



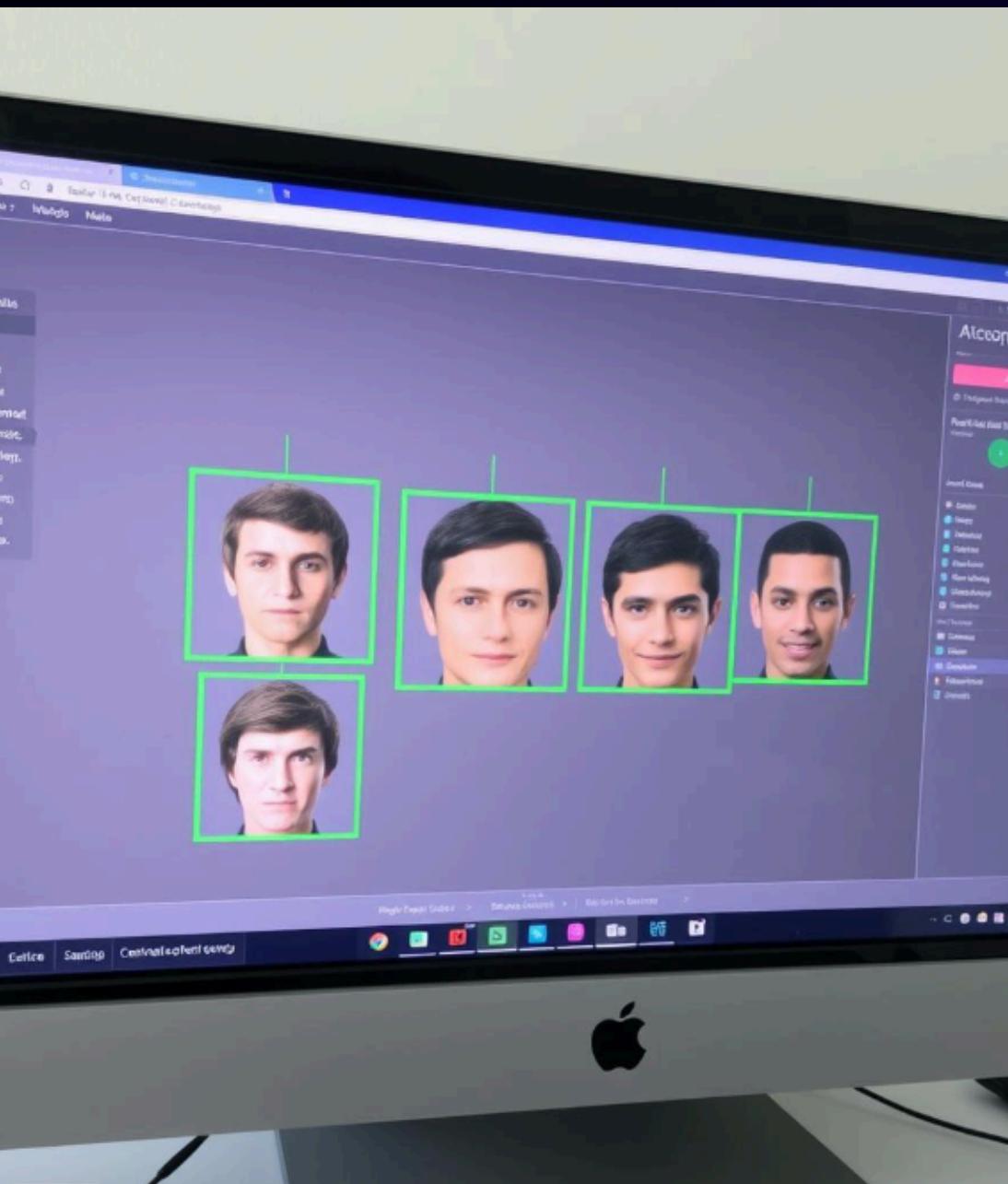
# REAL-TIME FACE RECOGNITION SYSTEM

Программа для идентификации лиц в реальном времени

Zhuldyz Kabibolla

# ЦЕЛЬ РАБОТЫ

Создание программного решения для распознавания лиц в режиме реального времени на языке Python, направленного на демонстрацию возможностей компьютерного зрения в задачах биометрической идентификации с возможностью дальнейшего усовершенствования для повышения эффективности систем видеонаблюдения.



РАСПОЗНАВАНИЕ ЛИЦ - ЭТО АВТОМАТИЧЕСКАЯ ЛОКАЛИЗАЦИЯ ЧЕЛОВЕЧЕСКОГО ЛИЦА НА ИЗОБРАЖЕНИИ ИЛИ ВИДЕО И, ПРИ НЕОБХОДИМОСТИ, ИДЕНТИФИКАЦИЯ ЛИЧНОСТИ ЧЕЛОВЕКА НА ОСНОВЕ ИМЕЮЩИХСЯ БАЗ ДАННЫХ. ИНТЕРЕС К ЭТИМ СИСТЕМАМ ОЧЕНЬ ВЕЛИК В СВЯЗИ С ШИРОКИМ КРУГОМ ЗАДАЧ, КОТОРЫЕ ОНИ РЕШАЮТ.

ТЕХНОЛОГИЧЕСКИ СИСТЕМЫ ИНОГДА МОГУТ СИЛЬНО ОТЛИЧАТЬСЯ В ПЛАНЕ РАСПОЗНАВАНИЯ ЛИЦ, НО ВСЕ ОНИ ИМЕЮТ ПРИМЕРНО ОБЩИЕ ПРИНЦИПЫ РАБОТЫ.

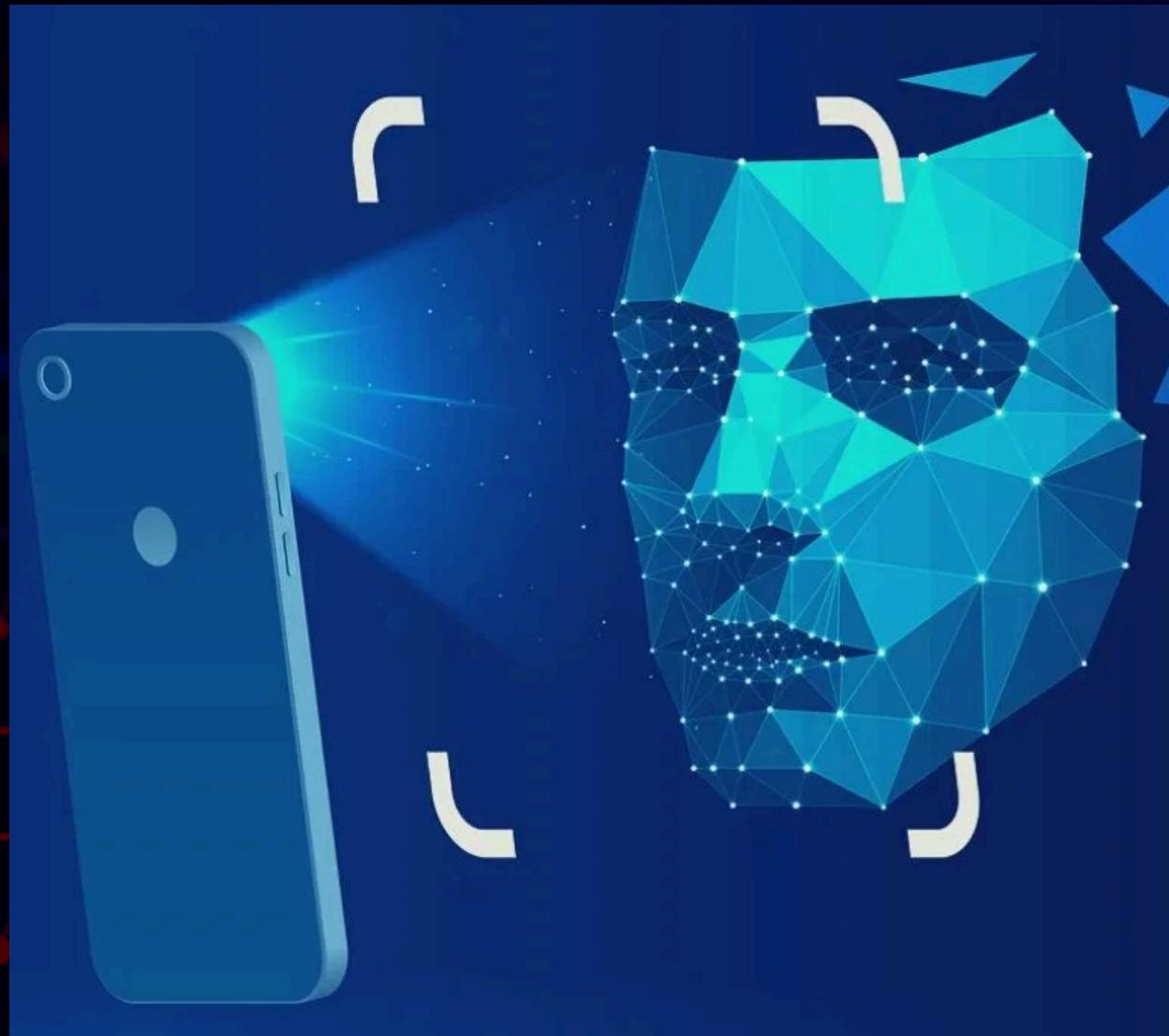
- ШАГ 1: ОБНАРУЖЕНИЕ ЛИЦА
- ШАГ 2: АНАЛИЗ ЛИЦА
- ШАГ 3: КОНВЕРТАЦИЯ ИЗОБРАЖЕНИЯ В ДАННЫЕ
- ШАГ 4: ПОИСК СОВПАДЕНИЙ





## ГДЕ ИСПОЛЬЗУЕТСЯ РАСПОЗНАВАНИЕ ЛИЦ?

- ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ В МЕСТАХ БОЛЬШОГО СКОПЛЕНИЯ ЛЮДЕЙ;
- СИСТЕМЫ ОХРАНЫ
- ВЕРИФИКАЦИЯ БАНКОВСКИХ КАРТ;
- ОНЛАЙН-ПЛАТЕЖИ;
- КРИМИНАЛИСТИКА;
- В КАЧЕСТВЕ РАЗБЛОКИРОВКИ ТЕЛЕФОНА – СЕЛФИ, СНЯТОЕ ВЛАДЕЛЬЦЕМ ТЕЛЕФОНА НА ФРОНТАЛЬНУЮ КАМЕРУ, СРАВНИВАЕТСЯ С ЗАРАНЕЕ ЗАГРУЖЕННЫМ ФОТО-ЭТАЛОНом.



# ЧТО ДЕЛАЕТ ПРОГРАММА?

- Определяет лица с веб-камеры.
- Сравнивает их с заранее загруженными фото.
- Фиксирует, сколько раз появилось каждое лицо.
- Выводит результаты в удобном графическом окне Tkinter.
- Работает в реальном времени.

## ИСПОЛЬЗУЕМЫЕ БИБЛИОТЕКИ

1. OpenCV – захват видео и отображение.
2. face\_recognition – распознавание лиц.
3. time – для контроля частоты подсчёта.
4. tkinter – графический интерфейс.





# ЛОГИКА РАБОТЫ ПРОГРАММЫ



Загрузка изображений известных людей из папки known\_faces.



Запуск веб-камеры и захват кадров.



Поиск лиц в каждом кадре.



Сравнение найденных лиц с базой данных.



Вывод имени найденного человека и обновление статистики.



Отображение информации в графическом интерфейсе.



# ИСПОЛЬЗОВАНИЕ БИБЛИОТЕК

```
import cv2
```

захват видео с веб-камеры, отображение видео и обработка изображений (рамки, текст).

```
import face_recognition
```

обнаружение лиц, вычисление и сравнение их признаков (энкодингов).

```
import time
```

измерения времени между распознаваниями

```
import tkinter as tk
```

GUI  
Создаёт окно с таблицей, где отображаются имена и количество распознаваний;  
Отображает дополнительные метрики (уникальные лица, процент распознавания);  
Обрабатывает закрытие приложения через кнопку.

```
from tkinter import ttk
```



# РАБОТА ФУНКЦИЙ

SLIDE 6

```
known_face_encodings = []
known_face_names = ["Zhannur", "Abdurakhim", "Timur", "Zhuldyz",
known_face_images = ["known_faces/zhannur.jpg", "known_faces/abd
| | | | | | | | "known_faces/zhuldyz.jpg", "known_faces/dan
| | | | | | | | "known_faces/timur.jpg", "known_faces/zhuldyz.jpg"]
```

Загружаются изображения из папки, посчитает его энкодинг и добавит в k.f\_encodings. Лицо в камере сравнивает с этими энкодингами и выдаст имя, если найдёт совпадение.

```
face_counts = {name: 0 for name in known_face_names}
last_seen = {name: 0 for name in known_face_names}
unique_faces = set()
total_faces_detected = 0
```

Подготовка данных: загрузка изображения известных лиц, создание энкодингов.

1. переменные  
face\_counts, last\_seen, unique\_faces, total\_faces\_detected.

2. загрузка изображений  
проходит по списку фото, загружает их и ищет лицо

3. готовность к работе: когда все фото обработаны, выводится сообщение о готовности к работе с камерой

```
root = tk.Tk()
root.title("📊 Статистика лиц")
root.geometry("500x350")

frame = ttk.Frame(root)
frame.pack(pady=10)
```

Графический интерфейс - отображение итоговой статистики по итогам работы системы.

Цикл проходит по списку известных лиц и добавляет строку с именем и количеством появлений

```
metrics_frame = ttk.LabelFrame(root, text="📊 Метрики")
metrics_frame.pack(pady=10, fill="both", expand=True)
```

Вывод статистики работы системы распознавания лиц.

Создаются две текстовые метки:

1. Количество уникальных лиц (число разных людей, которые были распознаны).
2. Процент успешного распознавания (сколько лиц удалось идентифицировать).



# СТАТИСТИКА РАСПОЗНАВАНИЯ

```
def update_table():
    global total_faces_detected
    if total_faces_detected > 0:
        recognition_rate = (len(unique_1
```

Обновление статистики в реальном времени

- Обновляет таблицу с количеством распознанных лиц.
- Обновляет метрики (уникальные лица и процент распознавания).
- Запускает себя снова через 1 секунду, поддерживая актуальные данные.

Процент распознавания=(уникальных лиц/всего обнаруженных лиц)×100

Закрытие приложения и запуск обновления статистики

Добавляется кнопка Закрыть.

Освобождает ресурсы камеры.

Запускает обновление статистики в реальном времени (update\_table()).

```
def close_app():
    video_capture.release()
    cv2.destroyAllWindows()
```

## ОСНОВНОЙ ЦИКЛ ОБРАБОТКИ ВИДЕО

```
while True:
    ret, frame = video_capture.read()
    if not ret:
        print("Ошибка при захвате кадра с камеры!")
        break

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb
    face_encodings = face_recognition.face_encodings(rgb

    total_faces_detected += len(face_encodings)
    current_time = time.time()

    for (top, right, bottom, left), face_encoding in zip
```

Считывание поток с камеры в реальном времени.

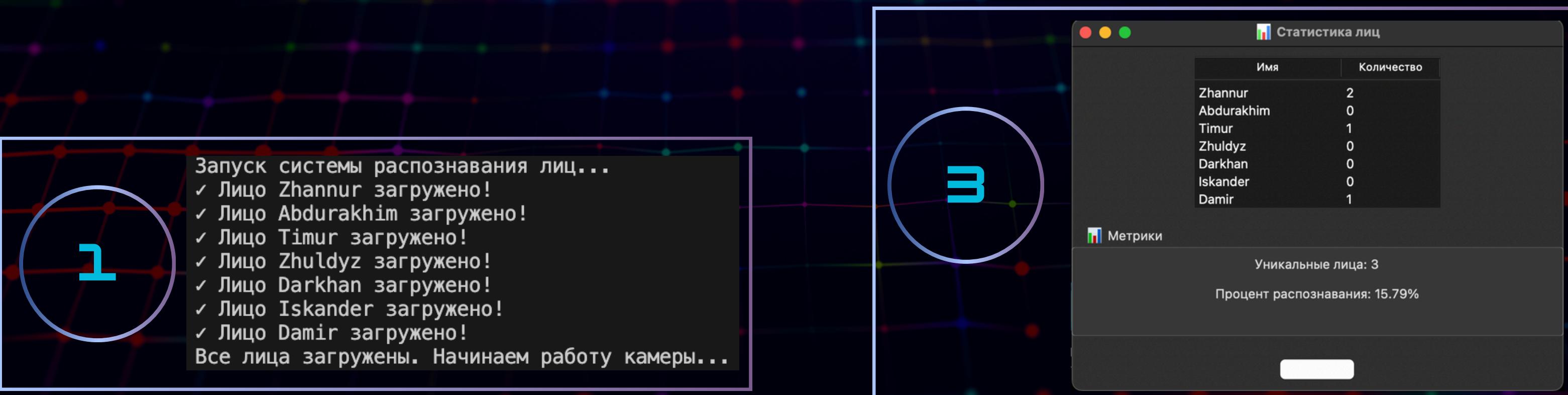
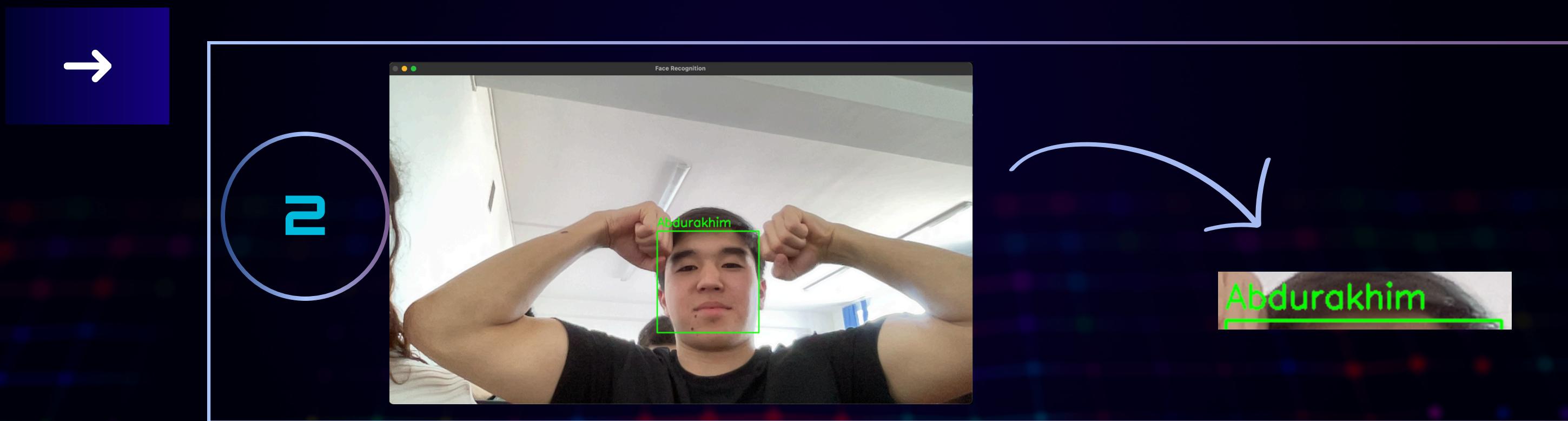
Определение лиц и сравнение с базой известных лиц.

Высчитывание уникальных лиц и процент успешного распознавания.

Вывод обработанного изображение с рамками и именами.

Обновление статистики

Завершение при нажатии "q" и запуск интерфейса.





# ПРИЗНАКИ ОБРАБОТКИ ИЗОБРАЖЕНИЯ

БИБЛИОТЕКИ **FACE\_RECOGNITION** и **DLIB** ИСПОЛЬЗУЮТСЯ ДЛЯ РАСПОЗНАВАНИЯ ЛИЦ.  
ОНИ ПОЗВОЛЯЮТ ОПРЕДЕЛЯТЬ КООРДИНАТЫ ЛИЦА, СОЗДАВАТЬ "ОТПЕЧАТКИ" И СРАВНИВАТЬ ИХ.

Компьютер воспринимает лицо как матрицу чисел, где:

- Каждый пиксель представляет интенсивность цвета.
- Для цветного изображения (RGB) у каждого пикселя есть три значения (Red, Green, Blue).

Выделение ключевых точек

Перед созданием вектора признаков алгоритм определяет ключевые точки лица:

- Глаза (внешние и внутренние уголки).
- Нос (переносица, кончик).
- Губы (верхняя и нижняя границы).
- Контур лица (линия подбородка).

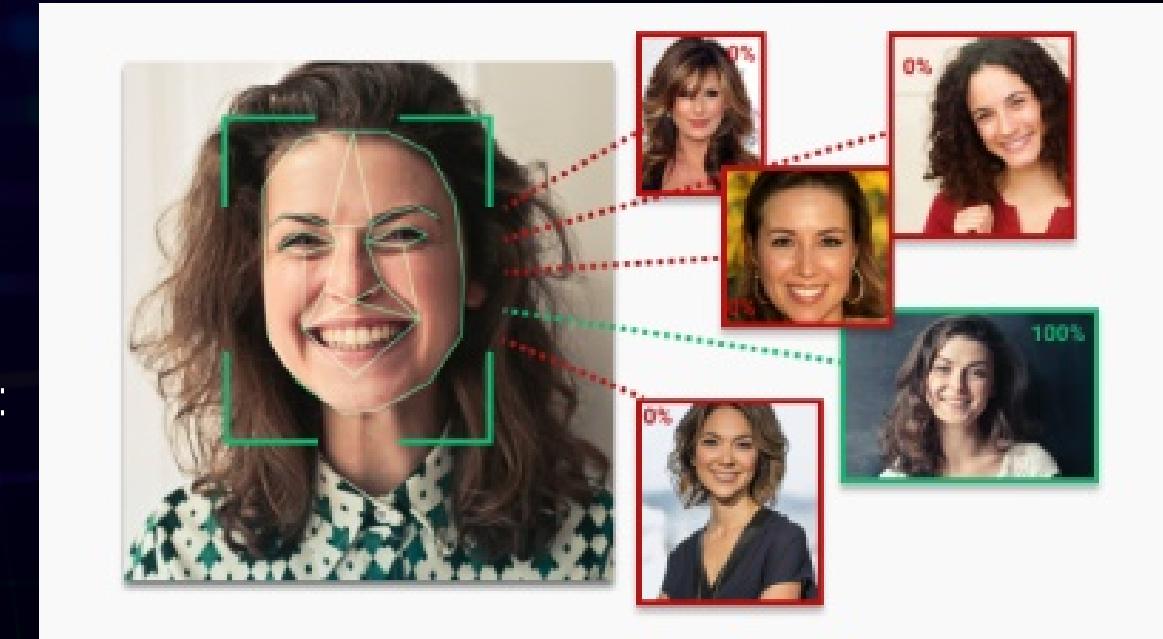
После нахождения ключевых точек создаётся 128-мерный вектор.

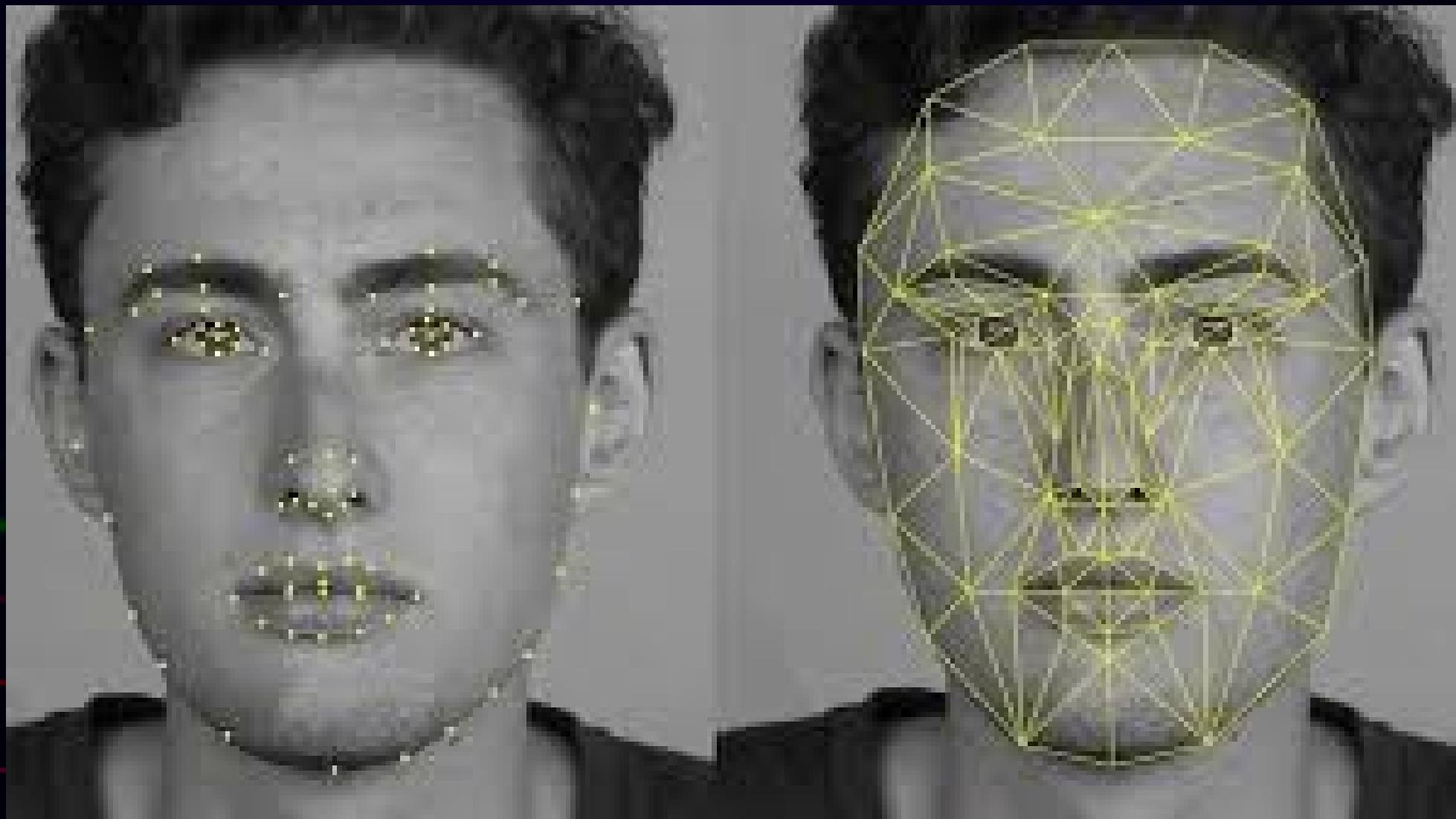
Он содержит информацию о соотношениях и особенностях лица.

Пример: [0.12, -0.34, 0.98, -0.56, ..., 0.87]

Сравнение лиц

- Программа вычисляет Евклидово расстояние между векторами.
- Если расстояние меньше порога (tolerance=0.5), лица считаются одинаковыми.





$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_{128} - y_{128})^2}$$

Где  $x$  — вектор известного лица,  $y$  — вектор нового лица.  
Если  $d < 0.5$ , то лицо считается распознанным.

Программа распознаёт лица в 4 этапа:

- 1 - Находит лицо на изображении (локализация).
- 2 - Извлекает 128-мерный вектор с ключевыми признаками.
- 3 - Сравнивает вектора по Евклидову расстоянию.
- 4 - Определяет, совпадает ли лицо с известными.



# ПРОБЛЕМЫ И ОГРАНИЧЕНИЯ



ИЗМЕНЕНИЯ ОСВЕЩЕНИЯ И РАКУРСА СОЗДАЮТ ТРУДНОСТИ. ЗАКРЫТИЕ ЛИЦА  
ОЧКАМИ ИЛИ МАСКОЙ ТАКЖЕ ВЛИЯЕТ НА РЕЗУЛЬТАТ. НИЗКОЕ КАЧЕСТВО  
ИЗОБРАЖЕНИЙ УХУДШАЕТ РАСПОЗНАВАНИЕ.

# ВЫВОД

В ходе выполнения проекта была успешно реализована система распознавания лиц в реальном времени с использованием языка программирования Python и библиотеки face\_recognition. Разработанное решение показало точность идентификации лиц на видеопотоке выше среднего и подтвердило эффективность применения современных инструментов компьютерного зрения в задачах биометрической безопасности. Работа продемонстрировала практическую значимость подобных систем в контексте автоматизации процессов контроля доступа, видеонаблюдения и других областей, требующих быстрой и надёжной идентификации личности.



СПАСИБО  
ЗА ВНИМАНИЕ