**Теория (Название)**

Задача поиска лиц на фотографии является одним из основных элементов разрабатываемого приложения. Данная задача наиболее затратна в плане требуемых ресурсов компьютера, однако, от ее реализации зависит успех и качество выполнения второй части, связанной с поиском конкретного лица.

Вышеупомянутая задача тесно связана с задачей преобразования входных данных в необходимый нам формат. Например, видеоролики необходимо обрабатывать покадрово, то есть стоит задача разбиения входного видеоролика на отдельные кадры. Каждый кадр должен отдельно обрабатываться на предмет наличия на нем лиц.

Для реализации описанной задачи предлагается создать два интерфейса – один для считывания входных данных, другой для реализации задачи поиска и распознавания лиц. Каждый из этих интерфейсов включает в себя набор чисто виртуальных функций, требующих реализации в классе-наследнике. В случае интерфейса для считывания входных данных под «наследником» подразумевается реализация для различных форматов видеозаписей. В случае интерфейса для задачи поиска и распознавания «наследником» является реализация интерфейса с использованием конкретной библиотеки распознавания лиц. Данная концепция позволяет абстрагироваться от конкретного формата входных данных и от определенной библиотеки, предназначенной для распознавания лиц.

Из множества имплементированных интерфейсов предполагается создать встраиваемую библиотеку, ориентированную на конечного пользователя. Поскольку реализация библиотеки является задачей программиста, а вопрос использования этой реализации интересует только конечного пользователя, то предполагается поставлять последнему реализацию в виде, во-первых, динамической (статической? Какой угодно?) библиотеки, закрытой для просмотра извне, и, во-вторых, двух интерфейсов, описывающих функции данной библиотеки, как, когда и в каком порядке они должны быть вызваны. Очевидно, что первым должен быть вызван конструктор той или иной имплементации интерфейса. Поскольку, как уже упоминалось, сама имплементация закрыта для конечного пользователя, а конструктор интерфейса не позволит нам сконструировать объект наследника, мы инкапсулируем конструктор интерфейса, сделав его недоступным, а для создания объекта будем использовать статическую функцию, принадлежащую интерфейсу и предназначающуюся для создания экземпляра класса наследника.

В рамках данной работы реализованы интерфейсы для формата видеороликов “.avi” и для библиотеки распознавания лиц Luxand FaceSDK.

**Доступные методы библиотеки**

Библиотека имеет 2 основных интерфейса – **Iloader** и **IFaceFinder**.

Рассмотрим интерфейс **Iloader**. Предлагается иметь нижеописанный набор функций.

**Пример использования библиотеки**

Для начала работы нам необходимо создать два объекта, один – загрузчик входного видеофайла, другой – «Искатель лиц», занимающийся поиском лиц на отдельных изображения и составлением базы:

ILoader\* loader = ILoader::createLoader();

IFaceFinder\* faceFinder = IFaceFinder::createFaceFinder();

Далее работа протекает с этими объектами. Каждый из них должен быть проинициализирован. Инициализация в данном случае – некая предварительная обработка, которая не относится к непосредственно созданию объекта. Она реализуется индивидуально, различные реализации загрузчиков, как и различные обработчики входных файлов могут иметь разные требования:

loader->init();

faceFinder->init();

Далее, загрузчику должен получить сам входной файл. Для этого вызывается соответствующая функция, в которую передается файл в виде пути к нему на жестком диске:

loader->loadFile("path/to/file");

Дальнейшая работа протекает в цикле. Загрузчик файла должен разбить видеозапись на последовательность кадров, причем, он должен иметь информацию об общем количестве кадров и о том, когда они подойдут к концу. Следовательно, нам необходимо выполнять цикл до тех пор, пока кадры не закончатся. Внутри цикла загрузчик предоставляет всю возможную информацию о кадре из видеозаписи – номер кадра; указатель на массив данных, представляющих из себя последовательность пикселей, считанных слева-направо снизу-вверх; длину и ширину картинки, а также ее глубину цвета. Все эти данные передаются в соответствующую функцию «Искателя лиц», где он уже делает свою работу – определяет лица на фотографии и создает по ним локальную базу «описаний лиц» (т.е. «описания лиц» в некотором предварительном формате – в виде указателей на объекты в памяти):

while (loader->hasFrameToRead()) {

const int curentDataFrameNumber = loader->getLastReadFrameNumber();

unsigned char\* data = loader->readNextFrame();

const int width = loader->getPictureWidth();

const int height = loader->getPictureHeight();

const int scanLine = loader->getSkanLine();

faceFinder->addImage(curentDataFrameNumber, data, width, height, scanLine, IFaceFinder::ColorDepth::Bit24);

delete[] data;

}

Внутри цикла мы обладаем возможностями внести дополнительные параметры обработки. Например, мы можем обрабатывать только каждый 10-й кадр:

if (curentDataFrameNumber % 10 != 0) {

continue;

}

После окончания обработки видеозаписи необходимо завершить работу как загрузчика, так и обработчика лиц. Во время завершения загрузки происходят действия постобработки, т.е. в локальной базе данных происходит поиск и объединение одинаковых лиц, которые, по каким-то причинам, были сохранены в разные элементы базы, а сама база из временного представления в памяти преобразуется в постоянное представление, содержащее только минимальную необходимую информацию и предназначенное для сохранения.

loader->finish();

faceFinder->finish();

Помимо вышеописанного, важным фактором является процедура «двойной верификации». При верификации лица №1 и лица №2 степень их схожести может оказаться ниже заявленного порога , определяющего принадлежность двух лиц одному и тому же человеку. Это может произойти даже если лица принадлежат одному человеку, но повернуты в разную сторону, находятся в неудачном ракурсе, освещение затемняет определенные части и т.п. Однако, существует лицо №3, степень схожести которого как с лицом №1, так и с лицом №2 является достаточно высокой для положительной верификации, из чего следует, что лицо №1 и лицо №2 тоже принадлежат одному и тому же человеку, однако, так как на момент сравнения лица №1 и лица №2 друг с другом они были определены как принадлежащие разным людям, они находятся в разных элементах базы данных. Для того, чтобы избежать подобного разделения лиц, принадлежащих одному человеку, процедура верификации проводится дважды, с двумя разными порогами и . В первый раз, с порогом – каждый кадр видеоролика проверяется на наличие на нем лиц и проводится процедура верификации всех найденные лиц со всеми элементами, уже сохраненными в базу. Если мера совпадения двух лиц и , равная значению µ(), оказывается достаточно высоким значением (то есть µ() ≥ ), то лица и считаются принадлежащими одному человеку. В случае положительной верификации лиц и происходит сравнение качества «описаний лиц» - того, что уже находится в базе и нового. Если находящееся в базе «описание лица» хуже, чем новое, то новое заменяет собой находящееся в базе. Если же находящееся в базе лучше, то сохраняется только порядковый номер кадра, на котором наше «худшее» описание лица находится. Поскольку во время первой процедуры верификации есть вышеописанная возможность разделения двух лиц, принадлежащих одному и тому же человеку на два элемента базы, то после окончания первой процедуры верификации сразу же запускается вторая процедура, в ходе которой каждый элемент базы данных сравнивается с каждым, с целью найти дополнительные совпадения. При этом, во время процедуры повторной верификации используется пороговое значение , которое, для достижение наилучшего результата, меньше значения . Составленная после процедуры повторной верификации база данных и есть финальный продукт библиотеки для текущего видеоролика.

В связи с тем фактом, что библиотек распознавания лиц на рынке очень большое количество и не существует единого стандарта, которого бы они придерживались, возникает вопрос создания универсального интерфейса, который подойдет для работы с любой из них. Это позволит нам создать решение, абстрагированное от конкретной библиотеки, или, иными словами, решение, которое подойдет для любой библиотеки. В таком случае, если по каким-то причинам мы захотим изменить используемую библиотеку распознавания лиц (например, по причине несоответствия текущей задаче или ухода с рынка старой библиотеки) мы сможем сделать это, не затрагивая само использование нашего приложения, а только переписав его внутреннюю часть.