Mobile Image Processing

- Examples of mobile image processing
- Android platform
- Class resources for Android
- Eclipse integrated development environment
- Augmentation of viewfinder frames
- Debugging with DDMS perspective
- Taking a device screenshot
- Creating an Android emulator
- Mixed programming with Android JNI
- OpenCV library for Android
- Past class projects
Recognizing video at a glance

1. User snaps a photo of screen.
2. Our system identifies video and frame within the video.
3. User resumes video on the phone.
Recognizing video at a glance
Mobile book spine recognition
Mobile book spine recognition

- Segment Spines
- Extract Features
- Query Vocab. Tree
- Check Geometry
- Send Query Frame
- Track Camera Pose
- Send Response
- Low Motion
- High Motion
- Viewfinder Frames
- Time
- Wireless Network
- Mobile Device
- Server

Signal Processing and Linear Systems
On-device image matching
On-device image matching

Motion-Adaptive Query Selection

Viewfinder Augmentation

Local Image Database

Local Database Search
Android manifest file
Android manifest file

```xml
...<application android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
    <activity android:name=".ViewfinderEE368"
        android:label="@string/app_name"
        android:screenOrientation="landscape"
        android:theme="@android:style/Theme.NoTitleBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

<uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.CAMERA"/>
...```

Set landscape orientation for viewfinder app
Set this activity as the application’s main activity
Declare permission to use the device’s camera
Real-time debugging with DDMS perspective
Real-time debugging with DDMS perspective

File system on device

Messages on the device

Device
Taking a screenshot of the device
Taking a screenshot of the device

Mean (R,G,B): 144.3, 123.4, 130.1
Std Dev (R,G,B): 86.85, 92.70, 90.43
Creating an Android emulator
“Color Histograms” app running on emulator

Mean (R,G,B): 121.2, 112.0, 110.9
Std Dev (R,G,B): 119.6, 120.3, 119.8
OpenCV: **Open Source Computer Vision**

- General Image Processing Functions
- Image Pyramids
- Segmentation
- Geometry descriptors
- Camera calibration, Stereo, 3D
- Features
- Utilities and Data Structures
- Transforms
- Tracking
- Fitting
- Machine Learning:
  - Detection
  - Recognition
- Matrix Math

http://docs.opencv.org
OpenCV for Android

- Port of OpenCV to Android platform
  - Over 2,500 optimized algorithms written in C/C++
  - Compiled with STL-enabled Android NDK
  - Enables popular CV functions to be used on mobile images/videos

- Differences from regular Android programming
  - Requires writing C/C++ code
  - Requires writing Java Native Interface (JNI) wrappers
  - Requires using Android NDK in addition to Android SDK

- Our Tutorial #2 helps to make the transition
  - How to set up OpenCV programming environment
  - How to write Android apps that call OpenCV functions
  - How to integrate NDK compilation into Eclipse IDE
“CVCamera” project

- Goals of this project
  - Learn how to incorporate C/C++ code into an Android project
  - Learn how to utilize OpenCV library functions
  - Learn how to draw feature keypoints on viewfinder frames
- Full source available on class website
“CVCamera” class hierarchy

CVCamera Activity (Java)

Manages user interface and program execution

CVCamera JNI (Java)

Interfaces with C/C++ code

Processor (C++)

Processes viewfinder frames

OpenCV Library
CV Camera class: menu options

```java
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add("FAST");
    menu.add("STAR");
    menu.add("SURF");
    menu.add("MSER");
    menu.add("Settings");
    return true;
}
```
public boolean onOptionsItemSelected(MenuItem item) {
    LinkedList<PoolCallback> defaultCallbackStack =
        new LinkedList<PoolCallback>();

    defaultCallbackStack.addFirst(myGLView.getDrawCallback());

    if (item.getTitle().equals("FAST")) {
        defaultCallbackStack.addFirst(new FastProcessor());
    } else if (item.getTitle().equals("STAR")) {
        defaultCallbackStack.addFirst(new STARProcessor());
    } else if (item.getTitle().equals("SURF")) {
        defaultCallbackStack.addFirst(new SURFProcessor());
    } else if (item.getTitle().equals("MSER")) {
        defaultCallbackStack.addFirst(new MSERProcessor());
    }

    myPreview.addCallbackStack(defaultCallbackStack);
    return true;
}
“CVCamera” class hierarchy

CVCamera Activity (Java)

Manages user interface and program execution

CVCamera JNI (Java)

Interfaces with C/C++ code

Processor (C++)

Processes viewfinder frames

OpenCV Library
JNI class: interface to C/C++ code

```java
static {
    try {
        System.loadLibrary("android-opencv");
        System.loadLibrary("cvcamera");
    } catch (UnsatisfiedLinkError e) {
        throw e;
    }
}

public final static native int DETECT_FAST_get();
public final static native int DETECT_STAR_get();
public final static native int DETECT_SURF_get();
public final static native int DETECT_MSER_get();
public final static native long new_Processor();
public final static native void delete_Processor(long jarg1);
```

Load libraries built by the Android NDK

Function prototypes for interface to C/C++ side
"CVCamera" class hierarchy

CVCamera Activity (Java)
- Manages user interface and program execution

CVCamera JNI (Java)
- Interfaces with C/C++ code

Processor (C++)
- Processes viewfinder frames

OpenCV Library
class Processor {
private:
    cv::StarFeatureDetector my_stard;
    cv::FastFeatureDetector my_fastd;
    cv::SurfFeatureDetector my_surfd;
    cv::MserFeatureDetector my_mserd;
    std::vector<cv::KeyPoint> my_keypoints;

public:
    Processor():
        my_stard(STAR detector parameters),
        my_fastd(FAST detector parameters),
        my_surfd(SURF detector parameters),
        my_mserd(MSER detector parameters) {
    }
    void detectAndDrawFeatures
        (int idx, image_pool* pool, int feature_type);
};

Different feature extractors stored as members
Initialize extractors in the class instantiation list
Function for detecting and drawing keypoints
Processor class: detect and draw feature keypoints

// Detect feature keypoints
Mat grey_im = pool->getGrey(idx);
Mat color_im = pool->getImage(idx);
my_keypoints.clear();
my_mserd->detect(grey_im, my_keypoints);

// Draw feature keypoints
vector<KeyPoint>::const_iterator it;
for (it = my_keypoints.begin();
it != my_keypoints.end(); ++it) {

  // Draw black circle
  Point2f pt = it->pt;
  circle(color_im, pt, it->size,
         cvScalar(0,0,0,0), 2);

  // Draw yellow circle
  pt.x += 1; pt.y += 1;
  circle(color_im, pt, it->size,
         cvScalar(255,255,0,0), 2);
} // iterator
Local feature keypoints extraction
Detecting edges, lines, and circles
Detecting edges, lines, and circles

// Extract Canny edges
C++: Canny(
    InputArray image, OutputArray edges, double threshold1, double threshold2
);

// Extract lines using Hough transform
C++: HoughLines(
    InputArray image, OutputArray lines, double rho, double theta, int thresh
);

// Extract circles using Hough transform
C++: HoughCircles(
    InputArray image, OutputArray circles, int method, double accumRatio, double minDist
);
Locally adaptive binarization
Locally adaptive binarization

// Perform locally adaptive binarization
C++: adaptiveThreshold(
    InputArray src, OutputArray dst, double maxValue,
    int adaptiveMethod, int thresholdType, int blockSize, double valOffset
);
Human face detection
Human face detection

// Call Viola-Jones face detector
vector<Rect> RectFaces;
((DetectionBasedTracker*)this)->process(*((Mat*)imageGray));
((DetectionBasedTracker*)this)->getObjects(RectFaces);
for (int nFace = 0; nFace < RectFaces.size(); nFace++) {
    rectangle(imageColor, RectFaces[nFace], cvScalar(255,255,0,0));
} // nFace
Class project: Foreign bill detector and converter

M. Digman and C. Elder, Spring 2013
Class project: Handwritten expression solver

S. Harvey and W. Harvey, Spring 2013
Class project: Lottery ticket checker

T. Zou and A. Gupta, Spring 2012
Class project: Optical music recognition and playback

Class project: mobile graph reader

What is the value at this point?

Segmentation of Image
Detection of Tick Marks
OCR of Text Labels

T. Jou, W. Ni, J. Su, Spring 2011
Class project: plant leaf classification

- Leaf Image
- Extract Contour
- Measure Classifier Similarity
- Compute Classifiers
- Classifier-Space Euclidean Dist. (17 dim.)
- Trained Leaf Species Classes

D. Knight, J. Painter, M. Potter, Spring 2010