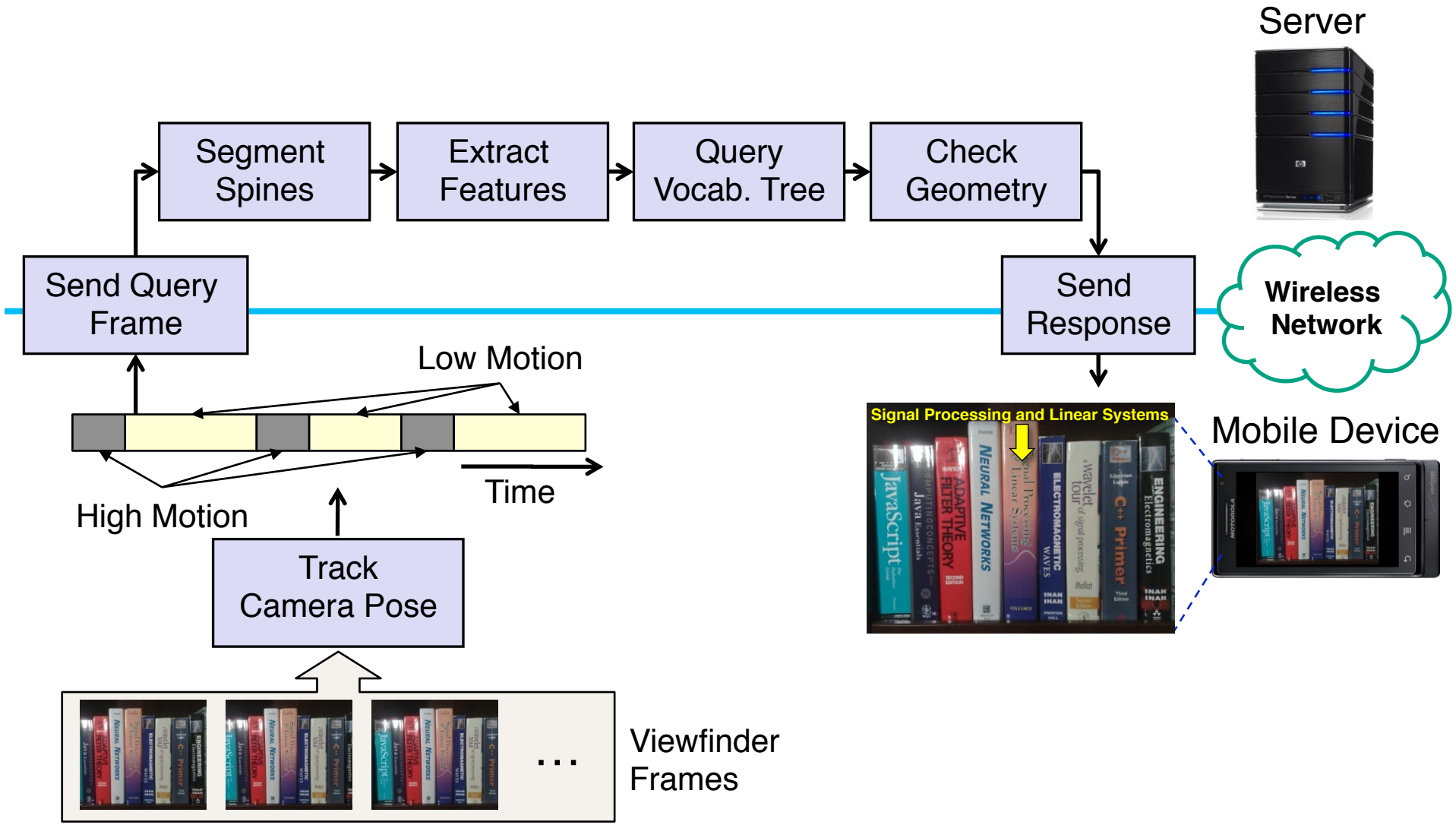
Mobile Image Processing

- Part 1: Introduction to mobile image processing on Android
- Part 2: Real-time augmentation of viewfinder frames
- Part 3: Utilizing optimized functions in the OpenCV library

Mobile book spine recognition

Mobile book spine recognition



Tracking local feature keypoints

???

OpenCV: Open Source Computer Vision

General Image Processing Functions

Geometric descriptors

Segmentation

Features

Transforms

Tracking

Machine Learning:
- Detection,
- Recognition

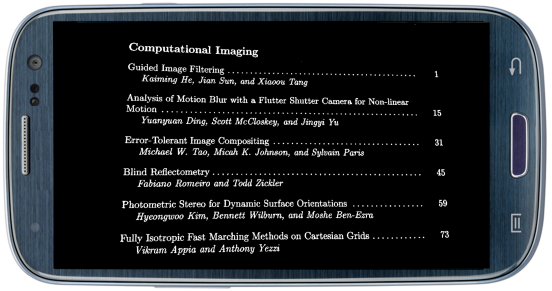
Matrix Math

Image Pyramids

Camera calibration, Stereo, 3D

Utilities and Data Structures

Fitting



http://docs.opencv.org

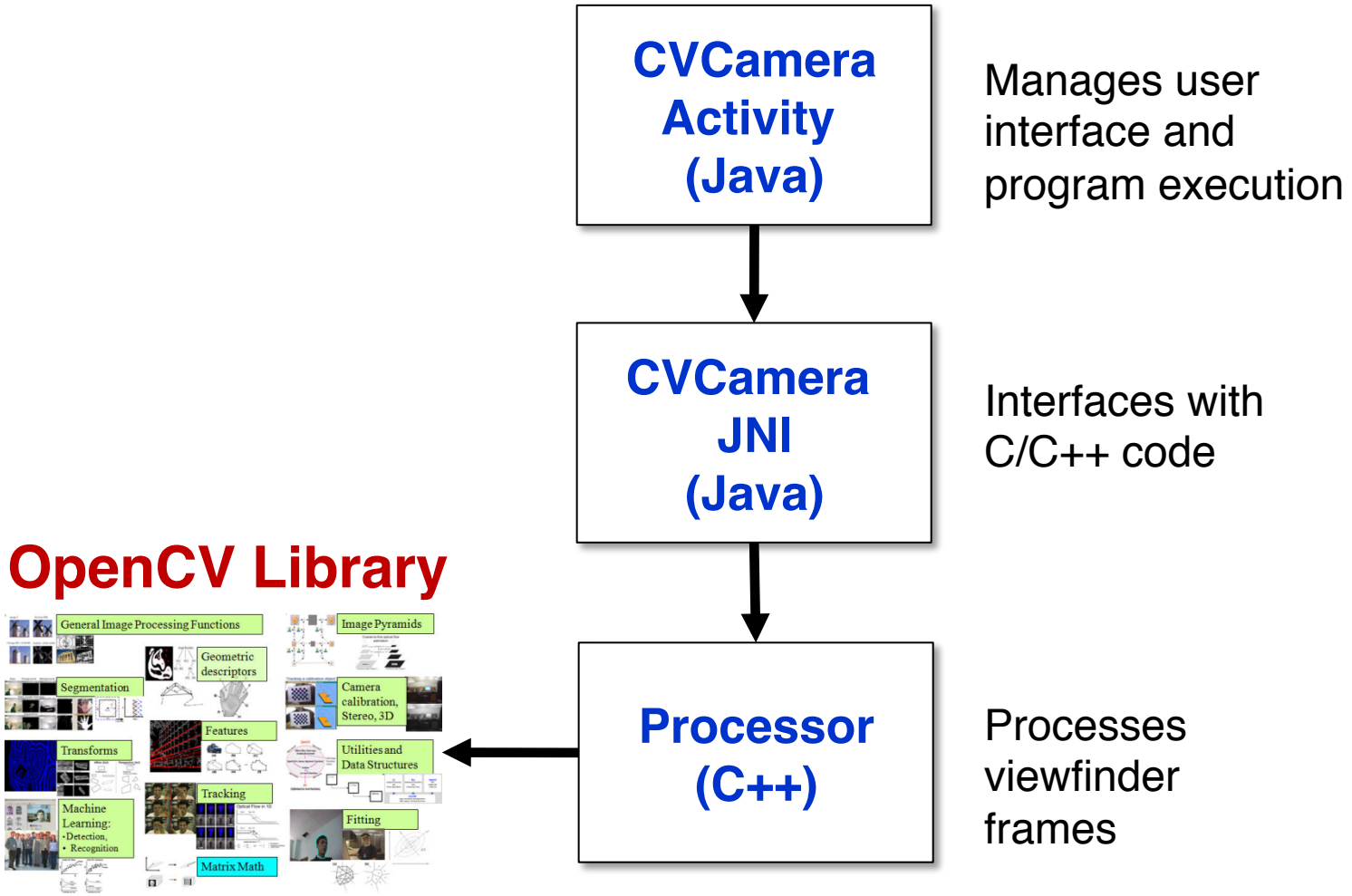
OpenCV for Android

- Port of OpenCV to Android framework
  - Over 2K 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 program
  - Learn how to utilize OpenCV library functions
  - Learn how to draw feature keypoints on viewfinder frames
- Step-by-step instructions available in Tutorial #2:
  - http://ee368.stanford.edu/Android

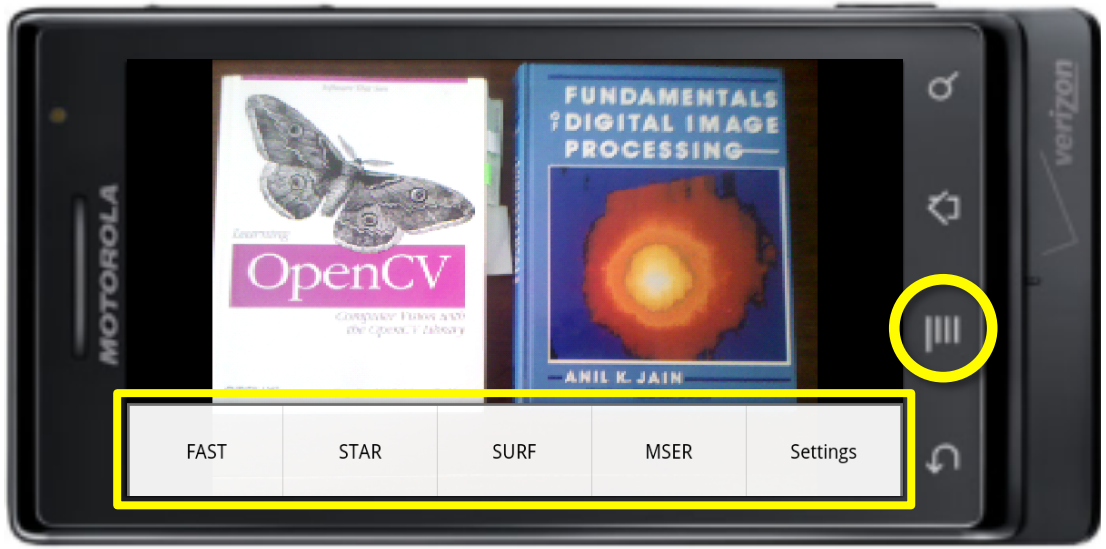
“CVCamera” class hierarchy



CVCamera class: menu options

```
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add("FAST");
    menu.add("STAR");
    menu.add("SURF");
    menu.add("MSER");
    menu.add("Settings");
    return true;
}
```

Java Code



CVCamera class: calling feature extractors

```
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;
}
```

Java Code

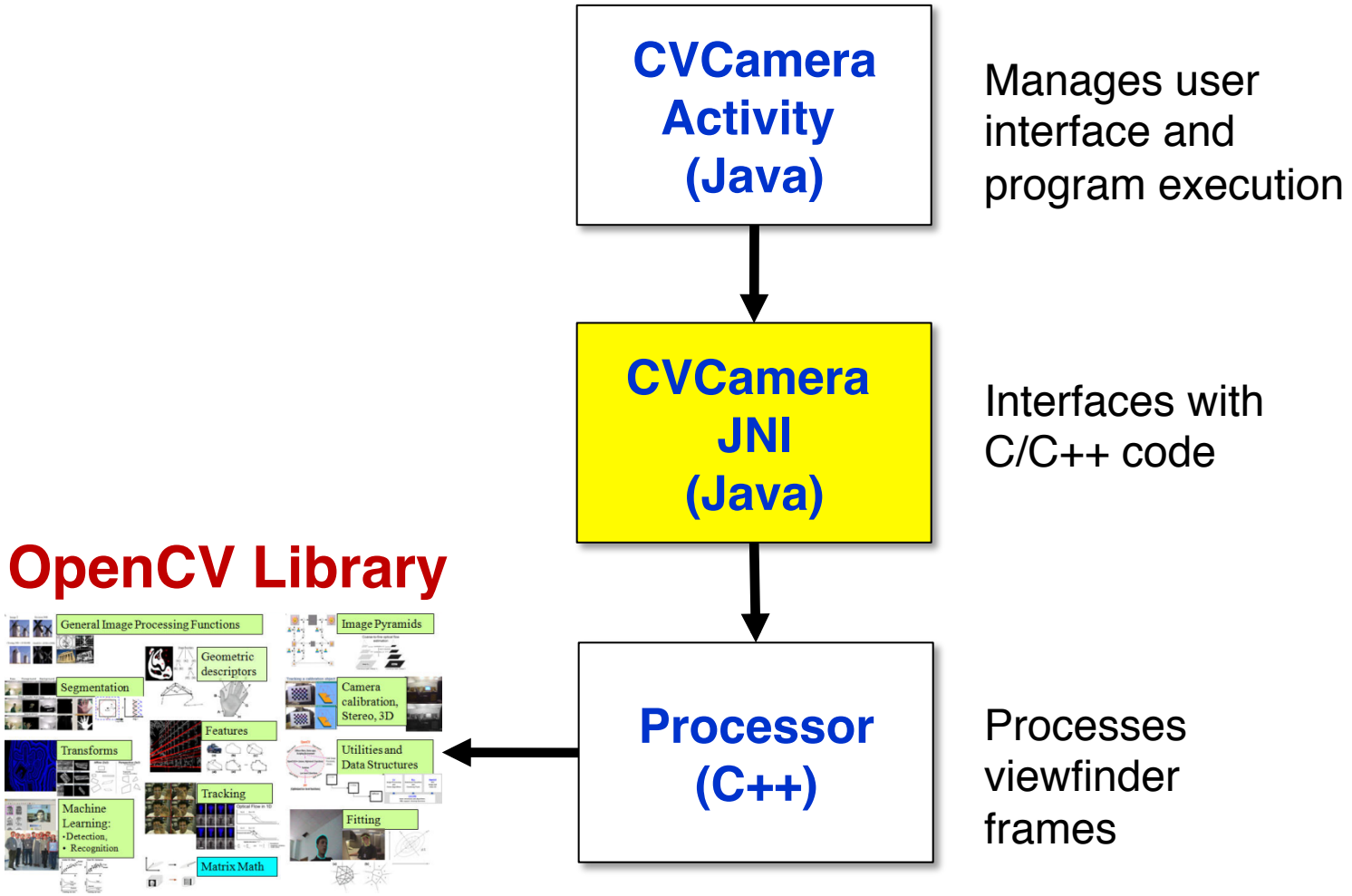
Populate a stack of callback functions

First callback function draws the frames

Other callback functions call feature extractors

???

“CVCamera” class hierarchy



JNI class: interface to C/C++ code

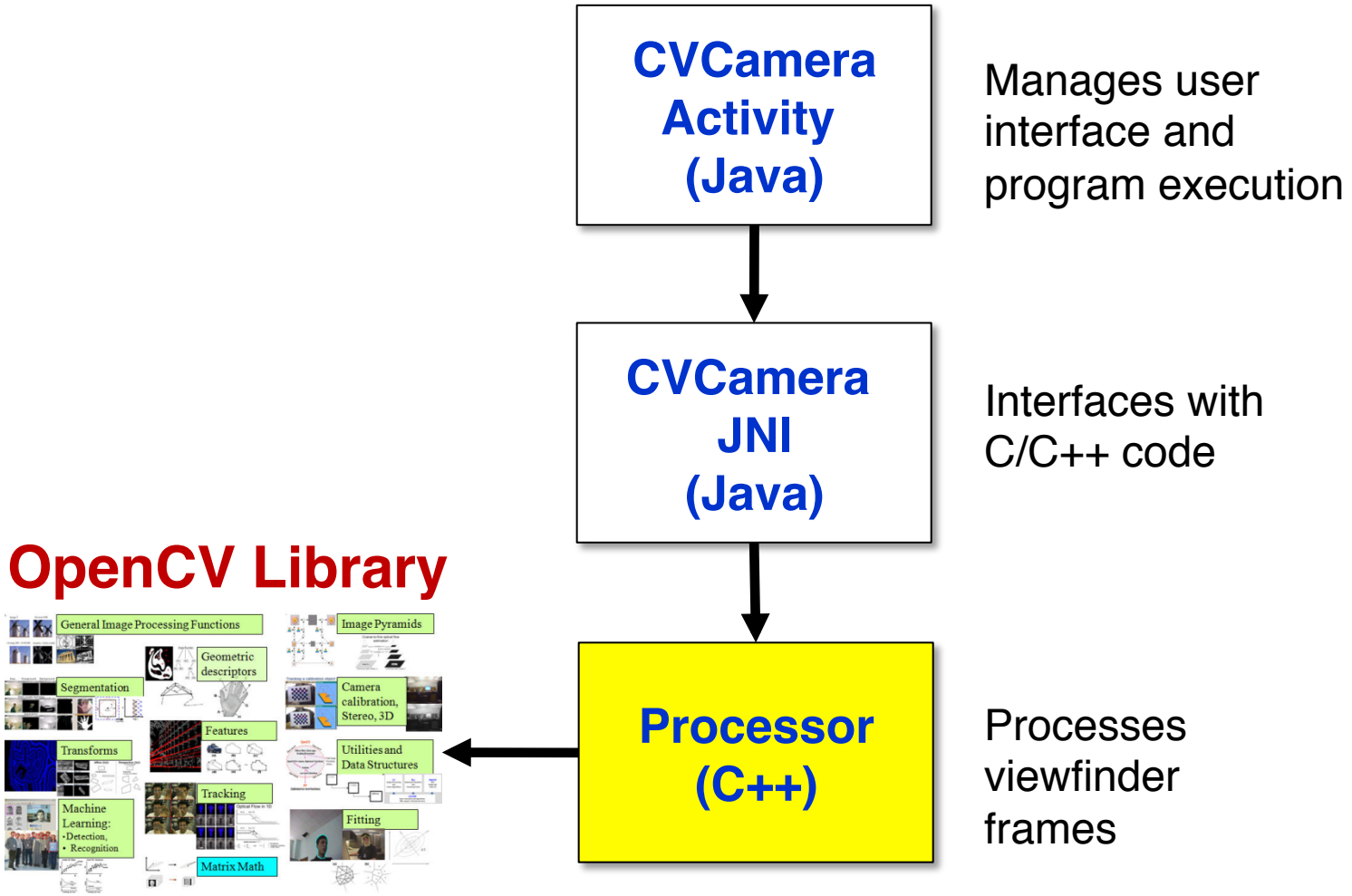
```
static {
   try {
         System.loadLibrary("android-opencv");
         System.loadLibrary("cvcamera");
   } catch (UnsatisfiedLinkError e) {
         throw e;
   }
}

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);
```

Java Code

Load libraries built by the Android NDK

Function prototypes for interface to C/C++ side

“CVCamera” class hierarchy



Processor class: initialize feature detectors

```
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);
};
```

C++ Code

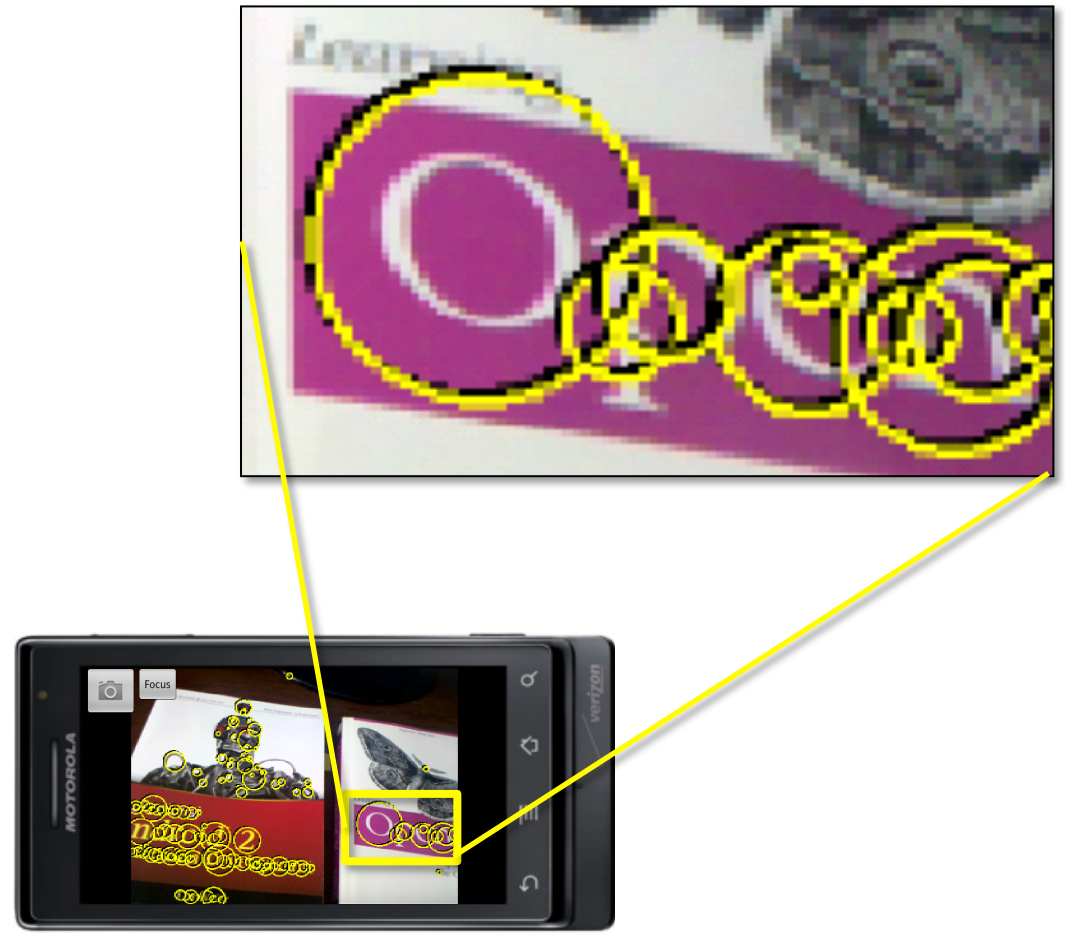
Different feature extractors stored as members

Initialize extractors in the class instantiation list

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
```

C++ Code



Local feature keypoints extraction

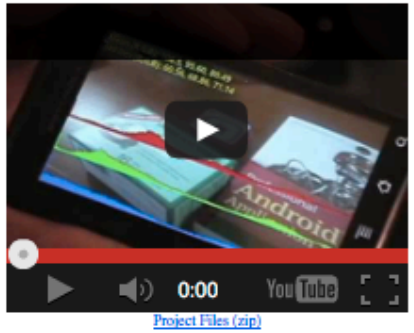
???

Other examples on class website

Histogram Equalization

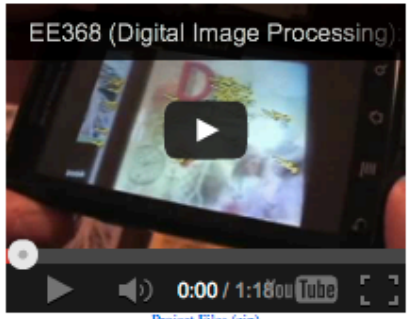
Project Files (zip)

Color Histograms



Project Files (zip)

Feature Tracking

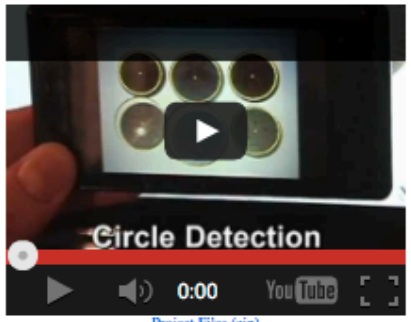


Project Files (zip)

Locally Adaptive Binarization

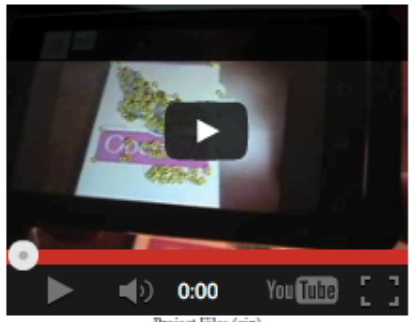
Project Files (zip)

Edges, Lines, and Circles



Project Files (zip)

Local Feature Keypoints



Project Files (zip)

Human Face Detection

Project Files (zip)

http://ee368.stanford.edu/Android

Detecting edges, lines, and circles

Detecting edges, lines, and circles

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

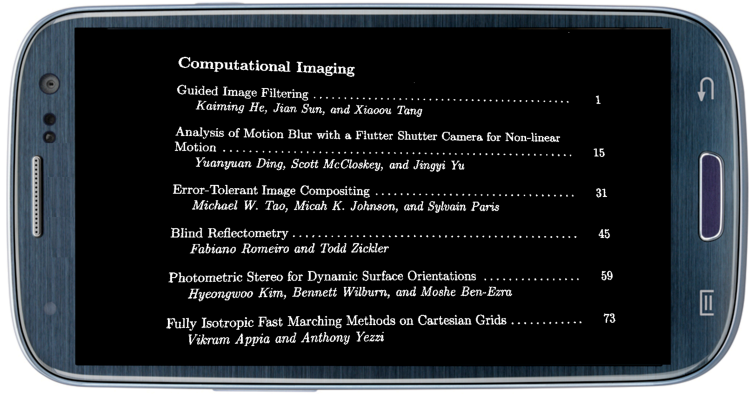
```
// Extract lines using Hough transform
C++: HoughLines(
    InputArray image, OutputArray lines, double rho, double theta, int thresh
);
```

```
// 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 thresholding
C++: adaptiveThreshold(
    InputArray src, OutputArray dst, double maxValue, int adaptiveMethod,
    int thresholdType, int blockSize, double valOffset
);
```

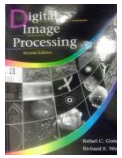


???

Server-client communications

1. Client takes an input image

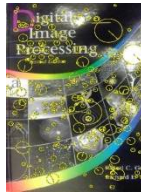
2. Send an image to server



HTTP

3. A PHP server invokes application

4. Server-side application processes the image



5. Client receives the processed result

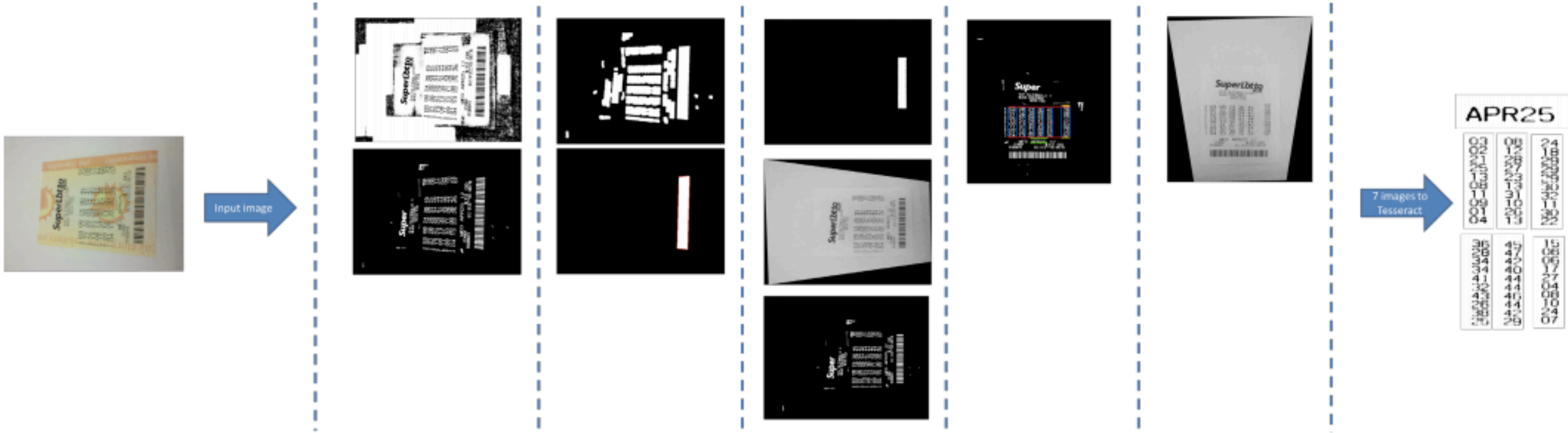
Server-client communications

- Covers every aspect of server-client interactions
  - How to upload an image from the phone to a server
  - How to invoke applications on the server after the image is received
  - How to download the result onto the phone from the server
- Available as Tutorial #3 on class website:
  - http://ee368.stanford.edu/Android

Class project: Previewing interior décor changes

M. Le, A. Zarraga, K. Zhu, Spring 2011
http://ee368.stanford.edu/Project_11

Class project: Mobile lottery ticket checker



T. Zou and A. Gupta, Spring 2012
http://ee368.stanford.edu/Project_12

???