МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по научно-исследовательской работе

Тема: Исследование и разработка алгоритмов замены лиц

Студент гр. 4304 _____ Козловский А.О.

Руководитель _____ Кухарев Г.А.

Санкт-Петербург
2019

ЗАДАНИЕ

НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент : Александр Козловский

Группа : 4304

Тема НИР:

Исследование и разработка алгоритмов замены лиц.

Задание на НИР:

1. Исследовать алгоритмы обнаружения и замены лиц.
2. Разработать программное обеспечение, для замены лиц на фото.

Сроки выполнения НИР: 10.09.2019 – 20.12.2019

Дата сдачи отчета: 20.12.2019

Дата защиты отчета: 25.12.2019

Студент _____ Козловский А.О.

Руководитель _____ Кухарев Г.А.

АННОТАЦИЯ

Задача НИР – разработать программу для замены лиц на фотографиях. Для решения этой задачи были изучены различные алгоритмы для обнаружения и замены лиц, настроены внешние библиотеки opencv и dlib в среде разработки Microsoft visual studio. Далее было разработано приложение для нахождения опорных точек и построения триангуляции Делоне. После чего было применено аффинное преобразование треугольников, и по полученной маске произведена замена лиц.

SUMMARY

The task was to develop a system for swapping faces on image. To solve the task there were studied different algorithms of image detecting and swapping, setting up external libraries like opencv and dlib for Microsoft visual studio. Then there was made an application for face landmarks detection and making Delaunay triangulation. Next, the affine transformation of triangles was applied, and then the face was replaced with the obtained triangles.

СОДЕРЖАНИЕ

Оглавление

ВВЕДЕНИЕ

Распознавание объектов изображения является очень актуальной темой в последнее время. Алгоритмы распознавания объектов распространяется практически во всех сферах жизни, от развлечения до обеспечения безопасности населения.

В сервисах GooglePlay и AppStoreбольшой процент приложений использует алгоритмы распознавания. Например, большую популярность имеют приложения, реализующие замену лиц, объединения двух лиц в одно, а также приложения, изменяющие возраст человека на фотографии.

Помимо приложений, носящих развлекательный характер, всё больше набирают популярность приложения, способные в будущем заменить работу человека. Так уже сейчас появляются качественные методы замены фона на фотографиях. То есть то, что пока делают, например, фотографы в детских садах и школах, фотографирующие ребёнка и заменяющие фон фотографии на фон из какого-либо мультфильма с помощью графических редакторов, уже скоро можно будет сделать самому, загрузив фотографию в приложение и нажав несколько клавиш.

Не меньшая польза в распознавании объектов имеется и для обеспечения безопасности. Так, в преддверии чемпионата мира по футболу в России, была запущена система распознавания лиц в метро, и на сегодняшний день она уже помогла в поимке преступников.

В данной дипломной работе пойдёт речь о разработке программного продукта для замены лиц. На данный момент сфера применения лежит в основном в области приложений для развлечения, однако потенциально замена лиц может иметь и более серьёзное применение, например, при съемках биографических фильмов. Благодаря замене лиц, персонаж может получить действительно ту внешность, которую имел его прототип из прошлого.

1. Выбор среды разработки и сторонних библиотек

1.1. Выбор сторонних библиотек

OpenCV (OpenSourceComputerVision) - это библиотека функций программирования, в основном ориентированная на компьютерное зрение в реальном времени. Первоначально разработанная Intel, позднее была поддержана WillowGarage, затем Itseez (который позже был приобретен Intel). Библиотека является межплатформенной и бесплатной для использования в рамках лицензии BSD с открытым исходным кодом.

OpenCV написан на C ++, но он по-прежнему сохраняет менее всеобъемлющий старый интерфейс C. Есть привязки в Python, Java и MATLAB / OCTAVE. API для этих интерфейсов можно найти в онлайн-документации. Обертки на других языках, таких как C #, Perl, Ch, Haskell и Ruby были разработаны для поощрения принятия более широкой аудиторией.
Все новые разработки и алгоритмы в OpenCV теперь разрабатываются в интерфейсе C ++.

Официально запущенный в 1999 году, проект OpenCV был первоначально инициативой IntelResearch для продвижения приложений с интенсивным использованием процессора, частью серии проектов, включая трассировку лучей в реальном времени и трехмерные стенды дисплея. Основными участниками проекта были ряд экспертов по оптимизации в IntelRussia, а также команда IntelPerformanceLibraryTeam.

OpenCV обладает огромным количеством обучающих ресурсов [1], которые могут помочь в процессе выполнения выпускной квалификационной работы (см. рис 1.1).

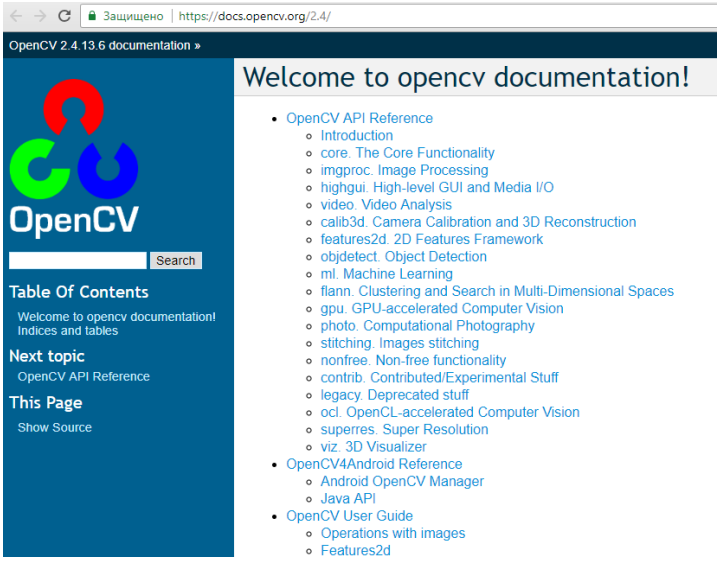

Рисунок 1.1 – Документация OpenCV

Dlib - это универсальная кроссплатформенная программная библиотека, написанная на языке программирования C++. На его конструкцию в значительной степени влияют идеи от проектирования по контракту и разработке программного обеспечения на основе компонентов. Таким образом, это, прежде всего, набор независимых программных компонентов. Это программное обеспечение с открытым исходным кодом, выпущенное под лицензией BSD [2].

С момента начала разработки в 2002 году, Dlib вырос до широкого спектра инструментов. По состоянию на 2016 год он содержит программные компоненты для работы с сетями, потоками, графическими пользовательскими интерфейсами, структурами данных, линейной алгеброй, машинным обучением, обработкой изображений, интеллектуальным анализом данных, анализом XML и текста, численной оптимизацией, байесовскими сетями и многими другими задачами. В последние годы большая часть развития была сосредоточена на создании широкого набора инструментов статистического машинного обучения, а в 2009 году Dlib был опубликован в журнале «Исследование машинного обучения». С тех пор он используется в широком диапазоне областей.

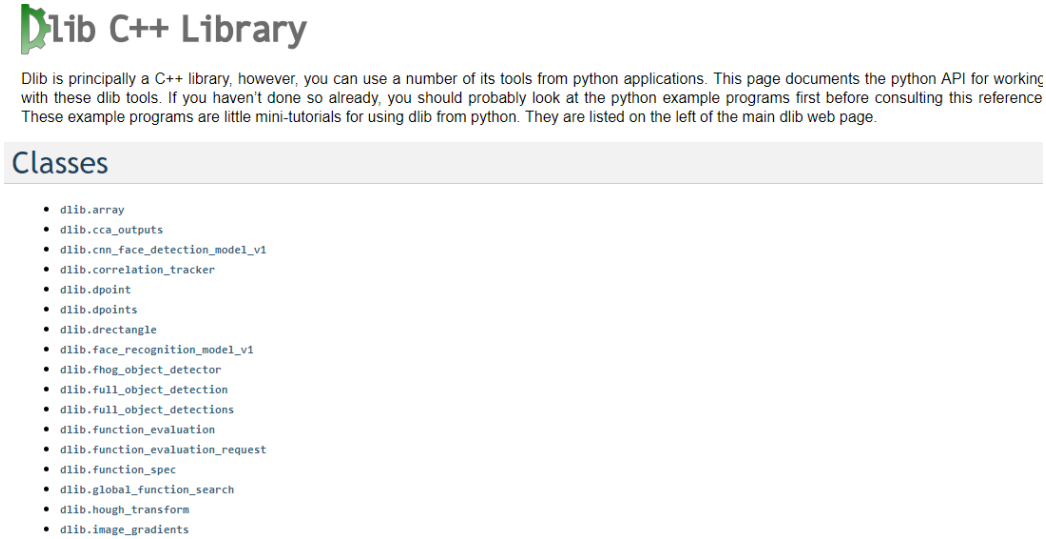

Рисунок 1.2 – Документация Dlib

1.2. Выбор среды разработки

Так как библиотеки OpenCV и Dlib изначально были разработаны на языке c++, то наиболее целесообразным будет использование c++ для разработки приложения для замены лиц.

Была выбрана MicrosoftVisualStudio, так как это одно из наиболее удобных средств разработки на c++.

2. АЛГОРИТМ ОБНАРУЖЕНИЯ И ЗАМЕНЫ ЛИЦ

Для реализации замены лиц было решено использовать библиотеки OpenCV и Dlib. Это open-source библиотеки с исходным кодом на c++, следовательно, будет логичнее делать проект, используя этот язык.

Предположительно, для замены лиц на двух фотографиях, необходимо будет проделать следующие шаги:

1) обнаружение лица с помощью алгоритма Виолы-Джонса;
2) в области лица, найденной в предыдущем пункте, необходимо найти опорные точки (к ним относятся овал лица, губы, нос, глаза и брови);
3) построение триангуляционной сетки по опорным точкам;
4) применение аффинных преобразований и перенос получившейся маски;
5) решения проблемы различий в освещении и оттенках кожи.

Программа должна поэтапно выполнить приведённые выше подзадачи. Сначала с помощью библиотеки OpenCV, реализующей метод Виолы-Джонса можно найти квадрат, содержащий лицо.

Далее с помощью библиотеки Dlib находятся опорные точки лица и заносятся в текстовый файл. Опорные точки стоит хранить в файле, для того, чтобы в дальнейшем их можно использовать для решения иных задач. Очередность точек должна соответствовать очередности точек у предиктора Dlib, что позволит, при необходимости, работать только с частью лица (например, глазами).

Функционал OpenCV должен помочь в построении триангуляции Делоне, необходимой в дальнейшем. Далее используя аффинные преобразования треугольников, должна быть сформирована маска для переноса лица.

2.1 Алгоритм Виолы-Джонса

Обнаружение объектов с использованием Haar каскадных классификаторов - эффективный метод обнаружения объектов, был впервые предложен Полом Виолой и Майклом Джонсом в статье «Быстрое обнаружение объектов с использованием расширенного каскада простых функций» в 2001 году. Этот подход основан на механизме обучения, где каскадная функция обучается из множества положительных и отрицательных изображений. Затем он будет использоваться для распознавания объектов на других изображениях.

Здесь мы будем работать с распознаванием лиц. Изначально для алгоритма требуется множество положительных изображений (содержащих лица) и негативных изображений (не содержащих лица) для обучения классификатора. Затем нам нужно извлечь из него функции. Для этого используются функции Haar, показанные на рисунке ниже. Каждая функция представляет собой одно

значение, полученное путем вычитания суммы пикселей под белым прямоугольником из суммы пикселей под черным прямоугольником.

Теперь все возможные размеры и расположения каждого ядра используются для вычисления множества функций. Для каждого вычисления функции нам нужно найти сумму пикселей под белым и черным прямоугольниками. Чтобы решить эту проблему, они представили интегральные образы. Это упрощает вычисление суммы пикселей, насколько большим может быть количество пикселей, при операции, состоящей всего из четырех пикселей.

Но среди всех этих функций, которые мы вычислили, большинство не имеют значения. Например, рассмотрим изображение ниже. Верхний ряд показывает две хорошие функции. Первая выбранная функция, похоже, фокусируется на том, что область глаз часто темнее области носа и щеки. Вторая выбранная функция полагается на свойство, что глаза темнее, чем у переносицы. Но те же самые окна, применяемые на щеках или в любом другом месте, не имеют значения. Итак, как мы выбираем лучшие функции из более чем 160000 функций? Это достигается с помощью Adaboost (см. рис 2.1).

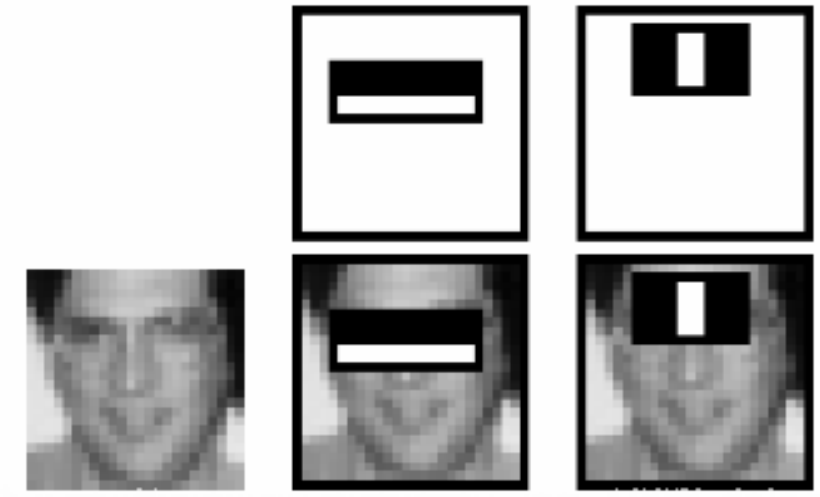

Рисунок 2.1 – Пример разбиения изображения

Для этого мы применяем каждую функцию на всех обучающих изображениях. Для каждой функции он находит лучший порог, который будет относить лица к положительным и отрицательным. Но, очевидно, будут ошибки. Мы выбираем функции с минимальной частотой ошибок, что означает, что они являются функциями, которые лучше всего классифицируют изображения лица и не лица. (Процесс не так прост: каждому изображению присваивается одинаковый вес в начале. После каждой классификации увеличивается количество ошибочных изображений. Далее опять выполняется этот же процесс, снова рассчитывающий коэффициенты ошибок и новые веса. Процесс будет продолжаться далее, пока не достигается требуемая точность или частота ошибок, или не будет найдено количество функций).

Конечный классификатор представляет собой взвешенную сумму этих слабых классификаторов. Он называется слабым, потому что он сам по себе не

классифицирует изображение, но в совокупности с другими он образует сильный классификатор. В документе говорится, что даже 200 функций обеспечивают обнаружение с точностью до 95%. Их окончательная настройка имела около 6000 функций.

Итак, теперь вы берете изображение. Возьмите каждое окно 24x24. Примените к нему 6000 функций. Проверьте, не лицо ли оно. Разве это не малоэффективно и требует много времени? Да. У авторов есть хорошее решение для этого.

В изображении большая часть области изображения является областью без лица. Поэтому лучше всего иметь простой способ проверить, не является ли окно областью лица. Если это не так, отбросьте его одним выстрелом. Не обрабатывайте его снова. Вместо этого сосредоточьтесь на регионе, где может быть лицо. Таким образом, мы можем найти больше времени для проверки возможной области лица.

Для этого они представили концепцию каскада классификаторов. Вместо того, чтобы применять все функции 6000 в окне, группируйте функции на разные этапы классификаторов и применяйте один за другим. (Обычно первые несколько этапов будут содержать очень мало функций). Если окно выходит из строя на первом этапе, отбросьте его. Мы не рассматриваем оставшиеся функции на нем. Если оно пройдет, примените второй этап функций и продолжите процесс. Окно, которое проходит все этапы, представляет собой область лица.

У детекторов авторов было более 6000 функций с 38 этапами с 1, 10, 25, 25 и 50 функциями на первых пяти этапах. (Две функции в приведенном выше изображении фактически получены как лучшие две функции от Adaboost). По мнению авторов, в среднем по одному фрагменту оценивается 10 признаков из 6000.

2.2 Нахождение опорных точек лица

Существуют различные детекторы лица, но все методы в основном пытаются локализовать и маркировать следующие области лица:

- рот;
- правая бровь;
- левая бровь;
- правый глаз;
- левый глаз;
- нос;
-челюсть.

Идентификатор лицевого ориентира, включенный в библиотеку dlib [3], представляет собой реализацию одномиллисекулярного выравнивания лица с ансамблем регрессионных деревьев, написанной Каземи и Салливаном (2014).

Этот метод начинается с использования учебного набора обозначенных лицевых ориентиров на изображении. Эти изображения помечены вручную,

указывая конкретные (x, y) - координаты областей, окружающих каждую структуру лица. Также используется вероятность на расстояние между парами входных пикселей.

Конечным результатом является детектор лица, который может использоваться для обнаружения ориентиров лица в реальном времени с использованием высококачественных прогнозов.

Описание детектора dlib:

Предварительно обученный датчик ориентировки лица в библиотеке dlib используется для оценки местоположения 68 (x, y) - координат, которые сопоставляются с лицевыми структурами на лице.

Индексы 68 координат можно увидеть на рис. 2.2.

Важно отметить, что существуют другие отличительные черты детекторов лицевого ориентира, в том числе модель с 194 точками, которую можно обучить набору данных HELEN.

Независимо от того, какой набор данных используется, та же структура dlib может быть использована для обучения предсказателя формы на входных данных обучения - это полезно, если вы хотите тренировать детекторы лицевого ориентира или собственные предсказатели собственной формы.

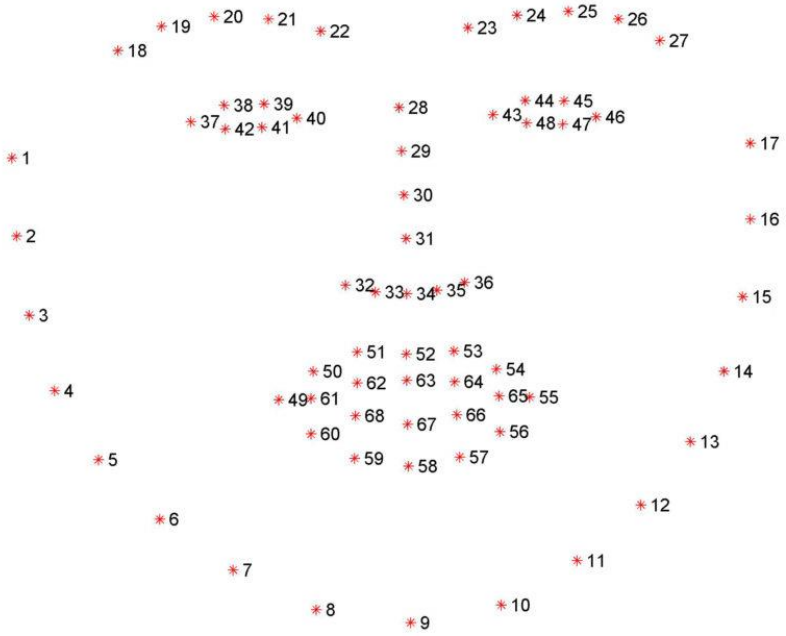

Рисунок 2.2 – Визуализация 68 опорных точек лица

2.3 Построение триангуляции Делоне

Что такое триангуляция Делоне? На множестве точек на плоскости, триангуляция относится к разбиению плоскости на треугольники, причем точки являются вершинами. Множество точек может иметь много возможных триангуляций, но триангуляция Делоне выделяется некоторыми хорошими

свойствами. В триангуляции Делоне треугольники выбираются так, чтобы ни одна точка не находилась внутри окружности любого треугольника. На рис 2.3 показана триангуляция Делоне в 4 точках A, B, C и D. В верхнем изображении, чтобы триангуляция была действительной триангуляцией Делоне, точка C должна находиться за пределами окружности треугольника ABD, а точка A должна находиться за пределами окружность треугольника BCD.

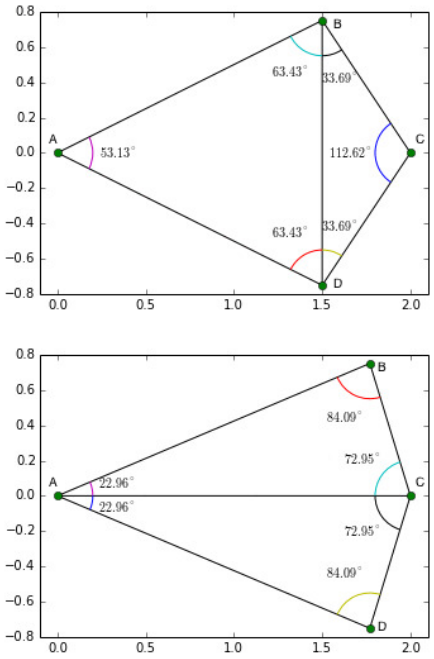

Рисунок 2.3 – Триангуляция Делоне в сравнении с обычной

Интересным свойством триангуляции Делоне является то, что оно не способствует «тощим» треугольникам (т.е. треугольникам с одним большим углом).

На рис. 2.3 показано, как изменяется триангуляция для выбора «жирных» треугольников при перемещении точек. В верхнем изображении точки B и D имеют x-координаты при x = 1,5, а в нижнем изображении они перемещаются вправо до x = 1,75. В верхних углах изображения ABC и ABD являются большими, а триангуляция Делоне создает грань между B и D, разделяющую два больших угла на меньшие углы ABD, ADB, CDB и CBD. С другой стороны, в нижнем изображении угол BCD слишком велик, а триангуляция Делоне создает ребро AC для разделения большого угла.

Существует множество алгоритмов для поиска множества точек триангуляции Делоне. Наиболее очевидным (но не самым эффективным) является начало любой триангуляции и проверка того, содержит ли окружность любого треугольника другую точку. Если это так, переверните края (как показано на рис 2.3) и продолжайте, пока не появятся треугольники, окружность которых содержит точку.

Любое обсуждение триангуляции Делоне должно включать диаграммы Вороного, потому что множество точек диаграммы Вороного математически двойственна множеству точек триангуляции Делоне.

Что такое диаграмма Вороного?

По заданному множеству точек в плоскости, диаграмма Вороного разбивает пространство так, что граничные линии равноудалены от соседних точек. На рис.2.4 показан пример диаграммы Вороного, рассчитанной по точкам, показанным черными точками. Заметьте, что каждая пограничная линия проходит через центр двух точек. Если вы соедините точки в соседних регионах Вороного, вы получите триангуляцию Делоне.

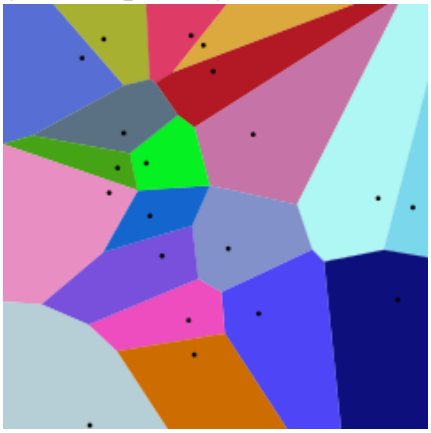

Рисунок 2.4 – Диаграмма Вороного

3. Разработка и тестирование приложения

Программа должна поэтапно выполнить приведённые выше подзадачи. Сначала с помощью библиотеки OpenCV, реализующей метод Виолы-Джонса можно найти квадрат, содержащий лицо.

Далее с помощью библиотеки Dlib находятся опорные точки лица и заносятся в текстовый файл. Опорные точки стоит хранить в файле, для того, чтобы в дальнейшем их можно использовать для решения иных задач. Очередность точек должна соответствовать очередности точек у предиктора Dlib, что позволит, при необходимости, работать только с частью лица (например, глазами).

Функционал OpenCV должен помочь в построении триангуляции Делоне, необходимой в дальнейшем. Далее используя аффинные преобразования треугольников, должна быть сформирована маска для переноса лица.

Далее на рисунках 3.1-3.6 отображён пример работы программы.

Рисунок 3.1-3.2 Исходные фото

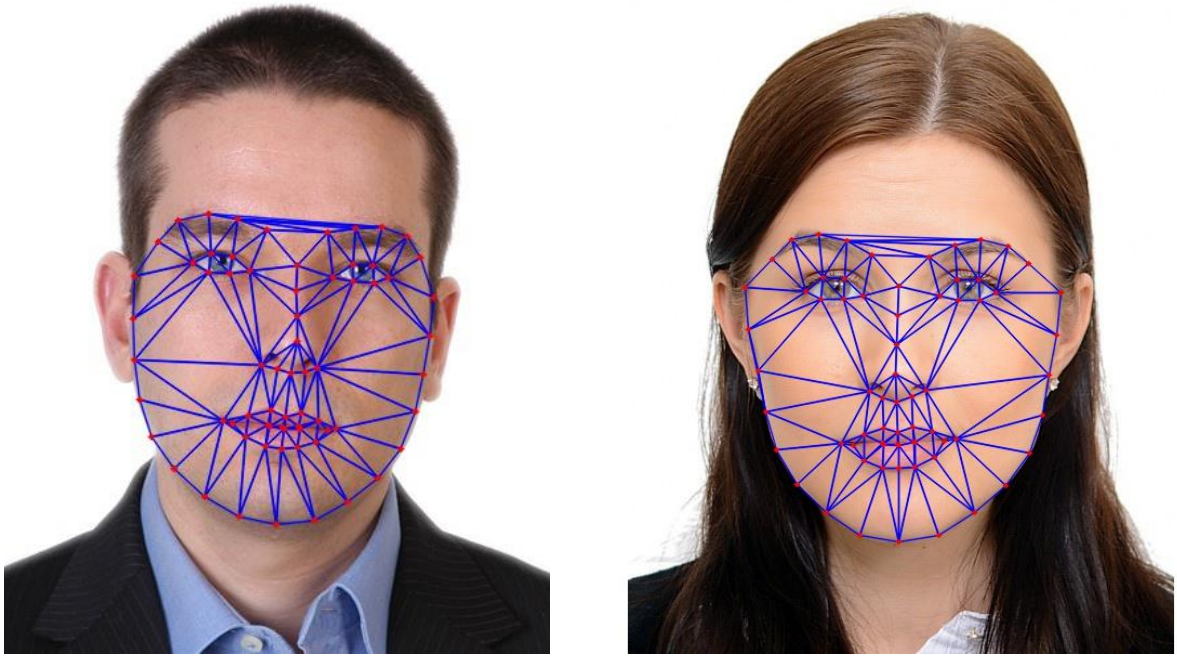

Рисунок 3.3-3.4 Результат нахождения опорных точек

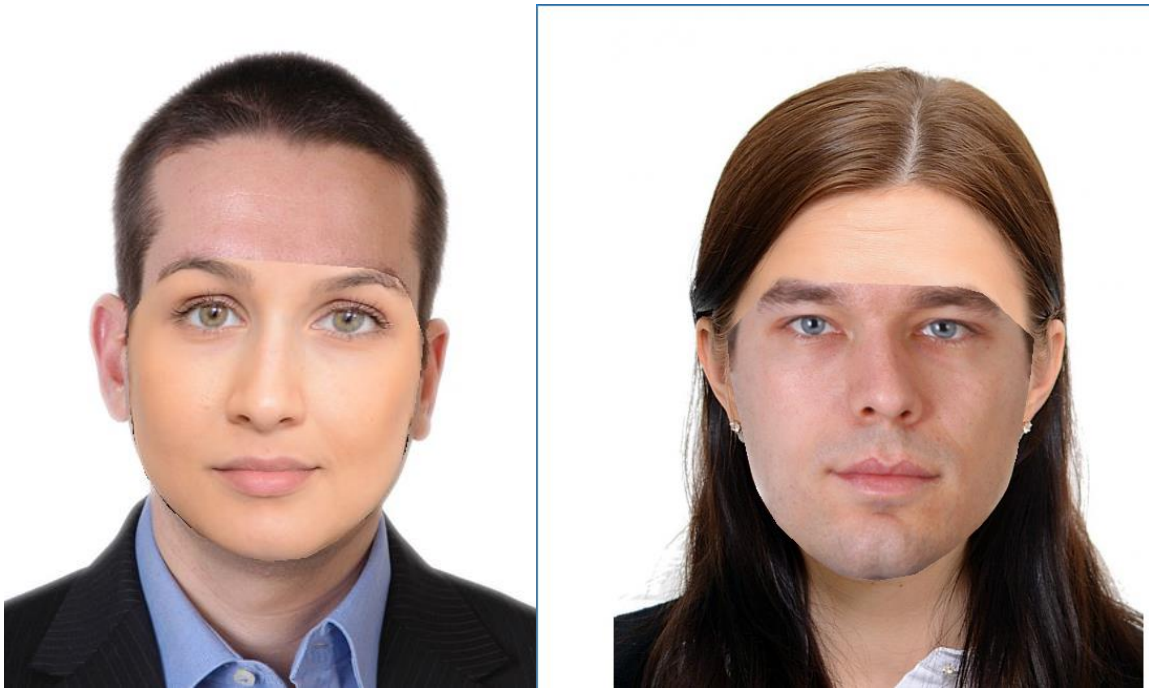

Рисунок 3.5-3.6 Результат замены лиц

Как видно из изображений 3.5-3.6, результат выглядит достаточно плохо, так как люди на фотографиях имеют разные оттенки кожи, а также фото сделаны при разном освещении. Для попытки решения данной проблемы будет использовано смешивание Пуассона, реализованное в функции seamlessClone

библиотеки opencv. Результат использования смешивания приведён на рисунках 3.7-3.8.

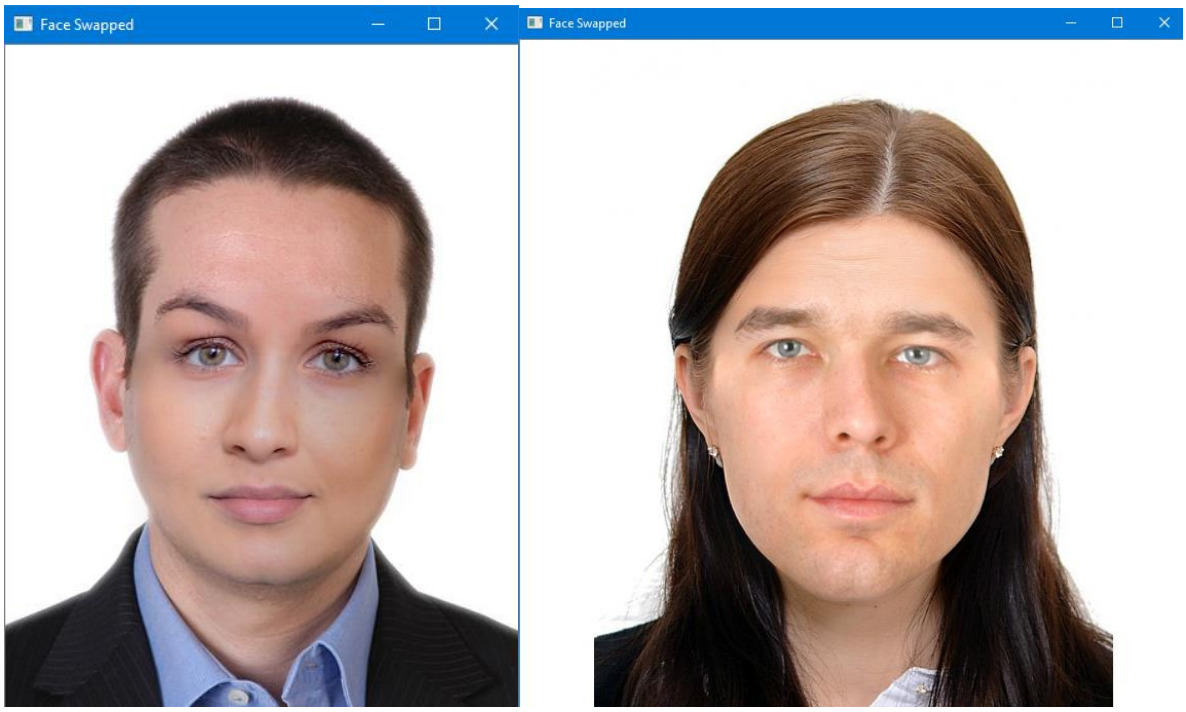

Рисунок 3.7-3.8 Результат замены лиц после смешивания Пуассона

Итоговый результат можно считать соответствующим поставленному в задаче.

ЗАКЛЮЧЕНИЕ

В процессе выполнения НИР были проведены исследования области алгоритмов обнаружения и замены лиц, приведены теоретические основы данной области. Были описаны вспомогательные библиотеки, помогающие в реализации работы. Реализовано и протестировано приложение, позволяющее производить замену лиц на фотографиях. Был улучшен первоначально задуманный алгоритм.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация библиотеки OpenCV [Электронный ресурс]. // URL: https://docs.opencv.org/ (дата обращения: 24.04.2018).

2. Документация библиотеки Dlib [Электронный ресурс]. // URL: http://dlib.net/ml.html/ (дата обращения: 28.04.2018).

3. LearnOpenCV [Электронный ресурс]. // URL: https://www.learnopencv.com/facemark-facial-landmark-detection-using-opencv// (дата обращения: 05.05.2018).

4. OpenCV установка и введение [Электронный ресурс]. // URL:http://robocraft.ru/blog/computervision/264.html/ (дата обращения: 12.05.2017).