# Детекција на лица и замена на лице

Ментор:
д-р Ивица Димитровски

Изработил:
Ангела Ангелеска 221083

Елена Талеска 223141

# Содржина

# 1. Апстракт

Оваа семинарска работа има за цел да ја претстави имплементацијата на алгоритмите за детекција и замена на лица преку демонстрација на соодветен код. Во имплементацијата се користат библиотеките OpenCV и dlib за обработка на слики и примена на овие алгоритми. Прво ќе биде даден вовед во концептот на детекција на лица, како што се основите на препознавање и анализа на лице, и нивната примена во компјутерската визија. Ќе бидат опишани основните алгоритми и методи што се користат во процесот на детекција и замена на лица, вклучувајќи техники за препознавање на лице, идентификација на карактеристични точки, и методи за комбинирање на лицата за создавање на нови визуелни прикази.

# 2. Вовед

Живееме во ера на брз напредок на технологијата, каде што дигиталното процесирање на слики игра клучна улога во различни аспекти на нашето секојдневие. Детекцијата на лица и замена на лица се две значајни области во оваа сфера, со широко распространети примени. Детекцијата на лица го вклучува процесот на идентификација и анализа на човечки лица во слики или видеа, што е основа за бројни модерни апликации, како што се системите за препознавање лица, видео надзор и слично.

Од друга страна, замената на лица вклучува методи кои дозволуваат трансформирање на едно лице во друго, создавајќи визуелни ефекти што се користат во филмови, игри и социјални медиуми. Со развојот на напредни алгоритми и техники, како што се откривање на карактеристични точки и замена на лица, овие технологии стануваат се попрецизни и подостапни. Напредокот во дигиталните ресурси и квалитетот на камери ја олеснуваат детекцијата и замената на лица, што отвора нови можности за креативност и иновации во визуелната уметност и медиумите. Овие технологии не само што ја подобруваат безбедноста и интеракцијата во дигиталниот свет, туку и овозможуваат нови форми на експериментирање и забавни искуства.

# 3. Алгоритми за детекција на лица

## 3.1. Dlib Frontal Face Detector

Детекторот на лица на dlib се базира на алгоритам за машинско учење наречен Histogram of Oriented Gradients (HOG), комбиниран со линеарен класификатор. Еве како функционира овој алгоритам:

1. Histogram of Oriented Gradients (HOG):

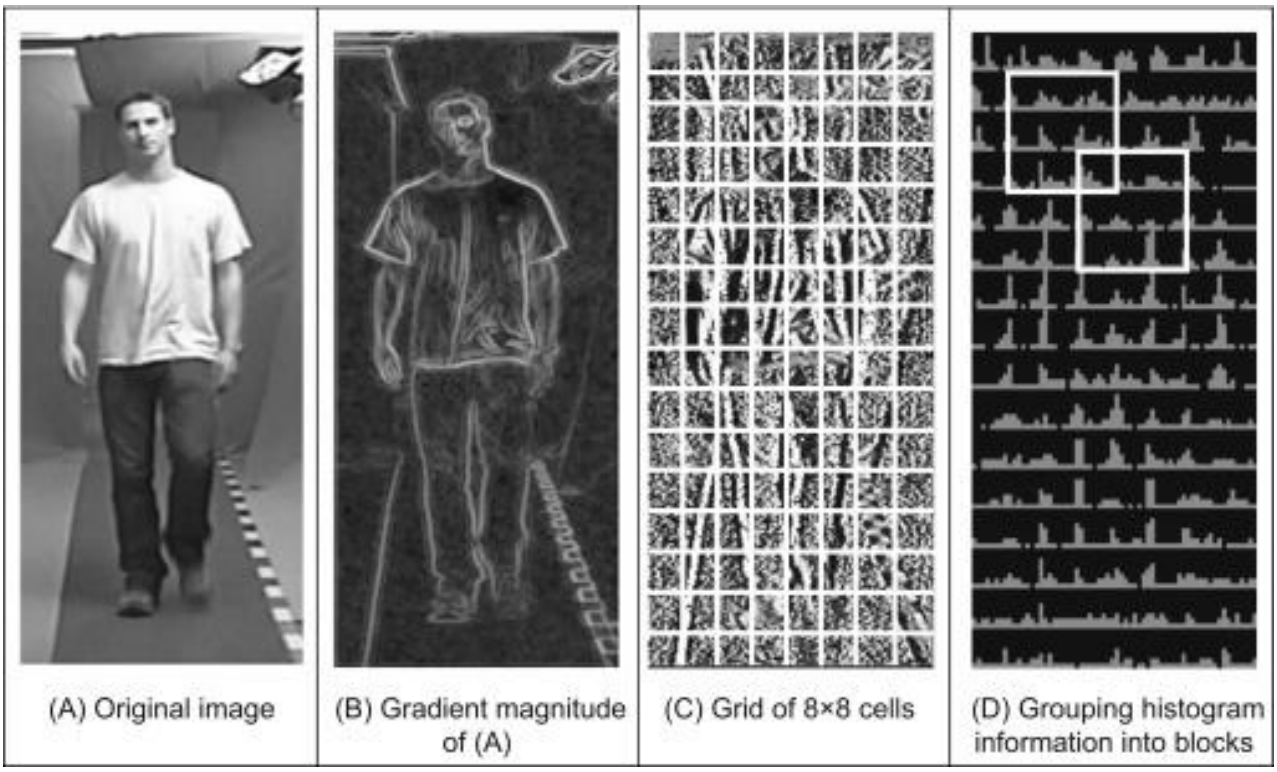
   HOG е техника за препознавање објекти која се фокусира на анализа на градиентите (промени во интензитетот на пикселите) на сликата. Процесот се состои од следниве чекори:

   - Пресметување на градиенти: Градиентите се пресметуваат за секој пиксел во сликата и покажуваат каде се наоѓаат рабови или текстури.
   - Делење на сликата: Сликата се дели на мали ќелии, обично 8x8 или 16x16 пиксели. Во секоја ќелија се прави хистограм на ориентираните градиенти.
   - Нормализација: Хистограмите на градиентите во секоја ќелија се нормализираат за да се осигура дека осветлувањето и контрастот не влијаат на препознавањето. Ова се прави со групирање на ќелиите во блокови (обично 2x2 ќелии).
   - Формирање на вектор на карактеристики: Нормализираните хистограми од сите блокови се комбинираат во долг вектор кој го претставува изгледот на објектот на сликата. Овој вектор се користи за идентификација на објекти, како што се лицата.

2. Линеарен класификатор

   Линеарниот класификатор е алгоритам кој користи математички формули за да направи одлука. Тој се обидува да одреди дали карактеристиките на сликата (на пример, векторот на карактеристики од HOG) припаѓаат на категорија, дали станува збор за лице, или не.

   Класификаторот е трениран на голем број на слики кои се означени како слики на кои има лица или не. Со овие примери, тој учи да ги препознава важните карактеристики што разликуваат лица од други објекти. Откако ќе биде истрениран, класификаторот користи математички формули за да пресмета дали новата слика содржи лице. Тој го прави ова така што ги проверува карактеристиките на сликата и ги споредува со шаблоните што ги научил.




Слика 1. Histogram of Oriented Gradients (HOG)

### 3.2. Eigenfaces

Eigenfaces е алгоритам за препознавање на лица базиран на метода наречена Principal Component Analysis (PCA). PCA е техника што се користи за намалување на димензионалноста на податоците и идентификување на најважните карактеристики.

### 3.3. Fisherfaces

Fisherfaces е алгоритам што се базира на Linear Discriminant Analysis (LDA), а се користи за препознавање на лица. LDA е техника што се користи за оптимизирање на разликувањето меѓу различни класи во податоците.

## 4. Алгоритам за замена на лица

Во оваа семинарска, разгледуваме алгоритам за замена на лица кој комбинира неколку познати техники од областа на компјутерската визија. Алгоритамот се состои од неколку фази, од откривање на лица до беспрекорно клонирање на лицето.

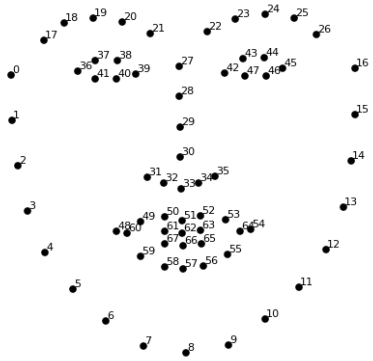
## 4.1. Детекција на лица

Алгоритамот Histogram of Oriented Gradients (HOG) се користи за препознавање на лица преку анализа на градиенти (нагли промени во бојата или светлината) на сликата. Методот ги дели сликите на мали делови наречени "ќелии" и ги мери правците на овие градиенти во секоја ќелија. Потоа создава хистограми на овие правци, што помага да се препознае структурата на лицето, како што се позициите на очите, носот и устата, без разлика на осветлувањето.

## 4.2. Откривање на значајни точки на лицата

По откривањето на лицето, се користи shape predictor од библиотеката dlib за идентификација на важните точки на лицето, наречени фацијални ознаки. Предикторот е трениран модел на машинско учење кој зема лице како влез и наоѓа 68 точки кои претставуваат специфични делови од лицето, вклучувајќи ги:

- брадата (1-16)
- веѓите (17-26)
- носот (27-35)
- очите (36-47)
- устата (48-68).



Слика 2. Фацијални ознаки

## 4.3. Делаунаова триангулација

Делаунаовата триангулација е математички метод што ги дели точките на лицето во мали триаголници. Целта е да се раздели лицето на триаголни делови што можат да се преместат или искриват, што овозможува адаптирање на лицето во друго лице.

## 4.4. Афинска трансформација

Афинската трансформација се користи за искривување на деловите од сликата, како што се триаголниците создадени со Делаунаова триангулација. Со оваа техника, може да се преместат, зголемат, намалат или искриват триаголниците од едно лице за да одговараат на соодветни триаголници од другото лице.

### 4.5. Беспрекорно клонирање на лица

По искривувањето и поставувањето на деловите од лицето, следниот чекор е беспрекорно клонирање, метод кој ги комбинира двете слики за да се направат преодите меѓу нив мазни и природни.

# 5. Имплементација

Оваа документација опишува програма за детекција и замена на лица базирана на Python. Примарната цел на програмата е да изврши замена на лица помеѓу две слики со извлекување на карактеристиките на лицето од изворната слика и интегрирање во целната слика. Програмата за да го постигне ова ги користи библиотеки како OpenCV, numpy, dlib и Tkinter.

## 5.1. Користени библиотеки

**a. OpenCV**

OpenCV (Open Source Computer Vision Library) е библиотека со отворен код во Python која обезбедува широк опсег на алатки за компјутерска визија и задачи за обработка на слики. Овозможува обработка на слики и видео во реално време и широко се користи во области како што се роботиката, машинското учење и вештачката интелигенција. Главни функционалности на оваа библиотека се:

- Обработка на слики и видео;
- Манипулација со слика;
- Филтри и трансформации;
- Замаглување и изострување на слики;
- Детекција на рабови;
- Детекција на карактеристики на слики (откривање на лице, очи, објекти);
- Машинско учење;

**b. NumPy**

NumPy (Numerical Python) е основна библиотека на Python што се користи за нумеричко пресметување. Обезбедува поддршка за низи, матрици и математички функции на високо ниво кои работат на овие структури, што ја прави суштинска алатка за научно пресметување, анализа на податоци и машинско учење. Главни функционалности на оваа библиотека се:

-Употреба на ефикасни повеќедимензионални низи;
-Математички и статистички операции;
-Генерирање на случаен број;
-Преобликување и манипулирање со низи;

**c. Dlib**

Dlib е моќен пакет со алатки со отворен код напишан во C++, кој се фокусира на обезбедување алгоритми за машинско учење и алатки за задачи како компјутерска визија, обработка на слики и анализа на податоци. Широко се користи за препознавање лица, откривање обележја на лицето и откривање објекти, но вклучува и алгоритми за машинско учење за општа намена, како што се машините

со векторски поддршка (SVM), стебла на одлука и модели за длабоко учење. Главни функционалности на оваа библиотека се:
- Откривање на лице: Dlib обезбедува најсовремен детектор за лице базиран на метод на Хистограм на ориентирани градиенти (HOG) или CNN (Convolutional Neural Network) базиран на длабоко учење. И двете можат ефективно да детектираат лица во слики и видео преноси.
- Откривање на обележја на лице: Dlib е добро познат по својот детектор за обележје на лицето со 68 точки, кој ги детектира клучните карактеристики на лицето како што се очите, носот, устата и вилицата. Ова е корисно за задачи како што се усогласување на лица, препознавање емоции и анализа на изразот на лицето.
- Препознавање на лице: Dlib поддржува препознавање лица со конвертирање на сликите на лицето во вектори со 128 димензии (вградувања). Овие вградувања потоа може да се споредат со користење на едноставни метрики на растојание за препознавање лица. Ова често се користи во врска со други библиотеки како OpenCV или рамки за длабоко учење.

**d. Tkinter**

Tkinter е стандардната библиотека за графички кориснички интерфејс (GUI) за Python, која овозможува креирање на прозорци и интерфејси со копчиња, менија, текстуални полиња, слики и други компоненти. Таа е лесна за употреба и многу популарна за мали и средни проекти поради својата едноставност и директна интеграција во Python. Tkinter користи објектно-ориентиран пристап за создавање GUI, при што се користат "widgets" како градбени блокови за прозорецот и неговите елементи.

## 5.2. Имплементација во Python

Во следниот дел од семинарската, ќе го опишеме кодот користен во Python за имплементација на алгоритамот за детекција и замена на лица. Секој чекор ќе биде детално објаснет.

Чекор 1: Вчитување на сликите

Првиот чекор се состои од вчитување на двете слики што ќе ги користиме за замена на лицата. Овој процес вклучува избор на слики од нашиот компјутер и нивно вчитување во програмата.

```
source_face = load_image("Target Face")
target_face = load_image("Source Face")
```

```python
def load_image(title):  2 usages
    root = tk.Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename(title=title, filetypes=[("Image files", "*.png;*.jpg;*.jpeg")])
    return cv2.imread(file_path) if file_path else None
```

Функцијата се повикува два пати за да вчита две слики: source_face – лицето што сакаме да го пренесеме и target_face – лицето на кое сакаме да го поставиме претходното.

Слика 3. Source face

Слика 4. Target face

Чекор 2: Подготовка на сликите

Подготовката на сликите е важна за правилна обработка. Овој чекор вклучува две главни активности: конвертирање на сликите во grayscale и создавање на празни маски.

1. Конвертирање на сликите во grayscale

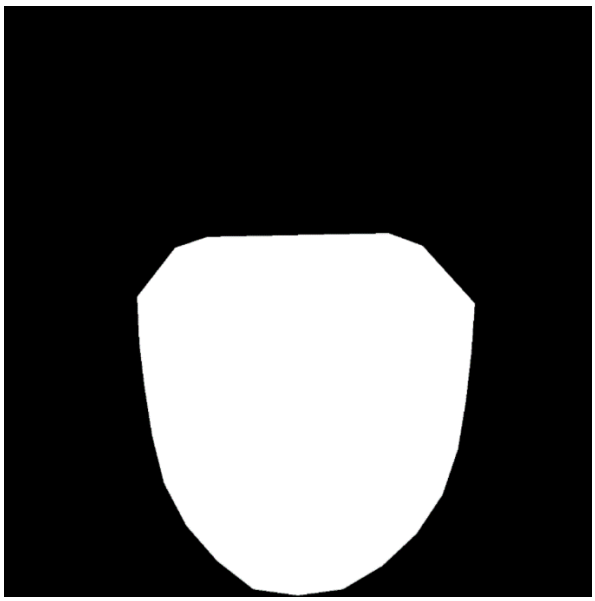
```python
img_gray = cv2.cvtColor(source_face, cv2.COLOR_BGR2GRAY)
img2_gray = cv2.cvtColor(target_face, cv2.COLOR_BGR2GRAY)
```

Конвертирањето на сликите во grayscale го олеснува процесот на детекција на лица, бидејќи компјутерот може полесно да го препознае лицето без да се грижи за боите. Сликата се претставува со различни нијанси на сива, што помага во детектирање на објекти како што се лица.

2. Создавање на празни маски

```
mask = np.zeros_like(img_gray)
```

Маската се користи за да се означи каде се наоѓа лицето на сликата. Кога ќе го најдеме лицето, маската ќе служи како шаблон за селектирање на определен дел од сликата. Прво, создавајќи празна маска (црна), а потоа ја пополнуваме со бела боја на местото каде што се наоѓа лицето.



Слика 5. Маска

Чекор 3: Детекција на лица и наоѓање на значајни точки

Во овој чекор користиме специјални алатки за да го пронајдеме лицето на сликата и да измериме каде се наоѓаат различните делови на лицето. Овие информации се клучни за успешно заменување на лицата.

1. Детекција на лица на сликата

```
detector = dlib.get_frontal_face_detector()
faces = detector(img_gray)
```

За да ги замениме лицата, прво мораме да знаеме каде се наоѓаат. Ова се постигнува со помош на детектор на лица.

2. Пронаоѓање на значајните точки на лицето

```
predictor = dlib.shape_predictor('resources/shape_predictor_68_face_landmarks.dat')
landmarks = predictor(img_gray, face)
```

Откако ќе ја идентификуваме областа на лицето, мораме да ги пронајдеме значајните делови т.н. "клучни точки". Користиме предиктор за да ги идентификуваме овие точки, што ни помага да ја разбереме структурата на лицето и да извршуваме операции како што е триангулацијата. Овие точки се важни за точно позиционирање на новото лице.

Слика 6. Значајни точки на лицето

Чекор 4: Креирање на конвексен обвив

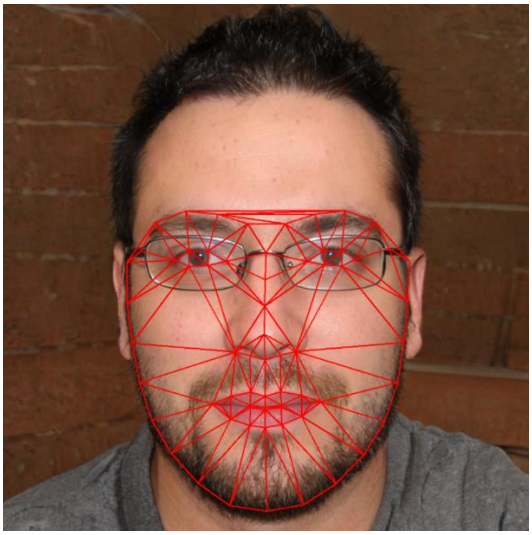
Откако ќе ги добиеме координатите на точките на лицето, следниот чекор е креирање на конвексниот обвив. Конвексниот обвив е полигонот што ја опфаќа најмалата конвексна област која ги содржи сите точки. Овој процес е важен за изградба на маската која ќе се користи за мапирање на карактеристиките на лицето.

Слика 7. Конвексен обвив

Чекор 4: Триангулација на лицата и пренос на триаголници

1. Триангулација на лицето на изворната слика (source_face)

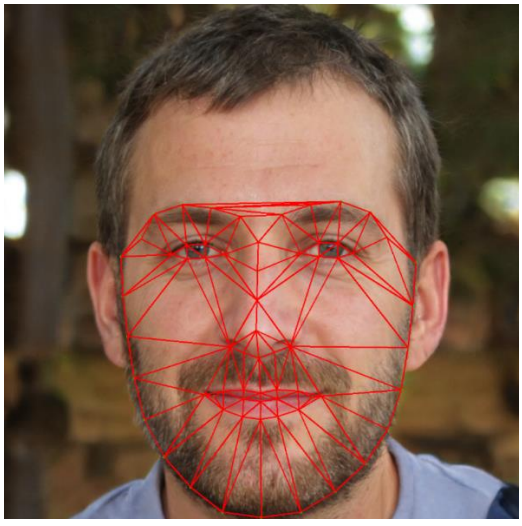
Триангулацијата го дели лицето на мали делови (триаголници), што овозможува поедноставен пренос на делови од едно лице на друго. Овој метод е особено корисен за пренос на обликот и структурата на лицето.



Слика 8. Триангулација на source face

2. Триангулација на лицето на целната слика (target_face)

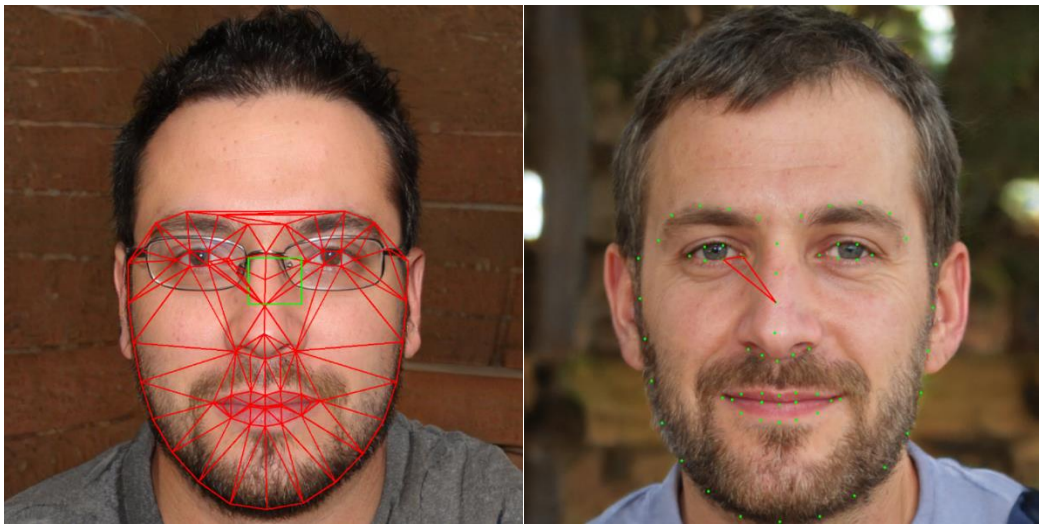
Исто како со изворната слика, триангулацијата на целната слика создава мали делови кои ќе се користат за замена. Ова обезбедува сигурност дека деловите од лицето се совпаѓаат правилно при преносот.



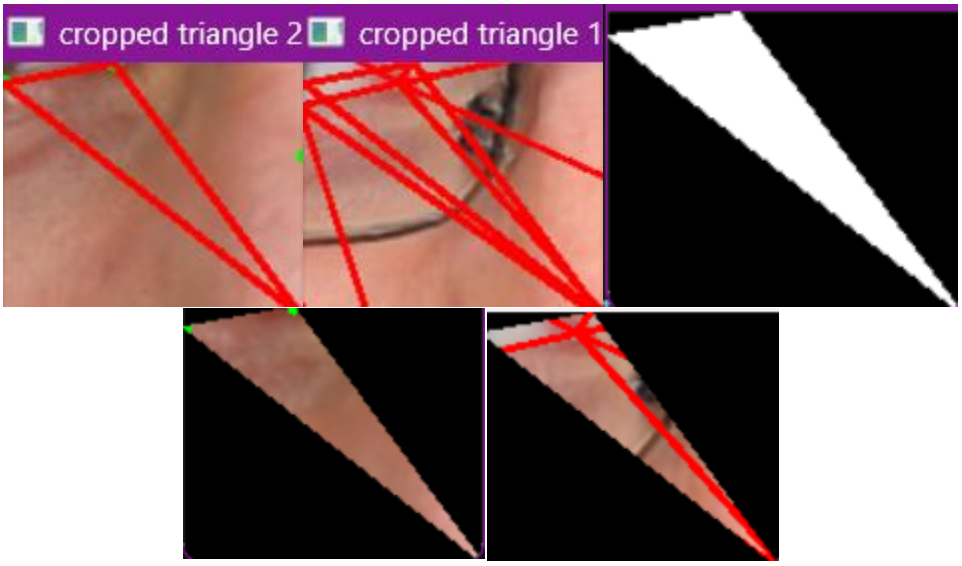
Слика 9. Триангулација на target face

3. Пренос на триаголници од изворната слика на целната слика

Овој чекор овозможува точен пренос на обликот и деталите од изворното лице на целната слика. Афинските трансформации се користат за искривување и адаптирање на деловите од изворното лице, така што тие ќе се совпаѓаат со формата на целната слика.



Слика 10. Триангулација




Слика 11. Триангулација

Чекор 5: Комбинирање на сликите и завршна обработка

Во овој чекор, секој триаголник од изворната слика се пренесува на соодветниот триаголник на целната слика преку афинска трансформација. Афинската трансформација е математички процес што дозволува прецизно менување на обликот, големината, и положбата на триаголникот, со цел да се вклопи во целната слика.

Слика 12. Завршна обработка на сликите

Слика 13. Влезни тест слики

Слика 14. Резултат

# Заклучок

Во оваа семинарска работа, го разгледавме процесот на детекција и замена на лица со помош на современи алгоритми и библиотеки како што се dlib и OpenCV. Применивме различни техники за детекција на лице, како што е Dlib Frontal Face Detector, кој користи Histogram of Oriented Gradients (HOG) и линеарен класификатор за точна идентификација на лицата во сликите. Овој детектор покажува висока прецизност и стабилност, што го прави идеален за примена во различни сценарија.
Следно, го разгледавме процесот на замена на лица, кој вклучува детекција на значајни точки на лицата, користење на Делаунаова триангулација за дефинирање на обликот на лицето и примена на Афински трансформации за создавање на природен изглед. Примената на овие техники овозможува создавање на убедливи и визуелно добри резултати, со значителни примени во области како што се анимација, видео игри и виртуелна реалност.
Имплементацијата на овие алгоритми во Python ја покажува моќта и флексибилноста на современите алатки за обработка на слики. Примената на библиотеките OpenCV и dlib значително го олеснува процесот на работа со слики и овозможува креирање на иновативни решенија за визуелни предизвици.
Заклучно, оваа семинарска работа не само што ја истражува теоретската основа на алгоритмите за детекција и замена на лица, туку и демонстрира практична имплементација и примена на тие алгоритми.

# Референци


[1] Face swapping – Opencv with Python
[2] Face swapping using face landmark detection