Правительство Росийской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образоваия

Национальный Исследовательский Университет - Высшая Школа Экономики

Факультет бизнес информатики

Отделение программной инженерии

Кафедра «Управление Разработкой Программного Обеспечения»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему **«Программа обнаружения лица на изображении на основе характеристических признаков»**

Студент группы №471ПИ

Тарасиков Александр Сергеевич

Руководитель ВКР:

профессор, доктор наук

Гостев Иван Михайлович

Москва

2013

Аннотация

В данной работе исследуется влияние ориентации лица человека в пространстве на работу алгоритма обнаружения лиц ViolaJones.

Целью данной работы является разработка способа доработки алгоритма обнаружения лиц для того, чтобы он допускал больший диапазон углов поворота лица во входном изображении.

Приводится обзор трех классификаторов обнаружения лиц (а именно, признаки Хаара, используемые в алгоритме Виолы-Джонса, локальные двоичные последовательности (LBP), собственные вектора). Предложена методика повышения устойчивости алгоритма к изменению положения лица. Описан метод сравнительного анализа производительности алгоритмов обнаружения лиц.

Результаты работы влкючают в себя теоретическую часть, представляющую собой анализ устойчивости алгоритма Viola-Jones к афинным преобразованиям; и практическую - программную реализацию доработанного алгоритма Виолы-Джонса.

Содержание

1 Введение

1.1 Актуальность

В связи с тем, что во всем мире участились случаи терроризма, для обеспечения безопасности на транспорте и в офисных помещениях, идентификация человека, особенно по лицу, приобретает важное значение.Решение задачираспознавания лица включает в себя этапы получения изображения,предварительной обработки, обнаружение лиц и идентификацияс учетом выявленных особенностей.

В данной работе будет рассмотрена задача *обнаружения лиц*, в которую не входит сопоставление лица с известным изображением из базы данных.

Решение задачи обнаружения лиц особенно важно при использовании систем видеонаблюдения (таких, как CCTV) и в охранных комплексах. В связи с ростом вычислительной мощности персональных компьютеров и мобильных устройств, обнаружение лиц набирает популярность как способ организации человеко-машинного взаимодействия. Социальные сети, такие, как Facebook, обнаруживают лица в загруженных пользователем фотографиях и предлагают ассоциировать их с аккаунтом пользователя в сети. Также, существуют приложения с использованием «дополненной реальности», такие, как видеоигры, где игрок может взаимодействовать с объектами виртуального мира посредством движений и жестов, фиксируемых камерой.

На сегодняшний день область применения алгоритмов обнаружения лиц динамически развивается. Данные алгоритмы находят применение в различных встраиваемых (embedded) системах, а условия применения данных систем обуславливают существенные различия в качестве изображений. Требования обработки в режиме реального времени делают невозможным пост-обработку изображений или привлечение оператора для, поэтому важно разрабатывать устойчивые к дефектам изображений алгоритмы, обладающие вычислительной эффективностью. Таким образом, задача обнаружения лиц представляет собой одно из приоритетных направлений развития алгоритмов машинного обучения и компьютерного зрения.

1.2 Цель работы

Одним из факторов, оказывающим существенное влияние на корректное обнаружение лица, является ориентация головы в пространстве. Будем называть алгоритм *устойчивым к поворотам* при определенном угле поворота в том случае, если он корректно обнару-

живает лицо при условии, что голова повернута вокруг вертикальной оси на такой угол.

Порог устойчивости для большинства наиболее часто используемых на данный момент алгоритмов находится в пределах 10 градусов, чего недостаточно для обнаружеия лиц без предварительного позиционирования их перед камерой. Это делает такие алгоритмы неприменимыми для задачи поиска лиц в системах CCTV или в потоке видео с камеры мобильного телефона.

Целью данной работы является разработка и реализация алгоритма с улучшенной устойчивостью к поворотам. Необходимым условием для определения направления исследований является анализ существующего алгоритма с целью получения точных численных значений диапазонов угла поворота, а также производительности (под *производительностью* будем понимать время, необходимое для обнаружения лиц на тестовых изображениях. поскольку абсолютная производительность зависит от многих факторов, таких, как конфигурация аппаратного и программного обеспечения, нас будет интересовать относительная производительность - то есть, значения, полученные при запуске различных алгоритмов на одной и той же машине в идентичной среде исполнения).

В данной работе предложен способ увеличения устойчивости алгоритма к поворотам, который заключается в предварительной обработке входного изображения для того, чтобы компенсировать поворот голов. Реализованы и проанализированы различные подходы к оценке таких обратных преобразований. В качестве базового алгоритма обнаружения используется алгоритм Виолы-Джонса [1]. Данный алгоритм выбран по причине того, что является наиболее часто используемым на сегодняшний день алгоритмом. Кроме того, он относительно прост в реализации, и может служить базисом для дальнейшей реализации многоступенчатых методов обнаружения лиц.

1.3 Этапы работы

Для реализации описанных выше задач, были выполнены следующие этапы.

- Поиск публично доступных баз изображений лиц, сделаных при различных углах поворота. Была выбрана база изображений института INRIA [5]
- Написание приложения для автоматической подготовки изображений к использованию на этапе сбора статистики. Подготовка включает в себя следующие этапы:
  - Получение архивов изображеий с соответствующих веб-сайтов
  - Приведение данных из различных баз к единому формату каталогов.

  - Конвертация изображений из различных форматов в требуемый. Данная задача была решена средствами библиотеки Qt, которая также используется для вывода изображения на экран.

- Реализация программы для сбора статистики по алгоритмам обнаружения лиц. Программа в автоматическом режиме исполняе алгоритмы, используя тестовые изображения в качестве входных данных, и генерировать отчет.

Реализация алгоритма Виолы-Джонса.

Заключительный этап - реализация устойчивых к поворотам головы алгоритмов обнаружения лиц. Он состоит из реализации двух описанных ранее подходов к увеличению устойчивости. Первой стадией реализуется алгоритм, который обрабатывает изображение с камерой для того, чтобы компенсировать изменение положения головы. Таким образом, большинство алгоритмов обнаружения лиц могут работать в большем диапазоне допустимых углов без внесения изменения в исходный алгоритм. Вторая стадия - разработка и реализация алгоритма, использующего набор инвариантных к поворотам признаков для поиска лиц.

1.4 Классификация алгоритмов обнаружения лиц

На данный момент существует множество алгоритмов обнаружения лиц. Некоторые из них доступны в виде статей с описанием математических моделей, другие - исключительно в виде программной реализации (напр., библиотека OpenCV, библиотека распознавания лиц ОС Android и др.)

В данной секции приведена классификация алгоритмов обнаружения лиц, а также описаны преимущества и недостатки каждой группы алгоритмов.

1.4.1 Каскады признаков-функций

Данная группа состоит из алгоритмов, которые используют в качестве признаков лиц критические значения некоторой функции от яркости двумерного изображения. Наиболее известными и часто используемыми являются следующие два алгоритма:

- Алгоритм П.Виолы и М.Джонса [1], использующий признаки Хаара - сумма яркостей пикселей.

- Алгоритм LBP (Local Binary Patterns) для каждого проверяемого пикселя изображения в 8-связной области сравнивает яркость пикселя из области с проверяемым

пикселем. В случае, если первое значение больше - пикселю области присваивается значение «1», в противном - «0». 8-битное число, получаемое обходом пикселей по часовой стрелке, называется Binary Pattern, и используется для сравнения со значениями в классификаторе.

Преимущества данной группы алгоритмов:

– Универсальность метода. Подход можно использовать для обнаружения произвольных объектов, не только лиц.

Недостатки данной группы алгоритмов:

– Большое время обучения классификатора. В процессе обучения применяется алгоритм AdaBoost, который требует перебора всех N классификаторов для каждого изображения в тестовой выборке. число N может достигать нескольких тысяч.

– Отсутствие структурной модели лица. Поскольку данные алгоритмы оперируют исключительно яркостью изображения, возможны ложные срабатывания на объектах, схожих по уровню яркости с лицами из тестовой выборки.

1.5 Собственные вектора и значения

Алгоритмы данной группы рассматривают входное изображение как матрицу в многомерном пространстве. Изображения из тестовой выборки представляются в матричном виде (обычно в виде вектора размерности $1xN$), нормируются путем вычитания из них усредненного изображения. После этого в матрицах, представляющих изображения, ищутся собственные вектора, описывающие отличие изображения от среднего. Процесс обнаружения лица заключается в сравнении расстояния вектора-изображения с критическим значением, хранимым в базе. Среди всех тестовых изображений, расстояние до которых меньше критического значения, ищется изображение с минимальным расстоянием. Таким образом, данная группа алгоритмов позволяет решить задачу *идентификации* человека. Основное различие между алгоритмами данной группы заключается в методике вычисления собственных векторов.

Основные алгоритмы данной группы:

– EigenFaces описан в [2]. Использует яркость пикселей как значение компонент векторов. Неустойчив к изменению освещения.

– FisherFaces - модификация алгоритма EigenFaces. Использует нормированную яркость пикселей. Более устойчив к изменению освещения.

Преимущества данной группы алгоритмов:

– Меньшее время обучения классификатора по сравнению с каскадами признаков. Разница может достигать нескольких порядков.

– Возможность идентификации человека по лицу

Недостатки данной группы алгоритмов:

– Низкий процент распознавания на лицах не из тестовой выборки. Поскольку собственные вектора вычисляются по тестовой выборке, расстояние для изображений не из выборки будет больше критического значения.

2 Математические модели

В данной главе дается описание используемых в данной работе методов. Описывается процедура анализа поведения алгоритмов обнаружения лиц при афинных преобразованиях входного изображения. Описывается алгоритм Viola-Jones, особенности его программной реализации, а также способ доработки для повышения устойчивости к поворотам лица вокруг вертикальной оси.

2.1 Сбор статистики

Теоретическую часть данной работы составляет сбор статистики и сравнение существующих алгоритмов на тестовом наборе данных.

Тестовый набор данных представляет собой фотографии лиц людей при разных углах поворота головы. В тестовом наборе есть фотографии нескольких людей. Для одного человека доступны фотографии с шагом поворота в $5°$ в диапазоне $(-90°; +90°)$

Для каждого алгоритма производится выполнение с использованием каждой фотографии в качестве входных данных. Результаты сгруппированы по углам поворота для каждого алгоритма. Внутри одной группы (то есть, результатов для одного алгоритма

при фиксированном угле на изображениях различых людей), считается процент распознанных изображений.

Таким образом, в результате формируется набор записей вида

<Название Алгоритма, № лица, угол поворота, распознано/нет, $\triangle T$*>*

Время работы алгоритма $\triangle T$ рассчитывается как разница между временем окончания и временем начала обработки (в миллисекундах) $T_{end} - T_{start}$.

2.2 Алгоритм Viola-Jones

Алгоритм обнаружения лиц с применением признаков Хаара был впервые описан П.Виолой и М.Джонсом [1].

2.2.1 Признаки Хаара

Признаки Хаара (Haar-Like features) [1] представляют собой двоичную аппроксимацию вейвлета Хаара. Каждый признак представляет собой двоичную маску, т.е., черно-белое изображение, как можно видеть на Рис. 1.

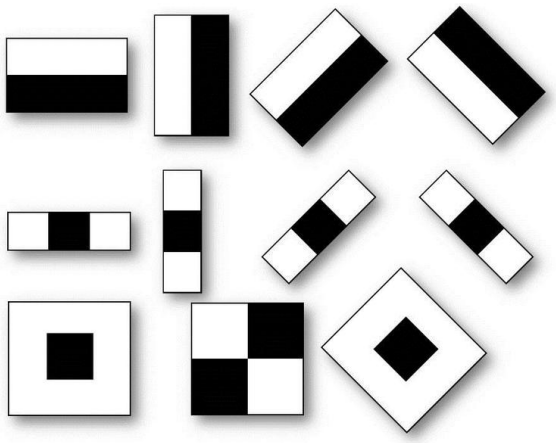

Рис. 1: Признаки Хаара

2.2.2 Каскад признаков

В данной реализации алгоритма признаки представлены набором прямоугольников. Для каждого прямоугольника задан *вес*, который вычисляется как сумма значений яркости пикселей, закрываемых данным прямоугольником. Такое представление обусловено тем, что возможно быстро вычислять сумму яркостей для прямоугольных областей, но не для фигур произвольной ориентации.

Признаки группируются в этапы (Stages). Для каждого этапа в каскаде хранится пороговое значение (threshold). Для каждого признака хранятся два критических значения *left* и *right*. Далее описан алгоритм обнаружения лиц, использующий такую структуру каскада признаков.

2.2.3 Интегральное изображение

Для того, чтобы за константное время (4 обращения к памяти) получить сумму пикселей внутри прямоугольного региона изображения, используется оптимизация, называемая интегральное преобразование изображения. Интегральным оно называется по аналогии с математической операцией интегрирования, вычисляющей сумму значений функции на отрезке. В данном случае функцией является яркость пикселя $I_{x,y}$

Интегральное изображение в точке с координатами $I_{x,y}$ представляет собой сумму значений яркости в точках, вертикальная и горизонтальная координаты которых меньше x и y. Пример интегрального изображения приведен на Рис. 2

Для вычисления значения в конкретной точке используется следующая формула:

$$I_{x,y} = I_{x,y-1} + I_{x-1,y} - I_{x-1,y-1}$$

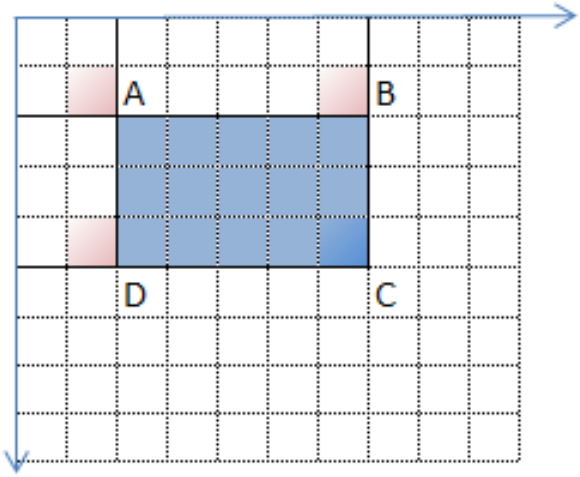

Рис. 2: Интегральное представление изображения

Сумма значений яркости пикселей внутри прямоугольника $ABCD$ можно вычислить по формуле

$$C - B - D + A$$

2.2.4 Обнаружение объекта

Алгоритм использует базу признаков (features) для обнаружения объектов. Базу возможно сгенерировать из всех возможных комбинаций Haar-Like features путем отсеивания «слабых» классификаторов, которые дают ошибку второго рода, то есть ошибочно не находят объект на изображении. Один из способов получения базы с применением алгоритма AdaBoost описан в [1] и [**?**]. В данной работе этап генерации базы не применяется, и реализуется только этап обнаружения объекта с применением готового набора признаков. В качесте базы используется каскад $haarcascade\_frontalface\_alt.xml$ из проекта $OpenCV$ [6]

Каскад признаков состоит из нескольких стадий (stages). Каждая стадия включает в себя набор признаков (features). В представлении, использующемся в каскаде из проекта $OpenCV$, признаки разбиваются на одноцветные прямоугольники (rects), каждому из которых назначен положительный или отрицательный вес.

Во время выполнения алгоритма, "окно"размером $W_h * W_w$ пикселей движется вдоль изображения по горизонтали и вертикали. Начальный размер окна выбирается равным размеру окна, записанному в каскаде-классификаторе. На каждом шаге размер окна увеличивается. Возможны два способа увеличения размера окна. Первый заключается в вычислении масштабирующего коэффициента и корректировке координат прямоугольников внутри признаков. Второй заключается в масштабировании самого исходного изображения. В данной работе использован первый способ, поскольку он вычислительно проще при программной реализации масштабирования изображений. При этом, размер размер окна на каждой итерации умножается на масштабирующий коэффициент $ITER\_SCALE$. Окно сдвигается по горизонтали на w_step_x пикселей, и на w_step_y пикселей по горизонтали. Данные переменные вычисляются следующим образом:

$$w\_step\_y = max(1, min(4, W_h/10))$$

$$w\_step\_x = max(1, min(4, W_w/10))$$

Для окна высчитывается нормировочный коэффициент

$$win\_norm = \frac{1.0}{W_h * W_w}$$

. Внутри окна считается сумма яркостей пикселей w_sum и сумма квадратов яркостей w_ssq пикселей с использованием интегрального представления изображений. Считается

математическое ожидание яркости пикселя $mean$, дисперсия var и стандартное отклонение

$$mean = w\_sum * win\_norm$$

$$var = (w\_ssq * win\_norm) - mean^2$$

$$stddev = \begin{cases} sqrt(var) & \text{если } var >= 0 \\ 1.0 & \text{иначе} \end{cases}$$

Для оптимизации введена пороговая константа $STDDEV\_MIN$, и если стандартное отклонение $stddev$ меньше этого значения, данное "окно"пропускается. Это сделано для того, чтобы исключить области с низкой яркостью, где отсутствует гарантинахожденияя объекта.

Для каждой *стадии* из каскада считается накопленный коэффициент sum_stage, и сравнивается с критическим значением, записанным в каскаде. В том случае, если значение меньше критического, стадия считается невыполненной, и для текущего окна обработка прерывается. В случае, если все стадии отработали корректно, считается, что в текущем окне обнаружен объект (лицо). При этом, введена проверка, что стандартное отклонение больше $STDDEV\_FACE$, что позволяет регулировать порог срабатывания.

Для каждого *признака* из каскада считается отнормированное значение $sum\_feature * win\_norm$ и сравнивается с отнормированным критическим значением, записанным в каскаде $feature- > threshold * stddev$. В случае, если значение меньше критического, к значению стадии прибавляется значение из левого поддерева признака, в противном- из правого. Значение $sum\_feature$ считается как сумма яркостей пикселей, которые попадают в прямоугольную область внутри признака, умноженная на вес области. Координаты области умножаются на win_scale.

В таблице приведены значения констант, используемых в данной реализации алгоритма.

ITER_SCALE	1.2
STDDEV_MIN	10.0
STDDEV_FACE	25.0

2.2.5 Код алгоритма

```
1 void VJFaceDetector::integral_image(const uchar *ptr,
2                                     VJFaceDetector::integral_t *out,
3                                     VJFaceDetector::integral_t *sum_sq,
4                                     unsigned _w,
```

```
5                                       unsigned _h)
6   {
7   #define IN(i, j) ptr[(4 * (i) * _w) + ((j) * 4)]
8   #define OUT(i, j) out[((i) * _w) + (j)]
9   #define SQR(i, j) sum_sq[((i) * _w) + (j)]
10
11      for (unsigned i = 0; i < _h; i++) {
12          for (unsigned j = 0; j < _w; j++) {
13              OUT(i, j) = IN(i, j);
14              SQR(i, j) = IN(i, j) * IN(i, j);
15
16              if (i > 0 && j > 0) {
17                  OUT(i, j) -= OUT((i - 1), (j - 1));
18                  SQR(i, j) -= SQR((i - 1), (j - 1));
19              }
20              if (i > 0) {
21                  OUT(i, j) += OUT((i - 1), j);
22                  SQR(i, j) += SQR((i - 1), j);
23              }
24              if (j > 0) {
25                  OUT(i, j) += OUT(i, (j - 1));
26                  SQR(i, j) += SQR(i, (j - 1));
27              }
28          }
29      }
30  }
31
32  inline VJFaceDetector::integral_t VJFaceDetector::sum(
33          VJFaceDetector::integral_t *arr,
34          unsigned x,
35          unsigned y,
36          unsigned w,
37          unsigned h,
38          unsigned _w,
39          unsigned _h)
40  {
```

```
41      if (x + w >= _w || y + h >= _h) {
42          return 0;
43      }
44
45      size_t i1 = _w * y + x;
46      size_t i2 = _w * (y + h) + x;
47
48      integral_t s1 = arr[i1];
49      integral_t s2 = arr[i2];
50      integral_t s3 = arr[i1 + w];
51      integral_t s4 = arr[i2 + w];
52      return s4 + s1 - s2 - s3;
53  }
```

```
1  bool VJFaceDetector::detect(const uchar *ptr, unsigned w, unsigned h,
2    FaceDetector::RectVector &out)
3  {
4      unsigned _w = w;
5      unsigned _h = h;
6      //buffer for the integral image
7      integral_t *integral = new integral_t[_w * _h];
8      integral_t *int_sqr = new integral_t[_w * _h];
9
10     std::fill(integral, integral + _w * _h, 0);
11     std::fill(int_sqr, int_sqr + _w * _h, 0);
12
13     unsigned win_h = _cascade->height;
14     unsigned win_w = _cascade->width;
15
16     double win_scale = 1.0;
17
18     //build integral image
19     integral_image(ptr, integral, int_sqr, _w, _h);
20
21     //scale image for each
22     while (qMin(win_h, win_w) < qMin(_w, _h)) {
23         double win_norm = 1.0 / (win_w * win_h);
24
25         unsigned w_step_y = qMax(1u, qMin(4u, win_h / 10u));
26         unsigned w_step_x = qMax(1u, qMin(4u, win_w / 10u));
27
28         //slide the classifier window across the image
29         for (unsigned wy = 0; wy < _h - win_h; wy += w_step_y) {
30             for (unsigned wx = 0; wx < _w - win_w; wx += w_step_x) {
31                 integral_t w_sum = sum(integral, wx, wy, win_w, win_h,
                       _w, _h);
32                 integral_t w_ssq = sum(int_sqr, wx, wy, win_w, win_h,
                       _w, _h);
33
```

```
34                  double stddev = 1.0;
35                  double mean = w_sum * win_norm;
36                  double var = (w_ssq * win_norm) - (mean * mean);
37                  if (var >= 0) {
38                      stddev = sqrt(var);
39                  }
40                  if (stddev < STDDEV_MIN) {
41                     continue;
42                  }
43
44                  bool failed = false;
45
46                  foreach (Stage *s, _cascade->stages) {
47                      double sum_stage = 0.0;
48                      foreach (Feature *f, s->features) {
49                          double sum_feature = 0.0;
50                          foreach (Rect *r, f->rects) {
51                              unsigned rx = r->x * win_scale;
52                              unsigned ry = r->y * win_scale;
53                              unsigned rw = r->w * win_scale;
54                              unsigned rh = r->h * win_scale;
55                              sum_feature += sum(integral, wx + rx, wy +
                                    ry,
56                                                          rw, rh, _w, _h) *
                                                             r->weight;
57                          } //Rects
58                          if (sum_feature * win_norm < f->threshold *
                                stddev) {
59                              sum_stage += f->left;
60                          }
61                          else {
62                              sum_stage += f->right;
63                          }
64                      } //Features
65
66                      if (sum_stage < s->threshold) {
```

```
67                          failed = true;
68                          break;
69                      }
70                  } //Stages
71
72                  if (!failed) {
73                      if (stddev > STDDEV_FACE) {
74                          FaceDetector::Rect r;
75                          r.x = wx;
76                          r.y = wy;
77                          r.w = win_w;
78                          r.h = win_h;
79                          out.append(r);
80                          qDebug() << "VJ: Object at " << wx << "," <<
                                wy;
81                      }
82                  }
83              }
84          } //scale window
85
86          win_h *= ITER_SCALE;
87          win_w *= ITER_SCALE;
88          win_scale *= ITER_SCALE;
89      }
90
91      delete integral;
92      delete int_sqr;
93
94      return out.count() > 0;
95  }
```

2.3 Модифицированный алгоритм Viola-Jones

В данной главе предлагается доработка алгоритма Viola-Jones для повышения устойчивости к поворотам лица вокруг вертикальной оси. Модифицированный алгоритм состоит из трех этапов.

2.3.1 Сегментация по цвету

Для выделения участков изображения, в которых содержится кожа, используется следующие условия, где R, G, B - цвета в диапазоне $[0; 255]$: $\{to \quad R >= 1.2 * G$
$G <= 8 * B$
$G >= 0.9 * B$
$R <= 2.8 * G(1)$Данные коэффициенты были выбраны на основе анализа результатов, представленных в работе, посвященной обнаружению участков кожи в условиях различной освещенности [3]. Пример работы алгоритма сегментации можно видеть на Рис. 3

После выделения областей, смежные пиксели, классифицированные как «кожа», объединяются в кластеры. Каждый кластер ограничивается прямоугольной областью. Внутри каждой прямоугольной области производится поиск лица (вторая и третья стадии алгоритма). После этого фильтруются области, занимающие менее 5 процентов высоты/ширины изображения (путем сравнения площади изображения и площади области). Рис. 4 показывает окончательный результат.

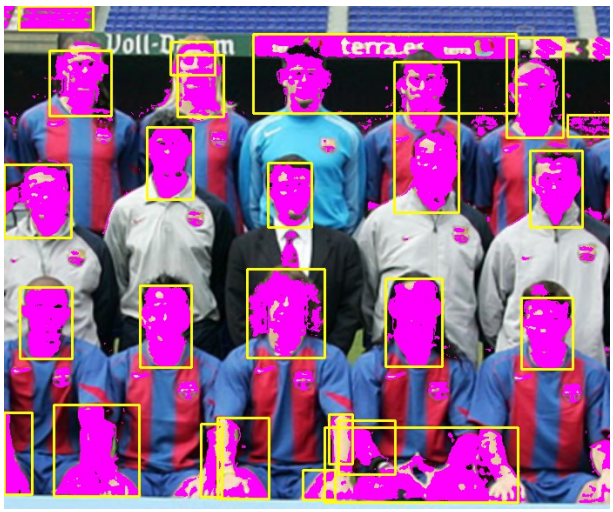

Рис. 3: Выделение регионов, принадлежащих коже

Изображение было взято из набора, предложенного для тестирования алгоритмов распознавания лиц на сайте University of Dundee [4]. Как можно видеть, алгоритм успешно выделяет области, в которых находятся лица. В то же время, есть и ложные срабатывания

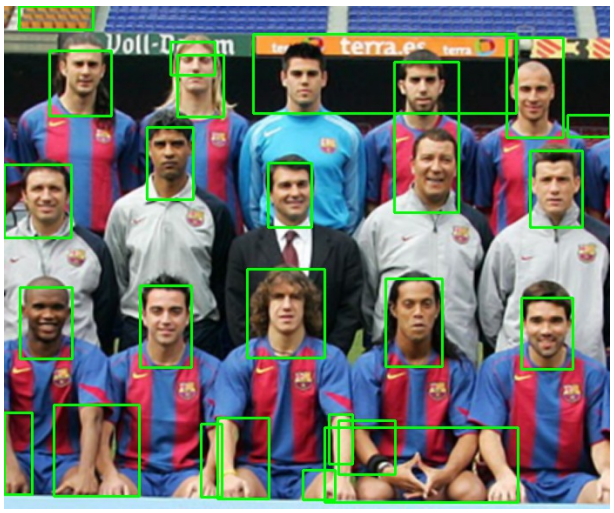

Рис. 4: Выделение областей, потенциально содержащих лица

на объектах, не являющихся лицами (руки, рекламный плакат), однако для нас важно то, чтобы не были пропущены области, где лицо есть. Кроме того, данная формула предназначена для обнаружения людей европеоидной расы на изображениях, сделанных при дневном свете. Поскольку каскады признаков, используемых для тестирования, были обучены по аналогичной выборке, данный этап не должен привести к ложной классификации области, содержащей лицо.

2.3.2 Выбор классификатора

В каждой из областей, потенциально содержащих лицо, которые были обнаружены на предыдущем шаге алгоритма, производится поиск частей лица с использованием каскадов признаков, обученных на поиск данной части.

– Выполняется поиск глаз. Используются каскады для обнаружения левого и правого глаза. Если обнаружен один глаз, то выполняется поиск лица с использованием каскада для профиля лица (вида сбоку).

– При обнаружении двух глаз выполняется поиск рта на изображении. В случае, если обнаружен рот, вычисляются точки-центры прямоугольных областей, ограничивающих глаза и рот. Вычисляются расстояния D_{right} и D_{left} от центра правого и левого глаза до центра рта. Вычисляется реальное расстояние между глазами D_{eyes}, и «ожидаемое» расстояние $D_{expected}$. «Ожидаемое» расстояние - это такое расстояние, которое должно быть между центрами глаз на изображении лица, смотрящего прямо в камеру. Предполагается, что изображение было повернуто только вокруг

вертикальной оси, и горизонтальные расстояния между частями лица не был нарушены. В таком случае, $D_{expected} = D_{left} * D_{right}/1.254/2$. Данный коэффициент был подобран как среднее значение анализом 10 изображений из базы. Производится горизонтальное масштабирование изображения с коэффициентом $\frac{D_{expected}}{D_{eyes}}$.

– Если не было обнаружено лицо при помощи каскада «профиль», или было получено отмасштабированное изображение после классификаторов «глаза», происходит построение виртуального изображения лица, содержащего аппроксимацию исходного изображения до поворота (третья стадия алгоритма).

2.3.3 Компенсация вращений

На данном этапе строится «виртуальное изображение», представляющее собой аппроксимацию изображения лица до поворота вокруг горизонтальной оси. Для этого берется изображение лица, представляющее собой двухмерную проекцию лица на плоскость камеры. Рассматриваем данное изображение как плоскость в трехмерном пространстве, назначая координату $Z = 0$. Будем вращать данную плоскость вокруг вертикальной оси с шагом в 5 градусов в диапазоне от -30 до $+30$ градусов включительно.

Предполагается, что комбинация вращения и масштабирования на предыдущем этапе сделают изображение более похожим на такое, которое мы могли бы получить в случае, когда лицо направлено в камеру, и алгоритм обнаружения лиц должен допускать больший диапазон углов поворота лица.

3 Анализ поведения алгоритмов

В данной главе описывается влияние афинных преобразований на обнаружение лиц лгоритмом Viola-Jones и модифицированным алгоритмом. В качестве тестовой базы используется база ICPR института INRIA [5].

3.0.4 Поворот вокруг вертикальной оси

Использовались изображения из базы ICPR. Из нее выбраны изображения 15 людей. Поворот головы вокруг вертикальной оси в диапазоне $[-90; +90]$ градусов с шагом в 15 градусов.

Результаты Viola Jones:

Угол, градусы	Число распознанных
-90	1
-75	2
-60	8
-45	12
-30	14
-15	14
0	14
+15	15
+30	15
+45	14
+60	10
+75	3
+90	2

Можно видеть, что при 60 градусах распознается более половины изображений, при 75 - меньше 20%. Таким образом, алгоритм стабильно работает при угле, меньшем 60%. При использовании модифицированного алгоритма, распозналось дополнительно одно изображение при угле 75% и два - при угле в 60%, при этом при меньшем угле поворота изменений нет. Это улучшение незначительно, но говорит о том, что начальное предположение о возможности аппроксимации верно. Дальнейшее направление - разработка более точного метода оценки поворота изображения.

3.0.5 Поворот вокруг оси Z

Тест состоит в повороте исходного изображения на 360 градусов инкрементами по 15 градусов. Результат: при поворотах вплоть до $+/-30$ градусов лицо обнаруживается. Поведение оригинального и измененного алгоритмов аналогично, так как модифицированный алгоритм компенсирует только повороты вокруг вертикальной оси.

3.0.6 Сдвиг (Shear)

Преобразование Shear вдоль осей X и Y задается следующими матрицами:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Тест Shear. Исходное изображение подвергнуто преобразованию Shear с λ от 0.0 до 1.0 с шагом 0.1. Результат - более 90% изображений перестают распознаваться при $\lambda >= 0.8$.

3.0.7 Масштабирование

Тест масштабирования. Исходное изображение последовательно увеличивается в 1.2 раза до тех пор, пока каждое из его измерений меньше 768 пикселей. На каждом масштабированном изображении запускается алгоритм.

Результаты тестирования для алгоритмов Viola-Jones и модифицированного алгоритма не отличаются от исходных. Объяснение - алгоритм Viola-Jones инвариантен к изменению размеров входного изображения, так как во время выполнения алгоритма окно классификатора последовательно масштабируется. Данный тест является частью методики анализа поведения алгоритма обнаружения лиц при афинных преобразованиях, и, несмотря на то, что алгоритм Viola-Jones показывает одинаковые результаты, запуск теста для этого алгоритма было необходимо для проверки корректности реализации алгоритма генерации тестов.

3.0.8 Параллельный перенос

Тест переноса. Исходное изображение последовательно увеличивается в 1.2 раза до тех пор, пока его длина и высота меньше 768 пикселей. На каждом масштабированном изображении запускается алгоритм.

Результаты тестирования для алгоритмов Viola-Jones и модифицированного алгоритма не отличаются от исходных. Алгоритм Viola-Jones инвариантен к сдвигу, так как во время выполнения алгоритма окно классификатора свдигаетс в горизонтальном и вертикальном направлениях.

4 Заключение

4.1 Результаты

В ходе выполнения данной работы были достигнуты все ожидаемые результаты:

– Описана методика тестирования алгоритмов обнаружения лица на предмет устойчивости к афинным преобразованиям

– Разработана программная реализация данной методики для автоматизированного тестирования

– Проведен анализ поведения алгоритма обнаружения лиц ViolaJones.

– Проведен анализ поведения доработанного алгоритма обнаружения лиц.

Несмотря на то, что доработанный алгоритм лишь незначительно увеличил процент распознавания изображений на границах интервала допустимых углов поворота для алгоритма Viola-Jones, введение сегментации изображения по признаку цвета кожи уменьшило количество ложных срабатываний алгоритма и сократило время выполнения алгоритма за счет того, что обрабатывается лишь часть изображения. Например, на приведенной ниже иллюстрации (Рис. 5), сравниваются результаты работы алгоритмов Viola-Jones и модифицированного алгоритма. Слева - результат работы алгоритма без предварительной обработки. Можно заметить большой квадрат, захватывающий несколько лиц. Это - ошибка распознавания - ложно срабатывание, чего не наблюдается на правой картинке, где области с лицами были размечены сегментацией по цвету до начала обнаружения. На основании данных результатов можно сделать вывод, что алгоритм Viola-Jones эффективно комбинировать с предварительной обработкой, и использование дополнительных классификаторов может повысить качество обнаружения лиц.

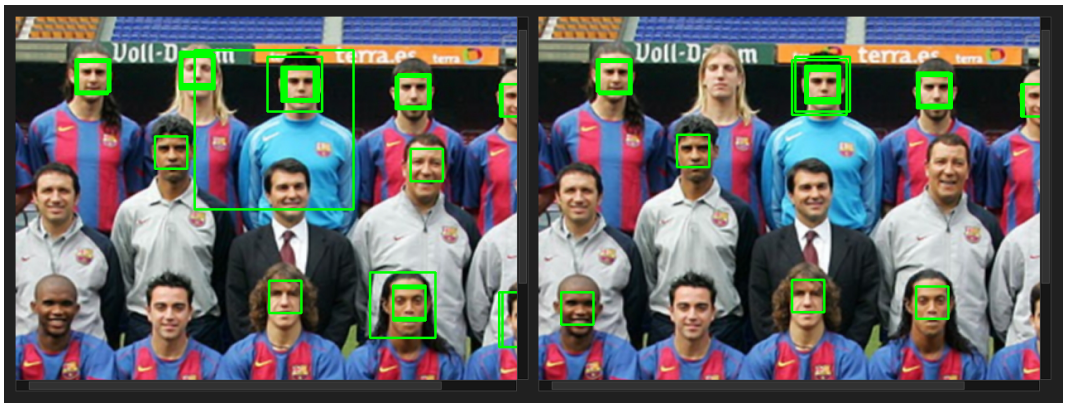

Рис. 5: Результат использования сегментации по цвету

4.2 Дальнейшая работа

По результатам анализа поведения алгоритмов, а также из наблюдений, сделанных в процессе работы, были определены следующие направления дальнейших исследований

- Анализ существующих исследований и разработка метода обнаружения кожи, учитывающий такие особенности, как раса, окружающее освещение.
- Построение линейной регрессионной модели яркости лица в зависимости от угла поворота
- Исследование поведения алгоритма EigenFaces

Список литературы

[1] Jones, M., Viola, P. (2001) *Robust Real-Time Face Detection.* URL: *International Journal of Computer Vision*, 57(2), 137-154, 2004. Available from: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf (ссылка проверена 28 мая 2013)

[2] Turk, M., Pentland, A. (1991) *Face Recognition Using Eigenfaces* URL: http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf (ссылка проверена 28 мая 2013)

[3] *A novel approach for human face detection from color images under complex background* http://visgraph.cs.ust.hk/biometrics/Papers/Face/pr2001-10-1.pdf (ссылка проверена 28 мая 2013)

[4] *Face Detection using AdaBoost and the OpenCV library* URL: http://www.computing.dundee.ac.uk/courses/ac52021/project2.html (28 мая 2013)

[5] *ICPR Face Database* URL: http://www-prima.inrialpes.fr/Pointing04/data-face.html (28 мая 2012)

[6] *OpenCV Open Source Computer Vision* URL: http://opencv.org (дата доступа 28 мая 2013)