Задача

С развитием технологий люди все чаще задумываются о защите информации на компьютерах и мобильных устройствах. В последнее время, как никогда раньше, важна безопасность данных. Так как уже сегодня компьютеры и телефоны уже стали обыденной вещью, уже многое из хранится на них, поэтому важно это всё защищать от различных злоумышленников. Обычно это решается установкой какого-то пароля. Но проблема, что это бывает не надежно. Но благодаря развитию науки и появлению новых алгоритмов можно исправить ситуацию.

Было создано приложение под мобильные устройства под управлением операционной системы Android, позволяющее авторизовать пользователя по его лицу, причем оно не просто фотографируется, а создается его 3D модель, что уменьшает риск обмана алгоритмов.

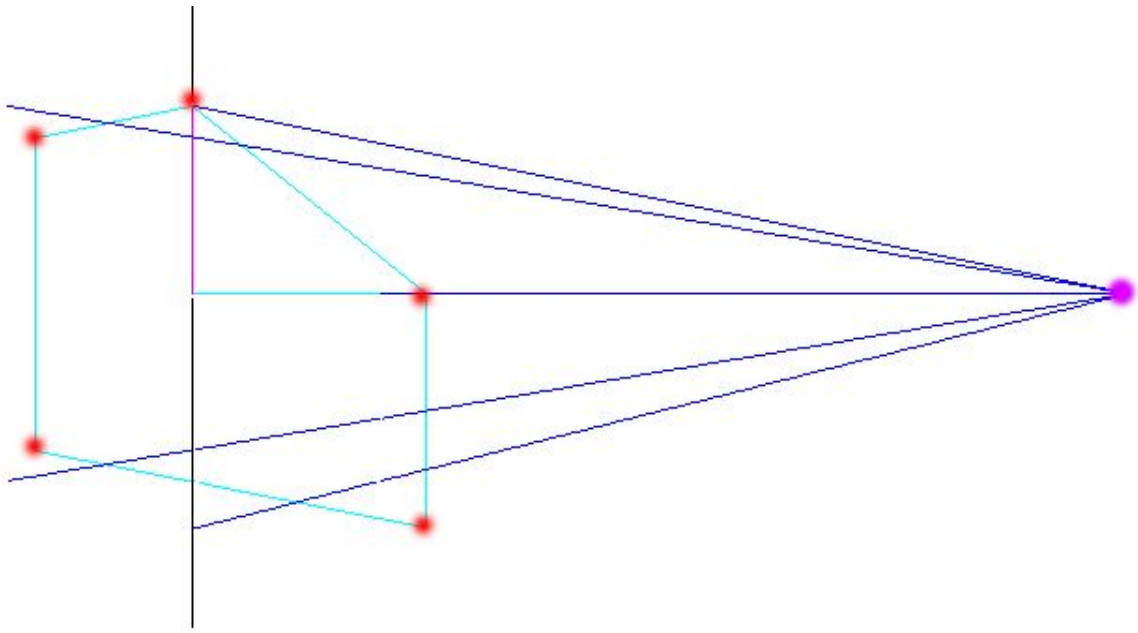
Таким образом, цель данной работы — разработка алгоритма, который обеспечит быструю и надежную авторизацию пользователя.

Поиск решения

Первые идеи были о том, чтобы использовать тени для получения 3D модели лица. Но данная идея была сложна в реализации, поэтому не нашла себе применения. Также была ещё идея пробовать трансформировать изображения объектов (например, сужать,

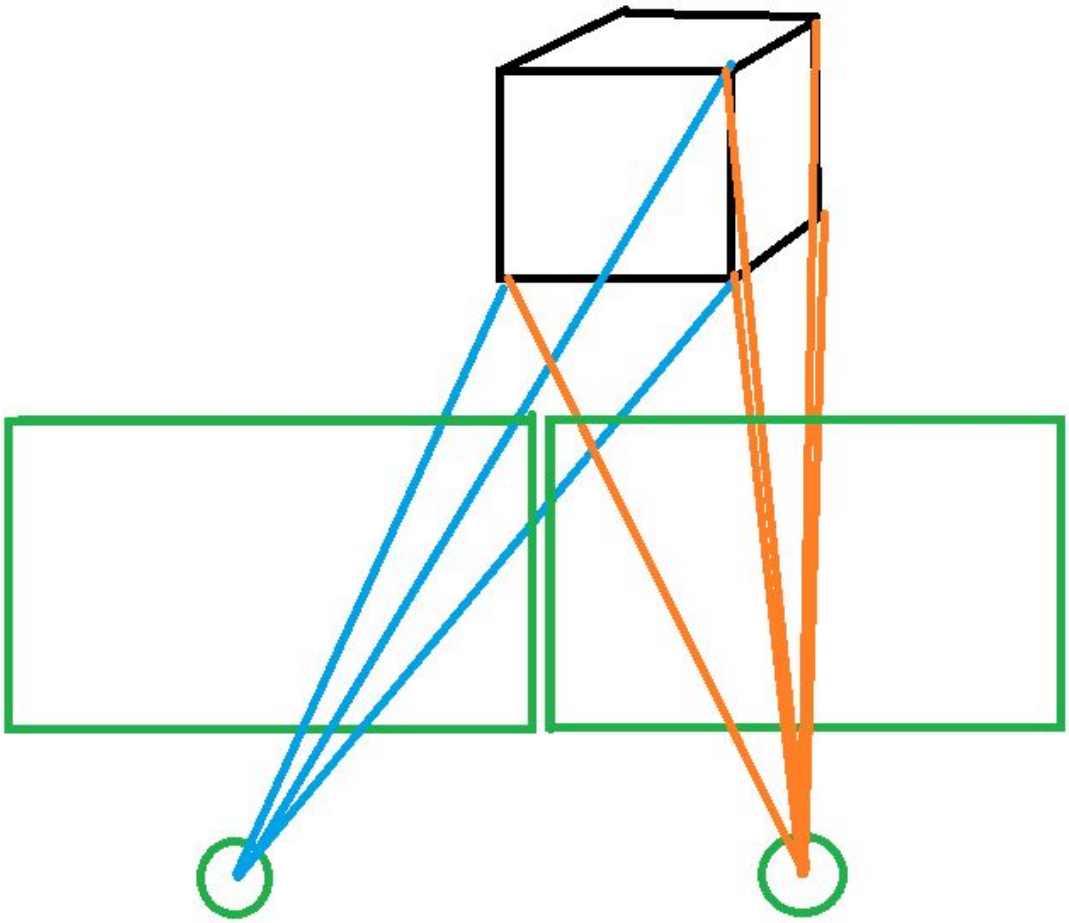
поворачивать) и сравнивать их с изображением того же объекта, но уже с другого ракурса. Проблема была с тем, что изображения объекта с разных ракурсов могут быть слишком сильно отличаться, чтобы можно было хоть что-то понять, что где, (пример: куб с разноцветными сторонами) да и к тому же объекты у нас довольно сложного и разнообразного вида, а такой алгоритм быстро их обработать не может.

В итоге пришла в голову мысль, а не попробовать ли для начала найти решение обратной задачи, то есть из трехмерного изображения получить двухмерное. Проецирование трёхмерных объектов в двумерные обычно начинают так: сначала проводят кучу линий между какой-то одной точкой и точками на границах объекта.



А затем между точкой и объектом ставят плоскость, через которую проходят все эти линии. Точки пересечения линий и плоскости и есть контур спроецированного трехмерного объекта.

В нашей задаче используется тот же принцип только наоборот: мы знаем где та точка, которую давайте назовём точкой камеры, и точки двухмерной проекции. Нам остаётся только получить два изображения с одного ракурса и с другого, который сдвинут вбок относительно первого, и найти пересечения этих прямых.



При создании алгоритма, конечно, происходили многие трудности. Самая сложная часть — это часть с поиском точки пересечения прямых. Проблемы были с тем, что прямые почти никогда не получаются пересекающимися из-за ошибок, связанных то с нечеткостью изображений, то с неточностью вычисления дробных чисел и так далее. Поэтому приходится искать общий перпендикуляр между двумя прямыми, и на нем уже выбирать какую-то точку.

Алгоритм

Для авторизации пользователя строится 3D модель его лица.

Весь алгоритм разбит на несколько частей.

Съемка фотографий

Для начала важно заметить, что когда происходит съемка, камера должна перемещаться строго вдоль одной линии, тогда все полученные изображения будут принадлежат одной плоскости, что необходимо для алгоритма.

При съемке также одновременно записываются данные с акселерометра. Так как акселерометр может дать только информацию об ускорении

телефона, приходится записывать время и вычислять пройденное расстояние по такой формуле:

$$S = \frac{at^2}{2},$$

где a — ускорение,

t — промежуток времени

Это расстояние затем используется для определения координат снимка.

Обработка изображения

Происходит упрощение фотографий.

Сперва изображения уменьшаются до размера 64x64 пикселей.

Потом они обесцвечиваются, то есть у каждого пикселя цвет меняется по такой формуле:

$$\begin{aligned} z &= max(r_0, g_0, b_0) \\ \mathrm{r} &= \mathrm{z} \\ \mathrm{g} &= \mathrm{z} \\ \mathrm{b} &= \mathrm{z} \end{aligned},$$

где r_0, g_0, b_0 — изначальные компоненты цвета,

r, g, b — красная, зеленая и синяя компоненты нового цвета

Что делает формула:

Также затем ещё повышается контраст. Это происходит по такой формуле:

$$r = f * (r_0 - 0.5) + 0.5$$
$$\mathrm{g} = \mathrm{f} * (\mathrm{g}_0 - 0.5) + 0.5$$
$$\mathrm{b} = \mathrm{f} * (\mathrm{b}_0 - 0.5) + 0.5,$$

где r_0, g_0, b_0 — изначальные компоненты цвета,

r, g, b — красная, зеленая и синяя компоненты нового цвета,

f — некоторое число.

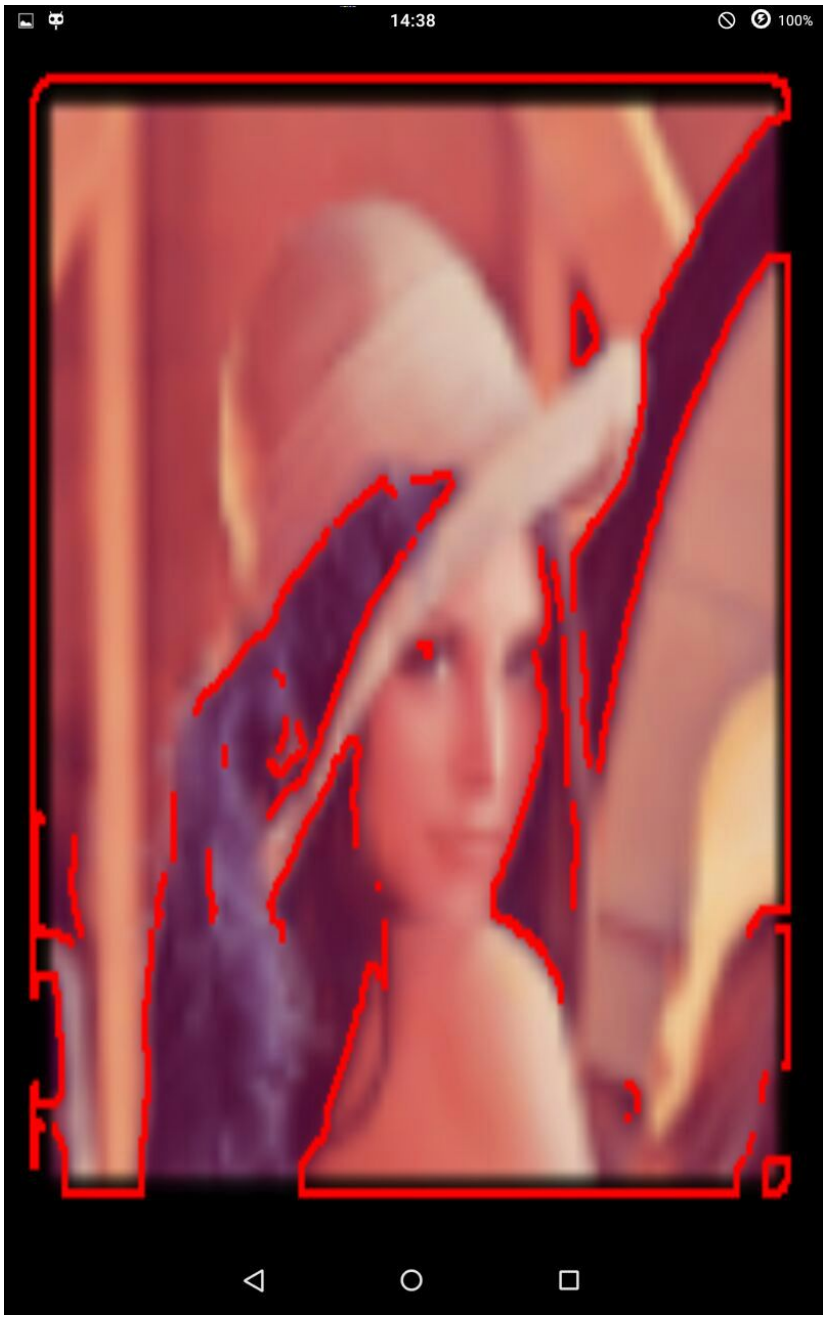
Если r, g или b стало меньше 0, то оно приравнивается к 0, а если больше 1 — то к 1.

Что делает формула:

Построение 3D модели из нескольких фотографий

Первым делом, алгоритм ищет места в изображениях, где есть границы каких-то ни было предметов. Это происходит по такому принципу: в цикле перебираются все пиксели слева направо, а потом также ещё сверху вниз. Алгоритм решает, что границы там, где отличаются яркости двух пикселей на число равное 0.5.

Пример работы поиска границ:



(это сейчас плохой пример, в программе поиск точек специально на самом деле чуток изменен, чтобы лучше работать на лицах)

После данной процедуры, начинается собственно создаваться 3D модель.

Для каждого изображения строится множество прямых, проходящих через точку, представляющее камеру в пространстве, и точку, где есть граница изображения.

Положение точек фотографии и камеры в пространстве определяется данными с акселерометра и параметрами линз в камере телефона.

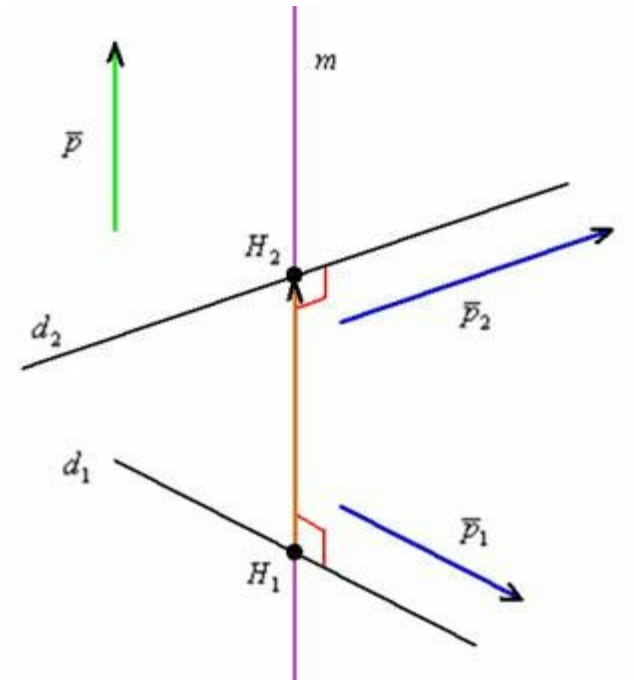
Уравнение прямой записывается таким образом:

$$AB: \begin{cases} x = x_A + (x_B - x_A) * t \\ y = y_A + (y_B - y_A) * t \\ z = z_A + (z_B - z_A) * t \end{cases},$$

где t — любое число, кроме 0.

После этого все эти прямые со всех фотографий собираются в кучу и находятся примерные точки пересечения прямых. Эти точки в итоге и будут представляют модель. Мы находим примерные точки пересечения прямых, так как вероятность настоящего пересечения двух таких прямых очень мала, и поэтому находится не сама точка пересечения, а любая из точек на общем перпендикуляре этих прямых.

Расстояние между прямыми равна длине общего перпендикуляра:



$$\rho(d_1; d_2) = \frac{\left|(\bar{p}_1 \cdot \bar{p}_2 \cdot \overline{M_1M_2})\right|}{\left|[\bar{p}_1 \times \bar{p}_2]\right|}$$

В качестве нужной нам точки мы выбираем $H_1(x_1, y_1, z_1)$. Замечаем, что раз H_1 принадлежит прямой d_1, то можно выразить координаты H_1 через параметры d_1 заменив параметр t на свой параметр t_1. Сделав тоже самое с другой точкой и использовав утверждение, что $\overline{H_1H_2}, \bar{p}$ — коллинеарны, можно получить систему уравнений, которая в программе решается методом Крамера.

Сравнение моделей

Если модели совпали на 50%, то авторизация считается успешной.

Сравниваются модели так: для каждой точки одной модели ищется такая точка другой модели, что расстояние между ними меньше 4. Затем вычисленное количество подошедших точек первой модели делится на количество всех точек первой модели.

Программа

Интерфейс

Если пользователь впервые пользуется программой, то ему нужно для начала сохранить 3D модель его лица. Чтобы сделать это, ему нужно открыть приложение и нажать на кнопку “Установить”.

Затем модель сохраняется в файл в формате json.

Когда нужно будет воспользоваться программой для авторизации, программа заснимет новую модель и сравнит её с сохраненной.

Применение программы в других программах

Эта программа предоставляет интерфейс для других Android приложений. По нему другие приложения могут использовать данный алгоритм авторизации в своих нуждах.

Используемые технологии

Программа написана на языке Scala. Этот язык примечателен тем, что благодаря ему можно легко обрабатывать данные. Также на нем можно

разрабатывать приложения под Android, операционной системы для телефонов, под которой запускается данная программа. Из дополнительных библиотек используется библиотека json4s, которая нужна для обработки файлов в формате json, а также библиотека scaloid, которая делает разработку приложения под Android более удобным в языке программирования Scala.

Промежуточные исследования

Есть другие проекты, которые также реализуют авторизацию пользователя по 3D модели лица человека. Один из них — это тот, который включено в списокк литературы внизу как №2. Алгоритм в этом документе существенно отличается. Там на лицо человека некоторое время светит проектор с изображением параллельных полосок, и эти полоски в итоге позволяют построить 3д модель, так как они искривляются, когда падают на лицо. Этот метод работает хорошо и эффективно, ему может даже хватить одной фотографии, чтобы понять, что как выглядит. Но в нем есть также и проблемы. Во-первых, необходимо дополнительное оборудование, которое может так светить, как нужно их алгоритму. Во-вторых, алгоритм всё-таки ошибается, когда изменяется положение лица или если есть очки, или когда человек улыбается.

Результаты работы

В экспериментах алгоритм всё-таки не всегда показывал себя не очень хорошо из-за различного рода проблем, например, связанных с различными вычислительными ошибками из-за неточностей данных, которые происходят в следствие обработки изображения, а также из-за сложности решения разнообразных особых случаев. Данному алгоритму также ещё не хватает реализации обработки различных изменений лица, например, улыбок. Но в теории, данное решение задачи довольно хорошее. Оно позволяет с минимальным количеством необходимых устройств решить задачу пользователя — защитить его данные.

Список литературы и ссылок

1. Взаимное расположение прямых в пространстве. http://mathprofi.ru/zadachi_s_pryamoi_v_prostranstve.html
2. Automatic 3D Face Authentication. C Beumier, M Acheroy - Image and Vision Computing, 2000. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.9190&rep=rep1&type=pdf
3. Краткий курс компьютерной графики: пишем упрощённый OpenGL своими руками https://habrahabr.ru/post/248611/
4. Android Developers http://developer.android.com/index.html