OpenCV Tutorial

Part IV
A Brief Guide to Memory Management
(and other Miscellaneous Functions)

02 December 2005

Gavin S Page
gsp8334@cs.rit.edu

Introduction

Why is Managing OpenCV Objects Important?

•Video, 30 frames per second
•Each frame is an image
•Images are arrays of pixels
•A 640x480 image is 307,200 pixels
•These must be represented in memory
•How much memory does your machine have?

A Top-Level OpenCV Type

*CvArr** is a function parameter for several OpenCV functions which accept arrays of more than one type. These are often *IplImage*, CvMat*, or CvSeq*.*

How Does OpenCV Represent Images?

IplImage: Structure from *Intel Image Processing Library*
In addition to representing the image data *IplImage* holds utilizes a subset of IPL data useful in IP/CV:
•*nSize* : Size of Image
•*nChannels* : Number of Image Channels (1-4)
•*width, height*
•ROI : Region of Interest (used in Tutorial II)
• and others (see documentation)…

Handling IplImage

Creating an IplImage

•*cvCreateImage*(CvSize size, int depth, int channels);
•header = *cvCreateImageHeader*(size,depth,channels);
cvCreateData(header);
•*cvCloneImage*(const IplImage* image);
•*cvLoadImage*(const char* filename, int iscolor=1);

•The first two functions are useful for creating a blank image of the specified parameters. A possible use is in functions that require a pointer to a result.
•The clone function performs an exact copy of the *IplImage** parameter.
•The load function loads an image from a file.

When allocating *IplImage* in a loop be sure to deallocate in the loop as well

Destroying an IplImage

•*cvReleaseImage(IplImage** image);*
•*cvReleaseData(*image);*
cvReleaseImageHeader(image);

• *cvReleaseImage* will work for the 3 single step creation functions
•The *cvReleaseData*/*cvReleaseImageHeader* combination is used when there is separate data and header information

An image header is initialized using *cvInitImageHeader.*

Utilizing IplImage

The *IplImage* structure makes it possible to target specific regions of an object for processing. This reduces overhead caused by working on the whole image. The selection can occur at both the channel and the region.

Setting the Region of Interest (ROI)

- cvSetImageROI(IplImage* image, CvRect rect);
- cvResetImageROI(IplImage* image);
- cvGetImageROI(const IplImage* image);

Setting the ROI of the image allows the user to select a rectangular region of the image to work with. This is useful after localizing objects for extraction and further processing. While the region is set the rest of the image will be ignored. Meaning any operation directed on the image will act on only the region (including *cvShowImage*).

Setting the Channel of Interest (COI)

- cvSetImageCOI(IplImage* image, int coi);
- cvGetImageCOI(const IplImage* image);

Setting the channel of the image allows the user to work with a particular layer of the image. i.e. The ‘R’ layer of an RGB image or the ‘V’ layer in the HSV format.
NOTE: Not all OpenCV functions support this.

The *CvRect* function is used to specify the region in *cvSetImageROI.*

cvRect(int x, int y, int width, int height);

Other Static Array Types

CvMat

•*cvCreateMat(int rows, int cols, int type);*
•*mat = cvCreateMatHeader(rows, cols, type);*
cvCreateData(mat);
•*cvCloneMat(const CvMat* mat);*
•*cvReleaseMat(CvMat** mat);*

OpenCV also has built in functions for mult-dimensional arrays (CvMatND) and sparse arrays (CvSparseMat).

OpenCV uses the *CvMat** as its general purpose matrix structure. It is managed in an equivalent style to *IplImage**

Specifying the type of a *CvMat* is done using the syntax
CV_<bit_depth>(S|U|F)C<number_of_channels>
i.e. CV_8UC1 for an 8-bit single channel unsigned

Gavin S Page
gsp8334@cs.rit.edu

Getting Matrix Information From an IplImage

In order for a matrix to be useful it must be populated with data. OpenCV makes it possible to fill a matrix with data from an *IplImage*.

Extracting Matrix Region

```
CvRect rect = cvRect(0, 0, 500, 600 );
CvMat* mt = cvCreateMat(500,600, CV_8UC1);
CvMat* sRect = cvGetSubRect(grayImage,mt,rect);
```

The actual parameters of the *cvGetSubRect* function are *(const CvArr* arr, CvMat* submat, CvRect rect)* .
This snippet illustrates how to copy matrix header information from the *IplImage*. The function does make use of ROI so this will be useful in specifying a target region.

Dynamic Arrays

CvSeq

•*cvCreateSeq(int seq_flags, int header_size, int elem_size, CvMemStorage* storage);*

CvMemStorage is a low-level structure used to store dynamic data objects.

CvMemStorage

• *cvCreateMemStorage(int block_size=0);*
•cvClearMemStorage(CvMemStorage* storage)

OpenCV uses the *CvSeq** to as its own representation for growable 1-d arrays. It is similar to *IplImage** with regard to the fact that it is a structure with multiple fields which are representative of the data content. This includes a pointer to *CvMemStorage* which actually holds the sequence.

The sequence is released by clearing the associated CvMemStorage structure.

Final Message

As with any C++ program it is important to destroy all memory that has been allocated.