Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 09.04.04 Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____/А.А. Замышляева
«____»_____2019 г.

Система непрерывной аутентификации для повышения защищенности рабочих мест пользователей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.04.04.2019.103.ПЗ ВКР

Руководитель работы,
доцент кафедры ПМиП
_____/С.М. Елсаков
«____»_____2019 г.
Автор работы
студент группы ЕТ-225
_____/А.А. Кудинов
«____»_____2019 г.
Нормоконтролер, ассистент
_____/Н.С. Мидоночева
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Кудинов А.А. Система непрерывной аутентификации для повышения защищенности рабочих мест пользователей. – Челябинск: ЮУрГУ, ЕТ-225, 60 с., 25 ил., библиогр. список – 41 наим., 1 прил.

Цель данной работы заключается в разработке и тестировании системы непрерывной аутентификации, которая позволит повысить уровень защищенности рабочих мест пользователей.

В работе выполнен анализ существующих методов аутентификации, а также рассмотрены предложенные методы непрерывной аутентификации.

Описаны математическая модель и алгоритмы распознавания лица для непрерывной аутентификации, а также методы защиты от фальсификации образа пользователя.

Проведено тестирование параметров разработанной системы непрерывной аутентификации, включающее в себя оценивание точности системы однократного распознавания лица, оценивание защищенности от фальсификации образа пользователя и вычисление общей точности распознавания лица.

Система реализована на языке программирования Python версии 3.3.0 с использованием библиотек Dlib и Keras. В приложении приведен текст программы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Пока пользователь во время сессии отсутствует на своем рабочем месте, доступ к его компьютеру или мобильному устройству может быть перехвачен через устройства ввода-вывода. Особенно это актуально для систем в средах с высоким риском, например, компьютеров в банках или управления ответственными объектами [38]. Стандартная защита заключается в следующем: фиксируется некоторый интервал времени, и, если в течение этого интервала не обнаружены действия пользователя, компьютер блокируется, и запрашивается повторный ввод пароля. В противном случае компьютер не блокируется, чем и может воспользоваться злоумышленник, произведя произвольный ввод.

Для повышения уровня защиты предлагается использовать систему непрерывной аутентификации, способную проводить проверку пользователя на подлинность регулярно через заданный интервал времени.

Цель данной работы заключается в разработке и тестировании системы непрерывной аутентификации, которая позволит повысить уровень защищенности рабочих мест пользователей.

Для достижения цели необходимо решить следующие задачи:

– обзор существующих методов аутентификации;

– изучение средств разработки;

– проектирование и разработка системы непрерывной аутентификации;

– тестирование разработанной системы непрерывной аутентификации.

Поскольку заставлять пользователя постоянно вводить пароль или вставлять токен бесперспективно, то единственно возможным пригодным фактором для непрерывной аутентификации является биометрия. При этом был выбран метод аутентификации через двухмерное распознавание геометрии лица с использованием стандартной веб-камеры. Предложены и разработаны способы защиты от наиболее распространенной атаки на данный метод,

заключающейся в фальсификации образа подлинного пользователя с помощью его фотографии.

В первой главе проанализированы существующие методы однократной аутентификации, предложенные ранее методы непрерывной аутентификации, на основе чего принято решение о выборе метода двухмерного распознавания геометрии лица.

Во второй главе приведены математические характеристики распознавания лица, рассмотрены особенности выбранного метода аутентификации и разработана защита от фальсификации образа подлинного пользователя.

И наконец, третья глава посвящена тестированию разработанной системы при различных условиях, приведены критерии и результаты этого тестирования.

1 СУЩЕСТВУЮЩИЕ МЕТОДЫ АУТЕНТИФИКАЦИИ

1.1 Однократная аутентификация

1.1.1 Факторы аутентификации

Чтобы доказать свою подлинность, пользователь должен предъявить один или несколько факторов аутентификации, которые могут быть следующими:

– фактор знания предполагает наличие некоторой известной только пользователю информации;

– фактор обладания подразумевает, что пользователь владеет некоторым уникальным предметом, содержащим необходимую характеристику (пластиковые карты, USB-токены и т. д.);

– фактор свойства предполагает использование некоторой физической особенности пользователя, например, отпечаток пальца, радужная оболочка глаза и т. д.

Ниже будут рассмотрены различные способы реализации этих факторов.

1.1.2 Многоразовый пароль

Аутентификация по многоразовым паролям обычно предполагает наличие у пользователя уникального идентификатора (логин) и, собственно, пароля, который тот может подбирать и изменять самостоятельно. Пара «логин-пароль» хранится в базе данных. Общий алгоритм такой аутентификации приведен далее.

1. Пользователь делает запрос на доступ в систему, вводит идентификатор и пароль.

2. Введенные уникальные данные поступают на сервер и сравниваются с эталонными.

3. Если данные совпадают с эталоном, аутентификация считается успешной, если нет – пользователь перемещается к первому шагу [2].

Такой метод в настоящий момент остается самым распространенным методом аутентификации благодаря наибольшей простоте в исполнении. Кроме того, запоминаемая секретная фраза очень удобна для пользователей, находящихся в постоянном перемещении и, таким образом, подключающихся из различных удаленных мест [6]. В остальном такая аутентификация полна недостатков:

– практика показывает, что пользователи обычно выбирают достаточно простые пароли, которые могут быть легко подобраны методом «грубой силы» (например, «qwerty», «1234»), или используют идентификатор (имя) пользователя, или же пароль является словом из какого-либо языка [20];

– пароль легко забывается либо теряется. Кроме того, злоумышленник может быстро получить его путем мошенничества (например, фишинг) либо путем применения насилия к владельцу [20];

– пароль можно подсмотреть либо перехватить при вводе;

– вследствие огромной популярности и простоты парольной аутентификации для этого метода существует наибольшее количество различных методик и утилит, позволяющих получить его [17].

Многоразовые пароли нередко используется в качестве второго фактора аутентификации. Наиболее распространенным примером является PIN-код, рассмотренный в [19].

1.1.3 Одноразовый пароль

Аутентификация с использованием одноразовых паролей [13] (или сокращенно OTP – от англ. One Time Password) предполагает необходимость нового пароля для каждого входа в систему. Защищенность такого метода зависит от конкретной реализации, однако главное преимущество перед многоразовыми паролями неизменно – получив доступ к сессии аутентификации и перехватив одноразовый пароль, злоумышленник не сможет использовать его повторно, поэтому OTP обеспечивает более высокую степень защиты.

Ниже перечислены различные реализации одноразовых паролей:

– математические алгоритмы;

– временная синхронизация;

– система запросов;

– использование SMS.

Первый подход использует одностороннюю функцию. Система одноразовых паролей начинает работу от некого начального числа, после чего генерирует пароли необходимое количество раз.

Злоумышленник, получивший такой пароль, не сможет использовать его по истечению небольшого периода времени или одного соединения. А чтобы получить следующий пароль в цепочке из предыдущих, необходимо вычислить обратную функцию. Это невозможно сделать, поскольку используется односторонняя функция.

Второй подход, как правило, связан с аппаратными токенами, содержащими точные часы, синхронизированные с часами на сервере. В качестве исходной строки выступают текущие показания этих часов. Обычно используется не точное указание времени, а текущий интервал с установленными заранее границами. Эти данные зашифровываются с помощью секретного ключа и в открытом виде отправляются на сервер вместе с именем пользователя. Сервер при получении запроса на аутентификацию выполняет те же самые действия: получает текущее время от своего таймера и зашифровывает его. После этого ему остается только сравнить два значения: вычисленное и полученное от удаленного компьютера. Такой подход используется, например, в токенах-брелоках RSA SecurID [12].

Система запросов требует от пользователя отправлять синхронизированные по времени запросы, например, путем ввода значения в программный токен. При этом появление дубликатов невозможно, поскольку обычно используется дополнительный счетчик (таким образом, при отправке двух одинаковых запросов будут сгенерированы разные OTP).

Для доставки одноразовых паролей зачастую используются SMS-сообщения (см. рисунок 1.1). Поскольку технология SMS используется во всех

мобильных телефонах и, следовательно, имеет низкую себестоимость, то такой подход очень удобен для пользователей. Тем не менее, SMS имеют слабую степень защищенности, поскольку в цепь доверия включаются мобильные операторы, и существует опасность перехвата и перенаправления сообщений.

Рисунок 1.1 – Схема использования SMS-сообщений

Несмотря на то, что OTP позволяют обеспечить гораздо более высокую степень защищенности, чем многоразовые пароли, у них есть свои слабые стороны, перечисленные ниже.

1. Уязвимость для атак типа «человек посередине», при которой подменивается сервер аутентификации, и пользователь будет отправлять данные злоумышленнику.

2. Для синхронных методов существует риск рассинхронизации информации на сервере и в программном или аппаратном обеспечении пользователя, когда данные о показаниях внутренних таймеров перестают совпадать друг с другом.

3. Одноразовые пароли также уязвимы и для «фишинга», т. е. получения доступа к паролю путем мошенничества. Однако в силу короткого времени действия пароля вероятность успеха для OTP гораздо ниже, чем для многоразовых паролей.

1.1.4 Токены

Токены представляют собой компактные устройства, которые пользователь может носить с собой. Они используются в большинстве существующих реализаций OTP, что позволяет отнести этот метод к двухфакторной аутентификации [30].

Существующие токены можно разделить на три типа: токены без подключения, токены с подключением (см. рисунок 1.2) и беспроводные токены.

Рисунок 1.2 – Токены-брелоки RSA SecurID

Токены без подключения никак не соединяются с компьютером пользователя, вместо этого они имеют встроенный экран для отображения сгенерированного одноразового пароля, который пользователь уже вводит вручную. Именно такой тип токенов используется чаще всего в двухфакторной аутентификации.

Токены с подключением не требуют от пользователя собственноручного ввода информации, поскольку им необходим физический контакт с компьютером. Как правило, для реализации таких токенов используются технологии смарт-карт и USB.

Беспроводные токены являются своего рода гибридом двух предыдущих типов, поскольку одновременно не требуют физического подключения, но при этом образуют логическую связь с компьютером клиента.

В качестве токена также может использоваться мобильный телефон, что существенно снижает затраты и обеспечивает серьезное удобство для пользователей.

Основной уязвимостью всех рассмотренных токенов является возможность кражи или утери предмета. Следует отметить, что случае использования двухфакторной аутентификации злоумышленник, заполучивший токен, не сможет им воспользоваться (поскольку в таких случаях необходимо также вводить PIN-код), что повышает степень защищенности системы.

1.1.5 Биометрия

Фактор свойства (или биометрический фактор) предполагает систему распознавания людей по одной или более физическим или поведенческим чертам. Биометрические методы очень удобны для пользователей, поскольку такой идентификатор невозможно украсть либо передать третьему лицу (но есть незначительная вероятность его утери или повреждения).

Любая биометрическая система оценивается по двум параметрам [39]:

– *FAR* (англ. False Acceptance Rate – коэффициент ложного пропуска) – частота возникновения ситуаций, когда система разрешает допуск незарегистрированному в системе пользователю;

– *FRR* (англ. False Rejection Rate – коэффициент ложного отказа) – частота отказа в доступе настоящему пользователю системы.

Обе характеристики получают путем расчетов на основе математической статистики. Как правило, попытка снизить одну из характеристик ведет к повышению другой. Следовательно, необходимо находить баланс между ними для достижения лучшей защищенности.

Методы биометрической идентификации можно разделить на две группы: статические, то есть основанные на неизменных в течение жизни чертах человека, и динамические, основанные на изменяемых поведенческих характеристиках [16].

К статическим биометрическим атрибутам относятся:

– отпечатки пальцев;

– геометрия руки;

– геометрия лица;

– термический образ лица;

– сетчатка глаза;

– радужная оболочка глаза;

– рисунок вен.

Динамические атрибуты включают:

– голос;

– рукописный почерк;

– клавиатурный почерк.

Статичность или динамичность атрибутов является одновременно как преимуществом, так и недостатком. При краже данных в системах, использующих статические атрибуты, пользователь не сможет изменить их (в отличие, например, от многоразового пароля) и будет постоянно находиться под угрозой взлома. Динамические же атрибуты могут изменяться непроизвольно в течение жизни, а следовательно, нуждаются в обновлении.

Далее перечисленные биометрические методы будут рассмотрены отдельно.

1.1.5.1 Распознавание отпечатков пальцев

Распознавание отпечатков пальцев (или дактилоскопия) основано на неповторимости рисунка кожи на пальцах (а также ладонях) рук, называемого папиллярным узором.

Существуют различные типы сканеров для распознания отпечатков:

– оптические;

– тепловые;

– емкостные;

– радиочастотные;

– сканеры давления;

– ультразвуковые [9].

Оптические сканеры имеют различные реализации оптического метода распознавания.

1. FTIR-сканеры (см. рисунок 1.3) используют эффект нарушенного полного внутреннего отражения (англ. Frusted Total Internal Reflection, FTIR). Эффект заключается в том, что при падении света на границу раздела двух сред световая энергия делится на две части – одна отражается от границы, другая проникает через границу во вторую среду. Доля отраженной энергии зависит от угла падения светового потока. Начиная с некоторой величины данного угла, вся световая энергия отражается от границы раздела (что называется полным внутренним отражением). В случае контакта более плотной оптической среды (поверхности пальца) с менее плотной в точке полного внутреннего отражения пучок света проходит через эту границу. Таким образом, от границы отразятся лишь пучки света, попавшие в определенные точки полного внутреннего отражения, к которым не был приложен папиллярный узор пальца. Для захвата полученной световой картинки поверхности пальца используется специальный датчик изображения.

Рисунок 1.3 – FTIR-сканер для отпечатков пальцев

2. Оптоволоконные сканеры (см. рисунок 1.4) представляют собой матрицу, в которой все волноводы на выходе соединены с фотодатчиками. Чувствительность каждого датчика позволяет фиксировать остаточный свет, проходящий через палец, в точке соприкосновения пальца с поверхностью матрицы. Изображение всего отпечатка формируется по данным, считываемым с каждого фотодатчика. Данный метод обладает более высокой надежностью и устойчивостью обмана, чем предыдущий, но достаточно сложен в реализации.

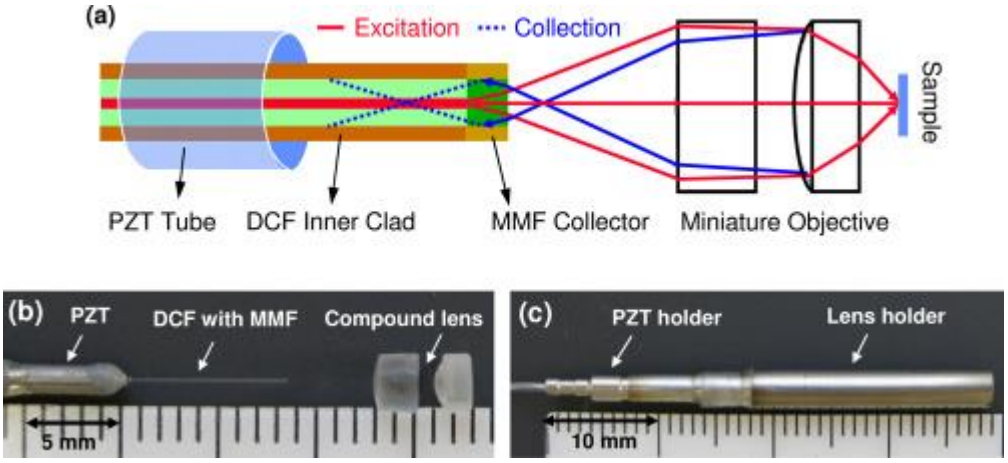

Рисунок 1.4 – Схема оптоволоконного сканера

3. Бесконтактные сканеры (см. рисунок 1.5) примечательны тем, что палец не контактирует с поверхностью сканера, а прикладывается к его отверстию, и несколько источников света подсвечивают его снизу с разных сторон. В центре сканера находится линза, через которую собранная информация проецируется на камеру, преобразующую полученные данные в изображение отпечатка пальца.

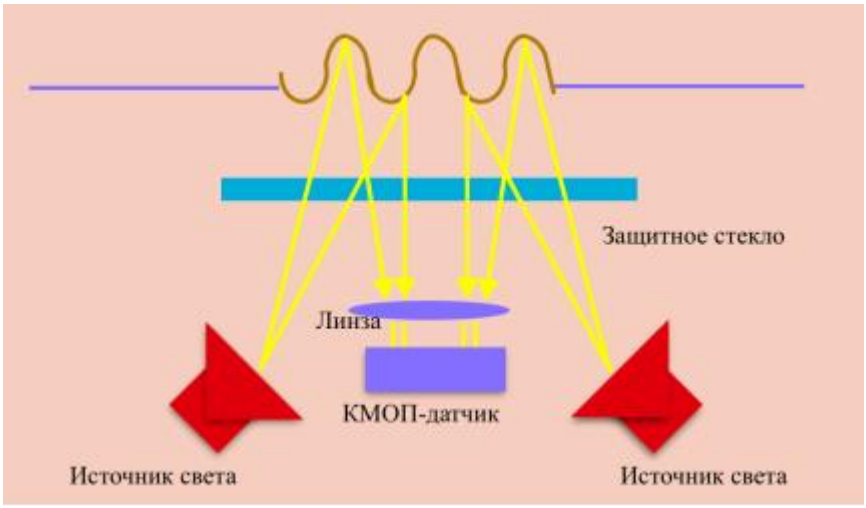

Рисунок 1.5 – Схема бесконтактного сканера

Оптические сканеры (кроме оптоволоконных) имеют неэффективную защиту от имитации пальца (муляжа). Кроме того, стандартные оптические сенсоры необходимо поддерживать в чистоте, поскольку загрязнение прозрачного окна сенсора ухудшит его работу.

В тепловых, или термосканерах (см. рисунок 1.6), используются датчики, которые состоят из пироэлектрических элементов, позволяющих фиксировать разницу температуры и преобразовывать ее в напряжение. При прикладывании пальца к сканеру по температуре прикасающихся к пироэлектрическим элементам выступов папиллярного узора и температуре воздуха, находящегося во впадинах, строится температурная карта поверхности пальца, которая в дальнейшем преобразуется в цифровое изображение. Тепловые сканеры не оставляют возможности использовать муляж, устойчивы к электростатическому разряду и хорошо работают в широком температурном диапазоне, однако их слабость в том, что изображение отпечатка быстро исчезает.

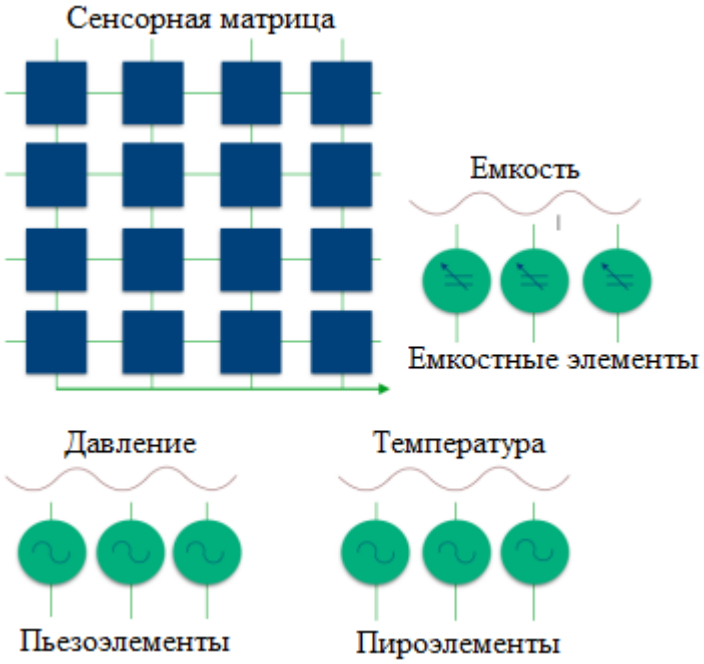

Рисунок 1.6 – Устройство термосканера

Емкостные сканеры (см. рисунок 1.7) используют эффект изменения емкости p-n-перехода полупроводника при соприкосновении гребня папилляр-

ного узора с элементом полупроводниковой матрицы. Такие сканеры достаточно экономичны, однако имеют низкую точность и, следовательно, надежность распознавания.

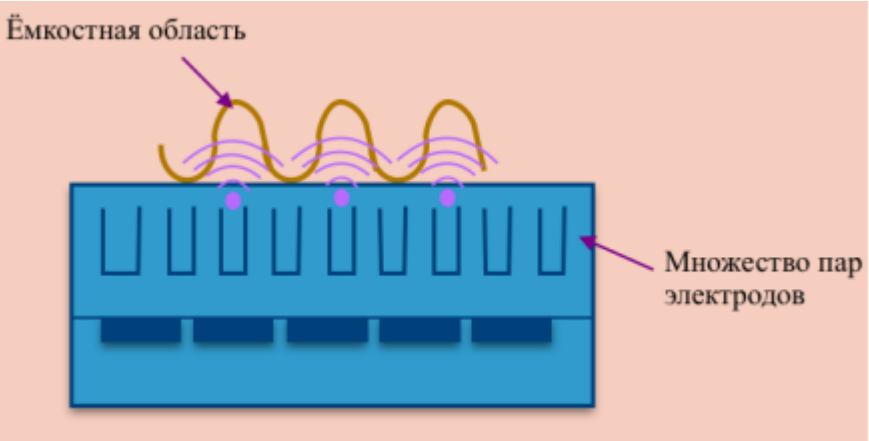

Рисунок 1.7 – Схема емкостного сканера

Радиочастотные сканеры (см. рисунок 1.8) содержат матрицу элементов, каждый из которых работает как миниатюрная антенна. Радиочастотный модуль генерирует сигнал низкой интенсивности и направляет его на сканируемую поверхность пальца. Каждый из чувствительных элементов матрицы принимает отраженный от папиллярного узора сигнал. Величина наведенной в каждой миниатюрной антенне ЭДС зависит от наличия или отсутствия вблизи нее гребня папиллярного узора. Полученная таким образом матрица напряжений преобразуется в цифровое изображение отпечатка пальца. Данный метод обеспечивает очень высокий уровень защиты, однако не очень удобен, поскольку неустойчив при плохом контакте пальца.

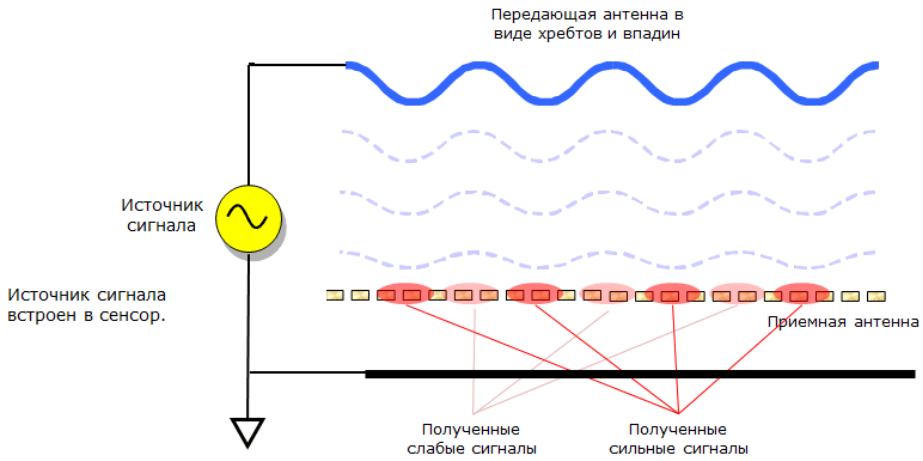

Рисунок 1.8 – Схема работы радиочастотного сканера

Чувствительные к давлению сканеры в своей конструкции используют матрицу пьезоэлектрических элементов, чувствительных к нажатию. При прикладывании пальца к сканирующей поверхности гребешковые выступы папиллярного узора оказывают давление на некоторое подмножество элементов матрицы. Впадины кожного узора никакого давления не оказывают. Таким образом, совокупность полученных с пьезоэлектрических элементов напряжений преобразуется в изображение отпечатка пальца. Такие сканеры имеют ряд слабых сторон, среди которых низкая чувствительность, опасность повреждения при излишних усилиях и низкая защита от муляжей.

Ультразвуковые сканеры (см. рисунок 1.9) сканируют поверхность пальца ультразвуковыми волнами. Расстояния между источником волн, гребешковыми выступами и впадинами папиллярного узора измеряются по отраженному от них эху. Данный тип сканеров обладает наибольшим потенциалом благодаря близкой к максимальной степени защиты, однако на данный момент имеет очень высокую стоимость.

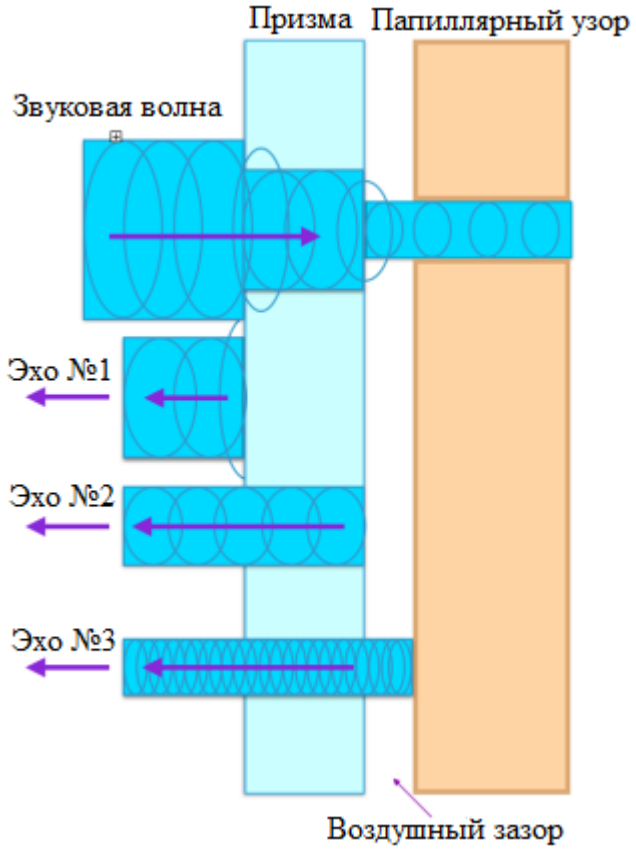

Рисунок 1.9 – Работа ультразвукового сканера

В целом, распознавание отпечатков пальцев на данный момент является наиболее распространенным биометрическим методом аутентификации. Дактилоскопия достаточно удобна в использовании, доступна многим пользователям (например, в смартфонах модели IPhone, Huawei и т. д.) повышает стандартный уровень безопасности. Однако среди остальных биометрических методов дактилоскопия проигрывает в надежности, поскольку подвержена фальсификации. Кроме того, папиллярный узор достаточно уязвим к повреждениям, например, мелкими царапинам и порезам.

1.1.5.2 Геометрия руки

Этот метод использует форму кисти руки, в частности, такие параметры, как изгибы пальцев, их длину и ширину, аналогичные параметры (а также высоту) тыльной стороны руки, расстояние между суставами и структура кости (см. рисунок 1.10). Также геометрия руки включает в себя мелкие детали (например, морщины на коже). С помощью сканера, который состоит из камеры и подсвечивающих диодов (при сканировании кисти руки, диоды включаются по очереди, это позволяет получить различные проекции руки), затем строится трехмерный образ кисти руки. Надежность аутентификации по геометрии руки сравнима с аутентификацией по отпечатку пальца.

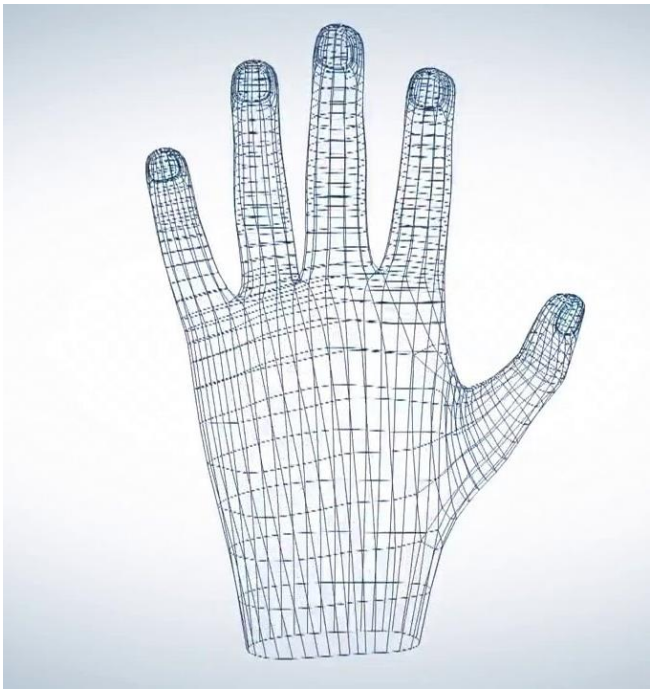

Рисунок 1.10 – Геометрическая схема руки

1.1.5.3 Геометрия лица

Данный метод имеет два направления – двухмерное и трехмерное распознавание лица [14].

Двухмерное (или 2D-) распознавание на данный момент считается наименее надежным биометрическим методом, поскольку очень чувствителен к условиям освещенности. При неравномерном освещении лица достоверность 2D-распознавания заметно снижается. Кроме того, серьезное влияние оказывают дополнительные элементы, такие как борода, усы, очки и т.п. [21]. Однако данный метод не требует дорогостоящего оборудования и позволяет распознавать пользователя даже на очень большом расстоянии от камеры. Это позволяет совмещать его с другими методами аутентификации, например, парольной.

3D-распознавание – уже гораздо более надежный метод, представляющий собой сложную математическую задачу. Она подразумевает построение трехмерной модели лица путем выделения контуров глаз, бровей, губ, носа, скул и других лицевых составляющих. После этого вычисляется расстояние между этими элементами, и на основе этого строится трехмерная модель лиц. Для определения уникального шаблона, соответствующего определенному человеку, требуется от 12 до 40 характерных элементов. Шаблон должен учитывать множество вариаций изображения на случаи поворота лица, наклона, изменения освещенности, изменения выражения. Диапазон таких вариантов варьируется в зависимости от целей применения данного способа (для идентификации, аутентификации, удаленного поиска на достаточно больших территориях и т. д.) [16].

В целом, 3D-распознавание лица обладает следующими преимуществами:

– отсутствие необходимости контактировать со сканирующим устройством;

– низкая чувствительность к внешним факторам, как на самом человеке (появление очков, бороды, изменение прически), так и в его окружении (освещенность, поворот головы);

– хороший уровень надежности (*FRR* и *FAR*), сравнимый с таковым у дактилоскопии.

Основным недостатком 3D-распознавания, особенно по сравнению с двухмерным, является высокая стоимость оборудования и высокий уровень технической сложности.

1.1.5.4 Термограмма лица

Метод основан на создании температурной карты (или термограммы) путем сканирования лица в инфракрасном свете. Исследованиями доказано, что термограмма лица является уникальной биометрической характеристикой, на точность которой не влияют пластические операции, старение организма, изменение температуры тела. Кроме того, данный метод, в отличие от геометрического, способен различать близнецов.

Тем не менее, на данный момент метод не имеет широкого распространения, поскольку не дает нужного качества аутентификации.

1.1.5.5 Сетчатка глаза

Сканер, использующий этот метод (см. рисунок 1.11), считывает рисунок капилляров на поверхности сетчатки глаза. Сетчатка имеет неподвижную структуру, неизменную во времени. Сканирование сетчатки происходит с использованием инфракрасного света низкой интенсивности, направленного через зрачок к кровеносным сосудам на задней стенке глаза. Из полученного сигнала выделяется несколько сотен особых точек, информация о которых сохраняется в шаблоне.

К преимуществам этого метода аутентификации можно отнести самую высокую надежность (*FAR* и *FRR* стремятся к нулю) среди существующих биометрических систем и бесконтактный способ снятия данных. Однако такие системы имеют высокую стоимость, малую доступность и отсутствие широкого

рынка распространения, а значит и низкую интенсивность развития. К тому же, подобные системы требуют четкого изображения и, как правило, чувствительны к неправильной ориентации сетчатки. Поэтому требуется смотреть очень аккуратно, а наличие некоторых заболеваний (например, катаракты) может препятствовать использованию данного метода.

Рисунок 1.11 – Сканер сетчатки глаза

1.1.5.6 Радужная оболочка глаза

Здесь используется уникальность признаков и особенностей радужной оболочки человеческого глаза. Радужная оболочка образовывается еще до рождения человека, и не меняется на протяжении всей жизни. По текстуре она напоминает сеть с большим количеством окружающих кругов и рисунков, которые могут быть измерены компьютером, и поскольку рисунок радужки очень сложен, это позволяет отобрать порядка 200 точек. Кроме того, радужная оболочка, в отличие, например, от отпечатков пальцев, защищена от повреждений роговицей.

Таким образом, системы, идентифицирующие радужную оболочку (см. рисунок 1.12), обладают очень высокой надежностью (уступающей только идентификации по сетчатке глаза), а также, как и распознаватели лица, могут

работать на приемлемом расстоянии (до нескольких метров). Кроме того, несмотря на то, что фальсификация для данного метода возможна, есть множество способов противодействия.

Из недостатков метода можно отметить относительно высокую стоимость оборудования, по сравнению с дактилоскопическим методом и распознаванием лица.

Рисунок 1.12 – Сканер радужной оболочки глаза

1.1.5.7 Рисунок вен

Рисунок вен представляет собой узор сети видимых кровеносных сосудов руки. Камера получает изображение сети сосудов с помощью инфракрасного света, отраженного гемоглобином в крови. Специальная программа на основе полученных данных создает цифровую свертку. Распознавание не требует контакта человека со сканирующим устройством (см. рисунок 1.13).

Согласно статистике, данный метод имеет очень высокую надежность, сравнимую с таковой у идентификации по радужной оболочки, однако рисунок вен подвержен изменению вследствие заболеваний, например, артрита, что ухудшает показатели *FAR* и *FRR*. Кроме того, препятствуют новизна и малая степень изученности метода, а следовательно, его низкая распространенность. Тем не менее, данный метод остается одним из наиболее перспективных для однократной аутентификации.

Рисунок 1.13 – Сканер рисунка вен

1.1.5.8 Голос

Этот метод использует различные комбинации частотных и статистических характеристик голоса. В зависимости от реализации, могут рассматриваться такие параметры, как модуляция, интонация, высота тона и т. п.

Одним из основных преимуществ распознавания по голосу является общедоступность, удобство и простота в применении – данный метод не требует иной аппаратуры, кроме общедоступных микрофона и звуковой платы. Однако человеческий голос достаточно сложен для распознания ввиду своей изменчивости – вследствие заболевания, возраста, перемены настроения. Кроме того, проблему представляет и необходимость учета шумов [3].

1.1.5.9 Рукописный почерк

Метод биометрической аутентификации по рукописному почерку основывается на специфическом движении человеческой руки во время подписания документов. Для сохранения подписи используют специальные ручки или восприимчивые к давлению поверхности. Этот вид аутентификации человека использует его подпись. Шаблон создается в зависимости от необходимого уровня защиты. Обычно выделяют два приведенных ниже способа обработки данных о подписи [15].

1. Анализ самой подписи, т.е. используется просто степень совпадения двух картинок. Недостатки такого метода очевидны – подверженность фальсификации и невозможность дважды сделать одну и ту же подпись делают данный метод бесперспективным.

2. Анализ динамических характеристик написания, т. е. для аутентификации строится свертка, в которую входит информация по подписи, временными и статистическими характеристиками ее написания. Данный способ очень удобен, поскольку не требователен к оборудованию и привычен для человека.

В целом данный метод не является надежным, поскольку четкий рукописный почерк формируется не у всех пользователей и является неустойчивой характеристикой.

1.1.5.10 Клавиатурный почерк

Клавиатурный почерк [8] – характеристика, которая описывается следующими параметрами:

– скорость ввода, т. е. количество введенных символов, разделенное на время печати;

– динамика ввода, характеризуется временем между нажатиями клавиш и временем их удержания;

– частота возникновения ошибок при вводе;

– использование клавиш, например, какие функциональные клавиши нажимаются для ввода заглавных букв и т. д.

Аутентификация по клавиатурному почерку может проводиться двумя способами:

– по набору ключевой фразы (что позволяет совместить данную аутентификацию с парольной);

– по набору свободного текста.

Временные интервалы между нажатием клавиш на клавиатуре и время удержания клавиш позволяют достаточно однозначно охарактеризовать почерк человека, что подтверждается рядом экспериментов [18]. Но данный метод подходит далеко не для всех пользователей, поскольку достаточный для распознания клавиатурный почерк сформировывается только для лиц с длительным опытом работы на клавиатуре. Для остальных пользователей вероятность ошибки довольно высока, что делает этот метод неприемлемым для массового распространения.

1.2 Непрерывная аутентификация

Непрерывная аутентификация предполагает постоянную проверку пользователя на идентичность во время сессии [24]. Главная ее особенность и преимущество перед другими системами заключается в том, что злоумышленник не сможет воспользоваться компьютером после того, как там авторизовался пользователь [10].

Следует отметить, что при реализации непрерывной аутентификации использование фактора знания или обладания бесперспективно, поскольку периодическая необходимость вводить пароль или предъявлять токен создает неудобства для пользователя и не может полноценно исключить вмешательство злоумышленника [25] (при этом такие методы могут быть использованы в качестве дополнительного фактора аутентификации в целях повышения безопасности [38, 40]).

Поэтому далее будут рассмотрены существующие реализации биометрической непрерывной аутентификации.

1.2.1 Клавиатурный почерк

В работах [11, 23] рассматривается непрерывная аутентификация путем анализа клавиатурного почерка. Вместо анализа динамики набора некой парольной фразы, используемой в статичной вариации этого метода, авторы

предлагают использовать в качестве базового материала для пространства признаков большой набор часто используемых комбинаций символов.

Таким образом, предлагаемый метод объединяет достоинства непрерывной аутентификации и достоинства идентификации клавиатурного почерка, рассмотренные в пункте 1.1.5.10. Однако основной недостаток остается тем же, что и ранее: клавиатурный почерк достаточно четко выражен только у ограниченного круга лиц, а следовательно, данный метод не является пригодным к массовому распространению.

1.2.2 Жесты на сенсорном экране

В работах [22, 35] предлагается распознавание движений пальцев пользователя на сенсорном экране мобильного устройства. Для сканирования используется специально разработанная цифровая сенсорная перчатка.

Такой подход очень удобен для пользователей мобильных устройств, однако достигнутый уровень *FAR* составляет 4,66% при *FRR* 0,13%, что дает достаточно низкий уровень безопасности. Кроме того, для потенциальных пользователей возникает необходимость приобретать саму сенсорную перчатку.

1.2.3 Движения компьютерной мыши

В [34] используется анализ движений компьютерной мыши, осуществляемых пользователем.

Для данного метода можно выделить следующие преимущества:

– возможно повсеместное использование благодаря отсутствию требований к оборудованию, необходима только компьютерная мышь;

– высокий потенциал надежности при дальнейшем развитии метода.

Однако имеются свои слабые стороны [32]:

– на данный момент достигнут недостаточный по сравнению со стандартными биометрическими методами уровень надежности;

– движения компьютерной мыши не очень сложно воспроизвести, таким образом, существует высокий уровень опасности фальсификации.

1.2.4 Геометрия лица

Авторы [36] создали систему непрерывного трехмерного распознавания лица через RGB-D камеру.

В данной реализации можно выделить следующие преимущества:

– достигнуто приемлемое соотношение *FAR* к *FRR*, что обеспечивает высокий уровень надежности;

– использование трехмерного распознания лица исключает возможность использования злоумышленником фотографии либо фальсификации.

Но следует отметить, что данная реализация не исключает недостатков статического геометрического распознавания лица, таких, как:

– ошибки при распознавании схожих лиц;

– проблемы при изменении лицевых признаков пользователя в результате осуществления пластических операций либо получения лицевых травм;

– необходимость использования RGB-D камеры влечет дополнительные затраты для пользователей.

Кроме того, схожий метод рассматривается в [33], однако в этом случае используется частичное распознавание лица через фронтальную камеру смартфона, в целях обнаружить нужные лицевые признаки при сокрытии или уходе за границы видимости пользователем.

1.2.5 Тоны сердца

В [1, 27] предлагается непрерывная аутентификация на основе акустических свойств сердца, тоны которого содержат уникальную информацию.

Для такой реализации можно выделить следующие особенности:

– тоны сердца не утрачиваются в течение жизнедеятельности;

– эти тоны очень сложно воспроизвести искусственно, что сводит риск фальсификации к минимуму;

– тоны сердца могут изменяться в течение жизни.

При данных особенностях можно сделать вывод, что аутентификация путем анализа акустики сердца не является универсальной. Дело в том, что помимо изменения сердечных тонов, данная методика имеет достаточно высокую сложность реализации, а главное – малодоступна для обычных пользователей ввиду необходимости приобретать дорогостоящее дополнительное оборудование.

1.2.6 Геометрия сердца

В [41] разработана идентификация путем определения формы, размера и динамики работы сердца. Ее можно считать своеобразным усовершенствованием пункта 1.2.4.

С точки зрения биометрической аутентификации проверка по сердцу имеет очень большой потенциал. Это обусловлено следующими особенностями:

– в отличие от пальца, сердце гораздо сложнее отделить от организма субъекта;

– по утверждению авторов, вред от облучения при сканировании не превышает 1% от смартфонов, что означает безопасность такого метода для организма;

– геометрические характеристики сердца практически невозможно подделать, сложно спрятать от системы;

– характеристики носителя очень трудно узнать потенциальному злоумышленнику;

– работающее сердце есть у каждого живого человека, в отличие, например, от пальцев, которые могут быть утрачены в результате несчастных случаев или насилия;

– форма сердца не изменяется в течение жизни, кроме как при серьезных сердечных заболеваниях;

– на данный момент не найдено двух людей с одинаковыми характеристиками сердца.

Тем не менее, сейчас данный метод представляет сложность для повсеместного использования, поскольку системы для распознания геометрии сердца будут обладать еще более высокой стоимостью, чем системы из пункта 1.2.4, хотя и остается одним из наиболее перспективных для непрерывной аутентификации.

1.3 Выводы по разделу

В ходе обзора были проанализированы существующие методы аутентификации в целом и различные реализации непрерывной аутентификации в частности. Были выделены их достоинства и недостатки.

Наибольшим уровнем надежности среди всех существующих методов аутентификации обладают биометрические системы, а именно:

– распознавание по сетчатке глаза;

– распознавание по радужной оболочке глаза;

– рисунок вен.

Однако все эти методики требуют приобретения дорогостоящего оборудования и не могут быть использованы для реализации непрерывной аутентификации.

Среди различных реализаций непрерывной аутентификации наибольшим потенциалом надежности обладает рассмотренная в [41] система распознавания геометрии сердца, но ее недостатком также является потенциально высокая стоимость для эксплуатации обычным пользователем, что будет препятствовать ее массовому распространению.

Поэтому в качестве основного фактора выбрано 2D-распознавание лица через веб-камеру. Можно выделить следующие преимущества данной реализации:

– для использования биометрической аутентификации необходима веб-камера, которая присутствует в большинстве смартфонов и ноутбуков либо может быть приобретена по достаточно низкой цене;

– лицо пользователя будет непрерывно распознаваться веб-камерой, и таким образом, злоумышленник не сможет воспользоваться устройством пользователя после авторизации последнего.

Ни один из факторов не имеет приоритета при прохождении аутентификации, а следовательно, будет актуально большинство недостатков парольных систем и систем 2D-распознавания лица, таких как:

– вероятность фишинга или перехвата пароля при вводе;

– вероятность потери или кражи пароля;

– возможность фальсификации лица пользователя;

– проблемы распознавания при изменении лицевых признаков пользователя.

Часть вышеперечисленных проблем необходимо разрешить при реализации выбранных методов.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ И АЛГОРИТМЫ РАСПОЗНАВАНИЯ ЛИЦА ДЛЯ НЕПРЕРЫВНОЙ АУТЕНТИФИКАЦИИ

2.1 Двухмерное распознавание лица

2.1.1 Процесс распознавания лиц

Обнаружение и распознавание лиц на данный момент является достаточно типовой задачей со множеством решений. Как правило, в этих решениях используются технологии машинного обучения, реализованные, например, при помощи нейронных сетей.

Большинство современных систем распознавания лиц, очень чувствительны к характеристикам предъявляемых изображений [31]. Поэтому перед непосредственным распознаванием, необходимо выполнять нормализацию исходных изображений. Процесс нормализации направлен на приведение изображений к единому стандартному виду, принятому в данной системе распознавания.

Распознавание лиц сводится к процессу, состоящему из двух этапов:

– локализация лица на изображении (т.е. обнаружение);

– выделение ключевых структур на поверхности лица.

2.1.2 Библиотека Dlib

Для реализации распознавания был использован алгоритм построения точек лица, описанный в [28] и реализованный в библиотеке Dlib. Для обозначения лица он локализует и маркирует следующие его области: правый глаз, левый глаз, правая бровь, левая бровь, нос, челюсть.

В работе используется предварительно обученная сверточная нейронная сеть ResNet. От сети отрезаются слои, отвечающие за классификацию, и остаются только сверточные слои, которые извлекают ключевые признаки из изображения, т. е. структуры лица. В Dlib использован модифицированный вариант этой сети – ResNet34.

Изображение помечается указанием конкретных (x, y)-координат, окружающих каждую вышеуказанную структуру лица. Всего таких точек – 68 (см. рисунок 2.1).

Результат работы сети – набор чисел, называемый дескриптором. Такие дескрипторы будут извлечены из оригинального изображения пользователя и из изображения, полученного с web-камеры.

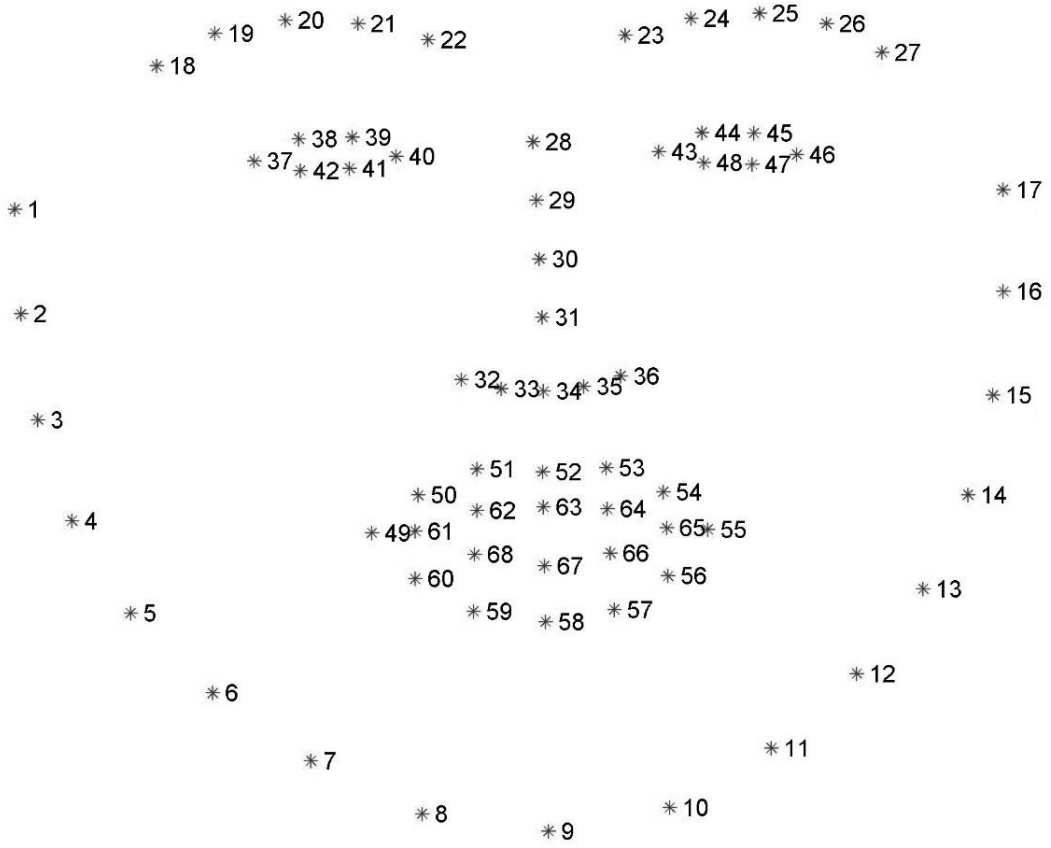

Рисунок 2.1 – Итоговая общая структура лица из 68 точек

Дескрипторы используются для распознавания лиц: для этого находится Евклидово расстояние между ними. Если это расстояние меньше некоторой константы $threshold_1$, то считается, что лица принадлежат одному пользователю, следовательно, лицо человека опознано. В противном случае пользователю будет запрещен доступ в систему, и ему будет необходимо снова авторизоваться для продолжения работы с ней.

2.2 Классификация фотографии

Стандартной атакой на систему распознавания лица является фальсификация образа подлинного пользователя и его демонстрация на камеру. В случае двухмерного метода это с наибольшей вероятностью будет фотография пользователя. Для защиты от этой атаки рассмотрены два разных метода.

Предположим, что злоумышленник зафиксировал фотографию перед веб-камерой для большей надежности. В этом случае, разница между дескрипторами лица на разных кадрах будет минимальной. Первый метод основан на этом предположении: вычисляется евклидово расстояние между предыдущими и текущим кадрами и сравнивается с константой $threshold_2$. Если это расстояние меньше $threshold_2$, значит, на изображение фотография пользователя, а не он сам.

Второй метод основан на различиях в движении живого человека и фотографии в кадре. Для реализации этого использована сверточная нейронная сеть, созданная с помощью библиотеки Keras.

2.3 Сверточная нейронная сеть

2.3.1 Принцип работы сверточных нейронных сетей

Сверточная нейронная сеть (от англ. convolutional neural network) – это тип архитектуры искусственных нейронных сетей, который был предложен французским ученым Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения (англ. deep learning). Использует некоторые особенности зрительной коры [29], в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определенного набора простых клеток. Таким образом, идея сверточных нейронных сетей заключается в чередовании сверточных слоев (англ. convolution layers) и субдискретизирующих слоев (англ. subsampling layers или англ. pooling layers, слоев подвыборки). Структура сети – однонаправленная (без обратных связей), принципиально многослойная. Для

обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) – любая, по выбору исследователя.

Эта архитектура получила свое название благодаря наличию операции свертки, в которой каждый из фрагментов изображения будет умножаться на матрицу свертки (так называемое ядро) поэлементно, результат при этом суммируется и будет записан в ту же позицию выходного изображения.

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причем каждая связь имеет свой персональный весовой коэффициент. В сверточной нейронной сети в операции свертки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале – непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свертки. Ее интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определенным углом. Тогда следующий слой, получившийся в результате операции свертки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и ее координаты, формируя так называемую карту признаков (англ. feature map). Естественно, в сверточной нейронной сети набор весов не один, а целая гамма, кодирующая элементы изображения (например линии и дуги под разными углами).

При этом такие ядра свертки не закладываются исследователем заранее, а формируются самостоятельно путем обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов ее пере-

двигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 ее сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации (англ. subsampling, англ. pooling, также переводимая как «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотненной карты признаков меньшей размерности. За счет данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

Рассмотрим типовую структуру сверточной нейронной сети (см. рисунок 2.2) более подробно. Сеть состоит из большого количества слоев. После начального слоя (входного изображения) сигнал проходит серию сверточных слоев, в которых чередуется собственно свертка и субдискретизация (пулинг). Чередование слоев позволяет составлять «карты признаков» из карт признаков, на каждом следующем слое карта уменьшается в размере, но увеличивается количество каналов. На практике это означает способность распознавания сложных иерархий признаков. Обычно после прохождения нескольких слоев карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становятся сотни. На выходе сверточных слоев сети дополнительно устанавливают несколько слоев полносвязной нейронной сети (перцептрон), на вход которому подаются оконечные карты признаков.

Слой свертки – это основной блок сверточной нейронной сети. Слой свертки включает в себя для каждого канала свой фильтр, ядро свертки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты

матричного произведения для каждого фрагмента). Весовые коэффициенты ядра свертки (небольшой матрицы) неизвестны и устанавливаются в процессе обучения.

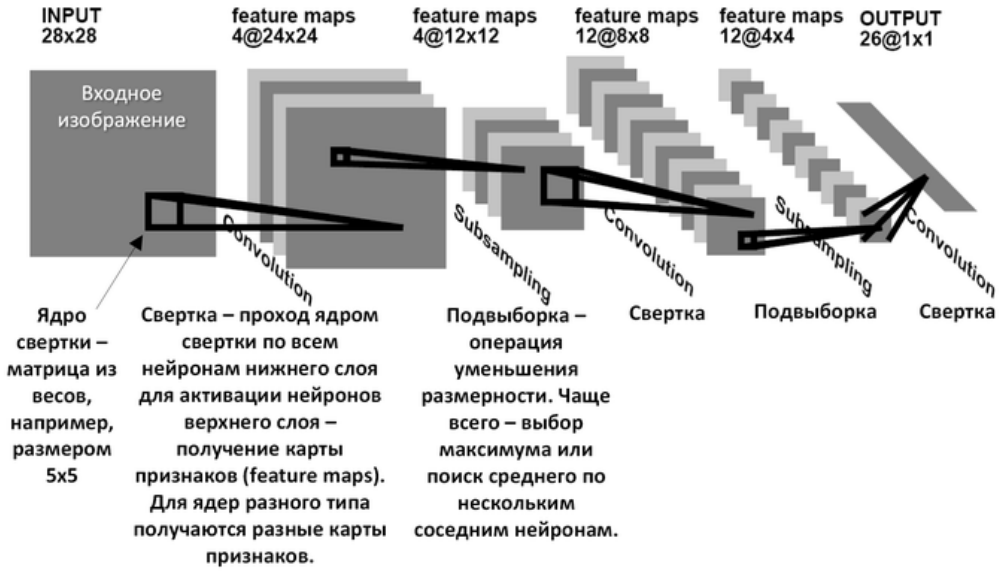

Рисунок 2.2 – Архитектура сверточной нейросети

Особенностью сверточного слоя является сравнительно небольшое количество параметров, устанавливаемое при обучении. Так например, если исходное изображение имеет размерность 100×100 пикселей по трем каналам (это значит 30000 входных нейронов), а сверточный слой использует фильтры с ядром 3x3 пикселя с выходом на 6 каналов, тогда в процессе обучения определяется только 9 весов ядра, однако по всем сочетаниям каналов, то есть 9×3×6 =162, в таком случае данный слой требует нахождения только 162 параметров, что существенно меньше количества искомых параметров полносвязной нейронной сети.

Слой пулинга (иначе подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пик-

сель, имеющий максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объем изображения. Пулинг интерпретируется так: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться. Слой пулинга, как правило, вставляется после слоя свертки перед слоем следующей свертки. Кроме пулинга с функцией максимума можно использовать и другие функции – например, среднего значения. Однако практика показала преимущества именно пулинга с функцией максимума, который включается в различные типовые системы.

После нескольких прохождений свертки изображения и уплотнения с помощью пулинга система перестраивается от конкретной сетки пикселей с высоким разрешением к более абстрактным картам признаков, как правило на каждом следующем слое увеличивается число каналов и уменьшается размерность изображения в каждом канале.

2.3.2 Характеристики созданной нейронной сети

Созданная сверточная нейронная сеть включает три слоя подвыборки и три слоя каскада свертки (см. рисунок 2.3). Сверточная часть сети предназначена для выделения характерных признаков на изображении. За сверточной частью следует полносвязная часть нейронной сети, которая отвечает за классификацию. Для этого были использованы два полносвязных слоя. Нейронная сеть использована для бинарной классификации изображений. Первым классом являются изображения лиц живых людей, вторым – изображения фотографий людей перед веб-камерой.

Слой исключения используется для предотвращения распространенной ошибки нейронных сетей, связанной с переобучением. Он случайным образом выбрасывает половину нейронов, позволяя оставшимся дообученным нейронам избегать ошибки.

Сеть принимает на вход изображение, на котором ранее было найдено лицо, а на выход сеть выдает один из двух классов: либо фотография, либо живой человек. Она была обучена на выборке от 500 до 3000 изображений каждого класса.

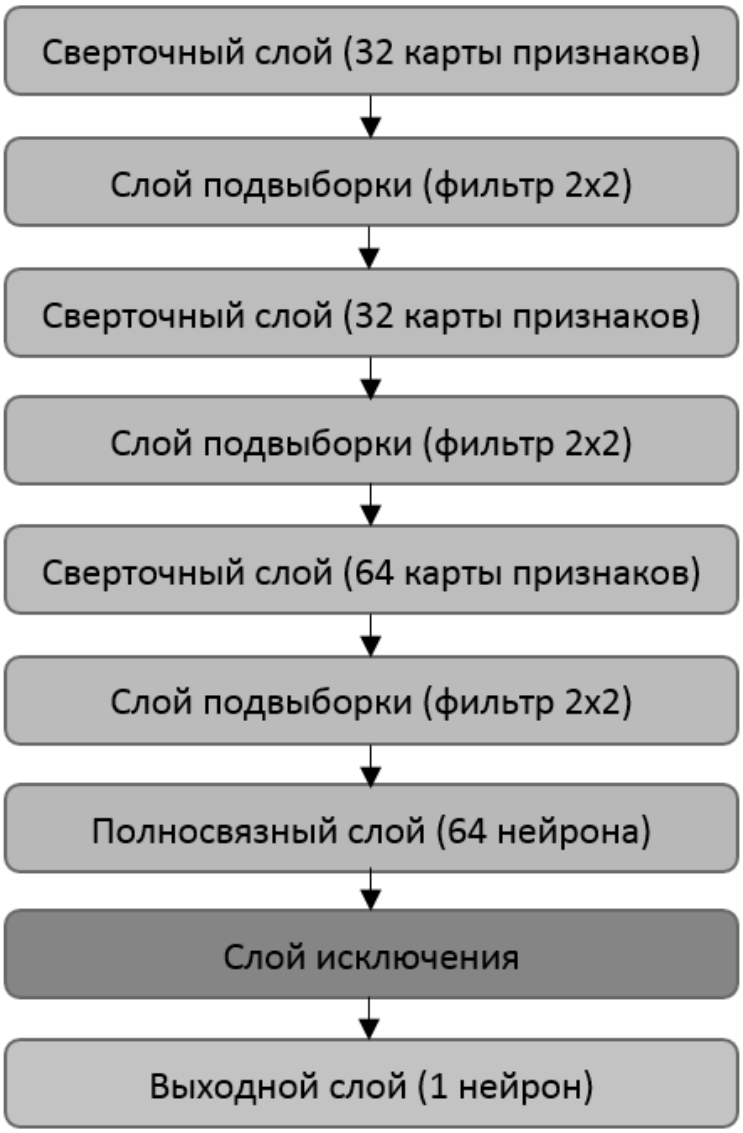

Рисунок 2.3 – Слои созданной нейронной сети

2.4 Выводы по разделу

В этом разделе рассмотрены математическая модель распознавания лица и метод классификации фотографии. Распознавание лица состоит из локализации лица на изображении и выделении на найденном лице ключевых структур, по которым можно сравнить два лица между собой.

Для этого использован алгоритм построения 68 точек лица, реализованный в библиотеке Dlib. Лицо помечается особыми координатами, окружающими каждую лицевую структуру.

Уязвимость в виде демонстрации фотографии пользователя на камеру преодолевается двумя разными методами: первый основан на возможной неподвижности фотографии в кадре, второй – на различиях в движении между живым человеком и фотографией. Для реализации второго метода использована сверточная нейронная сеть.

3 ТЕСТИРОВАНИЕ ПАРАМЕТРОВ ПРЕДЛОЖЕННОЙ СИСТЕМЫ НЕПРЕРЫВНОЙ АУТЕНТИФИКАЦИИ

3.1 Критерии тестирования системы

Чтобы оценить эффективность непрерывной аутентификации, использованы следующие критерии:

– False Accept Rate (FAR) означает вероятность того, что система примет чужого пользователя за своего, измеряется в процентах;

– False Reject Rate (FRR) означает вероятность того, что система примет своего пользователя за чужого, измеряется в процентах;

– Equal Error Rate (ERR) определяет точность системы, измеряется в процентах;

– нагрузка на оперативную память устройства;

– нагрузка на ЦПУ;

– продолжительность одной процедуры распознавания.

Для нахождения ERR необходимо составить графики FAR и FRR. ERR будет равен такому значению $FAR(threshold)$, при котором тот равен $FRR(threshold)$, т. е. ERR находится на точке пересечения этих графиков:

$$EER = FAR(threshold) = FRR(threshold) \quad (1)$$

3.2 Набор данных

В качестве набора данных использован видеоматериал от 11 разных видеоблогеров разной эмоциональной выразительности, разной освещенности и длиной от 10 до 220 минут.

Все видео были взяты из YouTube. Видеоблогеры разделяются на две группы. В первой группе [5] оказались блогеры с малой эмоциональной выразительностью, а также использующие затемненный фон. Во второй группе [4] находятся блогеры с большой эмоциональной выразительностью, использующие светлый или нейтральный фон.

3.3 Оценивание точности однократного распознавания лица

Перед тестированием каждый блогер был зарегистрирован в системе в качестве пользователя, под своим логином, и получил свой шаблон – набор из четырех взятых вручную изображений, из которых 3 изображения с лицом анфас и одно – полу-анфас к камере, для тестирования качества афинных преобразований точек лица. Для всех 4 изображений вычисляются дескрипторы d_1, d_2, d_3, d_4.

График *FAR* строится по следующему алгоритму, приведенному ниже.

1. *threshold* принимает значения 0,01, 0,02, ..., 1,0.

2. Вычисляются дескрипторы d_1, d_2, d_3, d_4 для шаблона каждого пользователя.

3. Система последовательно просматривает видео всех остальных блогеров и обрабатывает по 100 кадров от начала видео каждого блогера. На каждом кадре определяется лицо, и по нему вычисляется дескриптор $d_{current}$.

4. Находится Евклидово расстояние между каждым из $d_{1..4}$ и $d_{current}$. Результат – четыре значения e_1, e_2, e_3, e_4. Из них вычисляется среднее значения e по формуле:

$$e = (e_1 + e_2 + e_3 + e_4)/4. \quad (2)$$

5. Если $e <$ *threshold*, то значение считается ошибочным.

6. Отношение количества ошибочных значений к общему количеству значений и есть *FAR* для каждого из 100 шагов *threshold*.

7. По полученным 100 значениям *FAR* строится график *FAR*(*threshold*).

Для нахождения *FRR* шаблон пользователя сравнивается с видеоматериалом этого же пользователя, а ошибочными считаются значения, превышающие $threshold_1$. В остальном алгоритм нахождения *FRR* повторяет алгоритм для *FAR*.

Рассмотрим изменение *ERR*, меняя следующие параметры, способные оказать влияние на качество распознавания:

– шаблон пользователя;

– разрешение видео;

– освещенность (яркость) шаблона.

Стандартный тест был проведен на видео с разрешением 360p. Данный тест был выбран благодаря тому, что именно при данном разрешении достигается наибольшая точность среди всех возможных разрешений, как будет показано позднее. Результаты тестирования показаны на рисунке 3.1. *FAR* и *FRR* пересекаются при $threshold_1 = 0{,}63$, а *ERR* составляет 0,87%. Для сравнения, в методе трехмерного распознавания ошибка достигает 0,8%, в методе частичного двухмерного распознавания – 1,2%.

Как правило, в системах однократной аутентификации по геометрии лица *FAR* намного ниже *FRR* (например, 6,5% *FRR* и 0,1% *FAR*) [26]. Это оправданно, поскольку обычно важнее не пропустить постороннего человека, чем отказать в доступе подлинному пользователю. В случае непрерывной аутентификации очень вероятна ситуация, когда уже авторизованный пользователь во время работы в определенный момент может быть неверно опознан и получит отказ в доступе. Для решения этой проблемы предлагается держать *FRR* на балансе с *FAR*, это позволит сбалансировать ошибку.

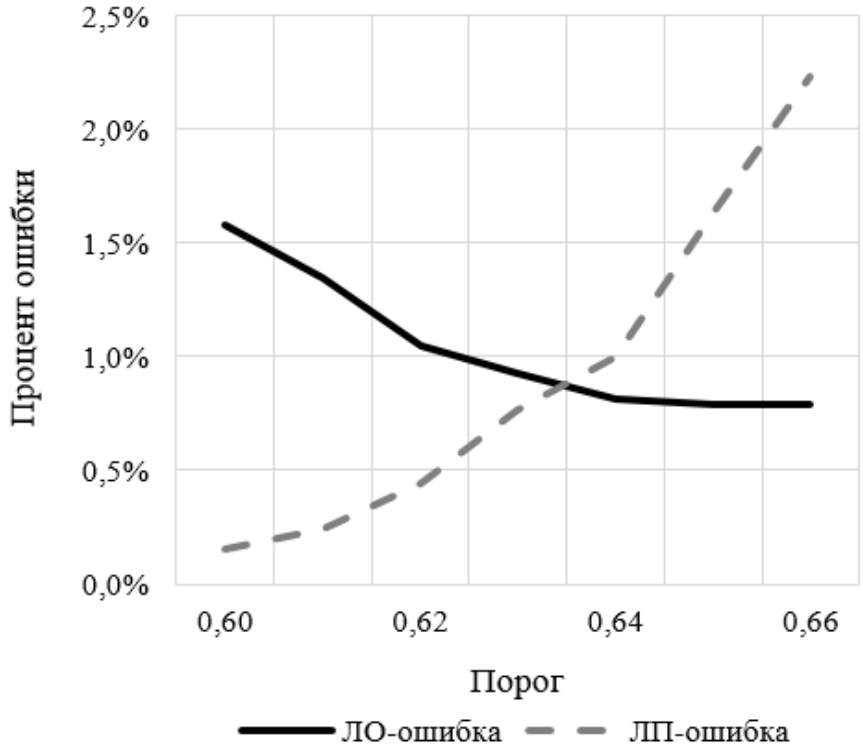

Рисунок 3.1 – Соотношение *FAR*/*FRR* для стандартного теста 360p (фрагмент)

Тестирование также было проведено для каждой из групп по отдельности (см. рисунки 3.2 и 3.3). Первая группа распознается гораздо лучше, чем вторая: *ERR* для первой группы составил 0,15% при $threshold_1$ = 0,59. *ERR* для второй группы достиг 1,31% при $threshold_1$ = 0,68. Отсюда следует, что эмоциональная выразительность и задний фон существенно влияют на точность распознавания.

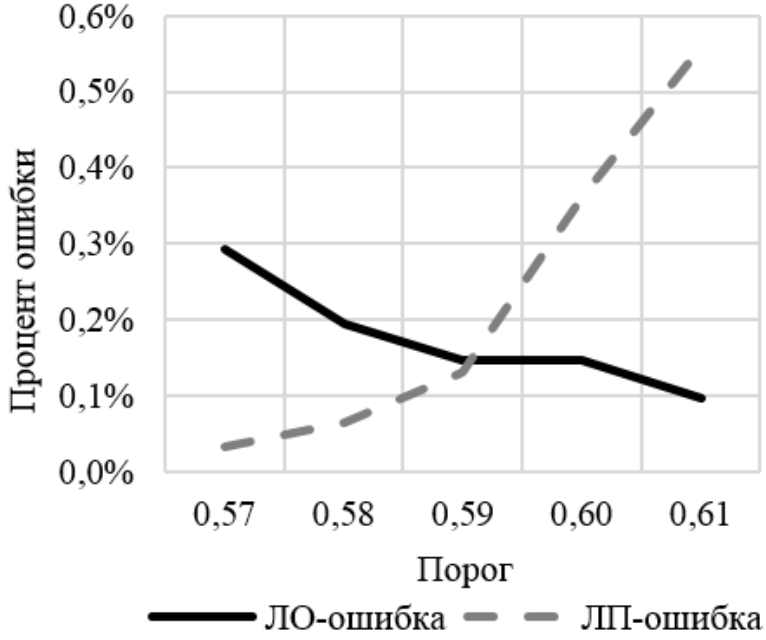

Рисунок 3.2 – Соотношение FAR/FRR для первой группы (фрагмент)

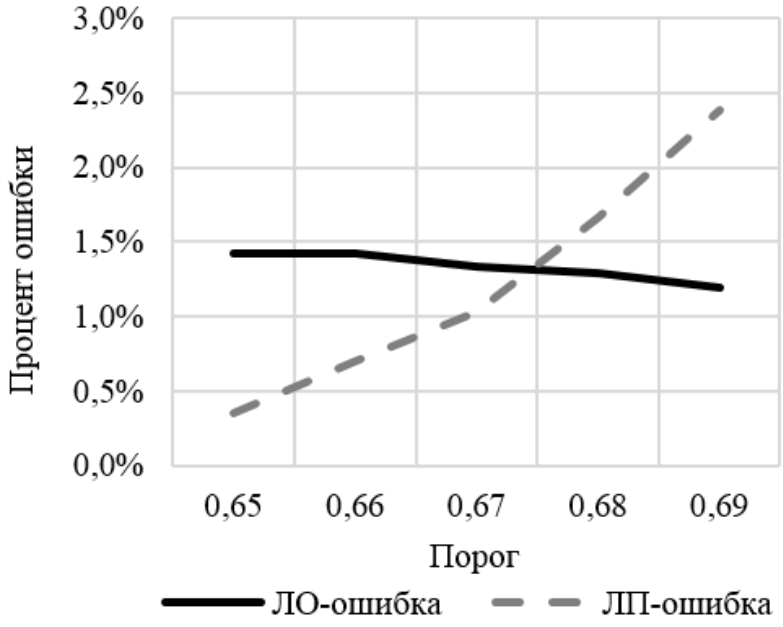

Рисунок 3.3 – Соотношение *FAR*/*FRR* для второй группы (фрагмент)

Далее было протестировано влияние различных пользовательских шаблонов на точность системы. Шаблоны были получены автоматически непосредственно из видеоматериала путем считывания каждого сотого обрабатываемого кадра как одного из четырех изображений. Результаты тестирования показаны на рисунке 3.4. Изменение шаблона почти не влияет на точность: достигнут *ERR* = 0,88%, при *threshold* = 0,61.

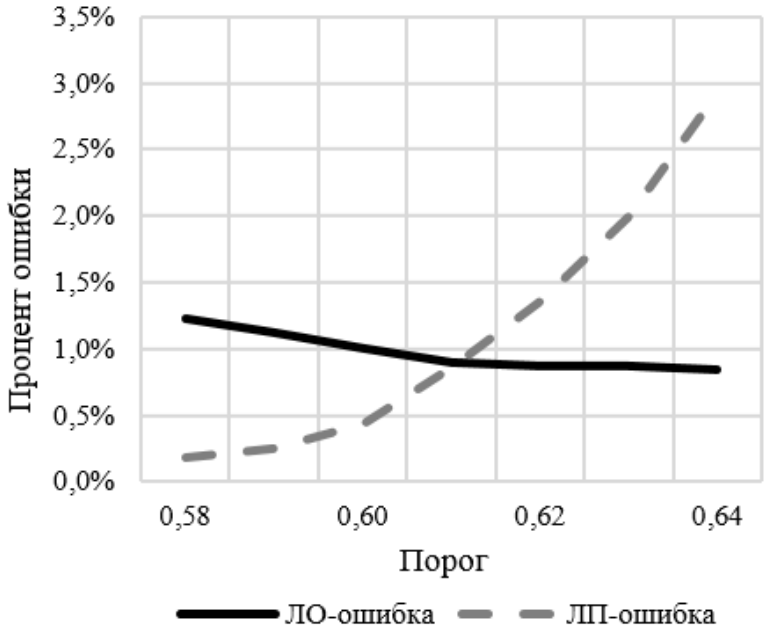

Рисунок 3.4 – Соотношение FAR/FRR для автоматически взятых шаблонов (фрагмент)

Теперь рассмотрим изменение разрешения. Для этого весь видеоматериал был последовательно протестирован в следующих разрешениях: 240p, 360p (стандартный тест), 480p, 720p. Результаты тестирования показаны на рисунке 3.5.

Тестирование демонстрирует, что самая высокая точность достигнута на стандартном тесте при 360p. При сильном снижении разрешения *ERR* резко увеличивается до 2,5% при 240p. Однако при повышении разрешения *ERR* также повышается: до 1,10% при 480p и 1,25% при 720p.

Было протестировано влияние освещения на качество распознавания. Для имитации разного освещения все шаблоны пользователей были взяты с различной яркостью: -50%, -25%, 0% (стандартный тест), +25%, +50%. Результаты тестирования показаны на рисунке 3.6.

Освещение отрицательно влияет на точность распознавания только при высокой или очень низкой интенсивности, обычно не используемых в реальности [7]. Увеличение яркости на 25% уменьшило *ERR* до 0,86%. Дальнейшее осветление шаблонов ведет к резкому уменьшению точности до 3,6%. Затемнение оказало незначительное влияние на *ERR* – 0,92% при уменьшении яркости на 25% и 0,96% при уменьшении на 50%.

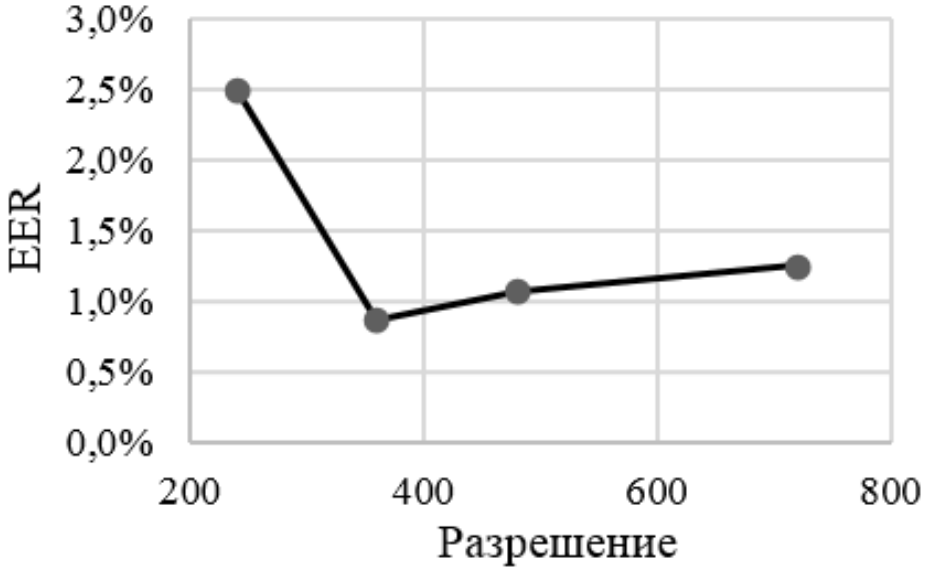

Рисунок 3.5 – Влияние разного разрешения видео на *ERR*

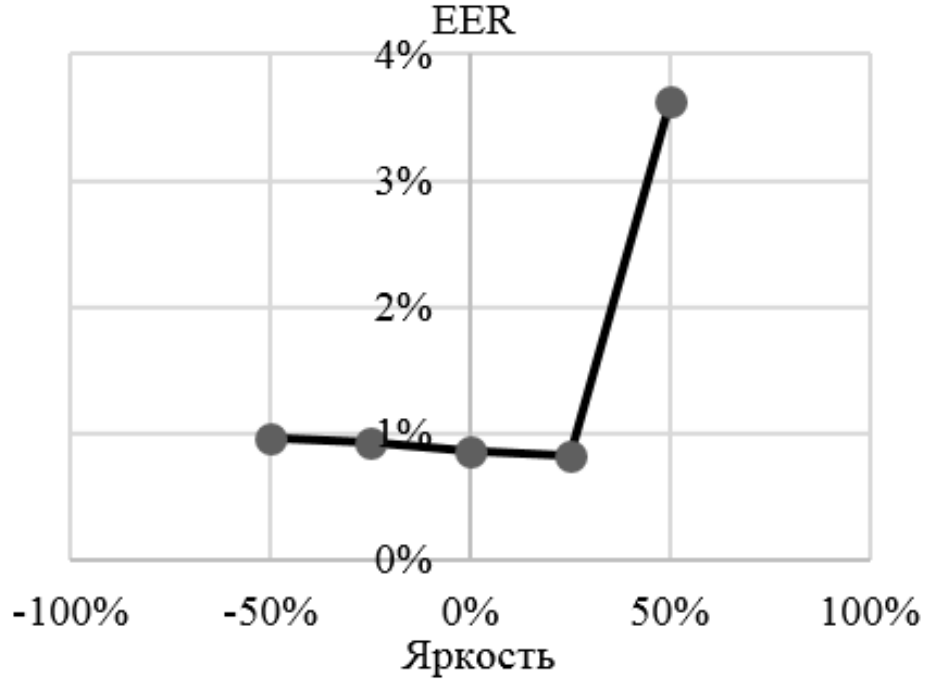

Рисунок 3.6 – Влияние уровня освещенности на *ERR*

3.4 Оценивание защищенности от фальсификации образа пользователя

Чтобы оценить точность обоих методов классификации фотографии, использованы *FAR* и *FRR*. *FAR* это вероятность того, что система ошибочно распознает фотографию пользователя как его реальное лицо, а *FRR* – вероятность распознавания настоящего, но малоподвижного лица пользователя как фотографии. Тестирование проводилось с использованием камеры «HDE 12 Megapixel». В качестве материала для тестирования использовались 250 кадров 10-ти разных живых людей и 250 кадров фотографий 10-ти людей перед веб-камерой.

Результаты для первого метода показаны на рисунке 3.7. Достигнутый *ERR* составляет 3,5% при $threshold_2 = 0{,}1$. Следует учесть, что ситуации, когда настоящий пользователь на рабочем месте сидит достаточно малоподвижно, могут возникать очень часто. Поэтому предлагается держать *FRR* выше, чем *FAR*, при $threshold_2$ не более 0,09, во избежание ложных распознаваний фотографии.

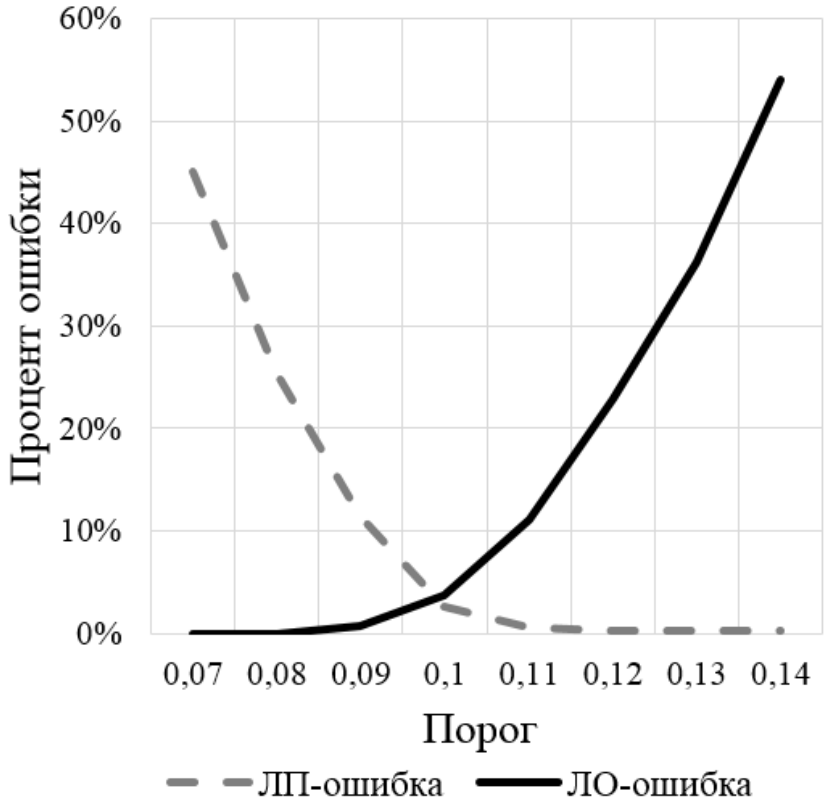

Рисунок 3.7 – Иллюстрация отношения *FAR* к *FRR* для тестирования качества классификации фотографии первым методом

Тестирование второго метода было проведено при обучающей выборке до 3000 кадров для обоих классов. Рисунок 3.8 показывает результаты тестирования при размере выборки в 500, 1000, 1500, 2000, 2500 и 3000 кадров.

Увеличение размера обучающей выборки повышает точность. При 3000 кадрах достигнут *ERR* = 1,3%. Следует заметить, что скорость роста точности уменьшается. Следовательно, дальнейшее увеличение размера выборки не будет иметь большого влияния на рост точности. Сравнение двух методов показывает, что при достаточно крупном размере обучающей выборки второй метод демонстрирует большую точность.

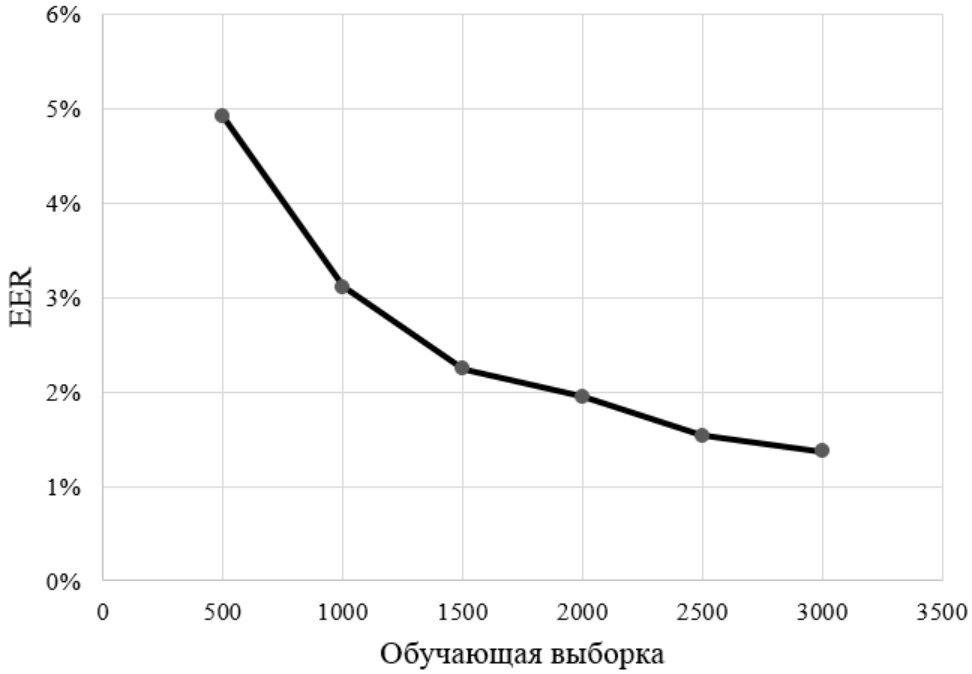

Рисунок 3.8 – Зависимость *ERR* от размера обучающей выборки.
Минимальное значение = 1,3% при 3000 кадрах,
максимальное = 4,9% при 500 кадрах

3.5 Общая точность распознавания лица

Рассмотрим общую точность системы, с учетом распознавания легитимного пользователя и распознавания фотографии. При считывании кадра с веб-камеры или с видео могут произойти следующие события:

– на входе изображение с лицом легитимного, подлинного пользователя (событие A);

– на входе изображение с фотографией легитимного пользователя (событие B);

– на входе изображение с лицом чужого человека (событие C).

Система распознавания работает по следующему алгоритму. Сначала она распознает лицо на легитимность, в зависимости от порога t_1. Если результат больше t_1, то человек считается чужим, отказ в доступе (событие D). Иначе она пытается распознать, находится ли лицо на фотографии, в зависимости от порога t_2. Если результат меньше t_2, считается, что на изображении – фотография и следует отказ в доступе (событие E). Иначе – пользователь считается живым и легитимным, следует получение доступа (событие F).

Пусть $FRR_1(t_1)$ – вероятность ложного отказа системы при проверке на легитимность, $FRR_2(t_2)$ – вероятность ложного отказа системы при классификации фотографии. Определим общую вероятность ложного отказа системы FRR_{total}. Это вероятность наступления любого из событий D или E при условии, что произошло событие A. Тогда на нее влияют следующие события:

– изображение содержит лицо легитимного пользователя, лицо опознано как пользователь, опознано как фотография (AE);

– изображение содержит лицо легитимного пользователя, лицо опознано как чужой человек (AD);

Тогда FRR_{total} вычисляется по формуле

$$FRR_{total} = P(D|A) + P(E|A), \quad (3)$$

где

$$P(D|A) = (1 - FRR_1(t_1)) \cdot FRR_2(t_2), \quad (4)$$

$$P(E|A) = FRR_1(t_1). \quad (5)$$

Отметим, что априорная вероятность события A (т. е. P(A)) не влияет на FRR_{total}.

Пусть $FAR_1(t_1)$ – вероятность ложного доступа системы при проверке на легитимность, $FAR_2(t_2)$ – вероятность ложного доступа системы при классификации фотографии. Определим общую вероятность ложного доступа сис-

темы FAR_{total}. Это вероятность наступления события F при условии, что произошло любое из событий B или C. Тогда на нее влияют следующие события:

– изображение с фотографией легитимного пользователя, лицо обнаружено, опознано как пользователь, опознано как живой человек (BF);

– изображение с лицом чужого человека, лицо обнаружено, опознано как пользователь, опознано как живой человек (CF).

Тогда FAR_{total} вычисляется по формуле

$$FAR_{total} = \frac{P(F|B)\cdot P(B)+P(F|C)\cdot P(C)}{P(B)+P(C)}, \quad (6)$$

где

$$P(F|B) = (1 - FRR_1(t_1)) \cdot FAR_2(t_2), \quad (7)$$

$$P(F|C) = FAR_1(t_1) \cdot (1 - FRR_2(t_2), \quad (8)$$

P(B), P(C) – априорные вероятности наступления событий B и C соответственно.

Не было найдено каких-либо примеров, в которых система бы ошиблась при распознании наличия или отсутствия лица. Поэтому FAR_0 и FRR_0 были взяты за 0,1%. Учитывая это, были взяты по всем значениям t_1 и t_2 и достигли оптимальных значений FAR_{total} = 1,02% и FRR_{total} = 1,06% при t_1 = 0,63 и t_2 = 0,1. Данное значение не актуально при использовании второго метода распознавания фотографии, поскольку для этого метода не существует порога.

3.6 Тестирование производительности

Тестирование производительности проводилось на ПК с 8 ГБ RAM с процессором Intel Core i5-3570K 3.40GHz, имеющим 4 ядра. Нагрузка была распределена на одно из ядер. Было взято видео продолжительностью 11 минут, в разных разрешениях.

Рассмотрим изменение среднего значения продолжительности процедуры распознавания при изменении разрешения видео. Время увеличивается почти линейно, от половины секунды при 240 пикселях, до 0.66 секунд при

360 пикселях и до 1.28 секунд при 720 пикселях. Следовательно, большее разрешение отрицательно влияет на продолжительность.

Проведено тестирование средней нагрузки на CPU и влияния разного разрешения на нее. Этот показатель тоже демонстрирует практически линейный рост при увеличении разрешения, от 22,7% при 240 пикселях до 71,6% при 720 пикселях.

Тестирование нагрузки на RAM показало, что разное разрешение мало влияет на этот показатель. Система уже при запуске использует не менее 230 МБ. Впоследствии потребление вырастает в среднем до 285 МБ (290 МБ в пределе) при обработке видео в 720 пикселей.

Рисунок 3.9 показывает относительное изменение всех трех показателей. Их значение при 360 пикселях было взято за 100%.

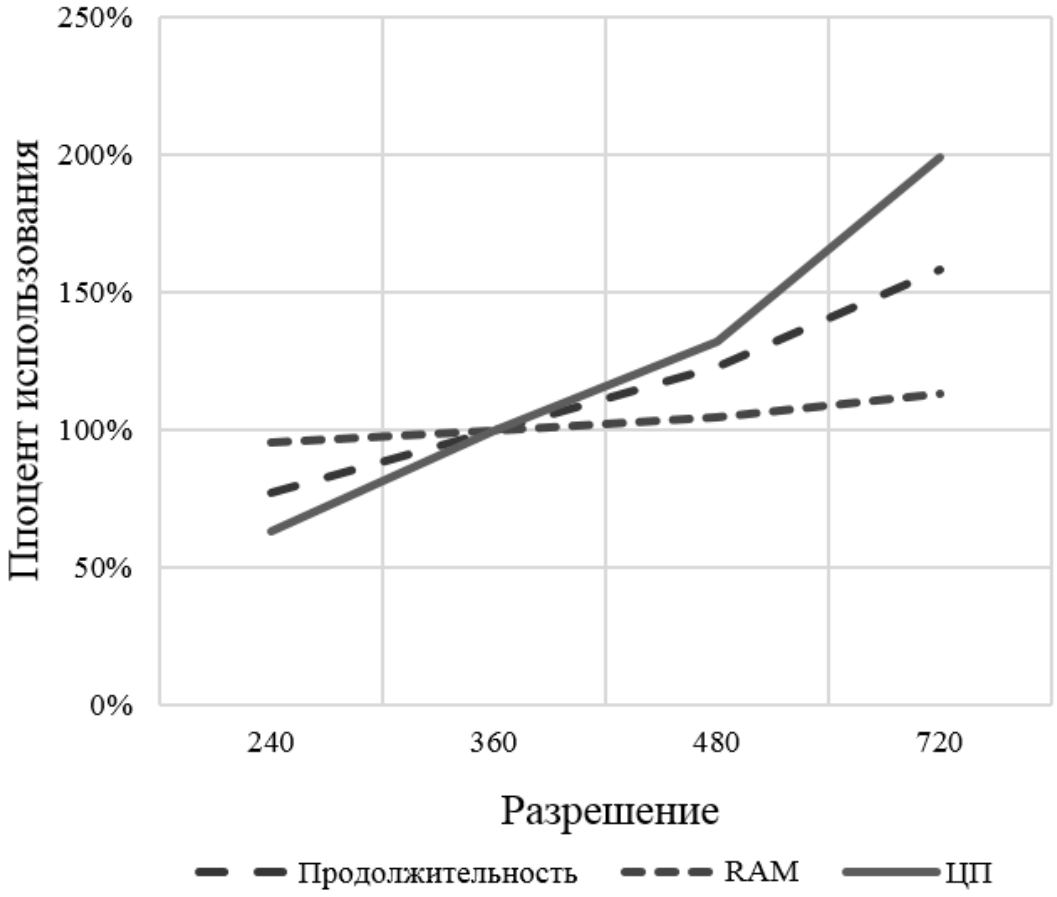

Рисунок 3.9 – Относительное изменение нагрузки на ЦПУ, потребления RAM и средней длины процедуры распознавания при разном разрешении видео.

При анализе результатов нагрузки на RAM и CPU следует учесть, что рассматривается нагрузка только на том временном отрезке, где происходит процедура распознавания. Временной интервал между ними не учитывается.

Производительность и время распознавания при использовании обоих методов классификации фотографии приблизительно одинаковое. Время выполнения первого метода – примерно 0,3 секунды, время выполнения второго метода – приблизительно 0,5 секунд.

Результаты тестирования производительности показывают, что использование системы в смартфонах неоправданно из-за более высокой нагрузки на процессор [37], и следовательно, крупных потерь энергии. Наиболее подходящие типы мобильных устройств – это ноутбуки и планшеты.

3.7 Выводы по разделу

В данном разделе было рассмотрено тестирование системы непрерывной аутентификации. Эта система была протестирована на видеоматериале от различных блогеров. Было выяснено, что разное разрешение видео оказывает влияние на точность, продолжительность процедуры распознавания, нагрузку на RAM и CPU. Наилучшая точность, выраженная через Equal Error Rate, равна 0,86% и достигнута для видео с разрешением 360p, при увеличении яркости шаблонов пользователей на 25%.

Была введена и протестирована способность системы отличать реальное лицо пользователя от его фотографии двумя разными методами. *ERR* первого метода составил 4%, второго метода – достигает 1,4%.

ЗАКЛЮЧЕНИЕ

В работе выполнен обзор существующих методов однократной и непрерывной аутентификации для повышения защищенности рабочих мест пользователей.

Предложен алгоритм распознавания лица с помощью широко распространенных веб-камер. Точность распознавания составила 0,9%. Время однократного распознавания составило 0,6 секунд на процессоре Intel Core i5-3570K 3.40Ghz.

В качестве основной угрозы предложенному методу аутентификации рассматривалась фальсификация образа пользователя с помощью фотографии. Предложено два алгоритма распознавания фотографии пользователя. Первый метод использует особенности представления изображения в библиотеке Dlib и позволяет достичь ошибки 3,5%. Второй метод использует сверточную нейронную сеть для распознавания фотографии пользователя и обеспечивает ошибку до 1,4%.

Предложенная система была протестирована на видеоизображениях различных блогеров. Для тестирования системы использовалось свыше 20000 изображений. Было протестировано влияние условий съемки на точность системы непрерывной аутентификации, при их изменении значительной деградации точности предложенной системы не наблюдалось.

Создаваемая нагрузка на оперативную память и процессор при распознавании изображения с веб-камеры позволяет рекомендовать предложенную систему непререрывной аутентификации для использования на стационарных компьютерах.

По результатам работы опубликована статья «Continuous Authentication System to Increase Security Level of User Workstations» в материалах конференции Global Smart Industry Conference (GloSIC 2018) и статья «Improved Continuous Authentication System with Counterfeit Protection» в журнале Journal of Computational and Engineering Mathematics в 2019 году.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Андреева, Е.А. Система аутентификации, основанная на использовании акустических свойств сердца / Е.А. Андреева // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2012. – № 2. – С. 153–156.

2 Аутентификация. Теория и практика обеспечения доступа к информационным ресурсам = Authentication. Theory and practice of ensuring access to information resources / Под. ред. А.А. Шелупанова, С.Л. Груздева, Ю.С. Нахаева. – М.: Горячая линия – Телеком, 2009. – 552 с.

3 Афанасьев, А.А. Непрерывная аутентификация диктора при ведении телефонных переговоров по низкоскоростным цифровым каналам / А.А. Афанасьев // Вопросы кибербезопасности. – 2016. – № 3. – С. 60–68.

4 Блогеры второй группы. – Дата обновления: 03.02.2019. URL: https://www.youtube.com/playlist?list=PLjd1aefITwnHiW9-9IB5V6tLDeVitb5Lh (дата обращения: 25.03.2019).

5 Блогеры первой группы. – Дата обновления: 03.02.2019. URL: https://www.youtube.com/playlist?list=PLjd1aewnGbaW0mBNna9e1qVBhNV4t0 (дата обращения: 25.03.2019).

6 Брагина Е.К. Современные методы биометрической аутентификации: обзор, анализ и определение перспектив развития / Е.К. Брагина, С.С. Соколов // Вестник Астраханского государственного технического университета. – 2016. – № 4. – С. 40–45.

7 Гладких, А.А. Базовые принципы информационной безопасности вычислительных сетей / А.А. Гладких, В.Е. Дементьев. – Ульяновск: УлГТУ, 2009. – 156 с.

8 Гузик, В.Ф. Биометрический метод аутентификации пользователя / В.Ф. Гузик, М.Н. Десятерик. // Известия Южного федерального университета. Технические науки. – 2000. – Т. 16. – №. 2. – С. 129–133.

9 Гуреева, О. Биометрическая идентификация по отпечаткам пальцев. Технология FingerChip / О. Гуреева // Компоненты и технологии. – 2007. – №. 69. – С. 176–180.

10 Еременко, А.В. Разграничение доступа к информации на основе скрытого мониторинга пользователей компьютерных систем: непрерывная идентификация / А.В. Еременко, Е.А. Левитская, А.Е. Сулавко // Вестник Сибирской государственной автомобильно-дорожной академии. – 2014. – № 16. – С. 40–47.

11 Ермошин, Р.В. Непрерывная аутентификация пользователя по клавиатурному почерку / Р.В. Ермошин // Экономика и социум. – 2017. – № 6 – 2. – С. 697–699.

12 Исхаков, А.Ю. Двухфакторная аутентификация на основе про-граммного токена / А.Ю. Исхаков, Р.В. Мещеряков, И.А. Ходашинский // Вопросы защиты информации. – 2013. – № 3. – С. 23–28.

13 Комаров, А. Современные методы аутентификации: токен и это все о нем / А. Комаров // T-Comm: Телекоммуникации и Транспорт. – 2008. – № 2. – С. 23–26.

14 Ложников, П.С. Аутентификация пользователей компьютера по клавиатурному почерку и особенностям лица / П.С. Ложников, А.Е. Сулавко, Е.В. Бурая // Вопросы кибербезопасности. – 2017. – № 4. – С. 24–34.

15 Мартынова, Л.Е. Исследование и сравнительный анализ методов аутентификации / Л. Е. Мартынова, М.Ю. Умницын, К.Е. Назарова // Молодой ученый. – 2016. – № 19. – С. 90–93.

16 Моржаков, В. Современные биометрические методы иденти-фикации / В. Моржаков, А. Мальцев // Безопасность. Достоверность. Информация. – 2009. – №. 83. – С. 44–48.

17 Никишова, А. В. Программный комплекс обнаружения атак на основе анализа данных реестра / А.В. Никишова, А. Е Чурилина // Вестник ВолГУ. Серия 10. Инновационная деятельность. – 2012. – № 6. – С. 152–155.

18 Сабанов, А. Г. Основные процессы аутентификации / А. Г. Сабанов // Вопросы защиты информации. – 2012. – № 3. – С. 54–57.

19 Шапиро, Л. Active Directory Domain Services. Двухфакторная аутентификация. Теоретические основы / Л. Шапиро // Системный администратор. – 2010. – № 7-8. – С. 100–105.

20 Шибанов, С.В. Сравнительный анализ современных методов аутентификации пользователя / С.В. Шибанов, Д.А. Карпушин // Математическое и программное обеспечение систем в промышленной и социальной сферах. – 2015. – №1. – С. 33–37.

21 Azar, C., Brostoff, G. System and method for providing secure access to an electronic device using continuous facial biometrics : pat. 8370639 USA. – 2013.

22 Feng, T. Continuous mobile authentication using touchscreen gestures / T. Feng // Homeland Security (HST), 2012 IEEE Conference on Technologies for. – 2012. – P. 451–456.

23 Feng, T. Continuous mobile authentication using virtual key typing biometrics / T. Feng // Trust, security and privacy in computing and communications (TrustCom), 2013. 12th IEEE international conference on. – 2013. – P. 1547–1552.

24 Fridman, L. Multi-modal decision fusion for continuous authentication / L. Fridman // Computers & Electrical Engineering. – 2015. – V. 41. – P. 142–156.

25 Gascon, H. Continuous Authentication on Mobile Devices by Analysis of Typing Motion Behavior / H. Gascon, S. Uellenbeck, C. Wolf // Sicherheit. – 2014. – P. 1–12.

26 Gupta, S. Advances and challenges in 3D and 2D + 3D human face recognition / S. Gupta, M.K. Markey, A.C. Bovik. // Pattern recognition in biology. – 2007. – P. 63–103.

27 Holz, C. Biometric touch sensing: Seamlessly augmenting each touch with continuous authentication / C. Holz, M. Knaust. // Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology. – 2015. – P. 303–312.

28 Kazemi, V. One millisecond face alignment with an ensemble of regression trees / V. Kazemi, S. Josephine // 27th IEEE Conference on Computer Vision and Pattern Recognition. – 2014. – P. 1867–1874.

29 King, D.E. Dlib-ml: A machine learning toolkit / D.E. King // Journal of Machine Learning Research. – 2009. – V. 10, №. Jul. – P. 1755–1758.

30 Labati, R.D., Sassi R., Scotti F. ECG biometric recognition: Permanence analysis of QRS signals for 24 hours continuous authentication / R.D. Labati, R. Sassi, F. Scotti // Information Forensics and Security (WIFS), 2013 IEEE International Workshop on. – IEEE, 2013. – P. 31–36.

31 Laganière, R. OpenCV Computer Vision Application Programming Cookbook Second Edition / R. Laganière // Packt Publishing Ltd. – 2014.

32 Locklear, H. Continuous authentication with cognition-centric text production and revision features / H. Locklear // Biometrics (IJCB), 2014 IEEE International Joint Conference on. – IEEE, 2014. – P. 1–8.

33 Mahbub, U. Partial face detection for continuous authentication / U. Mahbub // Image Processing (ICIP), 2016 IEEE International Conference on. – 2016. – P. 2991–2995.

34 Mondal, S. Continuous authentication using mouse dynamics / S. Mondal, P. Bours // Biometrics Special Interest Group (BIOSIG), 2013 International Conference of the. – IEEE, 2013. – P. 1–12.

35 Murmuria, R. Continuous Authentication on Mobile Devices Using Power Consumption, Touch Gestures and Physical Movement of Users / R. Murmuria, A. Stavrou, D. Barbara. // Research in Attacks, Intrusions, and Defenses. – Springer, Cham, 2015. – P. 405–424.

36 Pamplona Segundo, M. Continuous 3D face authentication using RGB-D cameras / M. Pamplona Segundo // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. – IEEE, 2013. – P. 64–69.

37 Patel, V.M. Continuous user authentication on mobile devices: Recent progress and remaining challenges / V.M. Patel // IEEE Signal Processing Magazine. – IEEE, 2016. – T. 33. №. 4. – P. 49–61.

38 Rajkumar, J. Using continuous face verification to improve desktop security / J. Rajkumar, S. Kumar, S. Zhang, T. Sim // Application of Computer Vision. – 2005. – Vol. 1. – P. 501–507.

39 Ross, A. An introduction to biometric systems / A. Ross, A.K. Jain, S. Prabhakar // IEEE Trans Circuits Syst Video Technol. – 2004. – №14. – P. 4–20.

40 Savvides, M., Jaech, A. Continuous Authentication, and Methods, Systems, and Software Therefor : pat. 14/052,200 USA. – 2013.

41 Song, C. Cardiac Scan: A Non-Contact and Continuous Heart-Based User Authentication System / C. Song // Biometrics Special Interest Group (BIOSIG). – 2017. – P. 15–21.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

Листинг программы

```
   import numpy.core.multiarray
   import numpy.core._methods
   import numpy
   import dlib
   import keras
   import cv2
   from skimage import io
   import sched, time, datetime
   from scipy.spatial import distance

   sp    =    dlib.shape_predictor('shape_predictor_68_face_land-
marks.dat')
   facerec  =  dlib.face_recognition_model_v1('dlib_face_recogni-
tion_resnet_model_v1.dat')
   detector = dlib.get_frontal_face_detector()
   s = sched.scheduler(time.time, time.sleep)

   wait = 1

   def registration():
       reader=open("Data.txt","r")
       print("Reg.Ваш логин:")
       InpRegLogin = input()
       if InpRegLogin == "exit":
           reader.close
           exit(0)
       for line in reader:
           if line[0:len(InpRegLogin)] == InpRegLogin:
               print("Занято")
               reader.close()
               registration()
       reader.close
       def InputPass():
           print("Ваш пароль:")
           InpRegPass = input()
           if InpRegPass == "exit":
               exit(0)
           if len(InpRegPass)<5:
               print("короткий")
               InputPass()
           else:
               return InpRegPass
       InpRegPass=InputPass()
       def SampleCreate():
           print("Посмотритесь в камеру, по готовности нажмите F")
           ReadyToShot = input()
```

```
            if ReadyToShot == "F" or ReadyToShot == "f":
                cap = cv2.VideoCapture(0)
                for i in range(30):
                    cap.read()
                ret, frame = cap.read()
                cv2.imwrite("FaceSamples/"+InpRegLogin+".jpg",
frame)
                cap.release
                writer=open("Data.txt","a")
                writer.write("\n"+InpRegLogin+" "+InpRegPass)
                writer.close
            else:
                SampleCreate()
        SampleCreate()
        main()
    def continiousCam(sc, face_descriptor1):
        global wait
        cap = cv2.VideoCapture(0)
        for i in range(30):
            cap.read()
        ret, img2 = cap.read()
        cap.release
        cv2.imshow("frame", img2)
        win1 = dlib.image_window()
        win1.clear_overlay()
        win1.set_image(img2)
        dets = detector(img2, 1)
        for k, d in enumerate(dets):
         #  print("Detection {}: Left: {} Top: {} Right: {} Bottom:
{}".format(
          #      k, d.left(), d.top(), d.right(), d.bottom()))
           shape = sp(img2, d)
           win1.clear_overlay()
           win1.add_overlay(d)
           win1.add_overlay(shape)

        if "shape" in locals():
            face_descriptor2      =      facerec.compute_face_de-
scriptor(img2, shape)
            a   =   distance.euclidean(face_descriptor1,   face_de-
scriptor2)
            if a > 0.6:
                print ("это не вы, вернитесь к камере")
                if wait == 1:
                    wait=0
                    time.sleep(3)
                    s.enter(0.3,   1,   continiousCam,   (s,face_de-
scriptor1))
                else:
                    main()
            else:
                print ("ок")
                wait = 1
```

```
                s.enter(0.3,    1,    continiousCam,    (s,face_de-
scriptor1))
        else:
            print ("это не вы, вернитесь к камере")
            if wait == 1:
                wait=0
                time.sleep(3)
                s.enter(0.3,    1,    continiousCam,    (s,face_de-
scriptor1))
            else:
                main()
    def continiousVid(sc, b, face_descriptor1, cap):
        global wait
        if cap == 0:
            cap = cv2.VideoCapture(0)
        else:
            pass
        for i in range(30):
            cap.read()
        ret, img2 = cap.read()
        cap.release
    #    cv2.imshow("frame", img2)
    #    win1 = dlib.image_window()
    #    win1.clear_overlay()
        dets = detector(img2, 1)
        for k, d in enumerate(dets):
      #     print("Detection {}: Left: {} Top: {} Right: {} Bottom:
{}".format(
     #          k, d.left(), d.top(), d.right(), d.bottom()))
            shape = sp(img2, d)
     #      win1.clear_overlay()
     #      win1.add_overlay(d)
     #      win1.add_overlay(shape)
        if "shape" in locals():
            face_descriptor2       =       facerec.compute_face_de-
scriptor(img2, shape)
            a   =   distance.euclidean(face_descriptor1,   face_de-
scriptor2)
            print (a)
        else:
            print ("1 ")
        u = datetime.datetime.now()
    #    print ((u-b).seconds)
        if (u-b).seconds < 160:
            s.enter(0.3, 1, continiousVid, (s, b, face_descriptor1,
cap))
        else:
            i=0
    def authorization():
        reader=open("Data.txt","r")
        print("Ваш логин:")
        InpAutLogin = input()
        if InpAutLogin == "exit":
```

```
            exit(0)
        print("Ваш пароль:")
        InpAutPass = input()
        if InpAutPass == "exit":
            exit(0)
        x=0
        for line in reader:
            if line == InpAutLogin+" "+InpAutPass:
                x=1
                break
        if x!=1:
            print("Неверные текстовые данные")
            reader.close()
            authorization()
        reader.close
        img1 = io.imread("FaceSamples/"+InpAutLogin+".jpg")
        #win2 = dlib.image_window()
        #win2.clear_overlay()
        #win2.set_image(img)
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
         #   print("Detection {}: Left: {} Top: {} Right: {} Bottom:
{}".format(
        #         k, d.left(), d.top(), d.right(), d.bottom()))
            shape = sp(img1, d)
        #    win2.clear_overlay()
        #    win2.add_overlay(d)
         #   win2.add_overlay(shape)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape)
        s.enter(0.3, 1, continiousVid, (s,face_descriptor1))
        s.run()
        AfterAutho()

    def AfterAutho():
        r=input()
        while r!="Exit":
            r=input()
        if r =="Exit":
            main()
    def info():
        print("R - регистрация")
        print("A - авторизация")
    def main():
        info()
        inputted = input()
        if inputted == "R" or inputted == "r":
            registration()
        else:
            if inputted == "A" or inputted == "a":
                authorization()
            else:
                exit(0)
```

```
      # if inputted != "R" or "r" or"A" or "a":
      #     exit(0)
   main()
   sp    =    dlib.shape_predictor('shape_predictor_68_face_land-
marks.dat')
   facerec  =  dlib.face_recognition_model_v1('dlib_face_recogni-
tion_resnet_model_v1.dat')
   detector = dlib.get_frontal_face_detector()
   s = sched.scheduler(time.time, time.sleep)

   wait = 1

   def registration():
       reader=open("Data.txt","r")
       print("Reg.Ваш логин:")
       InpRegLogin = input()
       if InpRegLogin == "exit":
           reader.close
           exit(0)
       for line in reader:
           if line[0:len(InpRegLogin)] == InpRegLogin:
               print("Занято")
               reader.close()
               registration()
       reader.close
       def InputPass():
           print("Ваш пароль:")
           InpRegPass = input()
           if InpRegPass == "exit":
               exit(0)
           if len(InpRegPass)<5:
               print("короткий")
               InputPass()
           else:
               return InpRegPass
       InpRegPass=InputPass()
       def SampleCreate():
           print("Посмотритесь в камеру, по готовности нажмите F")
           ReadyToShot = input()
           if ReadyToShot == "F" or ReadyToShot == "f":
               cap = cv2.VideoCapture(0)
               for i in range(30):
                   cap.read()
               ret, frame = cap.read()
               cv2.imwrite("FaceSamples/"+InpRegLogin+".jpg",
frame)
               cap.release
               writer=open("Data.txt","a")
               writer.write("\n"+InpRegLogin+" "+InpRegPass)
               writer.close
           else:
               SampleCreate()
```

```
        SampleCreate()
        main()
    def continiousVid(sc, b, face_descriptor1, face_descriptor2,
face_descriptor3, face_descriptor4, cap):
        global wait
        if cap == 0:
            cap = cv2.VideoCapture(0)
        else:
            pass
        for i in range(30):
            cap.read()
        ret, img2 = cap.read()
        cap.release
    #    cv2.imshow("frame", img2)
    #    win1 = dlib.image_window()
    #    win1.clear_overlay()
        dets = detector(img2, 1)
        for k, d in enumerate(dets):
     #      print("Detection {}: Left: {} Top: {} Right: {} Bottom:
{}".format(
     #           k, d.left(), d.top(), d.right(), d.bottom()))
            shape = sp(img2, d)
     #      win1.clear_overlay()
     #      win1.add_overlay(d)
     #      win1.add_overlay(shape)
        if "shape" in locals():
            face_descriptor5       =       facerec.compute_face_de-
scriptor(img2, shape)
            a1   =   distance.euclidean(face_descriptor1,   face_de-
scriptor5)
            a2   =   distance.euclidean(face_descriptor2,   face_de-
scriptor5)
            a3   =   distance.euclidean(face_descriptor3,   face_de-
scriptor5)
            a4   =   distance.euclidean(face_descriptor4,   face_de-
scriptor5)
            a = (a1+a2+a3+a4)/4
            if a>1:
                a = 1
            print (a)
        else:
            print ("1 ")
        u = datetime.datetime.now()
        if (u-b).seconds < 170:
            s.enter(1, 1, continiousVid, (s, b, face_descriptor1,
face_descriptor2, face_descriptor3, face_descriptor4, cap))
        else:
            i=0
    def info():
        print("R - регистрация")
        print("A - авторизация")
    def main():
        info()
```

```
        inputted = input()
        if inputted == "A" or inputted == "a":
            FAR()
        else:
            exit(0)
    def FRR():
        #5
        name = "Vilyan"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #1
        name = "Goblin"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
```

```
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("Goblin2Low360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #2
        name = "Zhukov"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
```

```
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("Zhukov2Low360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #3
        name = "Popov"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #4
        name = "Ulin"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
```

```
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #6
        name = "Bad"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
```

```
        #7
        name = "Shari"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #8
        name = "Eng"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
```

```
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
        #9
        name = "Book"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("Book2Low360.mp4")
        s.enter(0.9,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap, name))
        s.run()
    def FAR():
        #1
        name = "Goblin"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
```

```
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
```

```
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #2
        name = "Zhukov"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
```

```
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #3
        name = "Popov"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
```

```
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
```

```
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #4
        name = "Ulin"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
```

```
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #5
        name = "Vilyan"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
```

```
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #6
        name = "Bad"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
```

```
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
```

```
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #7
        name = "Shari"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
```

```
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #8
        name = "Eng"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
```

```
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
```

```
        cap = cv2.VideoCapture("BookLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        #9
        name = "Book"
        img1 = io.imread("FaceSamples/Bright-25/"+name+".jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor1  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"2.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor2  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"3.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor3  =  facerec.compute_face_descriptor(img1,
shape, 12)
        img1 = io.imread("FaceSamples/Bright-25/"+name+"4.jpg")
        dets_webcam = detector(img1, 1)
        for k, d in enumerate(dets_webcam):
            shape = sp(img1, d)
        face_descriptor4  =  facerec.compute_face_descriptor(img1,
shape, 12)
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ZhukovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("PopovLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("UlinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("VilyanLow360.mp4")
```

```
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("BadLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("ShariLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("EngLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
        b = datetime.datetime.now()
        cap = cv2.VideoCapture("GoblinLow360.mp4")
        s.enter(1,      1,      continiousVid,      (s,b,face_de-
scriptor1,face_descriptor2,face_descriptor3,face_descriptor4,
cap))
        s.run()
    #    print("R - регистрация")
    #    print("A - авторизация")
    main()
```