МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО**

**ОБРАЗОВАНИЯ**

### «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

## Институт ИТКН

## Кафедра инженерной кибернетики

## Направление подготовки: «01.03.04 Прикладная математика» Квалификация: бакалавр

## Группа: БПМ-18-3

Отчёт по курсовой работе

**по дисциплине**

**«Методы и средства обработки изображений»**

**на тему «Замена лица на изображении»**

### Студент

## /\_\_\_\_\_Евстигнеев С.А.\_\_\_\_\_/

подпись

(Фамилия И.О.)

### Руководитель

## /доцент, к.т.н. Полевой Д. В./

подпись должность, уч. степ. Фамилия И.О.

### Оценка:

**Дата:**

### Москва 2022

**Оглавление**

[Введение 3](#_Toc103594205)

[Содержательная постановка задачи 3](#_Toc103594206)

[Работа программы 3](#_Toc103594207)

[Описание алгоритма 4](#_Toc103594208)

[Оценка результата 7](#_Toc103594209)

[Эксперименты 10](#_Toc103594210)

[Заключение 14](#_Toc103594211)

[Список источников 15](#_Toc103594212)

# Введение

В наше время программы и приложения по замене лиц стали крайне популярными среди пользователей любых областей деятельности. Наиболее известным и крупным приложением с данным функционалом являлось приложение MSQRD (Masquerade, с англ. — «Маскарад»). С 13 апреля 2020 года оно прекратило функционирование, как отдельное приложение и было полностью интегрировано в Facebook и instagram. Данная работа повещена разработке подобного приложения с использованием методов и средств для обработки изображений и оценке качества получившейся работы.

# Содержательная постановка задачи

Требуется реализовать алгоритм, который заменит одно из лиц на заданной фотографии другим и произведёт математическую оценку эффективности данной замены.

# Работа программы

Сборка проекта осуществляется через среду разработки Visual Studio 2022. Для корректной работы программы требуется дополнительно установить библиотеки OpenCV и Dlib. Программа и исходные файлы доступны по следующей ссылке: https://github.com/Eragon7700/Image-processing.git

Основные файлы программы:

* main.cpp – запуск алгоритма, вывод результата и его оценка качества;
* 1.jpg – фотография, с которой берётся лицо для замены;
* 2.jpg – фотография, на которой производится замена лица;

Результирующее изображение показано на экране, файл с ним будет сохранён в той же папке по окончании работы программы. Два изображения с линиями оценки исходного и получившегося изображения отображаются после работы программы без сохранения в файл. Оценка результата выводится в консоль по окончании работы программы.

# Описание алгоритма

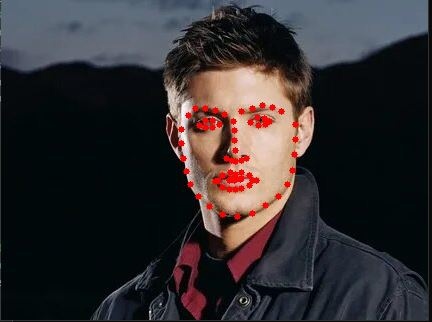
На примере следующий фотографий (1.jpg и 2.jpg соответственно) рассмотрим работу алгоритма.





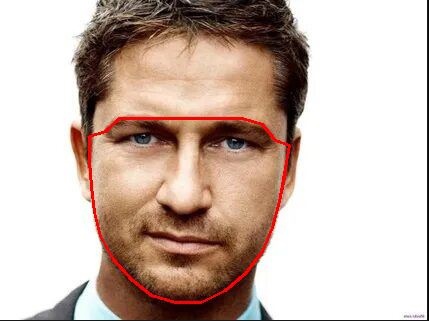
*Рисунок 1. Исходные изображения*

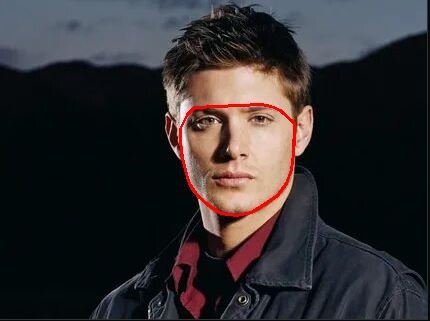
Для обнаружения лиц используется функции HOG в Dlib (гистограмма ориентированных градиентов). Она обнаруживает лицевые ориентиры на двух изображениях. Геометрия двух граней сильно различается, поэтому нужно слегка исказить исходную грань, чтобы закрыть целевую грань, но также необходимо убедиться, что она не искажается до неузнаваемости. Но в отличие от деформации лица, не требуется использовать все точки для выравнивания лица, нужны только точки на внешней границе лица, показанного на изображении.



*Рисунок 2. Обнаружение лиц и его ключевых точек*

Далее требуется найти выпуклую оболочку. В терминах компьютерного зрения и математики граница набора точек или фигур называется «оболочкой». Граница без углубления называется «выпуклой оболочкой». На рисунке 2 изображение показывает, что ориентиры лица, обнаруженные dlib, описаны красными точками, а на изображении 3 выпуклая оболочка точек показана красной линией.





*Рисунок 3. Выпуклая оболочка лица*

Следующим шагом в выравнивании является выполнение триангуляции Делоне точек на выпуклой оболочке с использованием функции *getTriangleList()*. Это позволяет разделить лицо на более мелкие части.

На последнем этапе выравнивания граней, учитывая соответствующий треугольник между исходной гранью и целевой гранью, мы аффинируем треугольник исходной грани на целевую грань. Для этого используем функции *getAffineTransform()* и *warpAffine()*. Но, как можно видеть, на изображении на рисунке 4, выравнивание граней и приклеивание одной грани к другой выглядит почти неестественно. Из-за разницы в освещении и тоне кожи между двумя изображениями видны швы.



*Рисунок 4. Выровненная и приклеена маска*

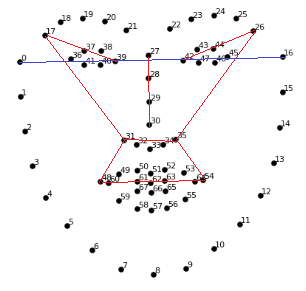
Следующим и последним шагом в замене лица на изображении идёт объединение двух изображений с использованием бесшовного клонирования. Реализован данный алгоритм с использованием поэлементного произведения изображения и маски с использованием функции *multiply()*. Результат представлен на рисунке 5.



*Рисунок 5. Итог работы программы*

# Оценка результата

Оценка эффективности работы алгоритма проверялась путём проверки пропорциональности масштабирования черт лица относительно его ширины. В работе проверялась сохранность пропорциональности 9 линий, соединяющих основные внутренние части лица. Данные линии брались, так как наибольшее зрительное искажение фотографии будет видно именно по ним. Список точек для анализа представлен в таблице 1. Нумерация взята с рисунка 6 [1].

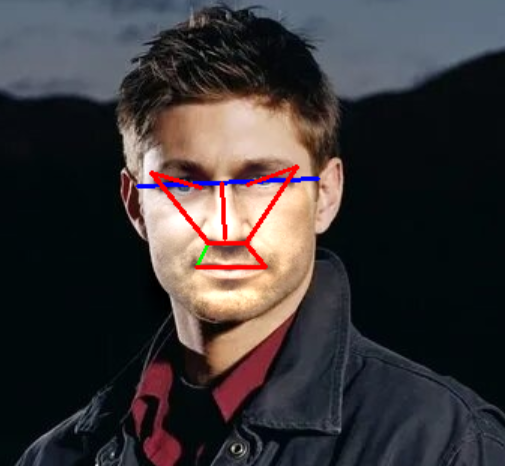
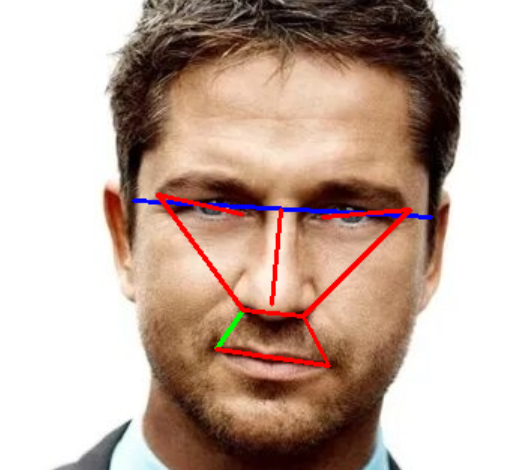


*Рисунок 6. Все точки лица и линии анализа.*

|  |  |  |  |
| --- | --- | --- | --- |
| Точка 1 | Наименование | Точка 2 | Наименование |
| 0 | TempleLeft | 16 | TempleRight |
| 17 | EyebrowLeftOuter | 39 | EyeLeftInner |
| 26 | EyebrowRightOuter | 42 | EyeRightInner |
| 27 | NoseTop | 30 | NoseTip |
| 31 | NoseLeftAlar | 35 | NoseRightAlar |
| 48 | MouthLeft | 54 | MouthRight |
| 17 | EyebrowLeftOuter | 31 | NoseLeftAlar |
| 26 | EyebrowRightOuter | 35 | NoseRightAlar |
| 48 | MouthLeft | 31 | NoseLeftAlar |
| 35 | NoseRightAlar | 54 | MouthRight |

*Таблица 1. Список анализируемых точек с их наименованиями [2].*

Сами линии представлены на рисунке 7, где синяя линия – основа для проверки масштабирования, зелёная – линия с наибольшим отклонением масштаба от полученной нормы.



*Рисунок 7. Линии проверки до и после и маска.*

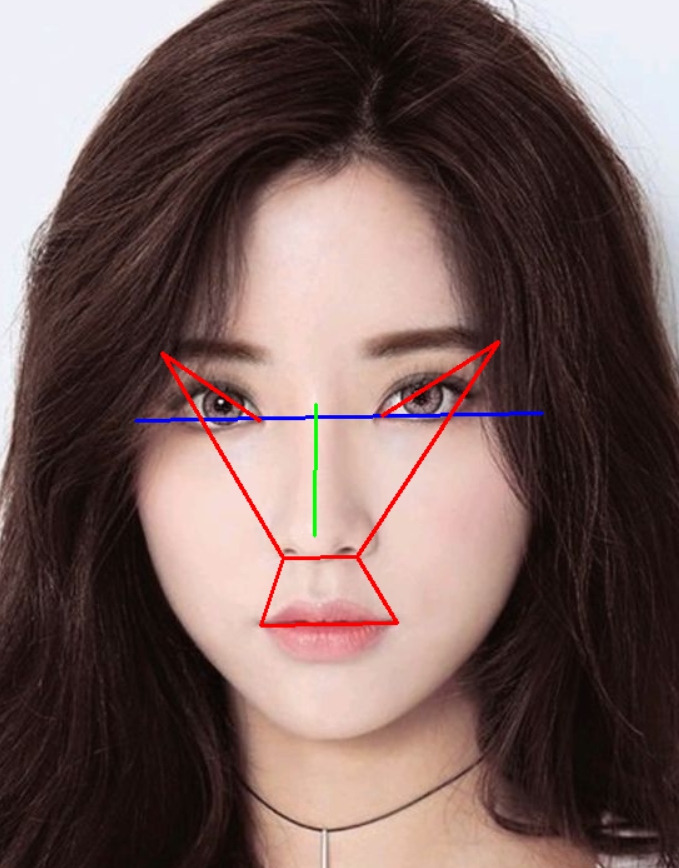
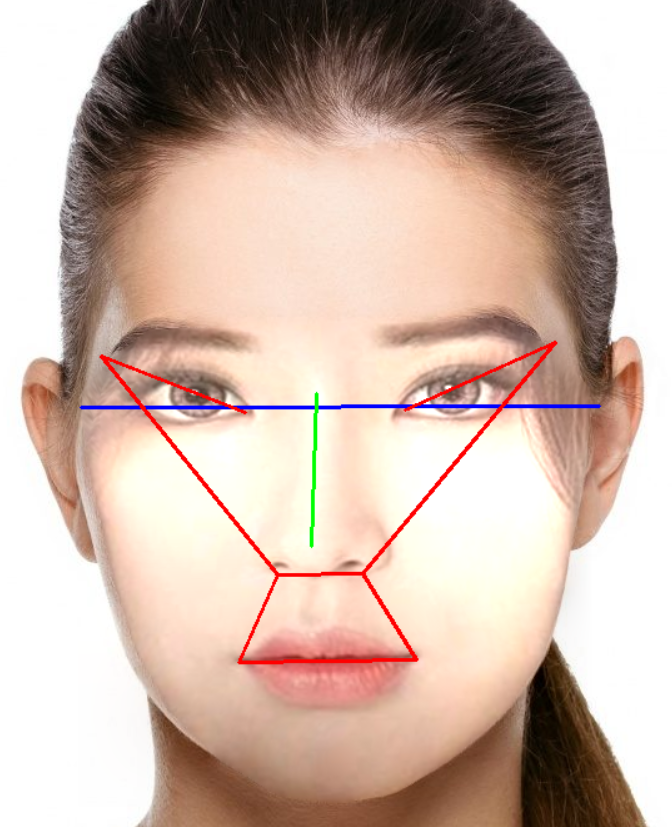
В качестве результата получаем 2 числа: среднее и максимально отклонение от нормы. Среднее отклонение рассчитывается, как среднее арифметическое, так как оно очень чувствительно к выбросам, а любой выброс может привести к искажению фотографии. Получившиеся отклонения переводятся в проценты и выводятся в консоль в формате «Абсолютная эффективность – 90%, средняя эффективность – 97%» (для данного примера)

# Эксперименты

Работа алгоритма так же проверялась на различных изображениях. В данном отчете представлены 4 группы изображений в формате «Было - Стало» с различными исходными данными: национальность, пропорции лица, наклон головы. Результаты представлены на рисунках 8 – 15.



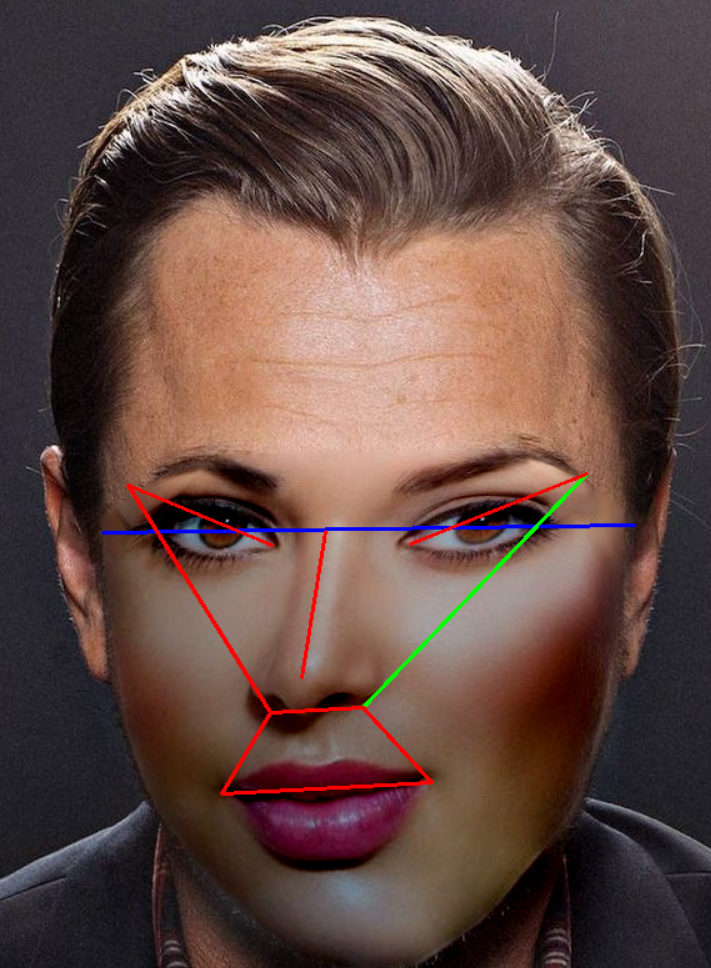
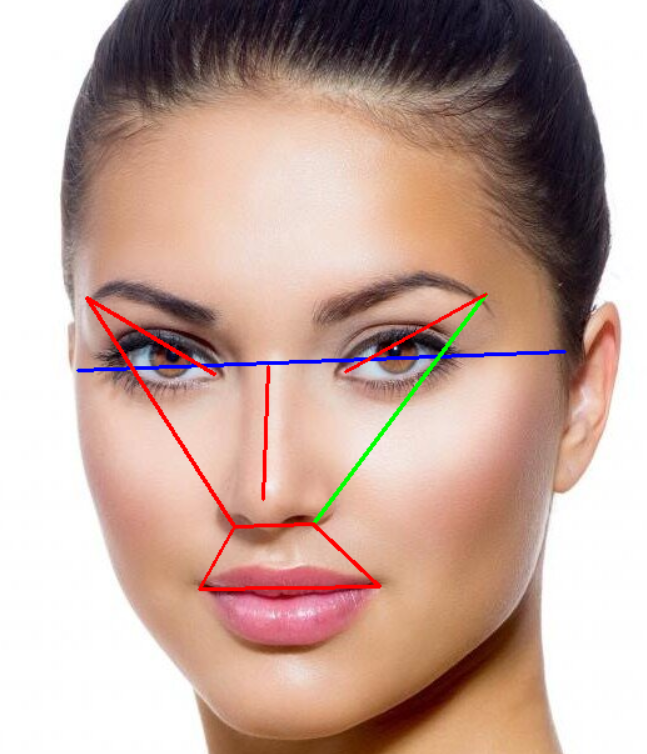
*Рисунок 8. Эксперимент 1. Разная форма лица и челка. Было.*

*Рисунок 9. Эксперимент 1. Разная форма лица и челка. Стало. Эффективность 89/94%.*

**

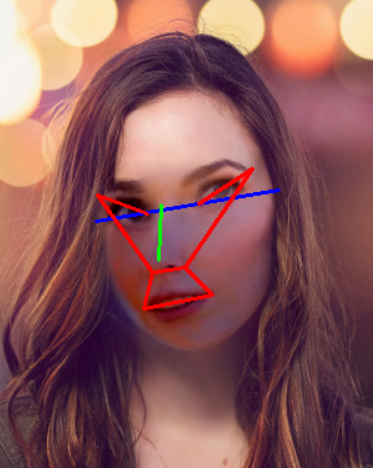
*Рисунок 10. Эксперимент 2. Поворот лица. Было.*



*Рисунок 11. Эксперимент 2. Поворот лица. Стало. Эффективность 82/93%.*

**

*Рисунок 12. Эксперимент 3. Наклон головы. Было*

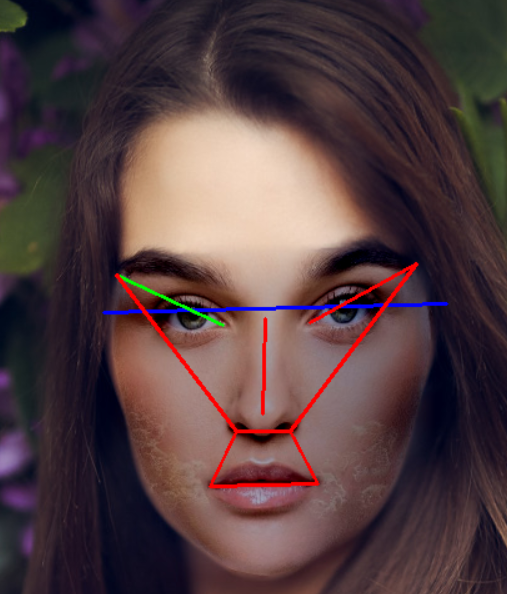
 

*Рисунок 13. Эксперимент 3. Наклон головы. Стало. Эффективность 83/92%.*





*Рисунок 14. Эксперимент 4. Было.*

**  

*Рисунок 15. Эксперимент 4. Стало. Эффективность 92/95%.*

# Заключение

В результате работы было создано приложение для замены лиц людей на фотографиях. Алгоритм показывает приемлемые результаты как с точки зрения математической оценки, так и с точки зрения восприятия глазом человека. Наилучше результат получается при отсутствии наклонов лица и одинаковом цвете изображений. Значительной зависимости от национальности, пола и фона не замечено (в пределах разумных допущении).

В качестве дальнейшего развития программы можно повысить качество слияния изображений и работу приложения в реальном времени.

# Список источников

[1] Асташенкова, Л. К. Распознавание ключевых точек лица на изображении человека / Л. К. Асташенкова, С. Н. Рощупкина, К. В. Кудринская. // Молодой ученый. — 2019. — № 26 (264). — С. 23-25.

[2] Определение лиц и их атрибутов //2022 // URL: https://docs.microsoft.com/ru-ru/azure/cognitive-services/face/concepts/face-detection