EMBEDDED SYSTEMS PROGRAMMING 2014-15

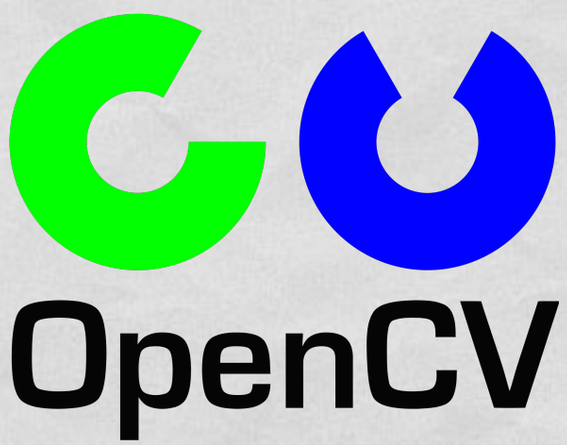
OpenCV

OPENCV

- "OpenCV" = "Open Computer Vision": library of computer vision algorithms, free and open source

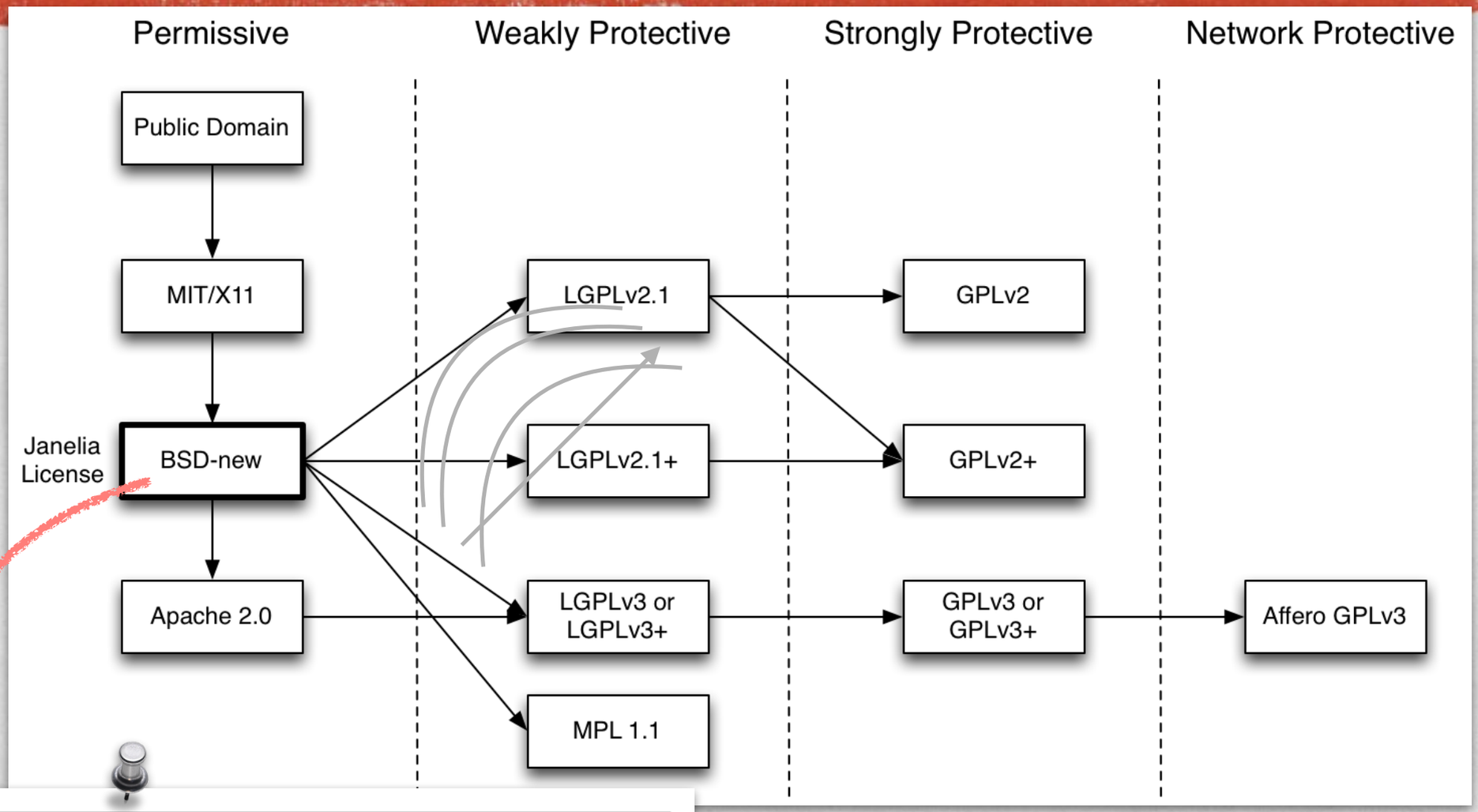
- Cover machine learning as well

- Supports Windows, Mac OS, Linux, Android, iOS



- Written C++;
  interfaces also in C, Python, Java, MATLAB

OPENCV: LICENSE (1/2)




```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of the <organization> nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.
```

OPENCV: LICENSE (2/2)

Modified BSD license ("BSD-new")

- Redistribution and use in source and binary forms, with or without modification, are permitted
- (Yes, the license does not require to publish modified source code)
- Redistributions must reproduce the OpenCV copyright notice, advertising material must not

OPENCV: APPLICATIONS

- Facial recognition
- Gesture recognition and human-computer interaction in general
- Augmented reality
- Motion tracking
- Mobile robotics

OPENCV: MODULES (1/4)

- **core**: contains all of the basic object types and their basic operations
- **imgproc**: basic transformations on images, filters and convolutional operators
- **highgui**: platform-independent GUI and media I/O functions to read/display images, or to get user input
- **video**: motion analysis and object tracking
- **calib3d**: camera calibration and 3D reconstruction

OPENCV: MODULES (2/4)

- **features2d**: algorithms for detecting, describing and matching keypoint features
- **objdetect**: algorithms for detecting specific objects (requires training)
- **ml**: algorithms for statistical classification, regression and clustering of data
- **flann**: clustering and search in multi-D spaces (mostly used by other OpenCV modules)

OPENCV: MODULES (3/4)

- **photo**: computational photography (inpainting, image denoising)
- **stitching**: image stitching pipeline
- **superres**: a few algorithms to enhance image resolution by exploiting the information in multiple frames
- **viz**: functions for 3D visualization

OPENCV: MODULES (4/4)

- **gpu**: CUDA-accelerated computer vision
- **ocl**: OpenCL-accelerated computer vision
- **contrib**: contributed/experimental functionality
- **nonfree**: non-free (e.g., patented) functionality
- **legacy**: deprecated functionality

CORE

- **Data structures** (points, arrays, matrices…)
- **Operations on data structures**
  including DCT, DFT, sorting, PCA, SVD, eigenvalues, eigenvectors
- **Clustering, partitioning**
- **Drawing functions**
- **XML/YAML file storage**

COMMON DATA TYPES

- **`Point`**: 2-D point
- **`Point3`**: 3-D point
- **`Scalar`**: 4-element vector
- **`Mat`**: *n*-dimensional dense array
  (i.e., *n*-dimensional dense matrix)
- In C++ they are template classes,
  in Java elements are doubles

IMGPROC (1/3)

- **Image filtering**
  - Blur, smooth, noise reduction, erosion/dilation
  - Computing image derivatives, edge detection
  - Convolution with a kernel
- **Image transformations**
  - Affine transform, perspective transform, resize
  - Rectification, compensation of lens distortion
  - Linear-polar space conversion, color space conversion
  - Flood fill, thresholding, image segmentation

IMGPROC (2/3)

- **Histograms**
- **Structural analysis and shape descriptors**
  - Contours: detection, calculation of lengths and areas
  - Image moments
  - Approximation of a polygonal curve
  - Fit with a line, bounding box, convex hull, circle, ellipse

IMGPROC (3/3)

- **Motion analysis and object tracking**
  - Accumulation of images (with weights)
  - Phase correlation
- **Feature and object detection**
  - Corner detection
  - Edge detection
  - Detection of lines and circles
  - Detection of an arbitrary template

HIGHGUI

- **Reading/writing image and video files**
  Several image formats: BMP, DIB, JPG, J2K, PNG, TIFF, PBM, … Supported video codecs are platform-dependent.
  Images are read from / stored to matrices

- **Video capturing** from cameras

- **GUI functions for windows and mouse**

VIDEO

- **Optical flow**
- **Rigid transform estimation**
- **Motion estimation** of a given silhouette
- **Motion splitting** into separate independent motions
- **Background/foreground segmentation**

FEATURES2D

- **Common data structures**
  E.g., the `KeyPoint` class: models a salient point
- **Feature detection** (several algorithms)
- **Blob detection**

OBJDETECT

- Haar feature-based cascade of boosted classifiers
- Discriminatively trained latent SVM

- Both detectors require training
- Both must must be applied repeatedly to the image at different positions and scales

ML (1/3)

- **Normal Bayes classifier**
  Assumes that feature vectors from each class are normally distributed

- **K-nearest neighbors** (K-NN)
  Classification outcome is determined by analyzing *K* of the nearest neighbors of the sample

- **Support vector machine** (SVM)
  Both 2-class and *n*-class ($n \geq 2$) datasets are supported

ML (2/3)

- **Expectation maximization** (EM)
  Iterative algorithm for maximum-likelihood estimates

- **Neural network**
  Feed-forward multi-layer neural network.
  Three common activation functions are supported

- **Decision tree**
  Both ordered and categorical variables are possible

ML (3/3)

- **Random trees, extremely randomized trees**
  Ensemble classifiers based on decision trees

- **Boosting**
  ML technique to addresses misclassified instances with a multi-step training procedure

- **Gradient boosted trees**
  Classification algorithm based on decision trees and boosting. *n*-class ($n \geq 2$) datasets are supported

OPENCV & JAVA

- Only a subset of modules: **core, imgproc, highgui, video, calib3d, features2d, objdetect, ml, photo.** (Modules mapped to Java packages)
- Only a subset of algorithms/functions in such modules
- Additional **utils** package for data types conversion. Additional **android** package: more about it later
- Documentation is lacking

CORE CLASS

- Available only in Java
- Part of the `core` package

- Collects, as static members, a large set of C++ functions for the manipulation of `Mat`s

CORE CLASS: EXAMPLES

- Basic, **element-wise operations**
  (e.g., calculating the absolute value of all `Mat` elements)
- **Matrix operations**
  (calculating determinant, eingenvalues, PCA, SVD, …)
- **Operations on pairs of `Mats`** (e.g., adding)
- **Transforms** (e.g., DCT, DFT)
- **"Image" operations** (e.g., flip a matrix along a row/column)

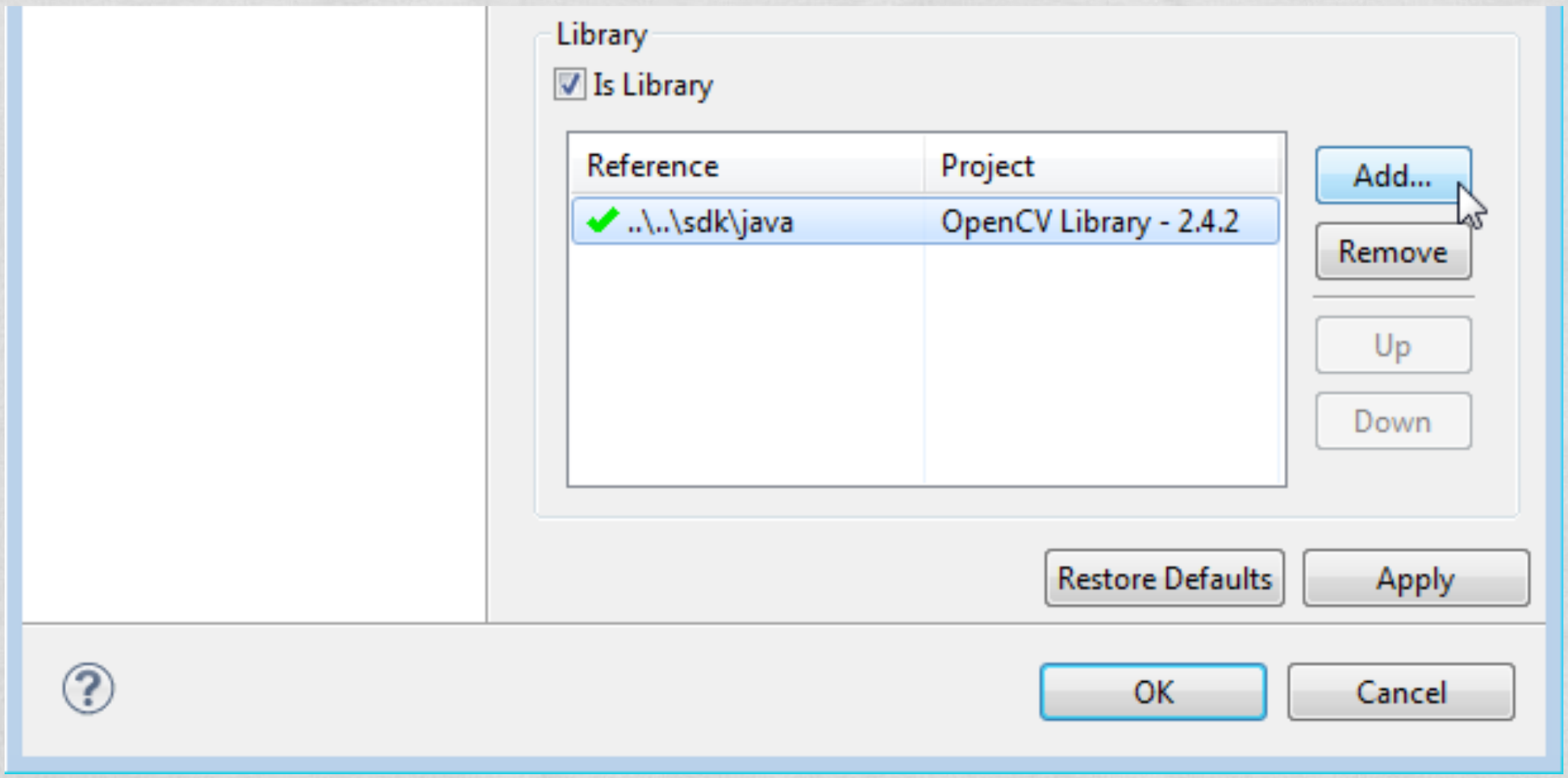
OPENCV & ANDROID

- **android** module
  - Interfaces and classes for initialization (more on initialization later)
  - Interfaces and classes to interact with the camera
  - `Utils` class for `mat` ↔ `Bitmap` conversion
  - `FpsMeter` class to calculate FPS and display result

- If the Java interface is limiting you, switch to the NDK and access the full-fledged OpenCV

OPENCV: ANDROID LIBRARY

- OpenCV library must be added to your project



OPENCV & NDK (1/2)

- Include OpenCV .mk file into `Android.mk`

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
include /path/OpenCV-2.4.11-android-sdk/sdk/native/jni/OpenCV.mk

LOCAL_MODULE    := HelloOCV_JNI
LOCAL_SRC_FILES := HelloOCV_JNI.cpp

include $(BUILD_SHARED_LIBRARY)
```

OPENCV & NDK (2/2)

- The file `Application.mk` must exist and contain variables `APP_STL`, `APP_CPPFLAGS`, `APP_ABI`

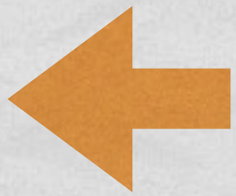
```
# The ARMv7 is significantly faster
# due to the use of the hardware FPU
APP_ABI := armeabi armeabi-v7a

APP_PLATFORM := android-8

APP_STL := gnustl_static
APP_CPPFLAGS := -frtti -fexceptions
```

ANDROID: INITIALIZATION

- **Static**: the app package contains all OpenCV binaries (actually, several copies of them: one for each supported platform)
  - Uses tens of MBs per app
  - App update required when a new OpenCV version is out
- **Dynamic**: OpenCV binaries contained in the auxiliary app **OpenCV Manager**



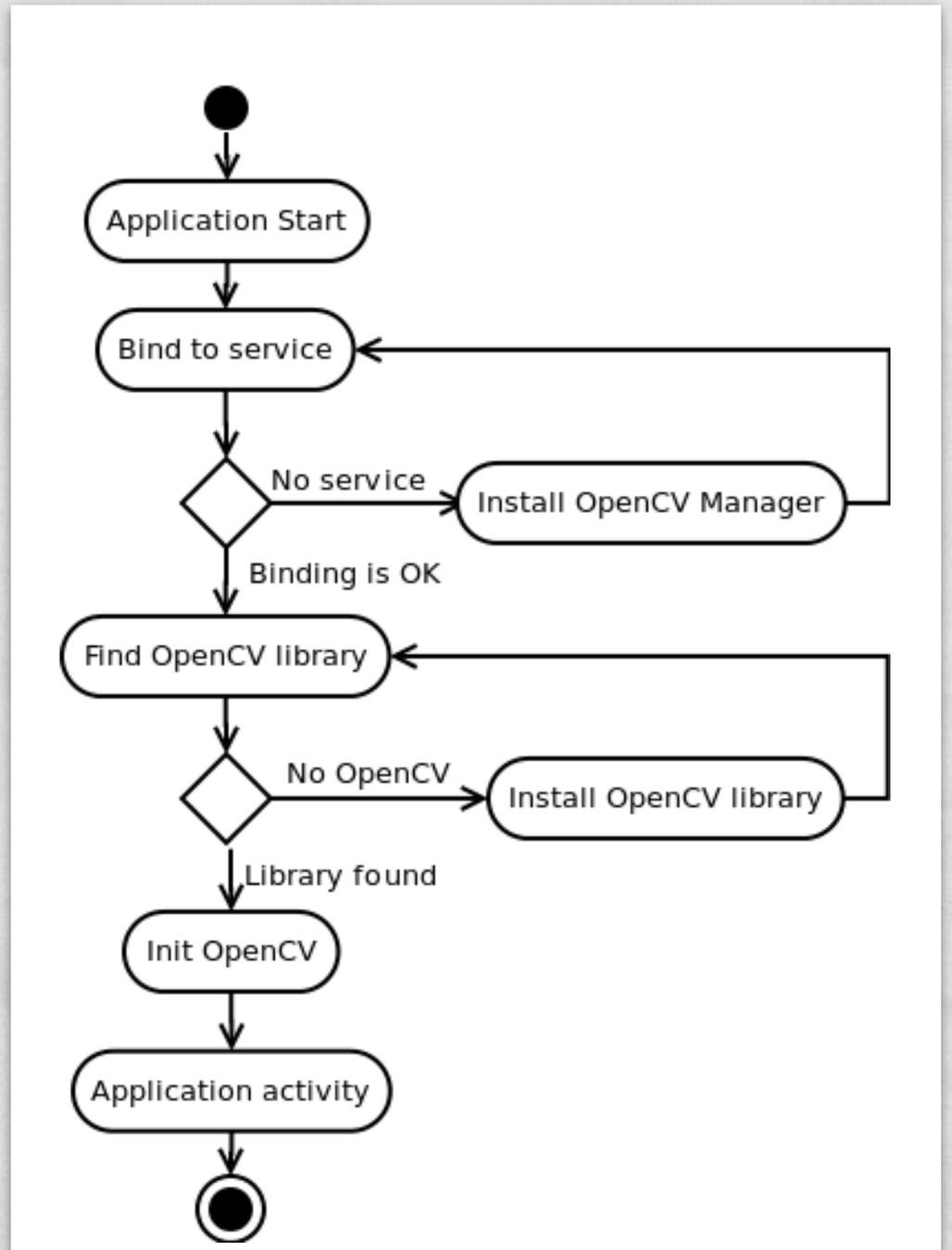
OPENCV MANAGER

- Contains all versions of OpenCV;
  automatic HW acceleration on supported platforms
- Supports both Java and native code

- Must be installed separately

- Available in the Play Store
  and in the OpenCV distribution package

INIT WITH OCV MANAGER



OPENCVLOADER CLASS

- Java class providing common initialization methods

- **`static boolean initDebug()`**
  Static initialization

- **`static boolean initAsync(String Version, Context AppContext, LoaderCallbackInterface Callback)`**
  Dynamic initialization with OpenCV version `version`.
  Returns true if initialization started successfully

LOADERCALLBACKINTERFACE

- Java interface that specifies how initialization must be managed

- **`void onManagerConnected(int status)`**
  Called after an attempt to connect to OpenCV Manager has been made. The initialization status can be `SUCCESS`, `INCOMPATIBLE_MANAGER_VERSION`, `INSTALL_CANCELED`, `MARKET_ERROR`, `INIT_FAILED`

- **`void onPackageInstall(InstallCallbackInterface Callback)`**
  Called when package installation is needed

BASELOADERCALLBACK CLASS

- Java class implementing `LoaderCallbackInterface`

- Designed to work inside an activity;
  use inside a service requires modifications

- Calls `Activity.finish()` method to exit in case of initialization failure

INIT: EXAMPLE (1/2)

```
public class MyActivity extends Activity implements LoaderCallbackInterface
{
private BaseLoaderCallback mOpenCVCallBack = new BaseLoaderCallback(this)
{
    @Override
    public void onManagerConnected(int status)
    {
        switch (status)
        {
         case LoaderCallbackInterface.SUCCESS:
         {
             Log.i(TAG, "OpenCV loaded successfully");
             // Create and set View
             mView = new puzzle15View(mAppContext);
             setContentView(mView);
          } break;
          default:
          {
             super.onManagerConnected(status);
          } break;
        }
    }
};
```

INIT: EXAMPLE (2/2)

```
/** Call on every application resume **/
@Override
protected void onResume()
{
    Log.i(TAG, "Called onResume");
    super.onResume();

    Log.i(TAG, "Trying to load OpenCV library");
    if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_6,
        this, mOpenCVCallBack))
    {
        Log.e(TAG, "Cannot connect to OpenCV Manager");
    }
}
```

REFERENCES

- OpenCV C++ API reference
- OpenCV Java API reference
- OCV4Android reference
- OpenCV cheat sheet (C++)

LAST MODIFIED: APRIL 27, 2015