OpenCV API

A Quick Introduction to the C Interface
By David Johnston
Warning: API Version 1.x
Primary OpenCV Interfaces

The 1.x API is based on C.
The 2.x API is based on C++. 
Goals

Very briefly outline the OpenCV installation process on Windows, OS X, and Linux.

Walk through 4 example programs which should be relevant to solving homework 2.

Point out some common API patterns and idioms.
Installing OpenCV
Windows 7 64-bit and Visual C++ 2010 Express


- Download prebuilt binaries and extract them to some desired location.

- Set the OPENCV_DIR environment variable:

  > setx -m OPENCV_DIR C:\opencv\build\x64\vc10
Windows 7 64-bit and Visual C++ 10.0 Express

- Add `%OPENCV_DIR%\bin` to your system path.
- Modify Visual Studio properties to find necessary files. The following describes this process in great detail:

docs.opencv.org/doc/tutorials/introduction/windows_visual_studio_OpenCV/windows_visual_studio_OpenCV.html
Install on Mac OS X

- Install XCode or the Apple Command Line Tools. Both are available with a (free) Apple ID.
- Install a package manager such as macports or homebrew.
- Install OpenCV.
Install on Linux

- Install via native package manager (ie. apt-get, yum, etc.)
Build on UNIX-Like System

cc `pkg-config --cflags --libs\opencv` -o foo foo.c
Four Code Demos
Some Constructors for Important Data Structures

\texttt{CvMat} \quad \texttt{cvMat(...)}

\texttt{CvMat*} \quad \texttt{cvCreateMat(...)}

\texttt{CvMat*} \quad \texttt{cvCreateMatHeader(...)}

\texttt{IplImage*} \quad \texttt{cvCreateImage(...)}

\texttt{CvSeq*} \quad \texttt{cvCreateSeq(...)}
Ex 1: Image Workflow

Loading, modifying, saving, and closing an image file.

See filter.c.
Macros

OpenCV defines a lot of macros.

- Most are prefixed with CV_*
- Many are function-specific
- The online documentation is usually pretty clear
In-Place Matrix Operations

cvNot(img, img);
Manual Memory Management

IplImage* img = cvLoadImage(...);

/* do something worthwhile */

cvReleaseImage(&img);
Ex 2: Basic Morphology and Color

Creating basic structuring elements and calling `cvErode()` and `cvDilate()`.

Using binary images (bit masks) and `cvSet()` to color regions of an image.

See `noteSeg.c`. 
Constructors of Helper Data Structures

Simple ideas wrapped inside a data type:

- CvPoint cvPoint(int x, int y)
- CvSize cvSize(int width, int height)
- CvScalar cvScalar(double d0, double d1, double d2, double d3)
- CvScalar cvScalarAll(double d)
- CvScalar cvRealScalar(double d)
- CvRect cvRect(int x, int y, int width, int height)
Ex 3: Horizontal and Vertical Projections

Use `cvGetRow()`, `cvGetCol()`, and `cvSum()` to perform very simple projections.
Sort of Object Oriented

The most important data structure is arguably *IplImage*.

*Figure 3-1. Even though OpenCV is implemented in C, the structures used in OpenCV have an object-oriented design; in effect, IplImage is derived from CvMat, which is derived from CvArr*
Sort of Object Oriented

The `IplImage` and `CvMat` data structures are just metadata which provide interfaces to the underlying image data.

See `projections.c`. 

The `IplImage` and `CvMat` data structures are just metadata which provide interfaces to the underlying image data.

See `projections.c`. 

The `IplImage` and `CvMat` data structures are just metadata which provide interfaces to the underlying image data.

See `projections.c`.
Ex 4: Custom Morphology

Creating custom structuring elements using `cvCreateStructuringElementEx()` and an `int[]`. 
Warning: Pointer Arithmetic is Imminent!

Examples and discussion can be found in the text (highly recommended reading):

- cvMat and IplImage: pp. 31 - 47
- cvSeq: pp. 222 - 234

See customStructElem.c.
Questions?
CvSize size = cvSize(600, 400);
IplImage* img = cvCreateImage(
    size, IPL_DEPTH_8U, 3
);
cvSet(img, cvScalarAll(0), NULL);
Why isn't every call to these helper functions a memory leak?
"The keyword inline is a request to the compiler to insert a function's machine code wherever the function is called in the program."

- Page 106 of *C in a Nutshell*, by Prinz and Crawford

Automatic variables declared in an inline function become automatic variables in the calling function.