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 ТЕКСТОВОЙ ИНФОРМАЦИИ ВЕБИНАРОВ

Работу выполнил _____Д.А. Ляхов
(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 содержит 46 страниц, 6 рисунков, 2 таблицы, 5 источников.

ВЕБИНАР, КОРРЕЛЯЦИЯ ИЗОБРАЖЕНИЙ, ПРЕОБРАЗОВАНИЕ ФУРЬЕ, ЛОКАЛИЗАЦИЯ ТЕКСТА, TESSERACT, РАСПОЗНАВАНИЕ ТЕКСТА

Объектом исследования являются методы выделения текстовой информации из изображений.

Цель курсовой работы – программная реализация процедур, вычисляющих основные критерии оценки вебинаров.

В результате работы были реализованы процедуры, реализующие

- поиск указки;
- выделение слайдов;
- распознавание блоков текста на слайдах.

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 стоит задача оценки качества контактной работы, реализуемой посредством вебинаров, в ходе дистанционного обучения.

Для решения этой задачи разработана математическая модель, которая включает систему из 32 показателей качества организации и проведения вебинара (перечень которых представлен в приложении), 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. В числе показателей, которые могут быть автоматически при помощи некоторого алгоритма входят: колличество и качество информации на слайдах, использование указки, точность заявленной длительности мероприяти и тд.

Основой для анализа данных показателей является три основных аспекта:

- поиск указки;
- выделение слайдов;
- распознавание блоков текста на слайдах.

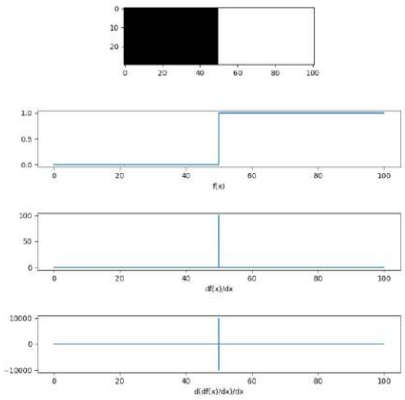
Для решения вышеперечисленных задач в ходе курсовой работы проведен анализ следующих процедур над изображениями:

- поиск границ объектов;
- поиск вхождения одного изображения в другое;
- вычисление коэффициента схожести изображений;
- локализация текста.

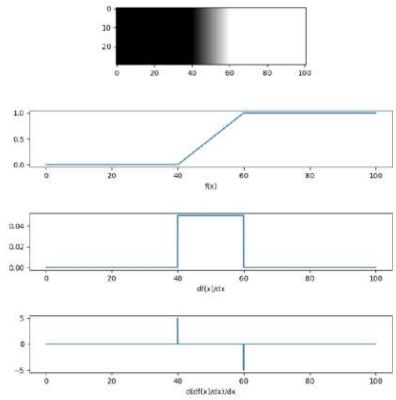
Программно разработаны процедуры, реализующие основу для дальнейшего развития приложения автоматического вычисления основных критериев оценки вебинаров. В реализации используется бесплатная библиотека машинного зрения OpenCV.

1 Методы определения границ объектов на изображении

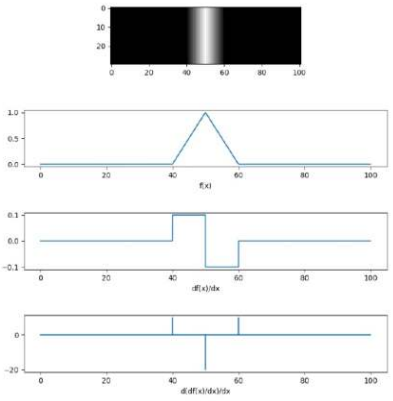
Границы объектов на изображении характеризуются изменением яркости в некотором направлении. Выделяют три вида границ: идеальные, размытые и крышевидные. На рисунке 1.1 изображены вертикальные границы трех видов.



а) Идеальная граница



б) Размытая граница



в) Крышевидная граница

Рисунок 1.1 – границы объектов: изображение, функция яркости, первая и вторая производные

Рассмотрим вертикальную размытую границу. Тогда зафиксировав ординату, получим дискретную функцию от одной переменной $g(x)$.

Рассмотрим её первую и вторую производные: первая производная равна нулю в областях, где интенсивность постоянна и равна константе на границе, причем константа тем больше, чем уже граница. Вторая производная не равна нулю только в координатах начала и конца границы $g(x)$, в этих точках она равна бесконечности. Для случая крышевидной границы, первая производная положительна на подъеме и отрицательна на спуске границы. Вторая производная не ноль в трех точках, причем граница помещается между первой и последней не нулевой точкой второй производной. Отсюда следует вывод: зная направление границы, её координаты находятся из производной функции интенсивности по этому направлению.

В общем случае, если заранее неизвестны направления границ объектов, за направление берут направление максимального роста интенсивности то есть градиент изображения, если представлять его как дискретную функцию от двух переменных.

Градиент есть вектор частных производных:

$$\nabla f(x,y) = grad[f(x,y)] = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{1.1}$$

На практике производные вычисляются численно, разностными методами. Если принять расстояние между соседними в строке и соседними в столбце пикселями за единицу, компоненты градиента с точностью $O(1)$ вычисляется по формулам:

$$g_x(x,y) = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{1.2}$$

$$g_y(x,y) = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \tag{1.3}$$

Что равносильно свертке изображения [5] с ядрами

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \end{bmatrix} \tag{1.4}$$

На практике для вычисления градиента используются оператор Собеля:

$$\begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{1.5}$$

Конечное изображение вычисляется по формуле

$$G = |G_x| + |G_y| \tag{1.6}$$

Где G_x и G_y - результат свертки изображения с первым и вторым ядром оператора Собеля соответственно.

В результате применения оператора Собеля на изображении белым цветом выделены предполагаемые границы. Градиентный метод Собеля выделения границ широко используется в прикладных задачах благодаря своей простоте и высокой точности. К недостаткам метода Собеля можно отнести зависимость от шумов: при достаточно высоком их содержании алгоритм не способен адекватно определить границы объектов.

2 Методы сравнения изображений

2.1 Сравнение гистограмм

Метод сравнения гистограмм - один из самых простых и быстрых способов сравнения двух изображений. Алгоритм основан на предположении о том, что похожие изображения имеют похожие цвета.

Гистограмма это график или функция распределения элементов цифрового изображения с различной яркостью. Гистограмма определена на множестве всех возможных значений яркостей чёрно-белого изображения, значение функции равно количеству пикселей, яркость которых равна аргументу гистограммы. В общем случае значение яркости - n мерный вектор. Обычно для сравнения цветных изображений используются каналы HS цветового пространства HSV. Это основано на том, что при сравнении цвета разной яркости не различают, а поэтому канал V, отвечающий за яркость цвета игнорируют.

Для сравнения гистограмм в openCV используется одна из метрик. Сравнение работы алгоритма с разными метриками на тестовых рисунках 2.1 представлено в таблице 2.1.

Correlation CV_COMP_CORREL

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \overline{H}_1)(H_2(I) - \overline{H}_2)}{\sqrt{\sum_I (H_1(I) - \overline{H}_1)^2 \sum_I (H_2(I) - \overline{H}_2)^2}}$$

$$\overline{H}_k = \frac{1}{N} \sum_J H_k(J)$$

Chi-Square (CV_COMP_CHISQR)

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

Intersection (method=CV_COMP_INTERSECT)

$$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I))$$

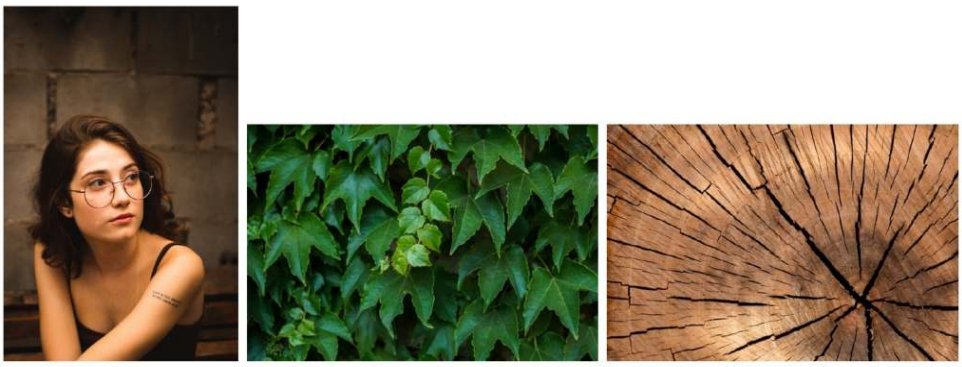
Bhattacharyya distance (CV_COMP_BHATTACHARYYA)

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H}_1 \overline{H}_2 N^2}} \sum_I \sqrt{H_1(I) H_2(I)}}$$

Где I в общем случае вектор, сумма идет по всем возможным векторам.

Достоинства данного метода - инвариантность относительно формы и размера сравниваемых изображений.

Главный недостаток метода сравнения гистограмм - разделение изображений весьма условно. Каждому изображению подобно всё множество изображений, составленных из тех же пикселей, расположенных в произвольном порядке. Поэтому данный метод не подходит, например, для задачи поиска курсора так как существует вероятность того, что на изображении, кроме курсора, могут находиться элементы, гистограммы которых очень близки к гистограмме курсора.



а) girl.jpg б) leafs.jpg в) wood.jpg

Рисунок 2.1 – изображения, используемые в сравнительном тесте

Таблица 2.1 – сравнение результатов сравнения изображений по гистограмме

	girl.jpg сравнить с girl.jpg	girl.jpg сравнить с leafs.jpg	girl.jpg сравнить с wood.jpg	wood.jpg сравнить с leafs.jpg
CV_COMP_CORREL	1.000000	-0.028190,	0.639221	-0.018233
CV_COMP_CHISQR	0.000000	47.621748,	131.495723	23476.763941
CV_COMP_INTERSECT	13.806632	0.057196,	6.012272	0.124236
CV_COMP_BHATTACHARYYA	0.000000	0.996709,	0.449109	0.987969

2.2 Совпадения по шаблону

Методы сравнения по шаблону используются для нахождения координат малого шаблонного изображения в большом. Алгоритм поиска таков: на первом этапе происходит нелинейная фильтрация изображения. Как и в остальных фильтрах, для фильтрации используется скользящее окно [5]. Размер скользящего окна равен размеру малого изображения. Все возможные подобласти размера окна из большого изображения поэлементно сравниваются с малым изображением посредством одной из метрик:

method=CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \sum_{x',y'} I(x+x',y+y')^2}}$$

method=CV_TM_CCORR

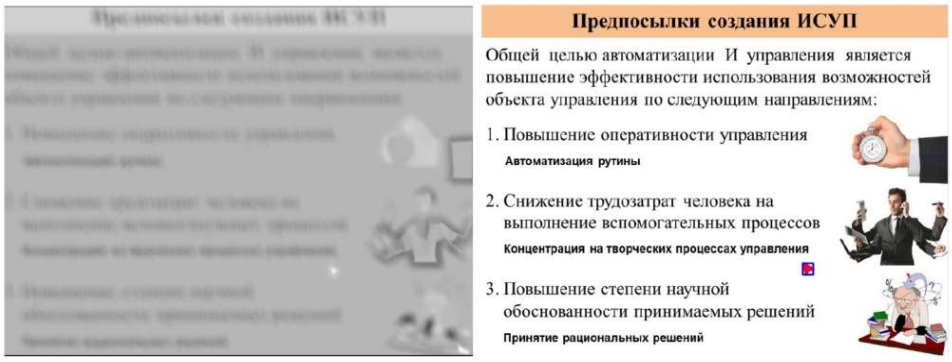
$$R(x,y) = \sum_{x',y'} (T(x',y') I(x+x',y+y'))$$

method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \sum_{x',y'} I(x+x',y+y')^2}}$$

На втором этапе полученная двумерная матрица нормируется, и в зависимости от используемого метода находится её минимальный или максимальный элемент. Иллюстрация работы алгоритма поиска по шаблону представлена на рисунке 2.2 .

а) Курсор б) Исходное изображение



в) Результат применения свёртки г) На исходном изображении локализован курсор

Рисунок 2.2 – пример работы алгоритма поиска изображения по шаблону

Достоинство метода - высокая точность нахождения фрагментов изображений.

Недостатки - высокая точность достигается только в случае, когда фрагмент того же размера и не повернут относительно шаблона.

Существуют также методы, инвариантные относительно размера и положения искомого элемента и перспективы. Их называются *методами совпадения по признакам*. Суть методов совпадения по признакам заключается в том, что из шаблонного изображения выделяются некоторые характерные признаки. После, в большом изображении находится объекты, схожие по признакам с шаблоном. Этот класс методов избыточен для решаемой в этой работе задаче.

3 Преобразование Фурье

Для анализа аналоговых и цифровых сигналов зачастую используются дискретные преобразования, переводящие сигнал из одного базиса в другой. Многие операции над сигналом значительно упрощаются в частотном базисе, позволяя ускорить алгоритмы машинного зрения. Рассмотренные ниже преобразования представляют сигнал в виде суперпозиции частот.

3.1 Непрерывное преобразование Фурье

Непрерывное преобразование Фурье имеет вид:

Прямое:

$$F(s) \equiv \mathcal{F}\{f(x)\}(s) \equiv \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i s x}\, dx \tag{3.1}$$

Обратное:

$$f(x) \equiv \mathcal{F}^{-1}\{F(s)\}(x) \equiv \int_{-\infty}^{\infty} F(s)\, e^{2\pi i s x}\, ds \tag{3.2}$$

Где $f(x)$ - непрерывная функция - сигнал вещественной переменной, $F(x)$ - непрерывная функция комплексной переменной - Фурье образ $f(x)$ [2].

Сверткой называют интегральное преобразование вида:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)\, g(x-y)\, dy \; = \int_{-\infty}^{\infty} f(x-y)\, g(y)\, dy \tag{3.3}$$

Корреляцией двух функций называют преобразование вида:

$$(f \star g)(x) = \int_{-\infty}^{\infty} f(y)\, g(x+y)\, dy \; = \int_{-\infty}^{\infty} f(x+y)\, g(y)\, dy \tag{3.4}$$

Доказана теорема свертки

$$(f * g)(x) \equiv \mathcal{F}^{-1}\{\mathcal{F}(f) \cdot \mathcal{F}(g)\} \tag{3.5}$$

Аналогично теореме о свертке, существует теорема о корреляция

$$(f \star g)(x) \equiv \mathcal{F}^{-1}\{\mathcal{F}^{*}(f) \cdot \mathcal{F}(g)\} \tag{3.6}$$

Где операция $\cdot$ – скалярное произведение, а $\mathcal{F}^{*}(f)$ - комплексно сопряженный образ Фурье функции f .

3.2 Дискретное преобразование Фурье

Известно, что компьютер оперирует исключительно конечным набором цифровых данных и не может напрямую обрабатывать аналоговые сигналы. Поэтому первым этапом компьютерной обработки любого сигнала является его дискретизация т.е представление в виде конечной последовательности точек аргумент- значение. Рассмотрим дельта функцию

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}, \quad \int_{-\infty}^{\infty} \delta(t)\, dt = 1$$

Имеющую свойства

$$\int_{-\infty}^{\infty} f(t)\delta(t)\, dt = f(0)$$

$$\int_{-\infty}^{\infty} f(t)\delta(t - t_0)\, dt = f(t_0)$$

Если функция $f(t)$ - дискретная, то рассматривается дискретная дельта функция

$$\delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases} \tag{3.7}$$

Тогда

$$\sum_{t=-\infty}^{\infty} \delta(t) = 1 \tag{3.8}$$

$$\sum_{t=-\infty}^{\infty} f(t)\delta(t - t_0) = f(t_0) \tag{3.9}$$

Рассматривая функцию, называемую серия импульсов (в англоязычной литературе train of impulses)

$$s_{\Delta T}(t) = \sum_{k=-\infty}^{\infty} \delta(t - k\Delta T) \tag{3.10}$$

И её образ Фурье

$$S(\mu) = \mathcal{F}\{s_{\Delta T}(t)\} = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta(\mu - \frac{n}{\Delta T}) \tag{3.11}$$

Можем построить математическую модель дискретизации функции $f(t)$ с соответствующим Фурье образом $F(s) = \mathcal{F}\{f\}(s)$ следующим образом

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) \tag{3.12}$$

Тогда рассматривая образ Фурье дискретной функции, используя теорему свертки 3.5, можно получить [2]

$$\tilde{F}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F(\mu - \frac{n}{\Delta T}) \tag{3.13}$$

Пусть значения $F(\mu)$ ограничены $[-\mu_{max}, \mu_{max}]$ т.е. частоты $f(t)$ ограничены (в англоязычной литературе band limited functions) и пусть частота при дискретизации удовлетворяет неравенству:

$$\frac{1}{\Delta T} > 2\mu_{max} \tag{3.14}$$

Из 3.13 и предыдущего неравенства очевидно, что $\tilde{F}(\mu)$ и $F(\mu)$ связаны соотношением

$$F(\mu) = H(\mu)\tilde{F}(\mu) \tag{3.15}$$

Где

$$H(\mu) = \begin{cases} \Delta T & \mu \in [-\mu_{max}, \mu_{max}] \\ 0 \end{cases} \tag{3.16}$$

Обратное преобразование $H(\mu)$ имеет вид

$$h(t) = \mathcal{F}^{-1}\{H(\mu)\}(t) = \int\limits_{-\infty}^{\infty} H(\mu)e^{2\pi i\mu t}d\mu \; = \int\limits_{-\mu_{max}}^{\mu_{max}} \Delta T e^{2\pi i\mu t}d\mu \; =$$

$$= \frac{\Delta T}{i2\pi t}\left[e^{2\pi i\mu t}\right]_{-\mu_{max}}^{\mu_{max}} = 2\Delta T\mu_{max}\frac{sin(2\pi t\mu_{max})}{(2\pi t\mu_{max})} = 2\Delta T\mu_{max}sinc(2t\mu_{max})$$

Отсюда следует вывод: допуская, что частота $f(t)$ ограничена по модулю значением μ_{max}, зная $\tilde{f}(t)$, которая получена сэмплированием из $f(t)$ с частотой не меньше, чем $2\mu_{max}$, можем приблизить $f(t)$ с любой точностью.

Вычислить $f(t)$ через $\tilde{f}(t)$ можно следующим образом:

$$\begin{aligned} f(t) &= \mathcal{F}^{-1}\{F(\mu)\} \\ &= \mathcal{F}^{-1}\{H(\mu)\tilde{F}(\mu)\} \\ &= h(t) * \tilde{f}(t) \end{aligned}$$

или

$$f(t) = 2\mu_{max}\Delta T \sum_{n=-\infty}^{\infty} f(n\Delta T)sinc[2\mu_{max}(t - n\Delta T)] \tag{3.17}$$

Если же $\Delta T = \frac{1}{2\mu_{max}}$:

$$f(t) = \sum_{n=-\infty}^{\infty} f(n\Delta T)sinc[(t - n\Delta T)/\Delta T] \tag{3.18}$$

Данный вывод называется *теоремой Котельникова или сэмплирования.* Этот подход широко используется в обработке изображений и звука. Например зная, что человек слышит звуки частоты от 16 до 20 000 Гц можно без какой-либо потери информации для человеческого слуха записывать звуки с частотой дискретизации 44100 Гц.

3.3 Двумерное преобразование Фурье

Все теоремы и свойства, рассмотренные выше, распространяются на случай функции n переменных. Рассмотрим двумерный случай. Для непрерывного сигнала $f(x, y)$ преобразование Фурье имеет вид

$$F(\mu, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z)\, e^{-i2\pi(\mu t + \nu z)} dz dt \tag{3.19}$$

$$f(t, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\mu, \nu)\, e^{i2\pi(\mu t + \nu z)} d\mu d\nu \tag{3.20}$$

Последовательность импульсов в двумерном случае будет выглядеть так:

$$S_{\Delta T \Delta Z}(t, z) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - m\Delta T, z - n\Delta Z) \tag{3.21}$$

Предполагая, что частоты функции $f(x, y)$ ограничены в $[-\mu_{max}, \mu_{max}] \times [-\nu_{max}, \nu_{max}]$ для “сжатия функции без потерь” достаточно взять

$$\Delta T < \frac{1}{2\mu_{max}} \tag{3.22}$$

$$\Delta Z < \frac{1}{2\nu_{max}} \tag{3.23}$$

Цифровые изображения хранятся как набор пикселей. Пусть дано изображение $M \times N$. Чтобы представить его в пространственном базисе,

можно, например, положить, что один пиксель есть единица измерения пространства. Тогда изображение моделируется формулой вида

$$\tilde{f}(t,z) = f(t,z)S_{\Delta T \Delta Z} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(t,z)\delta(t - m\Delta T, z - n\Delta Z) \qquad (3.24)$$

Где $\Delta t = \Delta z = 1$. Тогда соответствующие частоты будут изменяться в пределах $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ согласно 3.23.Таким частотам соответствуют периоды в пикселях: $[..., -3, -2] \cup [2, 3, ...]$. Подставляя выражение 3.24 в прямое преобразование Фурье 3.19, получим

$$F(\mu,\nu) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\mu x/M + \nu y/N)} \qquad (3.25)$$

Подставляя 3.25 в обратное преобразование 3.20, получим

$$F(x,y) = \sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1} f(\mu,\nu)e^{i2\pi(\mu x/M + \nu y/N)} \qquad (3.26)$$

откуда видно, что для восстановления исходного изображения достаточно всего $M \times N$ значений образа Фурье в точках $[0, M-1] \times [0, N-1]$. Формулы 3.25 и 3.26 называется прямым и обратным двумерным конечным дискретными преобразованием Фурье соответственно.

4 Приложения преобразования Фурье

4.1 Сжатие и увеличение изображений

Так как все Фурье образы - периодические функции как суперпозиция синусов и косинусов, то образ Фурье можно вычислить для любой точки пространства. Тогда в выражении 3.25 в суммах можем использовать любые пределы. Тогда, чтобы изменить размер изображения с $M \times N$ на произвольный $m \times n$, $m > 0$, $n > 0$, достаточно просто поменять пределы в формуле 3.25.

4.2 Свертка изображений

Как было показано выше (3.5), свертка двух функций в пространственном базисе эквивалентна по элементному произведению в частотном. В Частности, в случае изображений: вместо очевидного метода свертки, когда окно скользит по изображению и на каждой итерации вычисляется значение пикселя, можно перевести изображение и ядро фильтра в частотный базис посредством прямого преобразования Фурье, выполнить скалярное произведение двух образов и последним шагом преобразовать произведения образов в пространственный базис с помощью обратного преобразования Фурье.

Очевидный метод требует $MNmn$ операций для свертки изображения размера $M \times N$ с ядром размерности $m \times n$. Свертка же посредством преобразования Фурье, вместе со всеми преобразованиями требует $2MNlog_2(MN)$ Операций, что в случае большой размерности ядра значительно уменьшает число вычислений.

4.3 Алгоритм быстрой кросс кореляции

Данный алгоритм - разновидность поиска по шаблону [3]. Главное его отличие в том, что метрика представляет из себя функцию корреляции. Благодаря этому для нахождения шаблона в изображении можно использовать преобразование Фурье и значительно уменьшить количество вычислений. Рассмотрим метрику вида:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}][t(x-u,y-v) - t]}{\sqrt{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - t]^2}} \tag{4.1}$$

где

$$t'(x,y) \equiv t(x,y) - \overline{t}$$

$$f'(x,y) \equiv f(x,y) - \overline{f}_{u,v}$$

Тогда можем вычислить числитель как

$$\gamma_{num}(u,v) = \mathcal{F}^{-1}\{\mathcal{F}(f')\mathcal{F}^*(t')\} \tag{4.2}$$

В знаменателе второе слагаемое всегда константа. Первое слагаемое можно вычислить, используя только $3M^2$ вместо $3N^2(M - N + 1)$ операций, если использовать подход бегущей суммы (интегрирование изображений). Сравнивая методы поиска шаблона можно заметить, что методы быстрой кросс корреляция работает на порядок быстрее алгоритмов, не использующих преобразование Фурье.

Данный алгоритм был разработан специально для фильма “Форест Гамп”(1994) и использовался в большом количестве других проектов. Во время производства фильма было необходимо вырезать и заменить некоторые части изображения из видеопотока. Шаблон вручную

выделялся, а после автоматически отслеживался на протяжении всей сцены посредством алгоритма быстрой кросс-корреляции. Найденные координаты использовались для дальнейших спецэффектов.

Таблица 4.1 – сравнения времени работы стандартного и ускоренного алгоритма кросс корреляции. Тест предполагал координаты шаблона в кадрах из фильма “Форест Гамп”

Размер шаблона	Кол-во кадров	Время работы стандартного алгоритма	Время работы ускоренного алгоритма
168×86	896	15 часов	1.7 часа
115×200 150×150	490	14.4 часа	57 минут

5 Методы локализации текста на изображении

Проблема распознавания текстовых изображений появилась давно [4]. Системы распознавания текста из изображения (Optical Character Recognition) нашли своё применения во многих отраслях жизни:

- чтение регистрационных знаков транспортных средств;
- определение и чтение дорожных знаков;
- мобильное приложение для слабовидящих, распознающее и проговаривающее текст из изображения;
- ввод информации;
- оцифровка или архивация документов, видеопотока с камеры наблюдения и так далее.

На сегодняшний день алгоритмы локализации текста развиты достаточно хорошо. Тем не менее, они не могут дать абсолютно точный результат так как входные данные могут быть подвержены искажениям. Основные из них

- низкое разрешение;
- неравномерное освещение;
- геометрические искажения;
- дисторсия;
- сложный фон;
- шумы итд.

Задача распознавания текста не может быть поставлена точно: в некоторых ситуациях текст невозможно распознать физически, буквы не могут быть распознаны однозначно. Этот факт не уменьшает ответственности, возложенной на разработчика алгоритма распознавания

образов.

Первый этап большинства алгоритмов распознавания текста - его локализация. Главное отличие текста от фона и изображений - высокий контраст. Кроме того, текст чаще всего имеет некоторый шрифт, у которого есть постоянный размер. На изображениях условно выделяют два вида текстовой информации: текст, который нанесен на объекты съемки то есть надписи на транспорте, вывески, реклама и текст, который нанесен поверх изображения или из сканированного документа и так далее.

5.1 Градиентные методы локализации текста

Исходя из предположения о том, что символы имеют четкие границы, в отличие от другой информации, можно предположить, что пиксели, значения градиента в которых велико, являются границами изображенных символов. После того, как алгоритмы локализации текста, основанные на градиенте, найдут предполагаемые границы символов, эти границы группируются в текстовые блоки. Примеры реализации методов локализации текста, основанные на градиенте:

Исследователи Лии и Канканхалли активно используют производную по оси x. Кувано дополнил предыдущие подходы: он предполагал текстовые блоки по оси x и y отдельно, а затем объединял результаты, оставляя только пересекающие области. Для вычисления компонент градиента исследователи использовали ядра Собеля.

Для группировки границ в области, разработчики используют морфологические оперции: дилатацию и наращивание [5]. В некоторых работах на последнем этапе проводится верификация результата предыдущих этапов.

Исследователи Вонг и Чен используют в своем алгоритме только

производную по x. Они предполагают, что дисперсия производной в районе текстовых блоков будет выше, чем в других областях. Применяя к результату вычисления пороговую бинаризацию, предполагаются границы текстовых блоков. Кроме того, разные исследовали рассматривают разные каналы изображений, применяя к каждому из них операцию градиента.

Исследователь Ким в своём градиентном методе локализации номерных знаков кроме градиента изображения использует дисперсию градиента, плотность и дисперсию плотности границ символов. Этот метод основывается на предположении о том, что область номерного знака имеет большую дисперсию градиента, высокую плотность вычисленных градиентом границ и низкую дисперсию этой плотности.

Кроме того, некоторые исследователи используют для поиска областей текстовой информации нейронные сети. Исследователь Хуа на промежуточном этапе, после выполнения операции выделения границ, вычисляет углы границ символов с помощью нейронных сете и впоследствии использует эту информацию для улучшения результата работы алгоритма.

Семейство градиентных алгоритмов показывает высокую производительность, и при достаточно высоком качестве входных данных и малых искажениях работает точно.

5.2 Алгоритмы локализации текста, основанные на цветовых характеристиках изображения

В основе семейства алгоритмов, основанных на цветовых характеристиках изображения, лежат алгоритмы сегментации изображений по цвету. Предполагается, что текст имеет определенный цвет и яркость, что и отделяет его от фона. После сегментации, также,

как в градиентных методах, границы группируются и верифицируются. В общем случае, результат разделения изображения по цвету используется для выделения не только текста, но и любых объектов, схожих по цвету.

Например для сегментации, исследователь Миен использовал быстрый алгоритм сегментации по цвету, вычисляющий результат за один проход. Этот алгоритм можно описать следующим образом: сначала сравниваются пиксели, соседние в строке, а затем в столбце. На базе этой информации принимается решение, если ли в данной координате граница между цветом или нет.

Другой популярный подход к цветовой сегментации изображений - квантование цветов. Квантование по сути - разбиение диапазона отсчетных значений сигнала на конечное число уровней и округление этих значений до одного из двух ближайших к ним уровней. Главное преимущество квантования – высокая степень устойчивости к шумам.

Использование несколько цветовых пространств – распространенная практика при квантовании цветов. Исследователь Ли объединил результаты квантования из 27 различных цветовых индексов, рассматривая отдельные биты значений цветов как каналы каналы.

5.3 Алгоритм локализации текста, основанный на текстуре

Под текстурой подразумевают визуальные свойства каких-либо поверхностей или объектов.Предполагается, что текст имеет уникальную, отличную от фона текстуру. Текстурная информация может Человек может отличить текст от фона даже тогда, когда язык текста ему неизвестен. Он отличает символы по текстуре.

Описать текстуру изображения можно многими способами. Исследователь Шонг предполагает, что области, содержащие текстовую

информацию, содержат определенные высокие частоты, которые вычисляются при помощи дискретных преобразований. В своём алгоритме Шонг использовал дискретное косинус преобразование, которое по сути мало отличается от дискретного преобразования Фурье. Ву для выделения структуры изображения сворачивал изображение с гауссианой и после классифицировал области алгоритмом k- среднего.

Для выделение текстурных особенностей широко используются нейронные сети. Исследователь Юнг использовал многослойный перцептрон; нейронная сеть последовательно принимала на вход всевозможные малые области изображения и выносила вердикт: представлен символ или нет. Однако этот подход не типичен, чаще из изображения некоторым способом выделяются особенности, которые уже после классифицируются. Особенность изображения можно определит как некоторый объект, характерный данному изображению. Также, как и в других семействах алгоритмов, некоторые из представителей семейства текстурных алгоритмов рассматривают несколько каналов одного изображения, а затем объединяют результат. Нейронный классификатор обычно обучается на особенностях, выделенных из малых изображений.

Сравнивая семейства методов локализации текста, авторы статьи [4] пришли к выводу, что универсального алгоритма, который был бы лучше остальных по всем параметрам, нет. Семейство текстурных алгоритмов имеет ряд преимуществ: устойчивые к шумам, способные на самообучение и имеющие некоторый запас уверенности в результате, они требуют на порядок больше вычислений, чем алгоритмы, основанные на разделении цвета и градиенте, которые имеют на порядок меньшую точность в сравнении с первым.

Возможное решение этой задачи - композиция алгоритмов. На

первом этапе области выделяются градиентным методом или алгоритмом, основанным на цветовой разнице. А для верификации результатов использовать один из трудозатратных алгоритмов, работающих с текстурой.

Возможный подход к верификации быстрых алгоритмов локализации текста - попытка распознать полученный на предыдущем этапе текстовый блок.

6 Программная реализация

6.1 Постановка задачи

Для ускорения процесса оценки вебинаров необходимо выделить из видеопотока слайды презентации. На каждом слайде следует выделить и распознать текст. Реализация этой задачи разбилась на три подзадачи:

- сравнение двух изображений;
- поиск изображения указки;
- выделение слайдов из видеопотока;
- выделение текста из изображения слайда.

Для реализации указанных алгоритмов используется вспомогательная процедура сдвига видеопотока на величину shift и вычисления разности между текущим кадром и кадром, считанным до сдвига. Разность вычисляется как абсолютная разность между векторами - значениями пикселей. Будем называть разность изображений нулевой, если результат будет отличаться от черного изображения только шумом.

6.2 Алгоритм сравнения изображений

Рассматривается подзадача, заключающаяся в вычислении некоторой корреляции между парой изображений. Алгоритм сравнения изображений получает на вход два экземпляра изображений и возвращает вещественное число из $[0, 1]$, где 0 - абсолютно разные изображения, 1 - идентичные изображения. В программной реализации алгоритма сравнения изображений используется описанные выше методы сравнения совпадения по шаблону.

Алгоритм сравнения изображений:

1) на вход два изображения x, y;
2) если отношение max(ширина x, ширина y) / min (ширина x, ширина y) > 1.5 или max(длина x, длина y) / min (длина x, длина y) > 1.5 – то изображения разные, вернуть 0;
3) если max (кол-во не черных пикселей x, кол-во не черных пикселей y) / min (кол-во не черных пикселей x, кол-во не черных пикселей y) > 10 - изображения разные, вернуть 0;
4) вычислить относительное положение:
    а) длины сторон x меньше, чем у y;
    б) длины сторон y меньше, чем у x;
    в) одна из сторон x больше, чем у y и одна из сторон y больше, чем у x;
5) если а - $z = x$,$x = y$,$y = z$;
6) если в - добавить к элементу y белую границу справа на ширину x, снизу на длину x
7) с помощью бинаризации вычислить маску для x
8) используя метод совпадения по шаблону с метрикой TM_CCORR_NORMED найти координаты наиболее похожего на x окна в y
9) сравнить x и полученное на предыдущем этапе окно из y методом совпадения по шаблону с метрикой TM_SQDIFF
10) полученная маска размером 1×1 есть квадрат разности между яркостями пикселей изображения, вернуть 1 - яркость пикселя маски / 255.

6.3 Алгоритм поиска указки

Указку можно определить как изображение, постоянно или почти постоянно находящееся в видео потоке, меняющее свое положение и не меняющее форму. Для поиска указки и хранения его изображения удобно использовать класс, реализующий, кроме поиска указки, ряд вспомогательных функций

- процедура очистки чёрно-белого изображения разности кадров от шумов;
- процедура поиска границ областей в изображении.

Алгоритм поиска указки можно описать следующим образом:

1) на вход дан видеопоток;
2) задать величину, на которую схожи два эквивалентных изображения - 0.95;
3) пока разность между кадрами пустая - сдвинуть видеопоток и вычислить разницу между кадрами;
4) обвести белые области разности контурами;
5) инициализировать и заполнить цепь. Цепь есть вектор пар, второй элемент - изображение, вырезанное из видеопотока в момент после сдвига по контуру из предыдущего шага, первый элемент - число вхождений схожих изображений в видеопоток - изначально равно одному;
6) пока ни одно изображение в цепи не входит в видеопоток чаще, чем указанное число раз:
    а) сдвинуть видеопоток, обвести белые области разности контурами;
    б) попарно сравнить полученные на предыдущем этапе области с

элементами цепи;

в) если хотя бы одна пара элементов совпала - увеличить на одни число вхождений области в соответствующем элементе цепи;

г) если для изображения, вырезанного в текущий момент времени из видеопотока шаблон в цепи не найден - то это изображение включается в цепь, количество вхождений в видеопоток равно одному;

7) второй элемент звена цепи, первый элемент пары которого равен указанному выше числу принимается за изображение указки.

6.4 Алгоритм генерации слайдов

Алгоритм генерации слайдов вырезает слайды из видеопотока и пытается вырезать изображение курсора из слайда там, где это возможно.

Алгоритм поиска курсора в видеопотоке:

1) на вход дан видеопоток;
2) вычислить разность между кадрами;
3) сохранить кадр до свига видеопотока как текущий слайд;
4) сгенерировать маску курсора для текущего слайда;
5) пока маска не черная и количество не черных элементов разности меньше, чем 3 площади изображения курсора:
   а) вычислить разницу между кадрами;
   б) для текущего кадра вычислить маску курсора;
   в) вычислить маску для заполнения как маска кадра минус (маска кадра пересечение маска текущего слайда);
   г) вычислить новое значение маски слайда как пересечение маски слайда с текущей маской;
   д) из текущего слайда вырезать область по маске для заполнения

и вставить в изображение текущего слайда;

6) пока количество не черных элементов разности меньше, чем 3 площади изображения курсора вычислить разницу между кадрами;
7) сохранить текущий слайд в памяти.

6.5 Алгоритм выделения текста из изображения слайда

Алгоритм выделения текста, описанный в данной программной реализации относится к градиентным методам локализации текста. Для выделения границ изображения используется нестандартный оператор, состоящий из двух сверток. Ядро первой свертки аналогично второму ядру оператора Собеля. Таким образом, после первой свертки на изображении выделяются левые границы объектов. Ядро второй нестандартной свертки равно первому ядру нестандартной свертки, умноженному на минус один. Результатом второй свертки является изображение правые границы объектов. Выбор в пользу нестандартной свертки сделан на основе результатов сравнения работы нестандартного оператора и оператора Собеля.

Алгоритм выделения текста из изображения можно описать следующим образом:

1) на вход дано изображение;
2) размыть изображение по Гаусу с ядром $3 * 3$;
3) бинаризовать изображение с порогом 180;
4) изменить изображение посредством морфологической операции эрозия с прямоугольным ядром 1 * предполагаемая минимальная высота буквы.

5) отфильтровать изображение с ядром $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ для выделения левых границ объектов изображения
6) отфильтровать изображение с ядром $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ для выделения правых границ объектов изображения;
7) наложить оба отфильтрованных изображения друг на друга;
8) посредством морфологических операций оставить только те прямоугольные области ширины 1, длина которых входит в допустимый отрезок;
9) посредством морфологических операций объединить полученные области в предполагаемые текстовые блоки;
10) определить прямоугольные контуры вокруг предполагаемых текстовых блоков;
11) отфильтровать полученные контуры удалив те, которые по тем или иным причинам быть не могут контуром текстового блока.

Алгоритм выделения текста на последнем этапе требует выполнения процедуры фильтрации контуров. В программной реализации используются несколько работающих последовательно фильтров. Первый из них удаляет все вложенные друг в друга контуры, и те, чья площадь была меньше допустимой. Второй фильтр пытается, кроме этого, отличить текстовый блок от других изображений основываясь на том факте, что текстовый блок можно разбить на строки, строки можно разбить на слова, а слова разбить на символы. Формы символов, в свою очередь, имеют определенные ограничения. В реализации для каждой строки вычисляется

величина:

высота строки / среднее арифметическое ширины символа в строке.

После вычисляется медиана ряда полученных выше величин. Эта медиана m проверяется неравенством $1 \leq m \leq 3.5$. Если неравенство выполняется, то данный блок считается текстовым, иначе отбрасывается.

Для выделения строк текста для заданной области вычисляется вертикальная гистограмма как вектор, длина которого равна высоте выделенной области, а каждый элемент равен количеству не белых пикселей в соответствующей строке. Каждая непрерывная последовательность ненулевых элементов вектора определяет границей строки текста. Аналогичным способом в каждой найденной строке с помощью горизонтальной гистограммы вычисляются вхождения отдельных символов. Второй метод фильтрации мало устойчив к помехам и достаточно часто ошибается. Именно поэтому неравенство фильтра 2 проверяется не для каждой текстовой строки, а только для медианного случая. В силу того, что фильтр 2 работал некорректно, в конечной реализации алгоритма генерации слайдов он не использовался.

Последний фильтр использует для классификации библиотеку tesseract. Предполагаемый текстовый блок поступает на вход классификатора. Если классификатор выдает сообщение “Empty page” или текст, который по некоторым критерием не может содержаться на изображении, то область удаляется из множества текстовых блоков.

Библиотека tesseract достигает высокой точности в распознавании текстовых блоков, в то же время очень времязатратна: третий фильтр выполняется на порядок дольше двух предыдущих. Поэтому важно отфильтровать множество предполагаемых текстовых блоков как можно точнее до этапа распознавания слов.

6.6 Описание работы библиотеки tesseract

Tesseract - open-source OCR приложение, изначально разрабатываемое компанией HP в период 1984-1994 годов. В 2005 году HP выложила исходный код в открытый доступ[1].

Распознавание текста происходит в несколько шагов. На первом этапе изображение подвергается адаптивной бинаризации. Следующий шаг - анализ компонент связности. На этом этапе tesseract определяет контуры для каждой из компонент и полигонизирует их. Благодаря тому, что программа в дальнейшем рассматривает полигоны, а не изображение, вырезанное по контуру, она не зависит от цвета текста то есть черный текст на белом фоне распознается точно также, как белый текст на черном.

После приложение пытается найти строки текста. Строки не обязательно должны быть строго горизонтальными, tesseract может выделить строку, повернутую на некоторый угол. После tesseract пытается разделить слова на символы. На первой итерации программа вычисляет предполагаемую ширину символа, просто делит полигоны вертикальными линиями и передает результат в статический классификатор. После каждой удачной попытки определить символ то есть когда вероятность, что данная область и некоторый ascii символ совпадают высока, область передается в адаптивный классификатор (adaptive classifier) как набор обучающих данных. На следующих этапах классификации tesseract будет учитывать предыдущий опыт классификации, что сильно увеличивает точность. В случае неудачи tesseract пытается сегментировать слова другими методами (non-fixed- pitch). Сначала в полигонах определяются вогнутые вершины (concave vertices). С помощью статического классификатора, как и на предыдущем этапе, определяются пары найденный вершин, проведя

прямую через которые вероятнее всего отдельный полигон будет поделен на символы.

Если после этого некоторые слова так и не удалось распознать, tesseract предполагает, что символы на изображении составлены не из одного полигона, а из нескольких т.е. символы при адаптивной бинаризации разбились на несколько отдельных компонент связности. На этом этапе полигоны проходят через так называемый ассоциатор (associator), который основан на алгоритме направленного поиска A*. Ассоциатор ищет такую комбинацию изначальных компонент связности, что она уверенно определяется классификатором как символ. Авторы tesseract соглашаются с тем, что процесс “Разбить изображение на минимальные компоненты связности, а затем соединять их” может быть не оптимальным, но аргументируют выбор такой последовательности тем, что входные данные ассоциатора упрощаются, благодаря чему поиск может вестись быстрее. Когда данный алгоритм был впервые реализован в tesseract в 1989 году, точность распознавания разбитых на несколько областей символов программы была намного больше, чем у конкурентов.

После того, как слова распознаны, tesseract проводит лингвистический анализ: для каждого слова в словаре находится наиболее похожая аналогия. Аналогичное слово может отличатся от распознанного не только символами, но и их количеством.

Классификация символов проходит в два этапа. Из входа извлекаются особенности или фичи (features), которые сравниваются с особенностями символов, выделенных в процессе обучения. Классификатор имеет два вида особенностей: трехмерный вектора, содержащие координаты и углы, и четырехмерные вектора, которые хранят кроме углов и координат еще и длину. На первом этапе строится

список символов, на которые входной символ может быть похож, причем производится сравнения только трехмерных особенностей, что гораздо менее трудозатратно и позволяет на раннем этапе классифицировать вход как не символ. На втором этапе каждый из избранных символов представляется как логическая сумма произведений четырехмерных фич. Вычисляется расстояние между входом и каждым символом, на основе которого строится окончательное решение. Благодаря представлению, при вычислении расстояний tesseract распоряжается информацией о расстоянии между каждой особенностью в отдельности, а не глобальным расстоянием между контурами, что увеличивает точность классификации . Авторы tesseract опытном путем выявили, что при сравнении четырехмерных особенностей относительно короткие особенности резко отличаются у разбитых на части и целых символов. Поэтому для увеличения точности tesseract не использует относительно короткие особенности.

После более чем десятилетнего перерыва, tesseract отстает от современных коммерческих OCR приложений в точности распознавания текстов. Тем не менее, tesseract остается популярным среди разработчиков благодаря открытому исходному коду, достаточно высокой точности распознавания и активной поддержкой сообщества (в том числе разработчиками google).

6.7 Описание программного приложения

Программное приложение реализовано на языке c++ с использованием библиотеки машинного зрения OpenCV и состоит из трех модулей:

– модуль общих вспомогательных процедур обработки изображений

img_handler.h, img_handler.cpp;

– модуль, реализующий алгоритм распознавания текста из изображений slide_proc.h, slide_proc.cpp;
– модуль, реализующий алгоритмы поиска указки и генерации слайдов video_handler.h, video_handler.cpp.

6.7.1 Реализация алгоритма сравнения изображений

Релизация функции сравнения изображений находится в файле img_handler.h и имеет следующий вид:

```
double cmp(Mat x, Mat y);
```

Где:

– `Mat x, Mat y` - входные изображения.

Функция основана на алгоритме, описанном в главе 6.2.

6.7.2 Реализация алгоритма поиска указки

Реализация алгоритма поиска указки, для удобства, оформлена в виде класса, описанного в файле video_handler.h.

```
class Cursor {
Mat img;
...
    void filter_rects();
    void threshold_diff();
    ...
public:
    Cursor(){};
```

```
    void find_cursor(VideoCapture cap, int hit_lim, int shift);
    Mat get(){ return img;}
};
```

методы `filter_rects` и `threshold_diff` - вспомогательные, реализуют фильтрацию контуров и бинаризацю изображения разности соответственно. Основной интерес представляет метод поиска указки:

```
void find_cursor(VideoCapture cap, int hit_lim, int shift);
```

Где:

- `VideoCapture cap` - экземпляр видеопотока;
- `hit_limit` - верхняя граница количества вхождений изображения указки в цепь;
- `int shift` - величина сдвига видеопотока для вычисления разницы между изображениями.

Для запуска процедуры необходимо создать экземляр класса `Cursor`, используя конструктор по умолчанию. После от этого экзепляра следует вызвать метод `find_cursor`. Изображение курсора можно получить после работы метода `find_cursor` с помощю метода `get`. Результат работы алгоритма на тестовой записи вебинара представлен на рисунке 6.1.

Рисунок 6.1 – Результат работы алгоритма поиска указки на тестовом вебинаре

6.7.3 Реализация алгоритма генерации слайдов

Реализация алгоритма генерации слайдов описана в файле video_handler.h и имеет вид:

```
class Presentation {
...
public:
...
    Presentation(Mat cursor,Rect area);
    void generate(VideoCapture cap, int shift);
    void write_slides(string patch);
    vector<Mat> get_slides();
};
```

Для запуска процедуры генерации слайдов необходимо создать экземпляр класса `Presentation`. В конструктор необходимо передать два параметра: изображение указки и координаты слайда в видеопотоке в виде четырех координат. Далее от экземпляра класса `Presentation` необходимо вызвать метод `generate`, который имеет параметры:

- `VideoCapture cap` - экземпляр видеопотока;
- `int shift` - величина сдвига видеопотока для вычисления разницы между изображениями.

После работы метода `generate` слайды можно получить с помощью методов `write_slides` или `get_slides`. Первый получает в качестве параметра путь к папке, в которую сохраняются сгенерированные на предыдущем этапе слайды, а второй возвращает в качестве значения вектор изображений слайдов.

Пример работы процедуры генерации слайдов, в котором алгоритм вырезал изображение курсора можно видеть на рисунке 6.2.

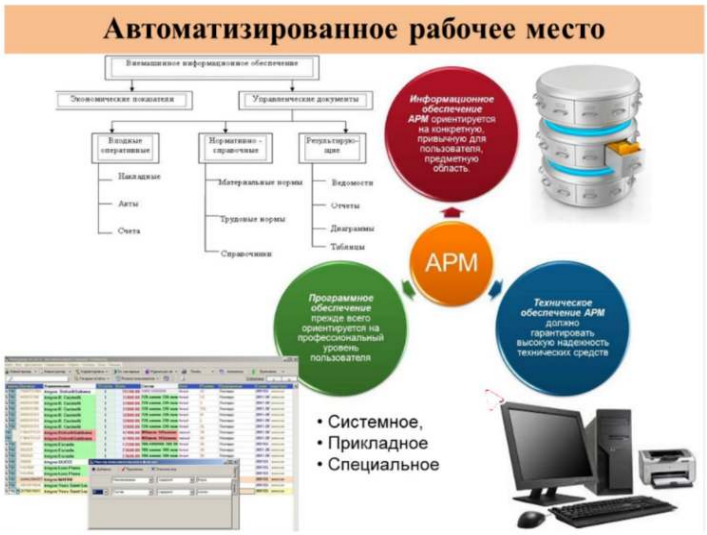


Рисунок 6.2 – один из слайдов, вырезанных из тестового вебинара с помощью алгоритма генерации слайдов

6.7.4 Реализация алгоритма выделения текста из изображения

Алгоритм выделения текста из изображения кроме вычисления координат текстового блока также распознает символы из этих текстовых блоков. Структура хранения координат символов, описанная в файле slide_proc.h, имеет вид

```
struct Text_block{
    Rect coord;
    string text;
};
```

Реализация алгоритма выделения текста из изображения описана в файле slide_proc.h и имеет вид:

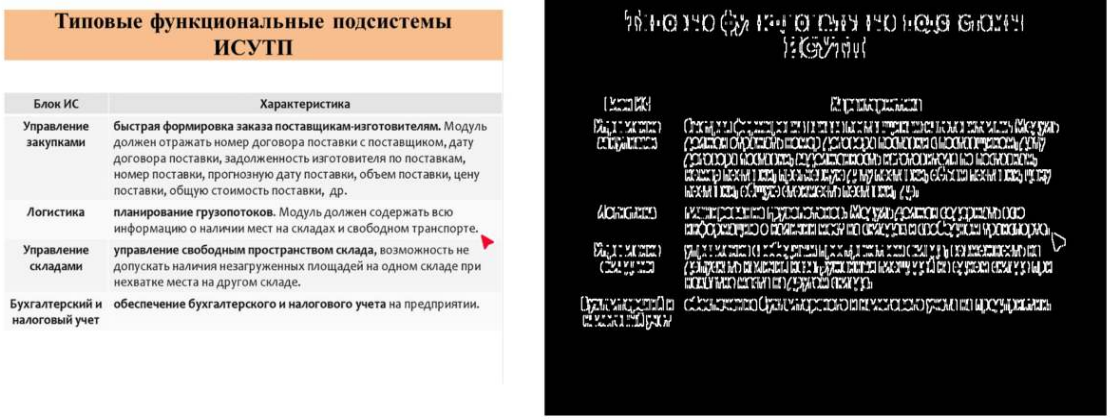
```
class Slide_proc{
```

```
    vector<Text_block> text_blocks;
public:
    Slide_proc();
    void proccess(Mat image);
    vector<Text_block> get();
};
```

Для запуска процедуры выделения текста из изображения необходимо создать экземпляр класса `Slide_proc`, используя конструктор по умолчанию. После следует вызвать метод `process`, который имеет параметры

- `Mat image` - входное изображение, из которого необходимо выделить текстовую информацию.

Результат работы указанного выше алгоритма можно получить методом `get`. Пример работы алгоритма выделения текста можно видеть на рисунке 6.3.



а) исходное изображение

б) результат работы алгоритма поиска границ

в) результат инверсии и первого этапа морфологических операций

г) результат второго этапа морфологических операций - объединения границ символов в блоки

д) результат операции выделения контуров текстовых блоков

е) результат фильтрации контуров из предыдущего этапа

Рисунок 6.3 – пример работы алгоритма выделения текста из изображения

ЗАКЛЮЧЕНИЕ

В курсовой работе были рассмотрены методы сравнения изображений, среди которых семейство методов сравнения по гистограмме, и семейство алгоритмов поиска по шаблону. Исследовано дискретное преобразование Фурье, изучены способы применения дискретных преобразований на практике. Рассмотрены алгоритмы локализации текста, основанные на градиенте изображения, квантовании цвета и текстурных особенностей текстовой информации. Программно были разработаны процедуры, реализующие основу для дальнейшей разработки приложения, которое в полуавтоматическом режиме оценивает вебинары по заданным критериям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Smith R. An Overview of the Tesseract OCR Engine – URL: https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/33418.pdf (12.03.2019)

2 Gonzalez R.C., Woods R.E. Processing digital Image 4 global edition – M: Pearson, 2017. – 993с.

3 Lewis J. P. Fast Normalized Cross-Correlation – URL: http://scribblethink.org/Work/nvisionInterface/nip.html (12.03.2019)

4 Jian Liang, David Doermann, Huiping Li. Camera-based analysis of text and documents: a survey. – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.9236&rep=rep1&type=pdf (12.03.2019)

5 Ляхов Д.А. Распознавание образов в условиях цветовой близости. Кубан. гос. ун-т. – Краснодар, 2018. – 30 с.