = k2 = 255;\ k3 = 1/2, & \text{если изображение } 8-\text{битное} \\ k1 = k2 = -65535;\ k3 = -1, & \text{если изображение } 16-\text{битное.} \\ k1 = k2 = k3 = 1, & \text{если изображение } 32-\text{битное} \end{cases}$$

1.3.3 Размытие (сглаживание) изображений

Размытие изображения достигается путем свертывания изображения ядром фильтра нижних частот. Применимо для удаления шумов. Он фактически удаляет высокочастотное содержимое (например, шум, края) из изображения. Таким образом, края немного размыты в этой операции.

(Имеются методы размытия, которые не размывают края). Можно выделить такие методы размытия как:

а) усреднение (простое размытие);

б) гауссово размытие;

в) медианное размытие.

Для выполнения операции сглаживания мы применим фильтр к нашему изображению (10).

$$out(i,j) = \sum_{k,l} in(i+k, j+l) * h(k,l), \tag{10}$$

где

$out(i,j)$ – значение выходного пикселя;

$in(i+k, j+l)$ – значение входного пикселя;

$h(k,l)$ – ядро преобразования (матрица зависит от вида преобразования).

Рассмотрим некоторые ядра преобразований.

Нормализованное ядро (Каждый выходной пиксель является средним значением его соседей ядра (все они вносят равный вес)) (11):

$$K = \frac{1}{K_w * K_h} * \begin{bmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ 1 & \dots & 1 \end{bmatrix}, \tag{11}$$

где

K – ядро преобразования;

K_w – ширина ядра;

K_h – высота ядра.

Гауссово преобразование (двумерное):

Элементы матрицы преобразования находятся из Гауссовой функции (12):

$$g(x,y) = A * exp\left(-\left(\frac{(x-x_0)^2}{2*\sigma_x^2} + \frac{(y-y_0)^2}{2*\sigma_y^2}\right)\right), \quad (12)$$

где

A – высота колокола;

(x_0, y_0) – сдвиг пика колокола от нулевой абсциссы;

(σ_x, σ_y) – размах колокола.

При $A = 1$, $(x_0, y_0) = (0, 0)$, получим график двумерной Гауссианы (колокола) в общей форме (рис. 1):

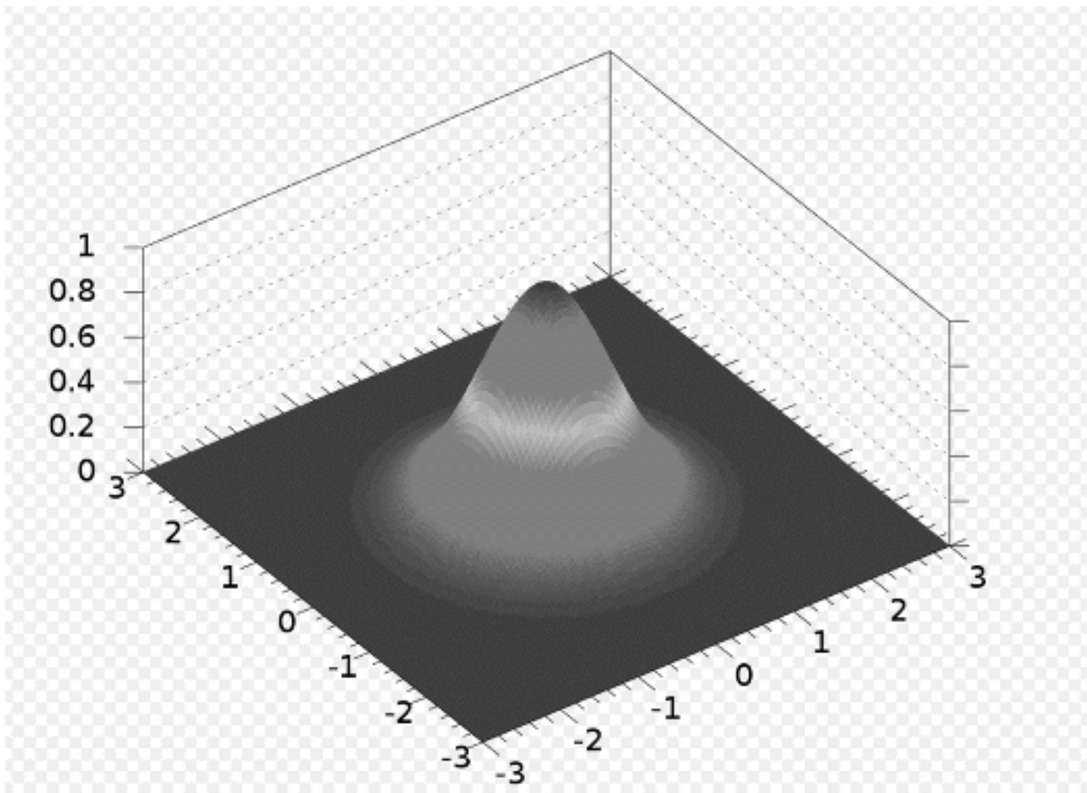

Рисунок 1 – Двумерная Гауссиана

Объем такой поверхности равен (13):

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)dxdy = 2 * \pi * A * \sigma_x * \sigma_y. \quad (13)$$

Медианное преобразование:

В этом преобразовании матрица не имеет значений. Необходимо лишь указать размеры этой матрицы (квадратной окрестности рассматриваемого пикселя в центре). После чего сортируется массив значений яркостей пикселей, попавших в окрестность и рассматриваемый пиксель, который в

центре окрестности, заменяется значением пикселя в центре массива (если число элементов массива четное, то выбирается среднее значение двух центральных пикселей).

Медианная фильтрация считается дольше, чем операция свертки, т.к требуется сортировка массива яркостей окрестных пикселей. Пути возможного ускорения:

а) использовать алгоритмы быстрой сортировки;

б) конкретная реализация для каждой размерности фильтра;

в) вообще не использовать сортировку, и считать через гистограмму окрестности точки.

Применяя описанные выше фильтры, получим размытое изображение, но размыты они будут по-разному в зависимости от выбранного метода. Простое размытие помимо подавления шума портит резкие границы и размывает мелкие детали изображения. Фильтр Гаусса меньше, чем простое размытие, размывает мелкие детали, лучше удаляет шумы. Медианный фильтр резких границ не портит, убирает мелкие детали, но изображение становится менее естественным. Фильтр выбирается в соответствии с задачей [3].

1.3.4 Алгоритм выделения границ Canny

Алгоритм состоит из пяти отдельных шагов:

а) сглаживание. размытие изображения для удаления шума;

б) поиск градиентов. границы отмечаются там, где градиент изображения приобретает максимальное значение;

в) подавление не максимумов. только локальные максимумы отмечаются как границы;

г) двойная пороговая фильтрация. потенциальные границы определяются порогами;

д) трассировка области неоднозначности.

Перед применением детектора, изображение преобразуется в оттенки серого, чтобы уменьшить вычислительные затраты.

Для подавления шума на изображении, используется размытие двумерным фильтром Гаусса.

Для поиска градиентов изображения часто применяют оператор Собеля. Это дискретный дифференциальный оператор, вычисляющий приближенное значение градиента яркости изображения. Результатом применения оператора Собеля на каждом пикселе изображения является либо вектор градиента яркости в этой точке (Gx, Gy), либо его норма ($\sqrt{Gx^2 + Gy^2}$). Оператор Собеля основан на свёртке изображения небольшими целочисленными фильтрами в вертикальном (14. а) и горизонтальном (14. б) направлениях. Ядра фильтров имеют вид (14):

$$MGx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad (14.\ а)$$

$$MGy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (14.\ б)$$

Угол вектора квантуется по 45° — именно такие значения необходимы для одного из следующих этапов.

Пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента. Значение направления должно быть кратно 45° (рис. 2):

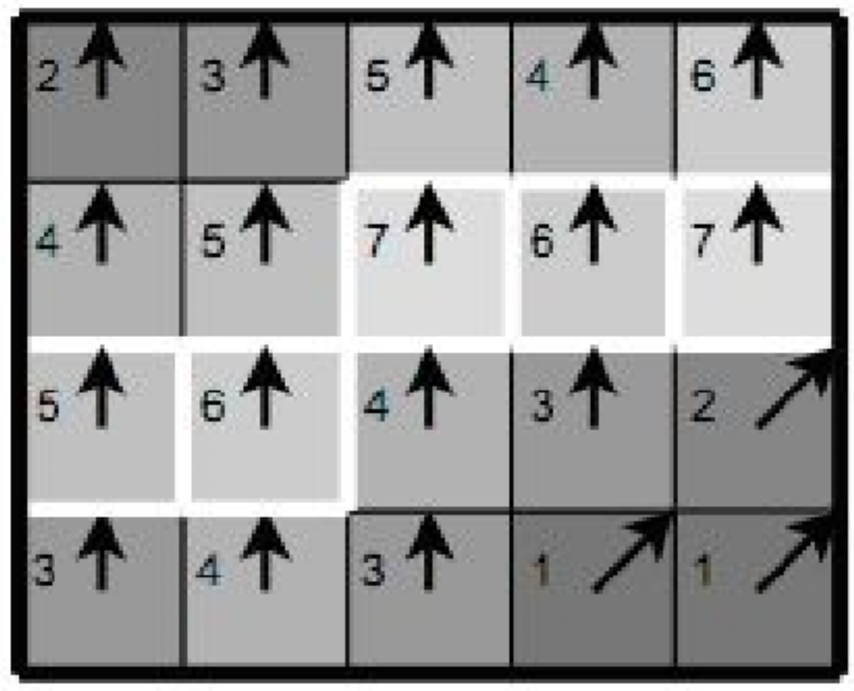

Рисунок 2 – Принцип подавления не максимумов

Почти все градиенты пикселей в примере на рисунке 2, направлены вверх, поэтому значение нормы градиента в этих точках будет сравниваться с ниже- и вышерасположенными пикселями (значения нормы градиентов пикселей, имеющие другое направление, будет сравниваться со значениями нормы градиентов по диагонали). Обведённые белым контуром пиксели останутся в результирующем изображении, остальные – будут подавлены. После подавления не-максимумов, края стали более точными и тонкими.

Следующий шаг — применение порога, чтобы определить находится или нет граница в данной точке изображения. Чем меньше порог, тем больше границ будет находиться, но тем более восприимчивым к шуму станет результат, выделяя лишние данные изображения. Наоборот, высокий порог может проигнорировать слабые края или получить границу фрагментами.

Выделение границ Canny использует два порога фильтрации: если значение пикселя выше верхней границы – он (значение яркости пикселя) принимает максимальное значение (граница считается достоверной), если ниже нижней границы – пиксель подавляется. Точки со значением, попадающим в диапазон между порогами, не изменяются (они будут уточнены на следующем этапе).

Таким образом, задача свелась к выделению групп пикселей, получивших на предыдущем этапе промежуточное значение, и отнесению их к границе (если они соединены с одной из установленных границ) или их

подавлению (в противном случае). Пиксель добавляется к группе границ, если он соприкасается с ней (с пикселем уже полученной границей) по одному из 8-ми направлений [4].

1.3.5 Метод хранения контуров изображения и некоторые операции над ними

После того, как были получены контуры, ими надо как-то оперировать и хранить. Храниться они будут в виде последовательности прямых (отрезков) определенной длины и указателя направления, следовательно, при работе с контурами, будут производиться действия с полученной последовательностью отрезков. Для получения и хранения такой последовательности отрезков, описывающая контуры изображения, часто используется цепной код Фримана.

Цепные коды применяются для конвертации границы в последовательность отрезков. В основе этого лежит 2х или 3х-битная решетка на 4 и 8 направления соответственно (рис. 3).

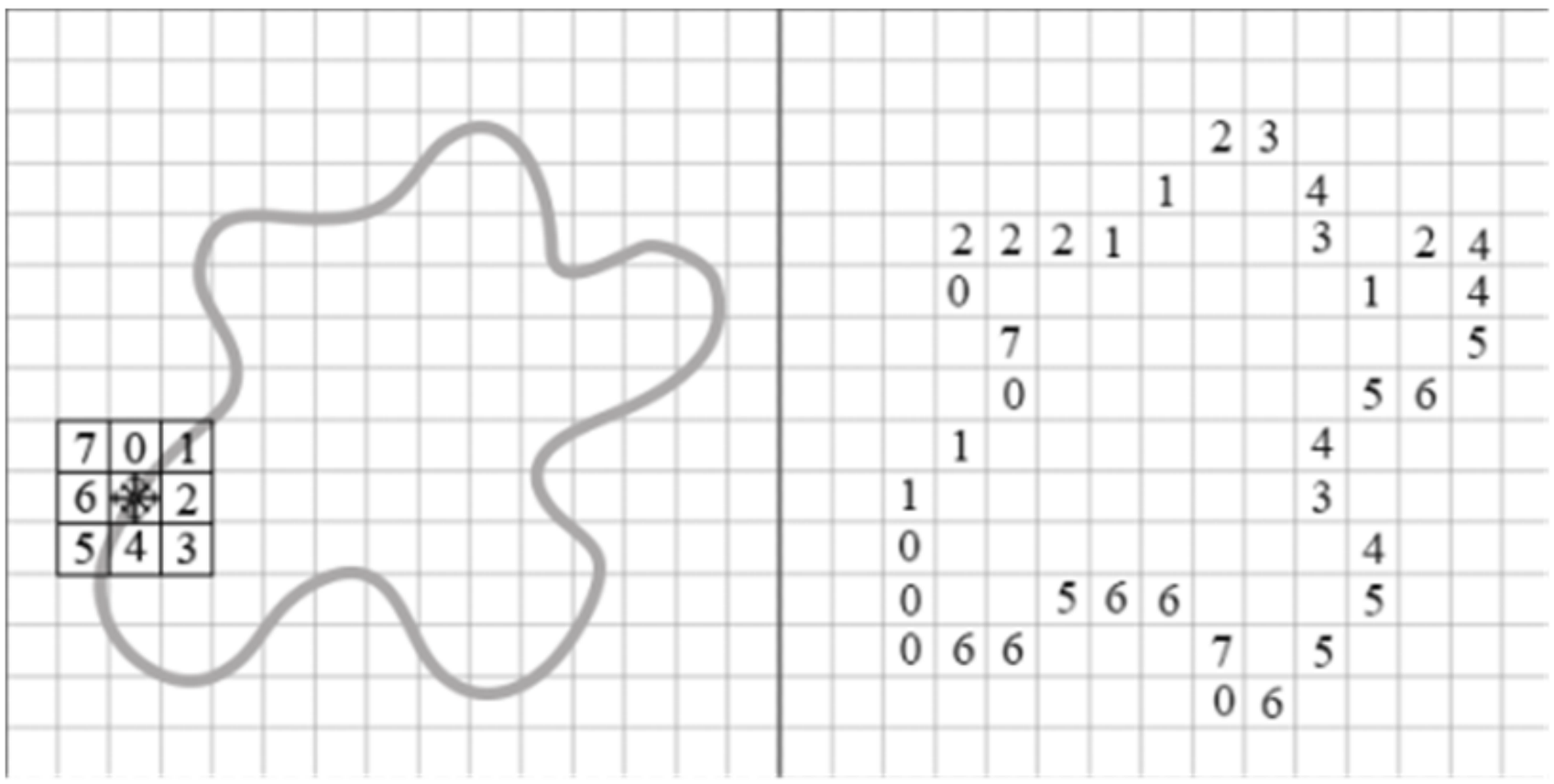

Рисунок 3 – Принцип цепного кода Фримана

Суть метода Фримена заключается в последовательной записи значения векторов между соседними дискретными участками контура. Значение

вектора находится в пределах от 0 до 7. Для начала работы алгоритма Фримена на изображение накладывается и назначается ядром любая ячейка, содержащая участок контура. Следующим шагом является поиск среди смежных ядру ячеек той, что содержит наибольший участок контура. После нахождения такой ячейки вектор между ней и ядром записывается в последовательность, и ядро переносится в найденную ячейку. Затем поиск повторяется, но в этот раз в него не входит предыдущая ячейка.

Когда ядром снова становится начальная ячейка, последовательность замыкается, а новым ядром назначается любая ячейка, содержащая участок ещё не описанного контура изображения. Результатом работы цепного кода Фримена является набор числовых последовательностей, каждая из которых описывает замкнутый контур.

Изменение начальной точки приведет к циклическому сдвигу вектор-контура, а изменение масштаба изображения можно рассматривать как умножение элементарного вектора на масштабный коэффициент [5].

1.3.6 Морфологические операции над изображениями

К основным морфологическим операциям относятся:

а) erode — размывание (операция сужения);

б) dilate — растягивание (операция расширения).

Операция расширения (Dilate) состоит из свертывания изображения с некоторым ядром, которое может иметь любую форму или размер, обычно квадрат или круг.

Ядро имеет определенную точку привязки, обычно являющуюся центром ядра.

Когда ядро сканирует изображение, мы вычисляем максимальное значение пикселя, перекрываемое ядром в обрабатываемом изображении, и заменяем пиксель нового изображения в позиции точки привязки этим максимальным значением. И так с каждым пикселем. Таким образом,

создается новое изображение, в котором светлые области были расширены. То есть, эта операция максимизации заставляет яркие области внутри изображения “расти” (поэтому эта операция и называется Dilate).

Операция сужения (Erode) аналогична операции Dilate, только в этом случае будут расширяться темные области изображения, то есть эта операция минимизации.

Синтаксис этих операции одинаковый:

Void Erode(Dilate) (CvArr in, CvArr out, IplConvKernel elem, int iter),

где

in – входное изображение;

out – изображение на выходе;

elem – ядро. По умолчанию = NULL, что соответствует использованию ядра 3x3 с точкой привязки по центру;

iter - число итераций (сколько раз повторить морфологическое преобразование). По умолчанию = 1.

Erode (размывание/сужение) изображения обычно используется для избавления от случайных вкраплений на изображении. Идея состоит в том, что вкрапления при размывании устранятся, тогда как крупные и, соответственно, более визуально-значимые регионы, остаются.

Dilate (расширение), используется для устранения шума и способствует объединению областей изображения, которые были разделены этим шумом, тенями, и т.д. Таким образом, применение небольшого растягивания должно «сплавить» эти области в одну.

Морфологические операции, чаще всего, применяются над двоичными изображениями, которые получаются после порогового преобразования (threshold) [6].

Для составления ядра используется функция:

Mat getStructuringElement (int shape, Size ksize, Point anchor),

где

shape – определяет тип ядра (MORPH_RECT – ядро вида прямоугольника, MORPH_ELLIPSE - ядро вида эллипса, MORPH_CROSS – ядро вида креста;

ksize – размер ядра;

anchor – положения центрального элемента. По умолчанию = центр ядра, т.е. (-1,-1).

2 Постановка задачи

Для решения задачи оценивания качества контактной работы, реализуемой посредством вебинаров, в ходе дистанционного обучения разработана математическая модель, которая включает систему из 32 показателей качества организации и проведения вебинара, позволяющих оценить с разных сторон качество вебинара 10-балльными экспертными оценками. Проблема заключается в существенных трудозатратах, которые несут эксперты при оценивании этих показателей. Кроме того, мнения экспертов субъективны, а задача поставлена максимально объективизировать процедуру оценивания, например, за счет минимизации влияния человеческого фактора в процедуре оценивания. В связи с этим актуальной является задача разработки такой компьютерной технологии, которая позволит оценить максимальное количество показателей без участия человека в автоматическом режиме.

Предварительный анализ показателей позволил выявить 17 таких, которые можно оценить без участия человека посредством компьютерного анализа аудиовидеозаписи вебинара.

В рамках представленной курсовой работы поставлена задача анализа видеозаписи вебинара, а точнее области конференции, для возможности получения оценки такого критерия как «Энергичность».

3 Функционал программы

3.1 Алгоритм выделение областей вебинара и поиск области конференции

Для решения поставленной задачи были использованы средства библиотеки OpenCV. Первым делом, для работы с областью конференции в вебинаре, ее необходимо сначала выделить из видеопотока вебинара (анализируются 2 последовательно взятых кадра). В программе установлен параметр, который указывает, какой интервал времени (в миллисекундах) должен быть между двумя последовательно взятыми кадрами вебинара, которые затем и будут проанализированы. Предполагается, что все области в вебинаре имеют заголовок с названием соответствующей области. По ним и определяется соответствующая область.

Результатом будет являться структура Regions, имеющая 3 поля типа Rect:

а) webcam (область конференции);

б) chat (область чата);

в) presentat (область презентации).

Все поля представляют собой прямоугольник с соответствующими координатами левого верхнего угла, шириной и высотой.

Для начала, прежде чем работать со слайдом видеопотока вебинара, необходимо произвести его предобработку. Алгоритм предобработки слайда:

а) слайд переводится с помощью порогового преобразования Threshold в бинарное изображение (CV_THRESH_BINARY);

б) получившееся изображение переводится в цветовое пространство HSV с помощью функции cvtColor;

в) в результирующем изображении проводим медианное размытие с ядром 3x3 для избавления от образовавшихся шумов.

После предобработки, на слайде находятся контуры. Результатом будет бинарное изображения, в котором содержатся лишь контуры изображения. Эту операцию проводит функция Canny. Ее синтаксис:

void Canny (CvArr image, CvArr edges, double threshold1, double threshold2, int aperture_size),

где

image — одноканальное изображение для обработки (в градации серого. В нашем случае, оно бинарное);

edges — одноканальное выходное изображение;

threshold1 — порог минимума;

threshold2 — порог максимума;

aperture_size — размер для оператора Собеля. По умолчанию = 3.

Для того, чтобы хранить полученные контуры, используется заранее созданный тип TContours, который представляет собой матрицу точек (элементов типа Point (двумерные точки)), которая будет хранить в себе координаты контуров изображения. Для того, чтобы как-то оперировать контурами, необходимо заполнить матрицу контуров. Для этого используется функция:

FindContours (CvArr image, CvMemStorage storage, int mode, int method, CvPoint offset),

где

image – исходное 8-битное одноканальное изображение (ненулевые пиксели обрабатываются как 1, а нулевые как 0). Для получения такого изображения из градаций серого можно, например, использовать функции cvThreshold() или cvCanny();

storage – хранилище найденных контуров (в нашем случае это элемент типа TContours);

mode — режим поиска (по умолчанию = RETR_LIST);

method – метод аппроксимации.

(по умолчанию = CHAIN_APPROX_SIMPLE);

offset – смещение, на которое сдвигаются точки контура (применимо, если контуры извлекаются из области и затем должны анализироваться в контексте целого изображения). По умолчанию = (0,0).

В качестве режима поиска также доступны:

а) retr_external (0) – найти только крайние внешние контуры. так как необходимо найти лишь внешние прямоугольные контуры областей вебинара, то именно этот режим и используется в программе;

б) retr_list (1) – найти все контуры и разместить их списком;

в) retr_ccomp (2) – найти все контуры и разместить их в виде 2-уровневой иерархии (внешние и внутренние);

г) retr_tree (3) – найти все контуры и разместить их в иерархии вложенных контуров.

В качестве метода аппроксимации также доступны:

а) chain_code (0) – цепной код фридмана;

б) chain_approx_none (1) – все точки цепного кода переводятся в точки (в виде двумерных координат);

в) chain_approx_simple (2) – сжимает горизонтальные, вертикальные и диагональные сегменты и оставляет только их конечные точки. в этом случае, например, прямоугольные контуры будут кодироваться с помощью четырех точек (4 угла). так как все области вебинара имеют прямоугольную форму, то именно этот метод и используется в программе.

Так как интересующие области вебинара имеют форму прямоугольников, то, для удобства работы, лучше оперировать не самими контурами, а минимальными прямоугольниками, которые их покрывают (это также исключит возможность работы с неровными фигурами контуров).

Для того, чтобы получить минимальный прямоугольник, покрывающий почти прямоугольный контур, используется функция:

RotatedRect MinAreaRect(CvArr points),

где

points — последовательность или массив точек (в нашем случае, контур, которые необходимо обвести).

Так как функция возвращает прямоугольник, который может быть повернут на несколько градусов, то работать с ним не получится. Чтобы это было возможно, необходимо, чтобы его стороны были строго параллельны сторонам слайда. У RotatedRect имеется функция BoundingRect(), которая возвращает такой минимальный прямоугольник, описывающий повернутый прямоугольник. Таким образом, прямоугольник был зафиксирован.

После того, как прямоугольники были получены, необходимо произвести их фильтрацию. Из предположения, что границы областей вебинара, являются самыми большими прямоугольниками, первичная фильтрация проходит по размеру полученных прямоугольников. Вторичная фильтрация будет проходить по заголовкам, т.е. по надписи над полученными прямоугольниками. Для проведения вторичной фильтрации, необходимо выделить в отдельное изображение и рассмотреть небольшие прямоугольники над полученными, которые должны содержать в себе название областей вебинара. После их получения, необходимо провести их предобработку, прежде чем будет произведена процедура поиска и распознавания в нем текста. Сначала, необходимо увеличить размер этих изображений. Это операция совершит функция:

void resize (Mat in_im, Mat out_im, Size size),

где

in_im – входное изображение;

out_im – выходное изображение;

size – размер выходного изображения.

После, изображение с помощью порогового преобразования Threshold переводится в бинарное изображение (CV_THRESH_BINARY). Получившееся изображение переводится в цветовое пространство Gray с помощью функции cvtColor. Процедуру поиска и распознавания текста на изображении проведем с помощью библиотеки Tesseract OCR.

Tesseract OCR — открытое ПО, автоматически распознающее и единичную букву, и сразу текст. Tesseract удобен тем, что он есть для любых ОС, стабильно работает и легко обучаем. Однако у него есть существенный недостаток: он очень плохо работает с замыленным, битым, грязным и деформированным текстом.

Алгоритм использования Tesseract для распознавания текста:

а) создается объект класса TessBaseAPI для работы tesseract с изображениями;

б) для созданного объекта устанавливается распознаваемый язык с помощью функции:

void init (string path, string language),

где

path – путь к файлу с данными распознаваемого языка (если все языки были подключены заранее, то значение = null);

language – строка с кратким названием используемого языка (в нашем случае, “rus”. если бы названия областей вебинаров были на английском, то “eng”).

в) установить изображение, текст на котором будет распознаваться с помощью функции:

Setimage (pix im),

где

im – входное изображение.

г) отправить изображение на обработку в tesseract с помощью функции string GetUTF8text().

В результате работы с Tesseract, будет получен текст на изображении. В процессе такого анализа, среди полученных строк будет строка «Конференция». Прямоугольник, в процессе обработки которого, была получена такая строка, и будет обозначать область конференции на вебинаре.

Аналогично будут получены области презентации и чата, в результате чего будет заполнена структура Regions. Информация о расположении областях вебинара (т.е. процедура выделения областей), будет обновляться в течении всего вебинара (установленный интервал времени). В дальнейшем работа будет происходить лишь в области конференции. Остальные области можно использовать для анализа уже других критериев. Результат работы выделения областей вебинара предоставлен на рисунке 4:

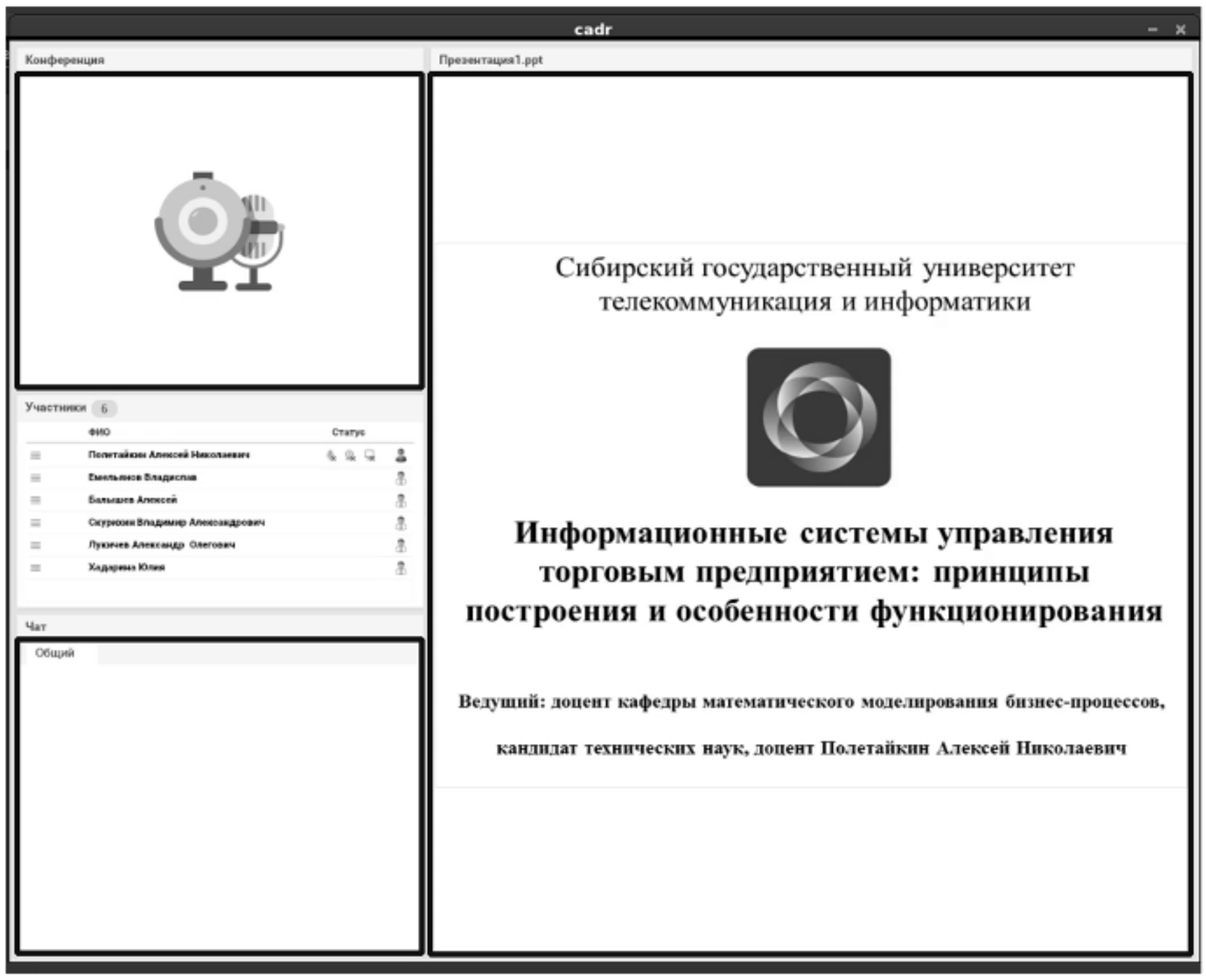

Рисунок 4 – Кадр вебинара с выделенными областями

3.2 Алгоритм обнаружение лица ведущего и подсчет движения головой

Для обнаружения лица ведущего в области конференции, используются каскады Хаара в OpenCV. Основаны они на методе Виолы-Джонса, который описан ниже (п. 3.2.1). Алгоритм поиска лица каскадами Хаара в OpenCV:

а) создается объект класса cascadeclassifier, позволяющий выполнить поиск лица методом виолы-джонса;

б) для работы класса требуется классификатор в формате xml, обученный распознавать какие-либо объекты, например, лицо, глаза и т.д. так как уже обученные классификаторы лежать вместе с дистрибутивом opencv, то остается лишь загрузить необходимый классификатор с помощью:

boolean load (string path + name),

где

path – путь к папке с классификаторами;

name – имя (с расширением xml) необходимого классификатора.

в) произвести поиск лица методом:

void DetectMultiScale(Mat im, MatOfRect matrect),

где

im – входное изображение, на котором ищется объект;

matrect – вектор прямоугольников (заранее созданный тип), содержащий найденное лицо (может и не одно).

После того, как лицо ведущего было найдено (оно содержится в прямоугольнике), его можно, например, обозначить, отрисовав прямоугольник. Для подсчета количества движений ведущего головой, также проходит работа с полученным прямоугольником.

В программе устанавливается 3 параметра, показывающие, какую амплитуду (в пикселях) движения головой по осям X, Y, Z, можно считать за движение.

Такая процедура выполняется через некоторый фиксированный промежуток времени.

Результат работы этой процедуры предоставлен на рисунке 5:

Рисунок 5 – На кадре вебинара выделено лицо ведущего

Основные принципы, на которых основан метод Виолы-Джонса, таковы:

а) используются изображения в интегральном представлении, что позволяет вычислять быстро необходимые объекты;

б) используются признаки Хаара, с помощью которых происходит поиск нужного объекта;

в) используется алгоритм AdaBoost (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;

г) все признаки поступают на вход классификатора, который даёт результат «верно», либо «ложь»;

д) используются каскады признаков для быстрого отбрасывания окон, где не найдено лицо.

Для того, чтобы производить какие-либо действия с данными, используется интегральное представление изображений. Интегральное представление позволяет быстро рассчитывать суммарную яркость произвольного прямоугольника на данном изображении, причем какой бы прямоугольник не был, время расчета неизменно.

Интегральное представление изображения – это матрица, совпадающая по размерам с исходным изображением. В каждом элементе ее хранится сумма интенсивностей (яркостей) всех пикселей, находящихся левее и выше данного элемента. Элементы матрицы рассчитываются по формуле (15):

$$L(x,y) = \sum_{i=0,\ j=0}^{i\leq x,\ j\leq y} I(i,j), \tag{15}$$

где

$I(i,j)$ - яркость пикселя исходного изображения.

К изображениям используется подход на основе сканирующего окна: сканируется изображение окном сканирования, а затем применяется классификатор к каждому положению окна. Система обучения и выбора наиболее значимых признаков полностью автоматизирована и не требует вмешательства человека, поэтому данный подход работает быстро [7].

Признак — отображение $f: X \rightarrow D_f$, где D_f – множество допустимых значений признака. Если заданы признаки $f_1 \dots f_n$, то вектор признаков $x = (f_1(x) \dots f_n(x))$ называется признаковым описанием объекта $x \in X$. Признаковые описания допустимо отождествлять с самими объектами. При этом множество $X = \bigcup_{i=1}^{n} D_{f_i}$ называют признаковым пространством.

В стандартном методе Виолы – Джонса используются прямоугольные признаки (они называются примитивами Хаара), изображенные на рисунке 6:

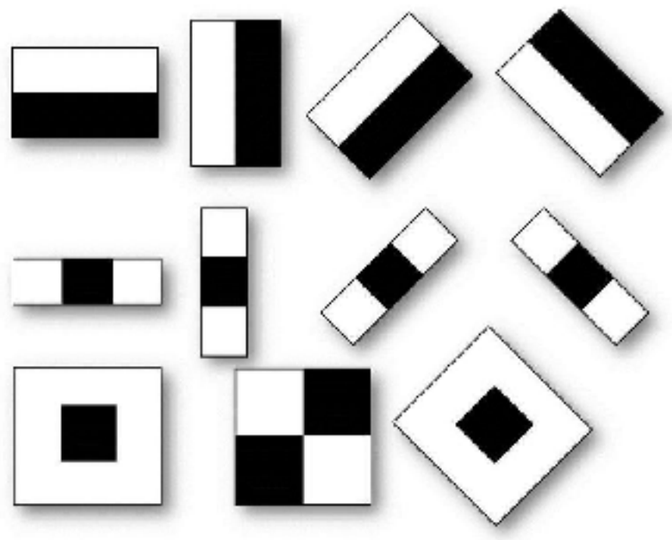

Рисунок 6 – Признаки Хаара

В расширенном методе Виолы – Джонса, использующемся в библиотеке OpenCV, используются дополнительные признаки (т.е. не только симметричные и центральные фигуры), изображенный на рисунке 7:

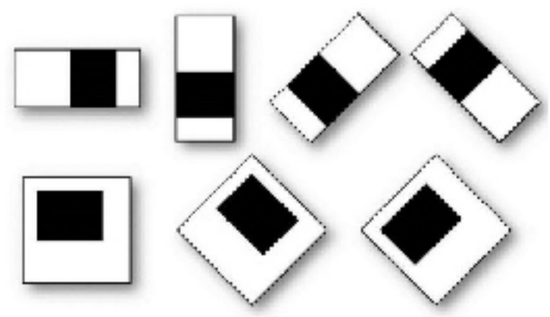

Рисунок 7 – Дополнительные признаки Хаара

Вычисляемым значением таких признаков будет (16):

$$F = X - Y, \tag{16}$$

где

X – сумма значений яркостей пикселей, закрываемых светлой частью признака;

Y – сумма значений яркостей пикселей, закрываемых темной частью признака.

Для их вычисления используется интегрального изображения.

Признаки Хаара дают точечное значение перепада яркости по оси X и Y.

Идея сканирования окна с признаками:

а) есть исследуемое изображение, выбрано окно сканирования;

б) при сканировании изображения в каждом окне вычисляется большое количество признаков, за счет изменения масштаба признаков и их положения в окне сканирования;

в) сканирование производится последовательно для различных масштабов;

г) масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки);

д) все найденные признаки попадают к классификатору, который и выдает ответ.

Чтобы классификатор выдавал правильный ответ, его надо обучить (в OpenCV он уже обучен). Для обучения используется метод AdaBoost. Алгоритм:

а) На вход алгоритма подается обучающая выборка размером m (вида (xi, yi)), где каждому признаку Хаара xi соответствует его классификация yi, т.е. +1 или -1 в зависимости от принадлежности его к лицу или не лицу соответственно;

б) На каждом шаге t=1..T (T – количество этапов обучения классификатора) происходит вычисление значения D_t (весов), для каждого из m примеров. В результате шага, получен слабый классификатор (признак Хаара) h_t, имеющий наименьшую ошибку ε_t для D_t. Затем переход к следующему шагу;

в) Комбинирование слабых классификаторов в один сильный классификатор F(x) (множество слабых классификаторов, которые наилучшим образом определяют лицо на изображении).

3.3 Работа с окружающей областью конференции

Для фиксирования движения ведущего всем остальным телом, кроме головы, используется остальная часть области конференции вебинара. Чтобы понять, как изменился кадр, по сравнению с предыдущим, используем функцию вычитания изображений:

void absdiff(Mat im1,Mat im2, Mat res),

где

im1, im2 – входные изображения, которые вычитаются;

res – выходное изображение.

В этой функции, происходит вычитание пикселей одного изображения из соответствующих пикселей другого изображения.

Полученное изображение перед анализом прошло предобработку. Сначала изображение было переведено в цветовое пространство Gray, а затем с помощью функции порогового преобразования перевелось в бинарное изображение. Для более точного получения результата (и лучшего контроля), была произведена морфологическая операция сужения изображения, т.е. были увеличены темные области прямоугольным ядром 5*5. Изображение, полученное в результате, показано на рисунке 8:

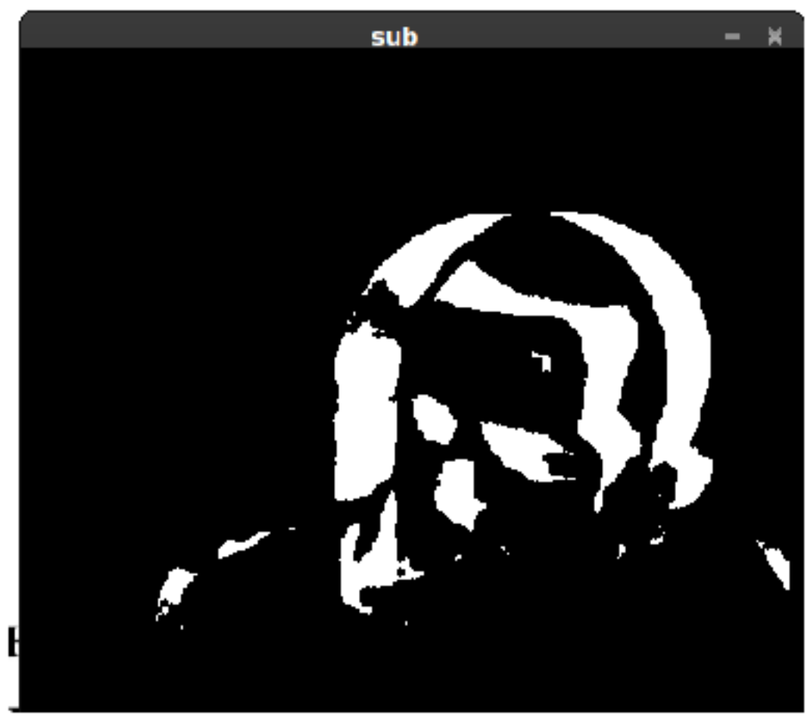

Рисунок 8 – Результат разности двух кадров вебинара

После завершения предобработки, все пространство, которое находится вокруг прямоугольника, описывающего голову ведущего, разделяется на 4 части: северная, южная, западная и восточная. Если хотя бы в одной из них количество ненулевых пикселей будет > 0 (это означает, что ведущий совершил движение), то счетчик движения увеличивается на 1. Количество ненулевых пикселей в изображении считается с помощью функции:

int countNonZero (Mat im),

где

im – входное изображение.

4 Результат работы программы

После проведения анализа видеопотока предоставленного вебинара, а точнее области конференции, было получено, что при установленных амплитудах 10, 10, 10 по осям X, Y, Z соответственно и интервала 1000 миллисекунд между анализируемыми кадрами, результат будет точнее (более верным) по сравнению с интервалами времени 100 или 3000 миллисекунд. Это так, так как при установлении малого интервала времени, движения проходят более плавно. Это означает, что амплитуда движения будет также малой, что приводит к потере (не учитывании) некоторых совершённых ведущим движений. При установлении большого интервала времени, также возможна потеря некоторых движений, так как они были совершены в не рассматриваемых кадрах. Таким образом, наиболее подходящие интервалы времени находятся, примерно, на отрезке $[500,\ 1500]$.

Так, при интервале 100(3000) миллисекунд, в течении всего вебинара ведущим было совершено 632(1048), 360(978), 462(698) движений головы по осям X, Y, Z соответственно и 545(899) движений остальным телом. При интервале 1000 миллисекунд, в течении всего вебинара ведущим было совершено 2286, 1878, 1414 движений головы по осям X, Y, Z соответственно и 1827 движений остальным телом. Для наглядного рассмотрения зависимости точности результата от выбранного интервала времени, была построена диаграмма на рисунке 9, отображающая такую зависимость:

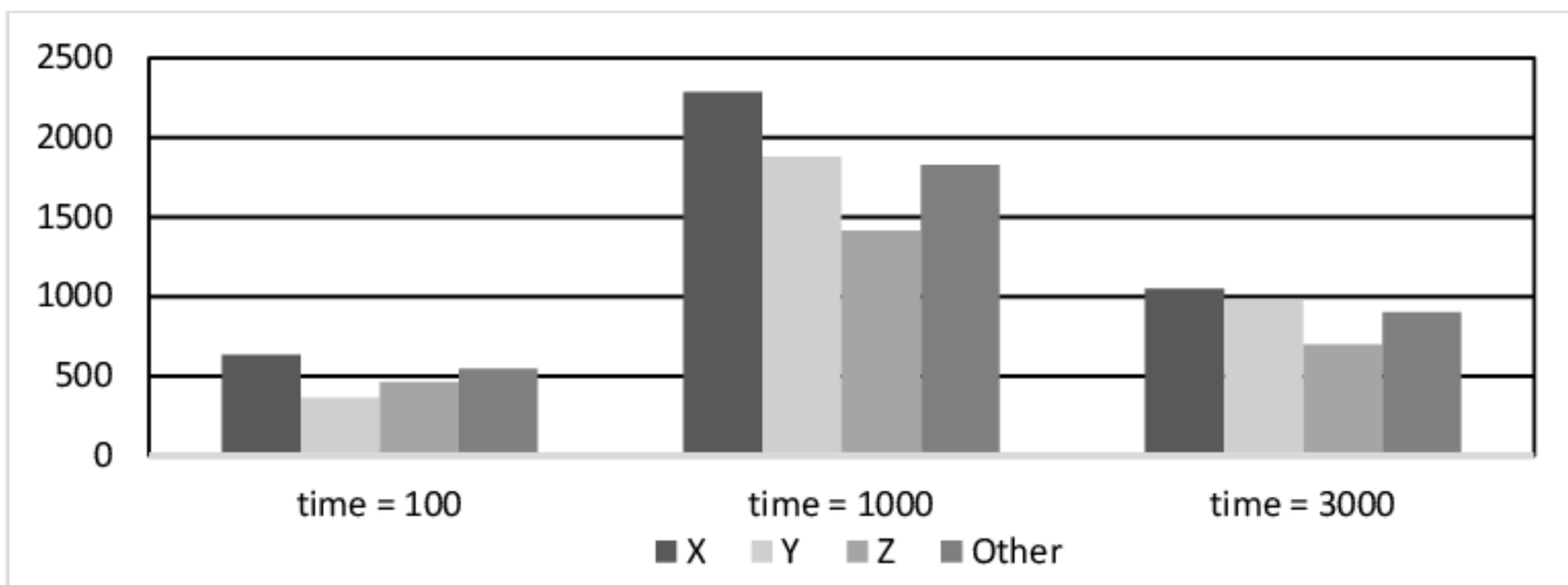

Рисунок 9 – Диаграмма зависимости

Также есть возможность просматривать вебинар (и его любые промежуточные изображения) параллельно с работой программы, например, для анализа других критерий.

ЗАКЛЮЧЕНИЕ

В курсовой работе был проведен анализ области конференции вебинара (области, где отображается ведущий вебинара), с целью частичной автоматизации процесса оценки эффективности вебинаров.

Также были изучены функции библиотеки OpenCV, и на основе этих функций, было создано программное приложение с целью минимизации влияния человеческого фактора в процедуре оценивания эффективности вебинаров, сделав ее максимально объективной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Как оценивать эффективность вебинаров. – URL: https://myownconference.ru/blog/index.php/kak-otsenivat-efectivnost-webinarow/ [2 апреля 2019].

2 Adrian Kaehler., Gary Bradski. Learning OpenCV. USA.: September 2008 – 543с.

3 OpenCV шаг за шагом. Обработка изображения. – URL: http://robocraft.ru/page/opencv/ [21 апреля 2019].

4 Детектор границ Канни. – URL: https://habr.com/ru/post/114589/ [2 мая 2019].

5 Описание и кодирование контуров. Цепной код Фримана. – URL: https://moluch.ru/archive/193/48447/ [20 мая 2019].

6 Морфологические преобразования. – URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html [10 мая 2019].

7 Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц. – URL: https://habr.com/ru/post/133826/ [26 мая 2019].