http://dx.doi.org/10.7236/JIIBC.2013.13.5.135
JIIBC 2013-5-16

화상 이미지내의 피사체 크기 계측을 위한 스마트 폰 어플 설계

Design of a Smart Phone App for Measuring Object Size in a Picture Image

오선진*

Sun-Jin Oh

요 약 하드웨어와 부품 소재 산업의 급속한 발전으로 휴대 단말의 진화가 거듭되고 있으며, 모바일 폰을 이용한 다양한 응용 기술들이 최근 스마트 응용의 주류를 이루고 있다. 특히 스마트 기기의 카메라 모듈을 이용한 흥미로운 어플들이 지속적으로 개발되고 있어 다양한 모바일 응용의 사용이 가능하게 되었다. 본 논문에서는 스마트 기기의 카메라 모듈로 촬영한 화상 이미지내의 특정 피사체의 실물 크기를 비교적 정확하게 계측하기 위한 스마트 폰 어플을 설계하고 구현하였다. 이때 피사체의 정확한 계측을 위해 촬영 상황에 따라 4가지 계측 모드로 구현하였고, 스마트 폰 기종이나 시스템 SW에 따라 촬영 결과 화상 이미지의 차이를 최소화할 수 있도록 0점 조절 모듈을 두어 조정하였다. 본 논문에서 구현한 어플의 계측 정확도는 모의실험을 통해 비교 분석하였다.

Abstract With a rapid growth of hardware and product material industries, evolution of handheld terminal technologies is progressed. Various applications using mobile phone are the major trend in recent smart applications. Especially, interesting applications using camera module in a smartphone are developed continuously, mobile users are able to use various useful mobile applications nowadays. In this paper, we design and implement smartphone app for measuring real size of the shooting object in the image taken by a camera module in a smartphone precisely. We implement 4 different shooting modes for precise measurement of objects and 0 point control module is developed to minimize the differences of shooting images taken by different types and systems of various smartphones. The smartphone app proposed and implemented in this paper is analyzed and evaluated by a simulation study.

Key Words : Smart measuring app, Android, Smartphone Camera

Ⅰ. 서 론

스마트 폰 기술의 급속한 발전과 보급에 힘입어, 스마트 폰을 이용한 다양한 어플들의 개발과 확산이 최근 스마트 응용의 주류를 이루고 있다. 이러한 스마트 응용들은 단순한 폰 기능에서부터 컴퓨팅 응용분야와 융합한 다양한 형태로 지속적으로 확대 발전되고 있으며, 최근에는 의학 분야나 여러 통신채널과 결합한 USN, 블루투스 기반 애드 혹 망 응용분야에 이르기까지 다양해지고 있고 앞으로도 그 발전 가능성은 무궁무진하다.[1] 특히

*종신회원, 세명대학교 정보통신학부
접수일자 : 2013년 7월 31일, 수정완료 : 2013년 9월 13일
게재확정일자 : 2013년 10월 11일

Received: 31 July, 2013 / Revised: 13 September, 2013 / Accepted: 11 October, 2013
**Corresponding Author: sjoh@semyung.ac.kr
Dept. of Computer & Information Science, Semyung University, Korea

컴퓨팅과 통신 기능이 융합한 응용분야나 스마트 폰에 장착된 주변 기기나 모듈들을 이용한 융합 어플, 그리고 취향이나 관심사가 같은 사람들의 사회적 모임을 연결해 주는 소셜 네트워크 관련 어플, 지리적 특징을 고려한 LBS같은 위치정보기반 어플, 개성이 중시되는 최근의 사회 풍조에 발맞춰 개인 라이프스타일이나 취향에 맞는 맞춤형 어플에 이르기까지 주로 실생활의 편이성과 효율성을 강조하는 기능성 어플들이 주류를 이루고 있다.[2] 또한 다양한 맞춤형 어플의 개발을 용이하게 할 수 있도록 인터페이스 역할을 하는 미들웨어에 대한 연구와 개발 또한 본격화 되고 있다.[3] 미들웨어는 비록 스마트 응용 개발 플랫폼 환경이나 휴대 단말의 사양이 다르더라도 약간의 수정 · 보완만으로 용이하게 스마트 어플을 개발할 수 있도록 하며, 최근 다양한 환경 하에서의 미들웨어의 설계와 연구가 활발하게 이루어지고 있다.[4]

본 논문에서는 안드로이드 기반 스마트 기기의 카메라 모듈을 이용하여 촬영되는 화상 이미지 내의 특정 피사체의 실물 크기를 비교적 정확하게 측정할 수 있는 피사체 크기 계측 앱을 설계하고 구현하였다. 피사체를 촬영하는 스마트 폰의 기종에 따라 대상 피사체의 결과 이미지에 다양한 영향을 미치는 특성[5]이 있으므로 계측 결과의 정확도와 신뢰도를 향상시키기 위해 높이와 넓이 측정모듈, 지정거리 측정모듈, 그리고 일반거리 측정모듈 등 4가지 계측 모드로 촬영 상황에 따라 적용할 수 있도록 설계하였으며, 특별히 촬영에 사용하는 스마트 폰 기종이나 시스템의 버전 그리고 카메라 모듈의 특성 및 액정화면의 해상도에 따라 결과 이미지에 차이가 발생하여 계측 정확도에 영향을 주게 되므로[5,6] 이를 조정하기 위한 0점 조절 모듈을 추가하였다. 본 논문에서 구현한 피사체 계측 어플은 모의실험을 통해 계측 정확도를 비교 분석하였다.

본 논문의 구성은 다음과 같다. 2장에서는 피사체 크기 계측 어플의 시스템 모델을 알아보고, 3장에서는 구성 모듈의 주요 동작과 알고리즘을 서술하였으며, 4장에서는 제안한 계측 앱의 모듈별 구현 결과와 모의실험을 통한 피사체 계측 정확도를 분석한 결과를 고찰하였고, 마지막으로, 5장에서 향후 연구과제와 함께 결론을 맺는다.

II. 시스템 모델

최근 스마트 응용 기술의 급속한 발전과 보급으로 많은 사용자들의 스마트 폰 응용에 관심과 참여가 이루어지고 있다. 본 논문에서 설계하고 구현하는 스마트 폰 카메라 모듈을 이용한 화상 이미지 내의 특정 피사체 크기 계측을 위한 스마트 폰 앱의 시스템 모델은 모바일 컴퓨팅 환경에서 스마트 폰이나 태블릿 PC에 내장된 카메라 모듈과 정전식 멀티 터치가 가능한 터치스크린을 기반으로, 스마트 기기의 컴퓨팅 기능을 이용한 융합 어플이다. 이 앱은 구글의 안드로이드 OS를 기반으로 하는 스마트 기기에 내장된 카메라 모듈을 통해 촬영된 화상 이미지 내에 특정 피사체의 길이와 피사체까지의 거리를 계측하는 어플로 MTB 등 다양한 분야에서 보조 계측장비로 이용될 수 있다. 어플 개발을 위한 소프트웨어 개발 툴로 이클립스 Juno Service Release 1을 사용하였고, 프로그래밍 언어는 최근 버전의 자바 JDK 1.7.0_25를 사용하였다. 여기에 안드로이드 API 함수 사용을 위해 Android SDK 4.2 (Jelly Bean)을 사용하였고 자바와 연동하기 위해 Android ADT를 설치하였으며, 구현한 앱의 실행 결과 확인을 위해 AVD 16-wvga.avd 에뮬레이터를 설치하여 사용하였다. 이 앱 개발을 위한 하드웨어 플랫폼으로는 3세대 인텔 코어 i5가 장착된 삼성 노트북에 32bit 윈도우 7 운영체제 하에서 웹 접속을 위한 Tomcat 7.0과 데이터베이스로 MySQL 5.5를 설치하여 사용하였다. 표 1은 본 논문에서 사용한 하드웨어와 소프트웨어 플랫폼 개발 환경을 보여준다.

표 1. 스마트 폰 어플 개발 플랫폼
Table 1. Development platform of a smartphone app.

Software Platform	
Dev. Tool	Eclipse Java Juno Service Release 1
언어	JAVA JDK 1.7.0_25
Plug in	Android ADT
SDK	Android 4.2 (Jelly Bean)
Emulator	AVD16-wvga.avd
Hardware Platform	
PC	3세대 Intel Core i5 Pr. Notebook
OS	Windows 7 - Home Premium Edition
웹 컨테이너	Tomcat 7.0
Database	MySQL 5.5

스마트 폰의 카메라 모듈을 이용한 화상 이미지 내의 정확한 피사체 계측을 위해 우선 시중에 시판 중인 스마

트 폰 카메라 모듈의 기종이나 시스템 버전에 따라 계측 결과에 영향을 미칠 수 있는 요소들을 배제하고, 스마트 기기에 장착된 카메라 모듈과 액정 화면의 해상도 등의 영향을 최소화하기 위해 대상 이미지를 촬영하기 전에 스마트 폰 카메라 모듈에 대한 0점 조절을 할 수 있는 모듈을 두어 스마트 기기상의 특성을 최소화 하였다. 또한 대상 이미지를 촬영하는 상황에 따라 촬영 결과에 영향을 줄 수 있고 이로 인해 계측 정확도가 저하되는 것을 방지하기 위해 각 촬영 여건과 상황에 따라 적용할 수 있도록 4가지의 계측 모듈로 구분하여 설계하였다. 그림 1은 본 논문에서 설계하고 구현한 스마트 폰의 카메라 모듈을 이용한 화상 이미지 내의 특정 피사체 크기 계측을 위한 어플의 주요 구성 모듈을 보여준다. 그림에서 보인 바와 같이, 앱 전체에서의 기기 특성 조절을 위한 0점 조절 모듈, 피사체 세로 높이위주 계측을 위한 높이 계측 모듈, 피사체의 좌우 폭을 주로 계측하기 위한 넓이 계측 모듈, 일정하게 정해진 거리에서 대상 피사체를 촬영하고 그 크기를 계측하기 위한 지정거리 계측 모듈, 그리고 아무런 거리의 제약 없이 기준자가 될 수 있는 물체와 함께 대상 피사체를 촬영하여 그 크기를 계측하는 일반 계측 모듈로 구성되어 있다.

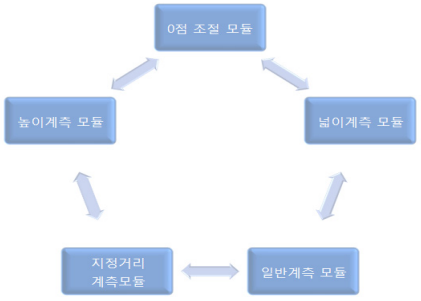

그림 1. 주요 모듈 구성도
Fig. 1. Diagram of major module components

Ⅲ. 스마트 피사체 계측 모듈

스마트 폰 카메라를 이용한 화상 이미지 내의 특정 피사체의 크기를 계측하는 어플의 주요 동작원리는 바로 스마트 폰에 내장되어 있는 카메라 모듈과 3축 자이로 가속도 센서를 이용하여, 촬영지점에서 화상 이미지까지의 거리와 피사체의 크기를 측정하는 것이다. 이때 가속 센서를 이용하여 촬영하는 스마트 폰 카메라와 대상 피사체가 지면에 닿는 선과의 기울기 각도와 카메라 모듈의 촬영 회전 각도를 측정하고, 대상 이미지를 촬영할 때 스마트 폰 카메라의 지면으로부터의 높이를 이용하여 촬영 지점에서 피사체까지의 거리를 산출하고, 촬영된 화상이미지가 표시된 액정 화면상의 좌표 값을 이용하여 해당 피사체의 크기를 환산하여 계측하게 된다.[6]

대상 이미지를 촬영하여 피사체 크기를 계측하기 전에 계측에 사용되는 스마트 폰의 카메라 모듈의 특성을 조절하여 크기 계측 정확도를 높이기 위해 제일 먼저 0점 조절 모듈을 실행하게 되는데 이 모듈은 처음 사용 시 한 번만 실행하면 된다. 앱 사용 시 기술적인 어려움을 최소화 하고자 이 모듈이 구동되면 음성을 이용한 안내 멘트가 실행되게 하였으며 계측의 정확도를 점검할 수 있도록 거리 측정 기능을 포함하였다. 우선 간단한 앱 사용에

<0점 조절 모듈>

```
Handler mHandler = new Handler() {
@Override
public void handleMessage(Message msg) {
 super.handleMessage(msg);
  // 제한시간을 알기 쉽게 하기 위하여 프로그래스바 사용
 mHandler.sendEmptyMessageDelayed(0, 1000);
  // 시간 감소
 timervalue++;
 if (timervalue == 10) {
  // 영점 조절 높이 가져옴
  AlertDialog.Builder alertDlg = new AlertDialog.Builder(
                                  zeromeasure.this);
  alertDlg.setTitle("현재 스마트폰의 높이를 입력하세요.");
  alertDlg.setView(linear);
  alertDlg.setPositiveButton("확인",
   new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,int which) {
     // TODO Auto-generated method stub
     name = edt1.getText().toString();
     if (edt1.length() != 0) {
      // 처음 값을 입력 할 때 테이블에 아무 값도 없을 시
      if (value.equals("")) {
       cv.put("height", name);
        db.insert("heightdatabasetable", null,cv);

                    // 중략

// 센서 리스너를 구현함으로서 센서 값들을 받아옴
SensorEventListener mSensorListener =
            new SensorEventListener() {
 public void onAccuracyChanged(Sensor arg0, int arg1) {
  // TODO Auto-generated method stub
 }
 // 실제 센서 값들을 받아옴
 public void onSensorChanged(SensorEvent event) {
  // TODO Auto-generated method stub
  float[] v = event.values;
  // 최종적으로 출력되는 거리 값
  double max;
  String strNumber = String.format("%.2f", v[1]);
  String strNumber2 = String.format("%.2f", v[2]);
  // 타입이 맞을시 값을 출력
  switch (event.sensor.getType()) {
  // 방향 센서 일시 동작
  case Sensor.TYPE_ORIENTATION:
   // 거리 구하는 공식
   max = maxheight / (Math.tan(90 * Math.PI / 180
             - Math.abs(v[1])* Math.PI / 180));
   String strNumber3 = String.format("%.1f", max);
   measure.setText("거리 : " + strNumber3);
   mTxtOrient.setText(" pitch(기울기) : " + strNumber
             + " , roll(회전) : " + strNumber2);
    break;
  }
 }
};
```

그림 2. 0점 조절 모듈 소스
Fig. 2. Source code of a 0 point control module.

<높이 측정 모듈>

```
// 실제 센서 값들을 받아옴
public void onSensorChanged(SensorEvent event) {
 // TODO Auto-generated method stub
 float[] v = event.values;
 // 최종적으로 출력되는 거리 값
 String strNumber = String.format("%.2f", v[1]);
 String strNumber2 = String.format("%.2f", v[2]);
 // 타입이 맞을시 값을 출력
 switch (event.sensor.getType()) {
 // 방향센서 일시 동작
 case Sensor.TYPE_ORIENTATION:
  if (count == 0) { // 거리 구하는 공식
   maxheight2 = Integer.parseInt(maxheight1);
   max = maxheight2 / (Math.tan(90 * Math.PI / 180 - Math.abs(v[1])
                * Math.PI / 180));
  }
  if (count == 1) {
   // 높이값
   height = max * (Math.tan(Math.abs(v[1]) * Math.PI / 180 - 90
                * Math.PI / 180)) + 150;
  }
  // 거리값 소수점 한자리 표시
  strNumber3 = String.format("%.1f", max);
  strNumber4 = String.format("%.1f", height);
  measure.setText("거리 : " + strNumber3 + "cm");
  hmeasure.setText("높이 : " + strNumber4 + "cm");
  mTxtOrient.setText(" pitch(기울기) : " + strNumber
                + ", roll(회전) : " + strNumber2);
  // 기울기 -90도와 회전 0도에 가까워지면 일시정지
  if (-88 >= v[1] && -92 <= v[1] && 2 >= v[2] && -2 <= v[2]) {
                background.pause();
  }
  // 아니면 계속 반복
  else {
                background.start();
  }
  break;
 }
};
```

그림 3. 높이 측정 모듈 소스
Fig. 3. Source code of a height measurement module.

대한 설명을 하고 이어 실제 이 앱을 이용하여 화상 이미지를 촬영할 때 카메라 모듈의 지면으로부터의 높이를 입력하게 한다. 그리고 나서 측정하고자 하는 피사체가 지면에 닿은 선에 카메라를 맞추도록 한다. 이때 카메라는 지면과 수평 상태에서 피사체와는 수직으로 유지해야 정확한 거리를 측정할 수 있으며, 카메라의 기울기와 회전 각도를 정확하게 유지할 수 있도록 그 각도가 허용 오차 범위인 2도 이상을 벗어나면 경고음과 함께 스마트 폰이 진동하도록 하였다. 이때 보통 2m - 3 m정도의 거리를 측정하는데 측정 결과가 실제거리와 상이하면 그 차이를 구해 입력하여 그 오차를 보정한다. 0점 조절 모듈의 또 다른 기능은 지정거리에서 길이가 알려진 기준자를 촬영한 후, 액정화면에 표시된 촬영 이미지의 양단을 선택하여 그 좌표 값을 산출하여 액정화면 상에서의 단위 좌표 당 실제 길이를 계산하게 되는데 이 값은 추후 지정거리 측정 모듈에서 피사체 계측 시 활용되게 된다. 그림 2는 0점 조절 모듈의 소스 일부를 보여준다.

그림 3과 그림 4는 각각 피사체의 높이와 폭을 측정하는 모듈의 소스 일부를 보여준다. 높이 측정 모듈에서는 대상 피사체의 지면과 닿은 부분을 조준하여 피사체까지

<넓이 측정 모듈>

```
// 실제 센서 값들을 받아옴
public void onSensorChanged(SensorEvent event) {
 // TODO Auto-generated method stub
 float[] v = event.values;
 // 최종적으로 출력되는 거리 값
 String strNumber = String.format("%.2f", v[1]);
 String strNumber2 = String.format("%.2f", v[0]);
 // 타입이 맞을시 값을 출력
 switch (event.sensor.getType()) {
 // 방향센서일시 동작
 case Sensor.TYPE_ORIENTATION:
  // 거리 구하는 공식
  if (count == 0) {
   maxheight2 = Integer.parseInt(maxheight1);
   max = maxheight2 / (Math.tan(90 * Math.PI / 180 -
              Math.abs(v[1]) * Math.PI / 180));
  }
  if (count == 1) { // 좌측 길이
   left = (Math.tan(360 * Math.PI / 180 - Math.abs(v[0])
            * Math.PI / 180)) * max;
  }
  if (count == 2) { // 우측 길이
  right = (Math.tan(Math.abs(v[0]) * Math.PI / 180)) * max;
  }
  if (count == 3) { // 최종 길이 계산
   last = left + right;
  }
  strNumber3 = String.format("%.1f", max);
  strNumber4 = String.format("%.1f", left);
  strNumber5 = String.format("%.1f", right);
  strNumber6 = String.format("%.1f", last);
  measure.setText("거리 : " + strNumber3 + "cm");
  lmeasure.setText( "좌측 : " + strNumber4 + "cm");
  rmeasure.setText("우측 : " + strNumber5 + "cm");
  lastmeasure.setText("최종 : " + strNumber6 + "cm" );
  // 기울기, 회전각도
  mTxtOrient.setText(" pitch(기울기) : " + strNumber
            + " , roll(회전) : " + strNumber2);
  // 기울기 -90도와 회전 0도에 가까워지면 일시정지
  if (-88 >= v[1] && -92 <= v[1] && 2 >= v[2] && -2 <= v[2]) {
            background.pause();
  }
  // 아니면 계속 반복
  else {
   background.start();
  }
  break;
 }
};
```

그림 4. 넓이 측정 모듈 소스
Fig. 4. Source code of a width measurement module.

의 거리를 구하고, 이어 피사체의 최상단을 조준하여 기울기를 측정하여 피사체의 최종 높이를 계산하게 된다. 한편, 피사체의 폭을 계측하는 모듈에서는 3단계로 이루어지는데 우선 대상 피사체와 닿은 지면을 조준하여 피사체 간 거리를 측정하고, 이어 피사체의 왼쪽 끝부분을 조준하여 피사체의 왼쪽 반의 길이를 구하고, 피사체의 오른쪽 끝을 향해 조준하여 오른쪽 반의 길이를 구하여 최종적으로 피사체의 폭을 계산하게 된다.[6,7]

그림 5는 2m - 3m로 지정한 거리에서 화상이미지를 촬영하여 피사체의 길이를 계측하는 지정거리 측정 모듈의 소스코드 일부를 보여준다. 이 모듈에서는 0점 조절 모듈에서 이미 산출한 액정 화면상의 단위 좌표 당 실제 길이 결과를 활용하게 된다. 이 모듈에서는 0점 조절 모듈의 거리측정 기능을 이용하여 피사체가 있는 곳으로부터 지정거리만큼 이동한 후 대상 이미지를 촬영한다. 그리고 나서 계측하고자 하는 특정 피사체를 액정화면 상

<지정거리 측정 모듈>

```
// 실제 센서 값들을 받아옴
public void onSensorChanged(SensorEvent event) {
 // TODO Auto-generated method stub
 float[] v = event.values;
 // 최종적으로 출력되는 거리 값
 double max;
 String strNumber = String.format("%.2f", v[1]);
 String strNumber2 = String.format("%.2f", v[2]);
 // 타입이 맞을시 값을 출력
 switch (event.sensor.getType()) {
 // 방향센서일시 동작
 case Sensor.TYPE_ORIENTATION:
  // 지정 거리 구하는 공식
  // max =(int) (150 / Math.tan(90-(int)v[1]));
  max = 150 / (Math.tan(90 * Math.PI / 180 - Math.abs(v[1])* Math.PI / 180));
  String strNumber3 = String.format("%.0f", max);
  int strnumber33 = Integer.parseInt(strNumber3);
  strnumber33 = strnumber33 / 10;
  measure.setText("거리 : " + strnumber33);
  mTxtOrient.setText(" pitch(기울기) : " + strNumber
                    + " , roll(회전) : " + strNumber2);
  // 기울기 -90도와 회전 0도에 가까워지면 일시정지
  if (-88 >= v[1] && -92 <= v[1] && 2 >= v[2] && -2 <= v[2]) {
            background.pause();
  }
  // 아니면 계속 반복
  else {
            background.start();
  }
  break;
 }
};
```

그림 5. 지정거리 측정 모듈 소스
Fig. 5. Source code of a predetermined distance measurement module.

에서 양단을 선택하여 이들 두 점간 좌표 값을 구하고, 이를 이미 산출한 액정화면상의 단위 좌표 당 실제 길이 결과를 적용하여 대상 피사체의 크기를 계측하게 된다.

그림 6은 일반 거리 측정 모듈의 소스코드의 일부를 보여준다. 이 모듈은 대상 화상이미지의 촬영을 할 때 미리 지정거리를 확보하기 힘들거나, 지금까지 피사체의 정확한 사이즈 측정에 장애요인이 되었던 스마트 폰 카메라의 물리적인 특징, 촬영영상을 보여주는 액정화면의 해상도나 기종에 따른 오차 발생 등으로 인한 문제점들을 모두 해소하여 보다 정밀한 화상 이미지내의 특정 피사체에 대한 계측이 필요한 경우에 사용하는 모듈로 화상 이미지와 함께 길이가 알려진 기준자(base)가 되는 물체를 동일 평면 내에 가져와 함께 촬영한다. 그리고 계측을 위해 우선 첫 번째 단계로 기준자의 양단 끝을 터치스크린을 통해 선택하여 좌표 값을 구하게 되는데 이때 양단의 끝을 손으로 터치스크린을 통해 선택이 어려우므로 상하좌우 방향키를 두어 선택 점을 정확히 피사체의 끝에 위치할 수 있도록 하였다. 이렇게 얻어진 좌표 값에 대한 기준자의 실제 길이를 입력하여 단위 좌표 당 실제 길이를 계산하게 된다. 이어서 실제 계측하고자 하는 대상 피사체의 양단의 끝을 선택하여 액정 화면상의 좌표 값을 구하고 이를 이미 산출한 단위 좌표 당 실제 길이를 을 적용하여 정확하게 피사체의 크기를 계측하도록 한다.

<일반거리 측정 모듈>

```
// XYcount가 2가아니라면 처음 두개의 점을 찍어달라고 요청 - 두번 계측
if (XYcount < 2) { // 기준자에 대한 계측과 대상 피사체 계측
  AlertDialog.Builder alertDlg2 = new AlertDialog.Builder(zeromeasrue_result.this);
  alertDlg2.setTitle("처음 두개의 점을 찍어주세요");
  alertDlg2.setPositiveButton("확인",
  new DialogInterface.OnClickListener() {
   public void onClick(DialogInterface dialog, int which) {
    // TODO Auto-generated method stub
   }
}).show();
  // 두 번째 점을 찍고 확인 버튼을 눌러야 하는데 누르지 않을 시
} else if (touchPoint[0][0] == 0 || touchPoint[0][1] == 0
                || touchPoint[1][0] == 0 || touchPoint[1][1] == 0) {
   AlertDialog.Builder alertDlg2 = new AlertDialog.Builder(zeromeasrue_result.this);
   alertDlg2.setTitle("확인 버튼을 클릭하시고 점을 한 번 더 찍어주세요");
   alertDlg2.setPositiveButton("확인", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,
     int which) {
    // TODO Auto-generated method stub
   }
  }).show();
  // 두 번째 점을 찍고 확인 버튼을 눌렀을 시
} else {
  // 커스텀 Activity에 있는 String를 받아옴, 두 점 사이의 거리 환산(cm)
            touch = Integer.parseInt(edt1.getText().toString());
  // 두번째 터치를 했을 때
  touchX4 = (touchPoint[0][0] - touchPoint[1][0]);
  touchY4 = (touchPoint[0][1] - touchPoint[1][1]);
  touchPoint[0][0] = 0;
  touchPoint[0][1] = 0;
  touchPoint[1][0] = 0;
  touchPoint[1][1] = 0;
  // 절대값 변환
  pitanum2 = (int) Math.sqrt((touchX4 * touchX4) + (touchY4 * touchY4));
  // 일반 거리 계측 결과값 게산
  x = (float) ((touch * pitanum2) / pitanum1);
  // float형인데 소수점 이하 자리 소수점 자르기
  String strNumber = String.format("%.2f", x);
  // 결과 값 출력 다이얼로그 박스
  AlertDialog.Builder alertDlg3 = new AlertDialog.Builder(zeromeasrue_result.this);
  alertDlg3.setTitle("피사체의 길이는" + strNumber + "입니다.");
  alertDlg3.setPositiveButton("확인", new DialogInterface.OnClickListener() {
   public void onClick(DialogInterface dialog,
    int which) {
   // TODO Auto-generated method stub
   }
  }).show();
  count++;
  if (count == 2) {
   count = 1;
  }
}
```

그림 6. 일반거리 측정 모듈 소스
Fig. 6. Source code of a general distance measurement module.

피사체의 크기 계측을 위해 화상 이미지를 촬영하거나 대상 피사체에 카메라 모듈을 조준할 때 계측 결과의 정확도에 가장 영향을 미칠 수 있는 요소는 화상 이미지를 촬영하는 자세이다. 즉, 대상 피사체를 향해 정면으로 카메라를 위치시켜야 하며 카메라의 기울기는 피사체와 수평을 이루어야 보다 정확한 피사체 크기 계측치를 얻을 수 있게 된다. 따라서 본 논문에서는 모든 계측 모듈에서 카메라 모듈을 사용하여 피사체 계측을 하고자 조준하거나 촬영할 때 카메라의 기울기 각도가 수평이 되도록 하고 피사체와는 정면으로 위치되도록 하기 위해 기울기 각도와 회전 각도를 실시간으로 산출하여 그 각도가 허용 오차 범위인 2도 이상의 오차가 발생하는 경우 비프 경고음을 울리게 하였고, 아울러 스마트 폰 전체에 진동을 주어 촬영 오차 가능성을 경고하게 하였다.

Ⅳ. 구현 결과 및 고찰

이 장에서는 본 논문에서 제안한 안드로이드 기반 스마트 폰 카메라 모듈을 이용한 화상 이미지 내의 특정 피사체의 크기를 계측하는 스마트 응용의 구현결과를 보여주고, 이를 국내에서 시판되는 주요 안드로이드 기반 스마트 폰 기종을 대상으로 모의실험을 통해 피사체 계측 정확도를 분석 · 고찰한 결과를 보여준다.

그림 7은 0점 조절 모듈을 실행한 결과를 보여준다. 왼쪽 그림은 화상 이미지를 촬영할 때 스마트 폰 카메라모듈의 지면으로부터 높이를 입력하는 것이고, 가운데 그림은 피사체까지의 거리를 측정하는 화면이며, 오른쪽 그림은 길이가 알려진 기준자를 지정거리에서 촬영한 후 그 이미지를 액정화면에서 선택하여 좌표 값을 추출한 후에 그 기준자의 실제길이를 입력하는 화면이다. 이 입력 값으로 단위 좌표 당 실제 길이를 계산하게 된다. 그림 8은 피사체의 높이와 폭을 측정하는 모듈을 실행한 화면을 보여준다. 피사체의 높이와 넓이를 계측하기 전에 우선 피사체까지의 거리를 측정하고 알고리즘에 따라 각각 높이와 넓이를 측정하게 된다.

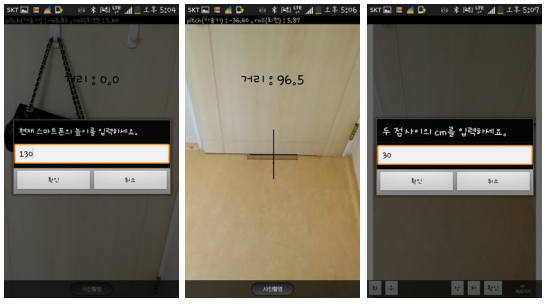

그림 7. 0점 조절 모듈 실행결과
Fig. 7. Execution results of a 0 point control module.

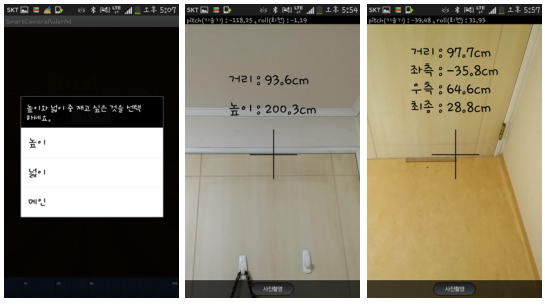

그림 8. 높이 · 넓이 측정 모듈 실행결과
Fig. 8. Execution results of a height & width measurement module.

그림 9는 지정거리와 일반 거리 측정 모듈을 실행한 결과 화면으로 지정거리 측정은 0점 조절 모듈에서 이미 산출한 액정화면의 단위 좌표 당 실제길이를 반영하여 촬영된 피사체의 실제 길이를 계측하게 되고 일반 거리 측정 모듈에서는 이미 길이가 알려진 기준자를 대상 이미지와 동일 평면에 두고 촬영하여 이 기준자를 통해 단위 좌표 당 실제길이를 계산하여 계측 대상 피사체의 길이 계산에 반영하게 된다. 그림 10은 본 논문에서 구현한 스마트 폰 카메라를 이용한 화상 이미지내의 특정 피사체를 계측하는 앱의 시작 화면과 어플 사용 방법을 설명하는 도움말 화면 그리고 0점 조절 모듈과 4가지 계측 모듈을 선택할 수 있는 메뉴화면을 보여준다. 이 앱의 사용자의 편의를 위해 각 단계별 사용방법에 대한 음성 안내 멘트가 나오도록 하였고 각 단계별 진행과 오류에 대해 진동 기능과 비프 경고음 기능을 이용하여 초보 사용자의 미숙한 사용으로 인한 계측 오류나 계측 정확도 저하를 막을 수 있도록 하였다.

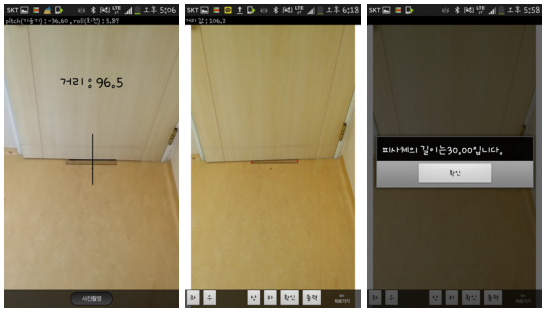

그림 9. 지정거리와 일반거리 측정 모듈 실행결과
Fig. 9. Execution results of a predetermined & general distance measurement module.

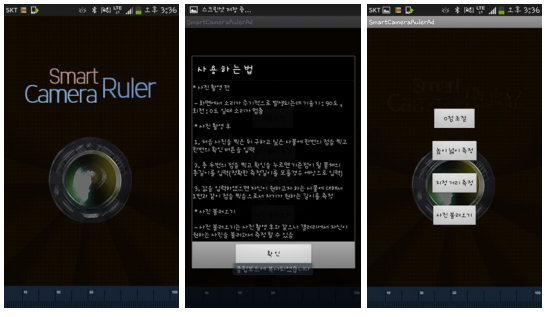

그림 10. 스마트 앱 메인 화면 실행결과
Fig. 10. Execution results of the main screen.

본 논문에서 제안한 어플의 계측 정확도를 알아보기 위해 국내에서 시판 중인 국내 3사의 안드로이드 기반 스

마트 폰 6개 기종을 대상으로 피사체 간 거리측정, 높이 측정, 넓이측정, 지정거리 측정, 그리고 일반거리 측정 모듈에 대한 모의실험을 실시하였다. 거리측정 모듈은 2m와 3m의 거리를 각각 20회씩 반복 측정하여 그 오차를 기록하였고, 높이와 넓이 측정 모듈에서는 길이가 알려진 피사체를 각각 20회씩 반복 측정하였으며, 지정거리 측정 모듈에서는 2m와 3m 거리에서 각각 20회씩 측정하여 0점 조절 모듈에서 계산한 단위 좌표 당 실제 길이를 적용하여 정확도를 비교하였다 그리고 마지막으로 일반거리 측정모듈에서는 임의의 거리에서 30cm 길이의 기준자를 사용하여 피사체와 동일 평면에서 함께 촬영한 후 기준자로부터 단위 좌표 당 실제길이를 산출하여 대상 피사체에 적용하는 방식으로 각 기종에 대해 총 20회씩 측정하여 그 계측 정확도를 산출하였다. 표2는 각 모듈에 대한 기종별 계측 정확도에 대한 오차율의 모의실험 결과를 보여준다. 표에서 보인바와 같이 전 기종에 걸쳐 피사체 계측에 대한 정확도는 97% 이상으로 매우 높게 나타났으며, 기종별로는 S사제품이나 P사제품보다 L사 기종에서 약간 더 높은 정확도를 나타내고 있음을 알 수 있다. 아울러, 각 촬영 계측 모듈별로 계측 정확도를 비교하면 높이와 넓이 측정 모듈이 상대적으로 지정거리나 일반거리 측정모듈에 비해 그 정확도가 낮게 나타나는데 그 이유는 높이나 넓이를 측정할 때에는 촬영 자세로 인한 기울기나 회전각도의 변화의 영향이 크게 초래되는데 기인한다고 사료된다. 전체적으로는 일반거리 측정 모듈의 정확도가 사용 기종에 상관없이 우수한 정확도를 보임을 알 수 있다.

표 2. 모의실험 결과(오차율 단위 %)
Table 2. Simulation results.

기종 / 모듈	S사		L사	P사		
	A	B	C	D	E	F
거리	0.41	0.27	0.23	0.73	0.84	0.76
높이	0.97	0.65	0.57	1.05	1.16	1.14
넓이	1.61	1.09	0.94	2.31	2.51	2.15
지정거리	0.44	0.34	0.33	0.58	0.61	0.53
일반거리	0.25	0.16	0.13	0.38	0.42	0.27

V. 결 론

본 논문에서는 안드로이드 기반 스마트 폰 카메라 모듈을 이용한 화상 이미지 내의 특정 피사체의 크기를 측정하는 앱을 설계하고 구현하였다. 제안한 어플의 피사체 사이즈 측정 정확도와 신뢰성을 높이기 위해 0점 조절 모듈, 높이와 넓이 측정모듈, 지정거리 측정모듈, 그리고 일반 거리 측정모듈 등 4가지 계측 모드로 설계하였으며, 모의실험을 통해 그 정확도를 비교 분석하였다.

모의실험 결과 본 논문에서 구현한 피사체 계측 앱은 사용 기종이나 특성에 무관하게 97% 이상의 계측 정확도를 나타냈으며, 특히 지정거리와 일반거리 측정 모듈에서 기종에 상관없이 99.4% 이상의 높은 정확도를 보였다. 다만 피사체 촬영 시 촬영 자세가 계측 결과에 영향을 크게 미치는 높이와 넓이 측정 모듈에서는 상대적으로 낮은 정확도를 보였는데 이는 피사체 촬영 시 자세로 인한 기울기와 회전 각도의 영향으로 판단된다. 향후 연구 과제는 피사체 계측 정확도에 영향을 미치는 촬영 자세의 영향을 최소화 하는 방안과 이 피사체 계측 어플의 다양한 응용에 관한 것이다.

References

[1] Geoff Dougherty, Digital Image Processing for Medical Applications, Cambridge University Press, 2009.

[2] Park, H., "Performance Analysis of the Tunnel Inspection System Using High Speed Camara", Korean Institute Of Information Technology, Journal of Korean Institute of Information Technology, Vol.11 No.4 pp. 1 - pp. 6, 2013.

[3] Oh, S., "Design of a Middleware for Android-based Smartphone Applications", Journal of the Institute of Webcasting, Internet and Telecommunication, Vol., 12, No. 2, pp. 111 - pp. 118, 2012.

[4] http://www.androidpub.com/1305

[5] Alan C. Bovik, Essential Guide to Image Processing, Elsevier, 2009.

[6] Oh, S.,"Design of Measurement Algorithms in the Smart CamRuler", Journal of the Institute of

Webcasting, Internet and Telecommunication, Vol. 13, No. 4, pp. 149 - pp. 156, 2013.

[7] https://play.google.com/store/apps/details?id=kr.sira.measure

※ 이 논문은 2012학년도 세명대학교 교내학술연구비 지원에 의해 수행된 연구임.

저자 소개

오 선 진(종신회원)

• 제 6권 제2호 참조
• 현재 세명대학교 정보통신학부 교수
<주관심분야 : 스마트응용, Big Data, MANETs, 모바일컴퓨팅, USN 등>