안드로이드 플랫폼에서 들로네 삼각망을 이용한 모핑 및 와핑 기법※

황기태
한성대학교 컴퓨터공학과
calafk@hansung.or.kr

Morphing and Warping using Delaunay Triangulation in Android Platform

Kitae Hwang
Dept. of Computer Engineering, Hansung University

요 약

고성능 스마트 폰 개발이 급속화 됨에 따라 PC 상에서 가능하였던 소프트웨어 기술을 스마트폰 상에 도입되고 있다. 본 논문은 안드로이드 플랫폼 상에서 두 개의 얼굴 이미지를 합성하는 모핑과 얼굴이미지를 사용자의 터치로 마음껏 변형하는 와핑을 구현한 오락 앱 응용프로그램을 개발한 사례를 소개한다. 본 논문은 모바일 단말기의 특성을 고려하여 간단한 LCD 터치로 제어점을 입력하고 이들로부터 들로네 삼각망 기법을 적용하여 이미지 합성 및 변환 단위를 구성하였다. 본 논문의 구현 사례는 안드로이드에서 이미지 모핑 및 와핑 기법을 활용하여 게임 등을 개발하고자 하는 경우 좋은 참고 사례가 될 것으로 예상한다.

ABSTRACT

According to rapid advent of the smartphone, software technologies which have been used on PCs are being introduced into the smartphone. This paper introduces an implementation of an application software under the Android platform using morphing and warping technology. It composes two face images with morphing technology and also transforms an original face image using warping technology for fun. For this, the image is triangulated by Delaunay Triangulation based on control points which are created by simple LCD touches on the mobile device. The implementation case of this paper will be a good reference for the development of applications like games using morphing and warping technologies over Android platforms.

Keywords : Smartphone(스마트폰), morphing(모핑), warping(와핑), Android(안드로이드)

접수일자 : 2010년 09월 13일, 일차수정 : 2010년 11월 05일, 심사완료 : 2010년 11월 23일
※ 본 연구는 한성대학교 교내 연구비를 지원받아 수행되었음

1. 서 론

최근 기존의 피처폰에 비해 높은 스크린 해상도와 빠른 처리 속도, 그리고 안정된 운영체제가 탑재된 스마트 폰 시장이 활성화되고 있다[1]. 속도와 스크린 크기의 한계 때문에, 피처폰이라고 불리는 안정적인 운영체제가 탑재되지 않는 기존의 휴대폰에서 실행되기 어려운 응용프로그램들이 스마트폰의 등장으로 모바일 단말기 상에서 가능하게 되었다.

구글에서 개발된 안드로이드 플랫폼[2]은 소스가 공개되어 있기 때문에 많은 개발자와 연구자들로부터 관심을 받고 있으며 스마트 폰 시장의 점유율을 높여가고 있다[3].

이 논문은 기존에 이미 있던 모핑(morphing)과 와핑(warping) 기술[4,5,6,7]을 활용하여 안드로이드 플랫폼을 탑재한 스마트폰에서 얼굴 합성과 변형을 실행하는 소프트웨어의 구현 사례를 소개한다. 이미지 처리의 핵심 응용 기술인 모핑은 하나의 영상에서 다른 영상으로 변화시키는 기술이며, 와핑은 얼굴에서 코를 길게 하거나 볼을 뚱뚱하게 하는 등 영상을 일그러뜨리는 기술로서 처음에서 일그러진 영상을 복구하는데 사용된 기술이었다.

모핑과 와핑 처리를 위해, 본 연구에서는 간단한 LCD 터치를 통해 제어점을 입력하고 들로네 삼각형(Delaunay Triangle)을 구성하고 이를 기반으로 어파인 변환(affine transformation)하는 방식을 사용하였다. 두 개의 얼굴을 합성할 때 가중치를 두어 10 단계로 한 얼굴에서 다른 얼굴로 변해가는 모핑이 가능하게 하였다. 사람의 얼굴과 원숭이의 얼굴을 합성하면 원숭이를 닮은 얼굴로 변형시킬 수 있다. 또한 사용자가 한 번의 LCD 터치를 통해 얼굴의 일부분이나 전체를 변형시킬 수 있도록 하였다.

본 연구는 인텔에서 개발하여 오픈 소스로 만든 OpenCV 라이브러리[8]를 부분적으로 이용하였다.

2장에서는 구현 사례를 이해하기 위한 기본 기술을 설명하고, 3장에서는 설계 내용을, 4장에서는 구현 내용과 시연을 기술하고 5장에서 결론을 맺는다.

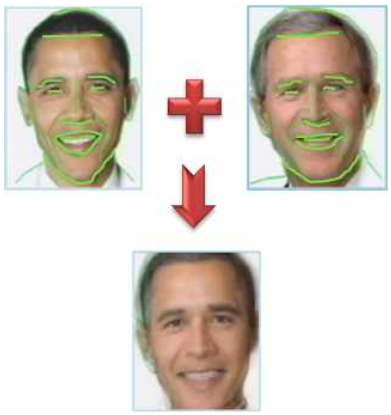
2. 관련 연구

2.1 이미지 처리 알고리즘

모핑과 와핑의 기반이 되는 기존 알고리즘을 소개한다.

2.1.1 특징 기반 모핑(Feature-Based Image Metamorphosis) 알고리즘

이 방법[4]은 삼각형이나 사각형을 사용하지 않고 [그림 1]과 같이 이미지 상에 그려지는 제어선이라고 불리는 섬세한 선을 사용한다. 입력 영상에 그려진 제어선의 각 픽셀을 목적 영상 상에 동일하게 그리며 그 다음 픽셀들은 제어선들을 참고하여 목적 영상에 정합된다.



[그림 1] 특징 기반 모핑 사례

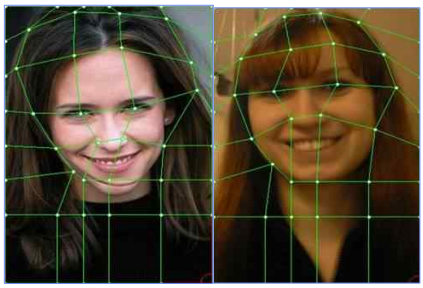
2.1.2 메시 와프(mesh warp) 알고리즘

메시 와프 알고리즘[5]은 [그림 2]와 같이 두 이미지 상에 동일한 개수의 x축과 y축으로 구성된 배열로 된 제어점(사각형 메시)을 가진다. 두 이미

지의 배열간의 보간법에 의해 새로운 x축 제어점이 탄생되는데 이때 보간법은 선형보간법에서 스플라인 함수까지 다양하다.

이렇게 새롭게 만들어야 할 제어점을 찾고 나서 새로운 영상의 y축을 재추출 한다. 이때 사용하는 알고리즘은 팬트(fant) 알고리즘을 사용한다.

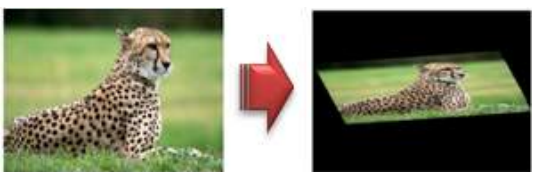
메시 와프 알고리즘은 PC 등의 데스크톱에서는 마우스를 이용하여 점들을 정교하게 대응시킬 수 있으므로 모핑의 효과가 뛰어나지만 기본적으로 많은 점을 찍고 이들을 움직여야하는 불편함이 존재할 수 있다.



[그림 2] 메시 와프 사례

2.1.3 어파인(affine) 변환

어파인 변환[9]이란 선형 변환과 이동 변환의 조합으로서 라인은 라인으로 맵핑되지만 투영 변환(projection transformation)과는 달리 평행선은 항상 평행을 유지하며 변환 비율 역시 보존되는 특징을 가진다. [그림 3]은 사각형 이미지를 평형사변형으로 어파인 변환한 경우이다. 사각형은 삼각형에 꼭지점이 하나 더 있는 경우이므로 어파인 변환으로 삼각형을 다른 형태의 삼각형으로 변환할 수 있다.



[그림 3] affine 변환 사례

2.1.4 이미지 변환과 들로네 삼각망

들로네 삼각망(Delaunay Triangulation : DT)은 평면상에서 주어진 제어점들로 구성되는 삼각형들의 독특한 망으로서, DT 내부의 어떤 삼각형의 경우에도 그 외접원 내 다른 제어점을 가지지 않는 삼각형을 구성된다. 들로네 삼각망은 컴퓨터의 많은 응용분야에서 적용되고 있지만 특히 컴퓨터 그래픽이나 비전 분야에서도 많이 사용되고 있다.

Xin[10]은 시간에 따라 제어점이 삽입되거나 삭제되는 등의 변화가 심한 이미지를 보다 자연스럽게 화면 출력하기 위해 이미지를 구성하고 들로네 삼각형에 제어점이 삽입되거나 삭제될 때 제어점에 변화를 줌으로써 몰핑 등의 이미지 변환이 매우 자연스럽게 하는 방법을 제안하였다. Yi[11]는 얼굴 인식을 위해 들로네 삼각망을 사용하였으며, Zi-Cai[12]는 최소 1000개 이상의 매우 작고 많은 삼각형들로 들로네 삼각망을 구성하여 얼굴 인식을 시도하였다. 그러나 현재 들로네 삼각망을 직접적으로 몰핑 및 와핑에 사용한 사례는 발견하지 못하였으며, 이러한 기존 연구들은 처리 속도가 빠른 컴퓨터를 대상으로 조밀하고 많은 들로네 삼각망을 정교하게 구성하고 있기 때문에 즐기기 위해 간단히 몇 개의 제어점을 입력할 수 밖에 없는 모바일 단말기 응용에 사용하기 힘들다.

2.2 OpenCV

OpenCV(Open source Computer Vision)란 1999년부터 인텔에서 시작된 프로젝트로서 실시간 이미지 처리를 위해 만든 공개 라이브러리이다. 현재 2.0 버전까지 개발되었으며, FreeBSD, 윈도우, 리눅스, 맥 운영체제 등에서 작동하며 C, C++ 버전을 제공한다.

3. 안드로이드 플랫폼에서의 모핑 및 와핑 시스템

3.1 기능 정의

본 논문에서 구현된 응용프로그램의 기능은 크게 3 가지로서 다음과 같다.

3.1.1 두 얼굴 이미지 합성

두 개의 얼굴 이미지를 합성하는 기능이다. 두 개의 얼굴 A, B 이미지에 대해 A에서 B까지 변하도록 모핑을 실행하며 A 얼굴에서 B 얼굴까지 11 단계의 모핑된 이미지를 만들어 낸다. 또한 총 11 개의 모핑된 이미지를 애니메이션하여 A 얼굴에서 B 얼굴로 변하는 애니메이션을 구현한다.

3.1.2 사용자 터치를 이용한 얼굴 이미지 변형

사용자가 임의의 얼굴 이미지를 로드하고 얼굴상의 임의의 점을 LCD 터치를 통해 다른 점으로 이동시키면 그에 따라 자연스럽게 이미지를 변형시켜 재미를 만들어 낸다.

본 논문에서는 LCD 터치로 간단히 모핑과 와핑을 위한 제어점을 입력한다.

3.1.3 부가 기능

카메라로 사진 찍기, 앨범으로부터 얼굴 이미지 로딩, 이미지의 확대 및 축소, 이미지 트리밍 등의 기능을 구현하여 모핑 및 와핑 작업을 하고자 하는 사용자에게 편의를 준다. 또한 모핑되거나 와핑되어 변형된 이미지의 저장할 수 있도록 구현한다.

3.2 모핑 알고리즘

본 논문에서는 사용자가 제어점을 비교적 쉽게 설정하고 설정된 제어점을 기반으로 삼각형을 구성하여 어파인 변환을 실시하는 방법을 사용하였다. 사용된 모핑 알고리즘은 크게 4 단계로 다음과 같이 작동한다.

단계 1) 제어점 결정

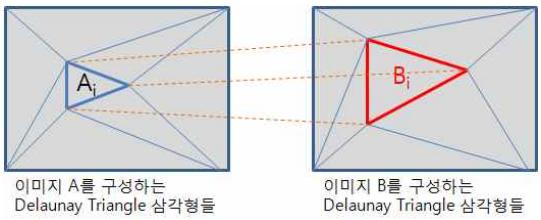
[그림 4]와 같이 각 삼각형을 구성하는 꼭지점을 제어점이라고 한다. 제어점은 모핑이나 와핑 시 어파인 변환을 하기 위해 구성되는 사각형(혹은 삼각형)의 꼭지점이다. 제어점은 터치를 통해 사용자가 입력한다. 이미지의 4 모서리 점은 디폴트 제어점이다.

단계 2) 두 이미지 사이의 제어점 대응

모핑시키고자 두 이미지 A, B에 대해서 하나의 이미지를 선택한다. 선택된 이미지에 대해 단계 1)의 방법으로 제어점을 찍으면 다른 이미지의 동일한 픽셀 위치에 대응하는 제어점을 자동으로 설정한다. 그리고 제어점 입력이 끝나면 각 이미지에 대해 제어점의 위치를 자유롭게 이동시킬 수 있다.

단계 3) 들로네 삼각형 구성

[그림 4]와 같이 사용자가 입력한 제어점을 연결하여 이미지를 삼각형의 집합으로 구성한다. 이 삼각형 구성하기 위해 1934년에 러시아의 Boris Delaunay에 의해 정립된 들로네 삼각망 알고리즘을 사용하며 이 삼각형을 들로네 삼각형(Delaunay Triangle)이라고 부른다. 본 논문에서는 간단히 삼각형이라고 부른다.



[그림 4] 제어점과 들로네 삼각형

단계 4) 가중치 기반 affine 변환

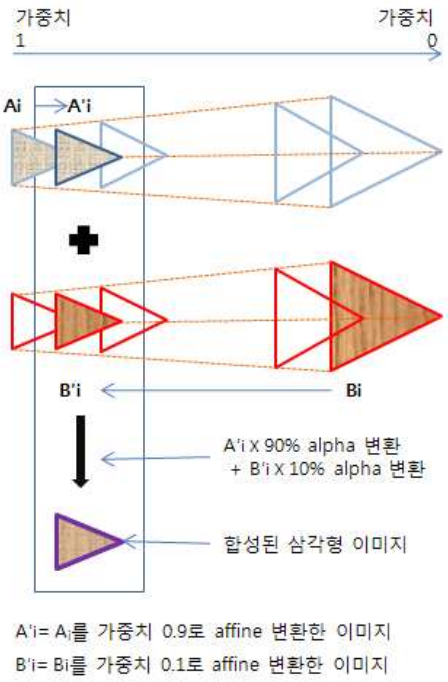
이미지 A와 이미지 B를 합성하기 위해 이미지 A의 각 삼각형과 이미지 B의 각 대응하는 삼각형을 같은 크기로 맞추어서 합성한다. [그림 4]에서 삼각형 Ai는 삼각형 Bi에 대응하므로, 삼각형 Ai를 삼각형 Bi의 크기로 변형하면 Ai의 원본 모양

이 틀어지게 되므로 합성된 결과는 Bi의 원본 이미지의 모양이 강하게 나타난다. 그 반대로 Bi를 Ai의 크기로 맞추어 변형하면 Ai의 원본 이미지가 강하게 나타나게 된다. 그러므로 본 연구에서는 가중치를 이용하여 이를 조절한다. 가중치를 0에서 1까지 변화시키며 가중치가 만일 1이면 합성된 이미지는 A 자체이며, 가중치가 0이면 합성된 이미지는 B 자체이다. 만일 가중치가 0.9이면 Ai의 크기와 모양에 90% 가깝고, Bi의 크기와 모양이 10% 반영되는 크기와 모양이 된다. 가중치를 적용하면 [그림 5]와 같다.

가중치를 기반으로 A의 각 삼각형을 모두 어파인 변환하여 변환된 A′이미지를 만들고, 가중치를 기반으로 B의 이미지를 어파인 변환하여 B′이미지를 만든다.

단계 5) 이미지 합성

가중치 값은 A′와 B′를 합성할 때 불투명도인 알파(alpha) 값으로 반영한다.

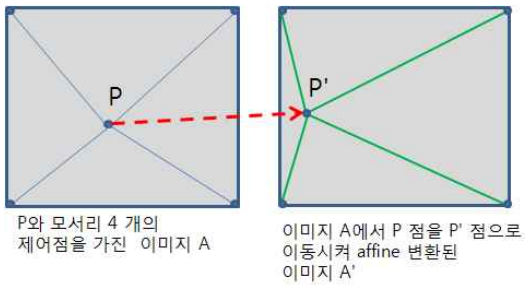


[그림 5] 가중치와 이미지 합성

즉 가중치가 높으면 A′의 불투명도를 높여 B′의 이미지가 보다 선명하게 합성되도록 한다. 0에서 1까지 가중치를 11단계로 구성하여 합성된 이미지는 총 11 개 만들어 진다. 이들을 순서대로 화면에 출력하면 A이미지에서 B 이미지로 모핑 결과가 애니메이션된다.

3.3 와핑 알고리즘

와핑은 모핑 과정의 일부분이며 대상 이미지가 하나이다. 와핑이란 이미지에서 임의의 점을 제어점으로 선택하고 제어점을 이동시킨 새로운 이미지를 만들어내는 것을 말한다. 와핑은 [그림 6]과 같이 사용자가 P 점을 제어점으로 선택하고 LCD 터치를 이용하여 P′ 점으로 이동시키면 이에 따라 이미지 A를 구성하는 삼각형의 모양이 변한다. P 제어점을 가진 원본 이미지 A의 각 삼각형을 어파인 변환하여 P′제어점을 가진 새로운 이미지 A′를 만들어낸다.



[그림 6] 제어점 P를 P'로 이동시킨 와핑

단계 1) 제어점 결정

사용자로부터 오직 하나의 제어점을 입력받는다.

단계 2) 들로네 삼각형 구성

입력된 제어점 P에 따라 들로네 삼각형을 구성한다. 4 개의 삼각형이 생긴다.

단계 3) 어파인 변환

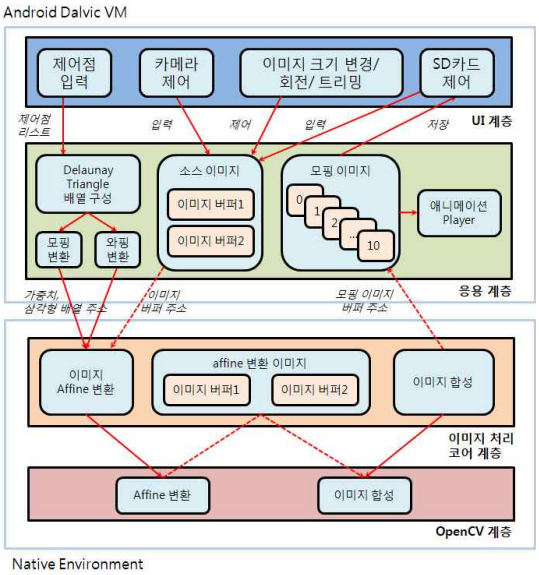
4 개의 삼각형에 대해서 각각 어파인 변환을 수행한다.

단계 4) 이미지 합성

4 개의 어파인 변환된 삼각형 이미지를 합쳐 와핑된 이미지를 완성한다.

3.4 소프트웨어 구성

본 논문에서 구현된 소프트웨어는 [그림 7]과 같이 크게 4 개의 계층으로 설계되었다. 상위 2 계층인 UI 계층과 응용 계층은 안드로이드 API를 이용하여 자바로 구현하고, 아래의 2 계층은 리눅스 API를 이용하는 C++ native 코드로 구현된다. 최하 계층은 OpenCV를 안드로이드 플랫폼에서 실행될 수 있도록 포팅한 C++ 코드를 컴파일하여 사용한다.



[그림 7] 소프트웨어의 구성

3.5 UI 계층

UI 계층의 목적은 터치를 통해 LCD로부터 사용자 입력을 받기 위한 것이다. 사용자가 카메라와 앨범으로부터 이미지를 선택하도록 하며, 선택된 이미지의 크기를 조절하고 트리밍할 수 있는 도구를 개발한다. 이 기능은 안드로이드 API를 이용하여 구현한다.

UI 계층에서 제일 중요한 기능이 제어점을 입력하는 것이다. 두 원본 이미지에 대해 제어점들을 설정하며 제어점들과 원본 이미지는 응용 계층 공간 상의 버퍼에 저장된다.

3.6 응용 계층

3.6.1 들로네 삼각형 배열 구성

[그림 4]와 같이 두 개의 원본 이미지에 대해 각각 들로네 삼각형을 구성한다. 삼각형 정보는 배열로 저장한다.

3.6.2 모핑 및 와핑

모핑을 위해서는 가중치 값을 0~1까지 0.1씩 증가시키면서 11번 이미지 처리 코어 계층의 이미지 어파인 변환 모듈을 호출한다. 이때 삼각형의 배열 주소, 가중치 값, 원본 이미지 버퍼의 주소를 전달한다. 와핑의 경우에는 한 번 어파인 변환 모듈을 호출한다.

3.6.3 애니메이션 player

총 11 개의 모핑된 이미지가 만들어지면 이들을 단순히 순서대로 화면에 그린다. 사용자는 이미지 A에서 이미지 B로 자연스러운 모핑을 보게 된다.

3.7 이미지 처리 코어 계층

3.7.1 이미지 어파인 변환

어파인 변환 알고리즘을 이용하여 이미지를 구성하는 삼각형 배열에서 각 삼각형을 가중치를 기반으로 크기와 모양을 변형시킨 삼각형을 만들어 낸다. 그리고 이들을 합쳐 하나의 이미지로 구성하고 어파인 변환된 이미지 버퍼 1과 2에 각각 저장한다.

이를 위해 OpenCV에 구현된 기본적인 어파인

변환 함수를 호출한다.

3.7.1 이미지 합성

어파인 변환된 이미지 버퍼 1과 2에 각각 저장된 이미지를 합성한다. 이때 가중치를 투명도 값인 알파 값으로 반영하며 OpenCV에 구현된 이미지 합성 함수를 호출한다.

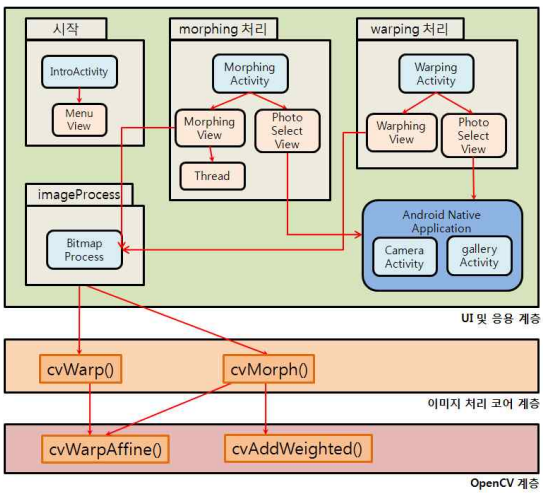
3.8 OpenCV 계층

OpenCV는 기본적으로 어파인 변환 모듈과 이미지 합성 모듈을 가지고 있으며 본 연구에서 이 두 기능을 사용한다.

4. 구 현

4.1 클래스 및 함수 구현

본 논문에서 구현한 소프트웨어의 실제 구현된 모듈은 [그림 8]과 같다.



[그림 8] 시스템 구현

UI 계층과 응용 계층을 합쳐서 안드로이드 API를 이용하여 자바로 구현하였으며, 이미지 처리 코어 계층은 C++로 구현하고 안드로이드의 NDK를 이용하여 응용 계층의 자바 코드와 연결하였다. OpenCV는 안드로이드에서 사용할 수 있는 C++ 버전의 소스를 컴파일하여 사용하였다.

안드로이드에서 Activity는 자체적인 생명 주기를 가진 응용프로그램 구성 기본 단위이며 하나의 응용프로그램은 여러 개의 Activity를 가진다. View는 GUI 화면 출력을 위한 필요한 기본 틀이다. Activity는 화면에 출력하기 위해 필요한 만큼 View를 가진다. UI 및 응용 계층에 구현된 자바의 주요 클래스는 다음과 같다.

① class BitmapProcess
11번 모핑 진행하고 와핑을 처리하는 클래스
② class MorphingActivity extends Activity
모핑을 위한 Activity
③ class PhotoselectView extends View
이미지 선택 뷰
④ class MorphingView extends View
모핑 처리 뷰
⑤ class MorphingView implements Runnable, OnClickListener
모핑을 수행하는 Thread
⑥ class WarpingActivity extends Activity
와핑을 위한 Activity
⑦ class WarpingView implements OnClickListener
와핑을 제어. 와핑 결과를 보여줌

다음은 C++로 구현된 이미지 처리 코어 계층의 두 함수의 원형을 보여준다.

① JNIEXPORT jintArray JNICALL cvMorph(JNIEnv *env, jclass cls, jintArray src_data, jintArray dst_data, jint src_width, jint src_height, jintArray src_triangle, jintArray dst_triangle, jintArray src_triangle2, jintArray dst_triangle2, jfloat weight)

두 이미지 버퍼와 삼각형 배열 정보를 받아 모

핑 수행. 수행 결과의 이미지 버퍼는 안드로이드 Dalvic VM 환경에 생성. 결과로 버퍼 주소 반환

② JNIEXPORT jintArray JNICALL cvWarp(JNIEnv *env, jclass cls, jintArray src_data, jint src_width, jint src_height, jintArray src_triangle, jintArray dst_triangle)

이미지 버퍼와 삼각형 배열 정보를 받아 와핑 수행. 수행 결과의 이미지 버퍼는 안드로이드 Dalvic VM 환경에 생성. 결과로 버퍼 주소 반환

4.2 모핑 구현 결과

[그림 9]는 구현된 응용프로그램에서 각 이미지에 총 15개의 제어점을 찍은 모습이다. 앞서 모핑 알고리즘의 단계 2)에서 설명한 바와 같이 왼쪽 그림에 제어점을 찍으면 자동으로 오른쪽 이미지의 동일한 위치에 대응하는 제어점이 찍히게 된다. 제어점 입력 후 오른쪽 이미지의 제어점을 이동시켰다.

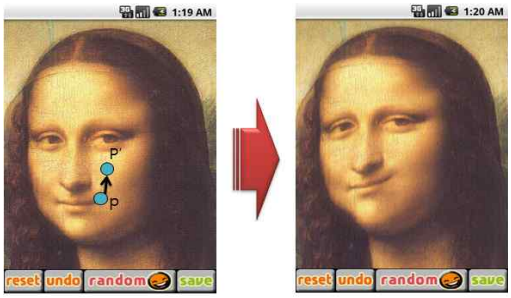
[그림 9] 두 이미지에 제어점이 설정된 화면

메뉴를 실행하면 제어점을 이용하여 11개의 모핑된 이미지가 생성된다. 그리고 이들을 애니메이션 player를 이용하여 재생하면 [그림 10]과 같이 재생된다.

[그림 10] 모핑 애니메이션

4.3 와핑 구현 결과

와핑은 한 개의 이미지를 대상으로 변형한다. 이미지를 로딩하고 LCD 상에 터치로 누르면 제어점(P)이 설정되며 터치로 누른 상태에서 임의의 위치로 드래깅하여 놓으면 마지막 위치가 이동된 제어점(P′)이 된다. 그 결과 [그림 11]과 같이 이미지가 변형된다.



제어점 P에서 P'로 이동된 와핑

[그림 11] 와핑에 의해 변형된 이미지

4.4 성능 평가

4.4.1 성능 평가 지수

모핑과 와핑은 많은 연산을 수반하기 때문에 처리 시간이 중요한 기준이 된다. 그러므로 본 연구에서 성능 평가 지수를 모핑과 와핑에 따른 처리

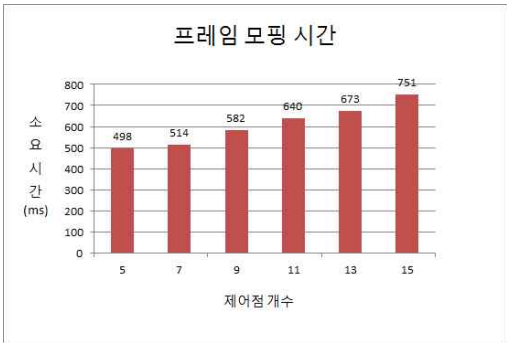
시간으로 설정한다. 이 시간은 제어점의 개수와 이미지의 크기에 주로 의존한다. 그러나 본 논문에서는 모핑과 와핑 전 이미지를 일정한 크기 740x480로 조절하기 때문에 사실상 성능에 가장 영향을 미치는 요소는 제어점의 개수이다. 그러므로 제어점의 개수에 따라 소요되는 모핑과 와핑 시간을 측정 평가하였다. [표 1]은 이미지 모핑 및 와핑을 구현한 Nexus One 단말기의 사양이다.

[표 1] 구현과 성능평가에 사용된 단말기, HTC Nexus One

시스템	프로세서	퀄컴 스냅드래곤 QSD8250 1GHz
	운영체제	안드로이드 2.2
디스플레이	크기	3.7인치
	해상도	WVGA (800 x 480)
	종류	SLCD

4.4.2 모핑 성능 평가 결과

모핑에 사용되는 제어점은 4 모서리에 하나씩 기본적으로 4개가 주어진다. 그러므로 1개 이상의 추가적인 제어점이 설정되어야 비로소 모핑의 의미가 있기 때문에 본 연구에서는 제어점의 개수를 5, 7, 9, 11, 13, 15개로 변화시키면서 시간을 측정하였다. 실험 결과는 10번 실측한 값의 평균을 이용하였다. 실험 결과 두 이미지 프레임을 모핑 처리하여 하나의 이미지 프레임을 만드는데 걸리는 시간은 [그림 12]과 같이 측정되었다. 제어점이 많으면 많을수록 더 정교한 모핑 효과를 낼 수 있지만 모바일 단말기에서 많은 제어점을 설정할 가능성은 낮아 보인다. 15개의 제어점을 설정한다고 했을 때 11개의 프레임을 만드는데 걸리는 시간은 약 8.3초 내외로서 그리 긴 시간은 아니다. 본 구현은 실시간 처리가 목적이 아니며 오락용이므로 이 시간은 충분한 시간이라고 판단된다.



[그림 12] 모핑 처리하여 한 프레임을 생성하는데 걸리는 시간

4.4.3 와핑 성능 평가 결과

와핑은 총 5개의 제어점에 대해서 이루어지며 [그림 11]에서 사용자가 제어점 P를 터치하여 P′로 옮기고 손을 떼는 시점에서 [그림 11]의 오른쪽 프레임이 완성되는 데까지의 시간을 측정하였다. 총 10번 실험한 값의 평균을 결과로 사용하였다. 이 결과 와핑 소요 시간은 약 239ms로 측정되었다. 이 시간은 사용자가 와핑을 시도하면 거의 즉각적으로 화면에 와핑 출력 효과를 얻을 수 있는 시간이다.

5. 결 론

본 논문은 새로운 이론 연구가 아니며, 안드로이드 플랫폼 상에서 모핑 및 와핑 기법을 이용한 얼굴 합성 및 변형 오락 응용프로그램을 개발한 사례를 설명하였다. 본 논문에서는 사용자가 LCD 상에서 터치로 간단히 직접 제어점을 찍고 터치로 제어점을 간단히 이동하여 원하는 제어점을 설정하면 제어점들을 연결하여 전체 이미지를 삼각형들의 연결로 재구성하는 들로네 삼각망을 활용하고 가중치를 두어 두 이미지의 참여도를 변화시키면서 어파인 변환을 실행하였다.

모핑 시간을 측정한 결과 740x480 크기의 이미지에 대해 15개의 제어점의 경우 한 프레임을 완

성하는데 약 0.75초 경과하였으며 11 개의 연속된 모핑 이미지는 8.3초 내외가 걸렸으며, 이미지 와핑은 약 0.239초로서 사용자가 터치를 통해 와핑을 시도하는 즉시 그 결과가 화면에 출력되었다.

본 연구 사례와 과정은 이미지 모핑 및 와핑을 이용하여 안드로이드 상에서 게임이나 오락 등의 응용프로그램을 개발하고자 하는 경우 좋은 참고 자료가 될 것으로 기대된다.

참고문헌

[1] 이환용, “스마트폰의 어플리케이션 미디어 플랫폼 동향”, 정보과학회지 28권 5호, pp. 40-47, 2010년 6월

[2] http://www.android.com

[3] 장윤정, 김철우, “스마트폰 시장의 진화와 안드로이드의 영향”, 정보과학회지 28권 5호, pp. 48-55, 2010년 6월

[4] Thaddeus Beier, Shawn Neely, “Feature-Based Image Metamorphosis”, ACM SIGGRAPH '92 Chicago, July pp.26-31, 1992

[5] Douglas Smythe, “A Two-Pass Mesh Warping Implementation of Morphing”, D. Dobb's Journal, No. 202, July 1993

[6] Wolberg, George, Digital Image Warping, IEEE Computer Society Press, Los Alamitos, CA, 1990

[7] Wolberg, G., “Digital Image Warping”, IEEE Computer Society Press, 1990

[8] http://www.opencv.org

[9] Randy Crane, A Simplified Approach To Image Processing, Prentice Hall, 1997

[10] Xin Liu; Rokne, J.G.; Gavrilova, M.L.; , “Smooth Morphing Delaunay Triangulation”, Voronoi Diagrams in Science and Engineering (ISVD), 2010 International Symposium on, pp.77-84, 28-30 June 2010

[11] Yi Xiao1 and Hong Yan1, “Facial Feature Location with Delaunay Triangulation/ Voronoi Diagram Calculation”, Proceedings of the Pan-Sydney area workshop on Visual information processing, pp.103-108, 2001

[12] Li, Z.-C.; Chiang, J. Y.; Suen, C. Y.; , “Face Transformation With Harmonic Models by the Finite-Volume Method With Delaunay Triangulation”, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol.PP, no.99, pp.1-12, 0

황 기 태 (Kitae Hwang)

1982 서울대학교 컴퓨터공학과 학사
1986 서울대학교 컴퓨터공학과 석사
1988 서울대학교 컴퓨터공학과 박사
1993 IBM Watson 방문연구원
1994 한성대학교 교수
2000-2001 University of California, Irvine 방문교수

관심분야 : 모바일 & 유비쿼터스 컴퓨팅, 게임 개발, 콘텐트 스트리밍