УДК 621.39

**С.Г. ХАЧАТРЯН**

# ОБ ОДНОМ АЛГОРИТМЕ ДЕФОРМАЦИИ ИЗОБРАЖЕНИЙ, ПОЛУЧЕННЫХ С КАМЕРЫ МОБИЛЬНЫХ УСТРОЙСТВ В СИСТЕМЕ “АНДРОИД”

Предлагается алгоритм преобразования (деформации) изображений, полученных из видеокамеры или фотокамеры мобильных телефонов в системе “Андроид”, с выводом преобразованного изображения на экран в режиме реального времени. Для обеспечения качественного режима реального времени производится переход из среды JAVA в среду C/C++ (применяются Android NDK и интерфейс JNI) и обратно.

***Ключевые слова***: камера, изображение, проектирование, обработка, интерполяция.

Платформа “Android” включает в себя поддержку различных камер и функций камеры, доступных на устройствах, что позволяет захватывать изображения и видео в своих приложениях. Для получения фотографии или видеоклипа достаточно использовать существующее системное приложение камеры [1, 2].

Начиная с версии Android 5.0 поддерживается новый интерфейс программирования приложений (API – application programming interface) android.hardware.camera2 для создания качественных фотографий и их последующей обработки. Доступ к камерам в системе обеспечивается функцией getCameraIdList() с применением функции openCamera(). Для фотосъемки нужно создать CameraCaptureSession и указать объект Surface для отправки сделанных фото. Для получения уведомления при создании снимков нужно создать обработчик событий CameraCaptureSession.CaptureCallback и настроить соответствующий запрос. После выполнения запроса обработчик получит вызов onCaptureCompleted() и передаст изображение в виде объекта CaptureResult [1, 2]. Данные из камеры получаются в виде одномерного массива в формате YUV.

На рис. 1 приведена общая схема алгоритма обработки одного кадра, полученного из камеры устройства.

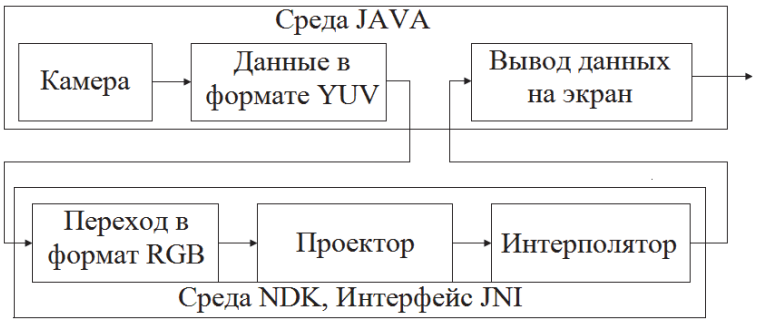


*Рис. 1. Общая схема алгоритма обработки одного кадра*

После того как получаются данные из камеры, нужно преобразовать их в формат RGB (Red, Green, Blue) для последующего вывода изображения на экран. В случае статического изображения это можно сделать на Java, но в случае видео, когда частота кадров равна 24 кадрам в секунду или более, необходимо повысить скорость обработки кадров. Поэтому лучше применять Android NDK (native development kit) – набор инструментов, которые позволяют реализовать часть программы, используя такие языки, как C/C++ [3]. При этом применяется интерфейс JNI (Java Native Interface) - стандартный механизм для запуска кода под управлением виртуальной машины Java. JNI написан на языках C/C++ или Assembler и скомпонован в виде динамических библиотек. Это даёт возможность вызова функции C/C++ из программы на Java и наоборот [3].

Из Java среды в C/C++ передается также результат работы проектора. В C/C++ происходит также интерполяция полученного кадра в формате RGB.

Для того чтобы использовать нативный код из Java, сперва следует определить нативные методы в Java классе.

Опишем создание собственного (нативного) JAVA метода. Собственный (нативный) метод создается путем добавления к его описанию спецификатора NATIVE, при этом он не должен иметь реализации. Спецификатор NATIVE сообщает компилятору, что реализация данного метода будет представлена в виде откомпилированного C/C++ кода, помещенного в библиотечный файл. Когда JVM встречает обращение к собственному методу, происходит вызов соответствующей C/C++ функции. Помимо описания собственного метода, JAVA код должен динамически загрузить библиотеку, содержащую C/C++ функцию с реализацией данного метода. Для этого в классе JAVA.LANG.SYSTEM существует метод PUBLIC STATIC VOID LOADLIBRARY (STRING LIBNAME), загружающий указанную библиотеку [3]. Следующий пример демонстрирует, как следует загружать библиотеку, объявленную в файле Android.mk, а также описание собственного метода.

```
static {
System.loadLibrary("YUVRGB-jni");
}
private native void YUVconvertRBG(int[] rgb, byte[] yuv, int width, int height);
```

Далее из соответствующего метода соответствующего JAVA класса нужно вызвать объявленную нативную функцию:

```
YUVconvertRBG(rgba, data, width, height);
```

Кратко опишем алгоритм работы проектора, который реализует центральное проецирование изображения на определенную поверхность.

Через каждую точку изображения и центр проецирования проводится прямая и определяется точка пересечения прямой с поверхностью. Полученная точка будет проекцией точки изображения на поверхность. Проектор однозначно отображает изображение на заданную поверхность. Заметим, однако, что проекции соседних точек изображения могут быть далеки друг от друга, т.е. не быть соседними. Далее полученные на поверхности проекции точек изображения параллельно проецируются на плоскость экрана.

Между проекциями соседних точек изображения могут оказаться “пустоты”, т.е. точки, которые не имеют прообраза на изображении. От пустот можно избавляться с помощью интерполяций. Однако нужно выбирать между скоростью обработки кадра (изображения) и качеством обработки. Исходя из требования высокой скорости обработки кадра, в предлагаемом алгоритме применяется линейная интерполяция. Далее проводится еще одна интерполяция для изменения размеров изображения на экране. Отметим, что для изображений с высоким разрешением можно сделать выбор в пользу скорости, так как в этом случае цвета *соседних* точек (*пикселей*) *сильно коррелированы*. В предлагаемом алгоритме две интерполяции объединены, и проводится одна интерполяция.

На рис. 2 приведен пример изображения прямоугольника и его центральной проекции на некоторую поверхность.

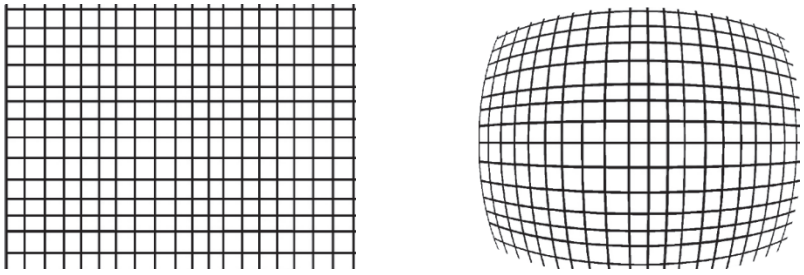

*Рис. 2. а) Оригинал “Прямоугольник” б) центральная проекция оригинала*

Приведем также примеры других деформаций изображений, не связанных с проецированием. Эти деформации определяются заданием произвольно выбранных разных отображений. На рис. 3 а – г приведены оригинал тестового изображения “Lena” и деформированные версии соответственно, полученные случайным выбором отображений и параметров деформации. Выборы определяются вложенным в алгоритме генератором случайных чисел и практически не повторяются.

а)

б)

в)

г)

*Рис. 3: а – оригинал, б, в, г - деформированные версии*

**СПИСОК ЛИТЕРАТУРЫ**

1. https://developer.android.com/guide/index.html
2. **Bill Phillips, Brian Hardy, Chris Stewart, Kristin Marsicano.** Android Programming // Big Nerd Ranch Guides. – 2015. -P. 600.
3. **Sylvain Ratabouil.** Android NDK Beginner's Guide. - Second Edition. - Packt Publishing, 2015.- 532p.

**Ս.Հ. ԽԱՉԱՏՐՅԱՆ**

## ՇԱՐԺԱԿԱՆ ՍԱՐՔԵՐԻ ՏԵՍԱԽՑԻԿԻՑ ՍՏԱՑՎՈՂ ՊԱՏԿԵՐՆԵՐԻ ՁԵՎԱՓՈԽՄԱՆ ՄԻ ԱԼԳՈՐԻԹՄԻ ՄԱՍԻՆ ԱՆԴՐՈՒԴ ՀԱՄԱԿԱՐԳՈՒՄ

Առաջարկվում է շարժական հեռախոսների տեսախցիկից կամ ֆոտոխցիկից ստացված պատկերների ձևափոխման ալգորիթմ՝ ձևափոխված պատկերի իրական ժամանակում էկրանին արտապատկերմամբ: Իրական ժամանակում բարձր որակ ապահովելու համար անցում է կատարվում Java միջավայրից C/C ++ միջավայր (կիրառվում են Android NDK և JNI ինտերֆեյսը) և հակառակը:

***Առանցքային բառեր.*** ֆոտոխցիկ, պատկեր, պրոյեկտում, մշակում, միջարկում:

**S.H. KHACHATRYAN**

## AN ALGORITHM FOR IMAGE DEFORMATION OBTAINED FROM THE CAMERA OF MOBILE DEVICES IN THE ANDROID SYSTEM

A conversion algorithm (deformation) of images obtained from the cameras of mobile telephones in the system Android with a converted image displayed on the screen in real time is proposed. To ensure high-quality real-time, a transition from the JAVA environment to the C / C ++ environment is carred out (Android NDK and JNI interface) and vice versa are applied.

***Keywords***: camera, image, design, processing, interpolation.