Design and Application of DSP

- Atom Processor-

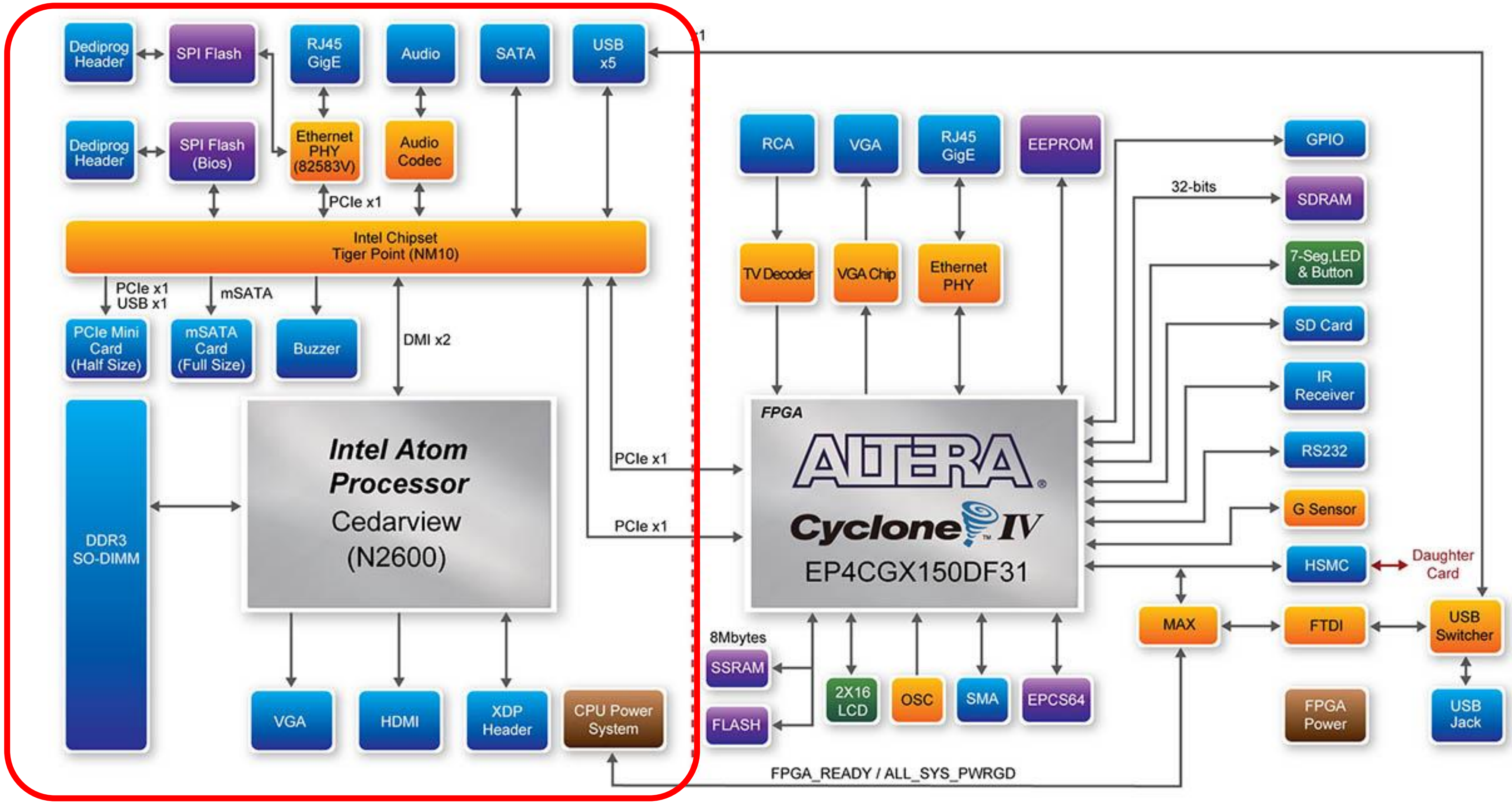
UNIVERSITY OF WASHINGTON
ELECTRICAL ENGINEERING

DE2i-150 Board

- CPU : Intel® Atom™ Dual Core Processor N2600(1M Cache, 1.6GHz)
  - Intel® Hyper-Threading Technology(4 execution threads)
  - Intel SpeedStep® Technology
  - Instruction Set : 64-bit
  - Instruction Set Extensions : SSE2, SSE3, SSSE3
  - Integrated Graphics
  - Graphics Base Frequency : 400MHz

- OS : Ubuntu Linux

Atom Processor

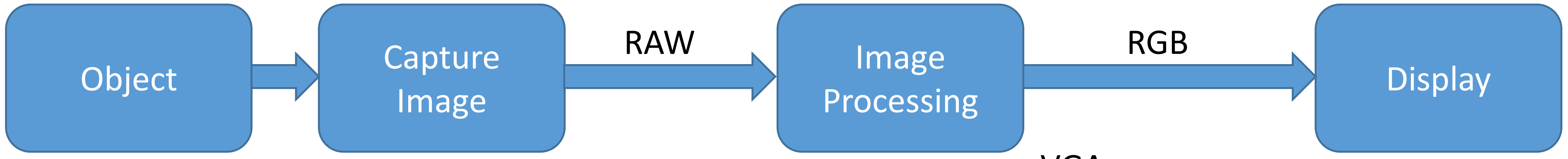


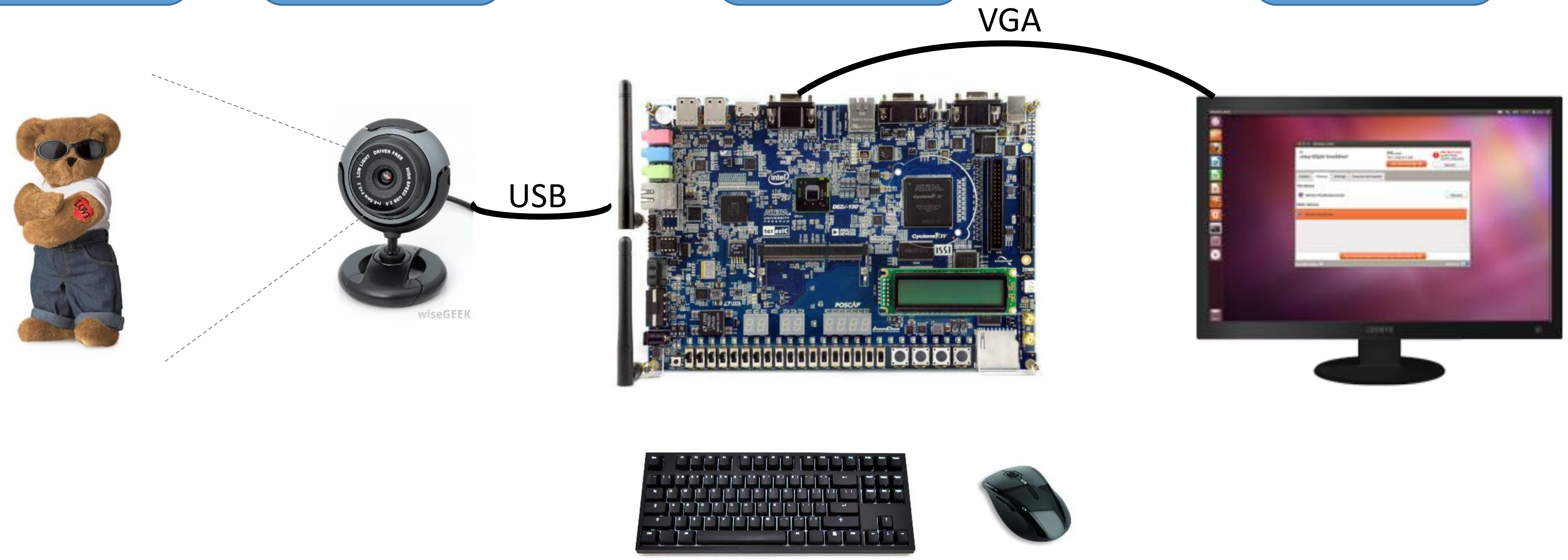
Embedded Linux

- Operating systems based on the Linux kernel are used in embedded systems such as consumer electronics
  - Set-top boxes, smart TVs, in-vehicle infotainment (IVI), networking equipment (such as wireless routers), machine control, industrial automation, navigation equipment, spacecraft flight software, and medical instruments in general

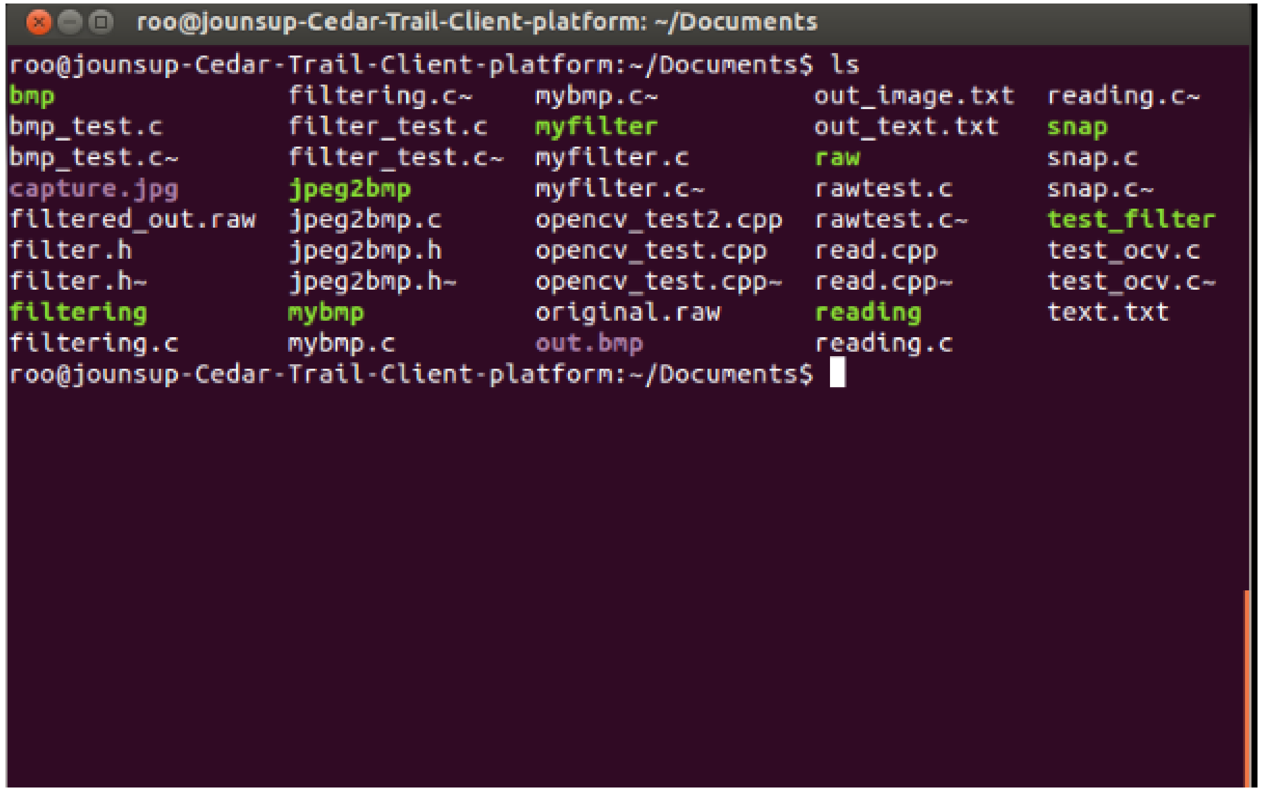
- Products
  - Google's Android (operating system) well-known type of embedded Linux
  - Maemo - embedded Linux for smartphones
  - Debian - used on Raspberry Pi
  - Emdebian Grip
  - BusyBox
  - OpenMoko
  - Familiar Linux
  - Mobilinux

Configurations





Terminal



roo@jounsup-Cedar-Trail-Client-platform: ~/Documents

```
roo@jounsup-Cedar-Trail-Client-platform:~/Documents$ ls
bmp               filtering.c~     mybmp.c~          out_image.txt  reading.c~
bmp_test.c        filter_test.c    myfilter          out_text.txt   snap
bmp_test.c~       filter_test.c~   myfilter.c        raw            snap.c
capture.jpg       jpeg2bmp         myfilter.c~       rawtest.c      snap.c~
filtered_out.raw  jpeg2bmp.c       opencv_test2.cpp  rawtest.c~     test_filter
filter.h          jpeg2bmp.h       opencv_test.cpp   read.cpp       test_ocv.c
filter.h~         jpeg2bmp.h~      opencv_test.cpp~  read.cpp~      test_ocv.c~
filtering         mybmp            original.raw      reading        text.txt
filtering.c       mybmp.c          out.bmp           reading.c
roo@jounsup-Cedar-Trail-Client-platform:~/Documents$ 
```

Keyboard shortcuts and special characters:
(^X means hold Ctrl key and press X)

key / character	**description**
Up arrow	repeat previous command(s)
Home/End or ^A/^E	move cursor to start/end of line
*	"wildcard", matches any file(s)
Tab	auto-completes a partially typed file/directory/command name
^C or ^\	kills the currently running process
^D	end-of-input; press this if a program is reading input from your keyboard and you want to notify it that you are finished
^Z	suspends (pauses) the currently running process; use fg or bg to resume it
^S	never ever press this; worst hotkey ever; totally locks up your shell until you press ^Q

Commands

directories	ls	list files in a directory
	pwd	displays the shell's current working directory
	cd	changes the shell's working directory to the given directory; can be a relative or absolute path
	mkdir	creates a new directory with the given name
	rmdir	removes the directory with the given name (the directory must be empty)
file operations	cp	copies a file/directory
	mv	moves (or renames) a file/directory
	rm	deletes a file
	touch	update the last-modified time of a file (or create an empty file)

sudo

- The sudo command is used to perform file operations on files that the Root User would only be allowed to change.
- An example would be trying to move one of your documents that another user accidentally moved to / back to your documents directory. Normally, to move the file, you would type
  - mv /mydoc.odt ~/Documents/mydoc.odt,
  - but you are not allowed to modify files outside of your home directory. To get around this, you would type
  - sudo mv /mydoc.odt ~/Documents/mydoc.odt.
  - This will successfully move the file back to its correct location, provided that you are not a standard user, who has less (administrative) ability than an administrator. Be aware, though, that by using the sudo command, you need to be extra careful. It is easier to damage your system by using the sudo command.

apt-get

- The *apt-get* command is a powerful command-line tool, which works with Ubuntu's *Advanced Packaging Tool* (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.
- Being a simple command-line tool, *apt-get* has numerous advantages over other package management tools available in Ubuntu for server administrators. Some of these advantages include ease of use over simple terminal connections (SSH), and the ability to be used in system administration scripts, which can in turn be automated by the *cron* scheduling utility.

Some examples of popular uses for the *apt-get* utility:

•**Install a Package**: Installation of packages using the *apt-get* tool is quite simple. For example, to install the network scanner *nmap*, type the following:

```
        sudo apt-get install nmap
```

•**Remove a Package**: Removal of a package (or packages) is also straightforward. To remove the package installed in the previous example, type the following:

```
        sudo apt-get remove nmap
```

- **Multiple Packages**: You may specify multiple packages to be installed or removed, separated by spaces.
- Also, adding the *--purge* option to `apt-get remove` will remove the package configuration files as well. This may or may not be the desired effect, so use with caution.

•**Update the Package Index**: The APT package index is essentially a database of available packages from the repositories defined in the`/etc/apt/sources.list` file and in the `/etc/apt/sources.list.d` directory. To update the local package index with the latest changes made in the repositories, type the following:

```
        sudo apt-get update
```

•**Upgrade Packages**: Over time, updated versions of packages currently installed on your computer may become available from the package repositories (for example security updates). To upgrade your system, first update your package index as outlined above, and then type:

```
        sudo apt-get upgrade
```

wget

- **Wget (download manager) – The non-interactive(non-gui) network downloader.**
- GNU **Wget** is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.
- **Wget** has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the server supports download-resume, it will instruct the server to continue the download from where it left off, pretty sweet for a non-gui command right?

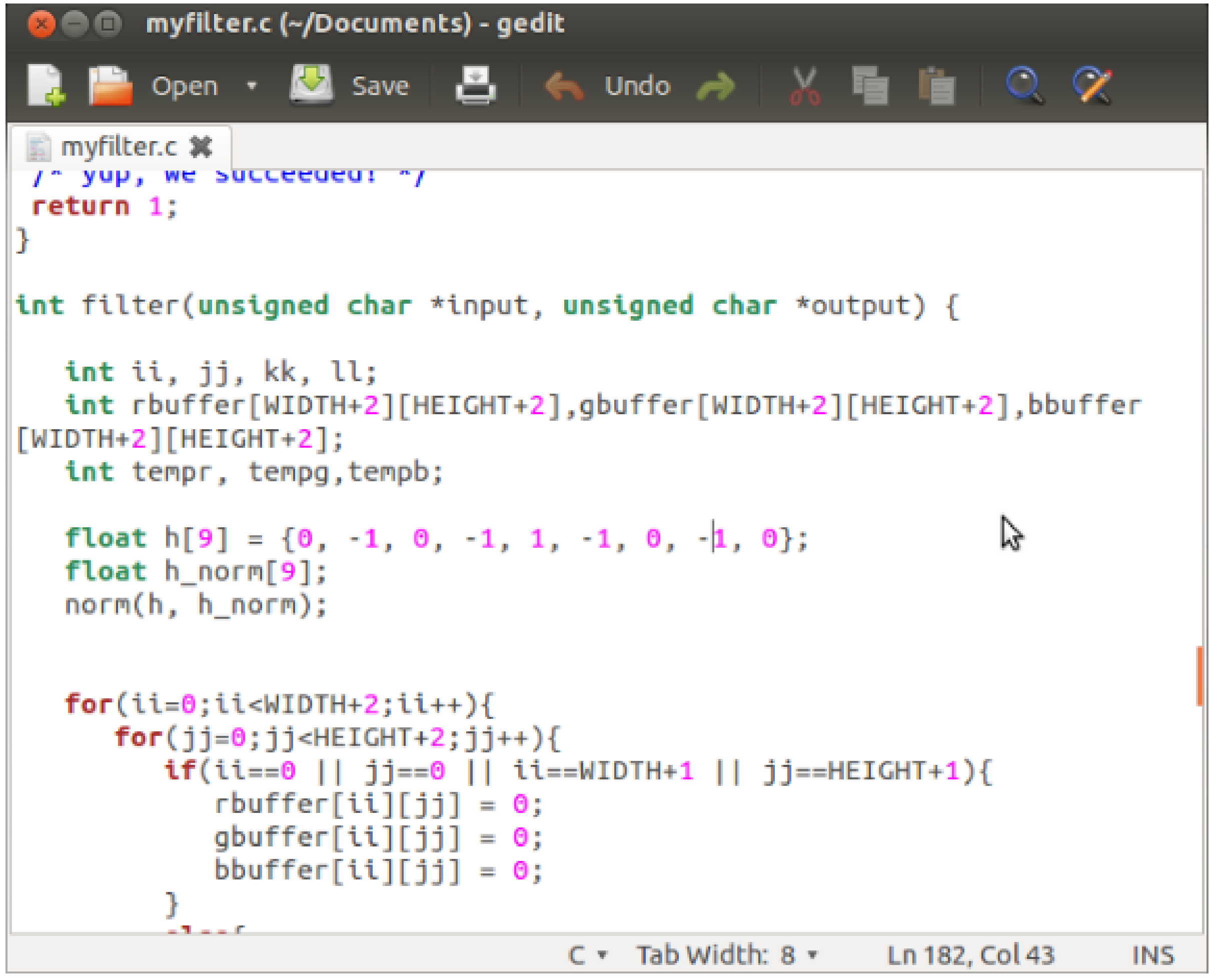
wget

- By default, Wget is very simple to invoke. The basic syntax is as follows:
  - **wget [option] [URL]**
- Whenever any downloading process is being carried out, wget will show:
  1. Progress of the download in percentage
  2. Amount of data downloaded
  3. Speed of the download process
  4. Time left to complete the download process.

How to download single file with Wget

- The folowing example explan how to download single file from internet and stores in current directory:
- wget http://releases.ubuntu.com/14.04/ubuntu-14.04-desktop-amd64.iso
- Downloading and saving single file with different name using Wget
- **wget -O [Preferred_Name] [URL]**
- **wget -O ubuntu_amd64.iso http://releases.ubuntu.com/14.04/ubuntu-14.04-desktop-amd64.iso**

gedit



```
 /* yup, we succeeded! */
 return 1;
}

int filter(unsigned char *input, unsigned char *output) {

   int ii, jj, kk, ll;
   int rbuffer[WIDTH+2][HEIGHT+2],gbuffer[WIDTH+2][HEIGHT+2],bbuffer
[WIDTH+2][HEIGHT+2];
   int tempr, tempg,tempb;

   float h[9] = {0, -1, 0, -1, 1, -1, 0, -1, 0};
   float h_norm[9];
   norm(h, h_norm);


   for(ii=0;ii<WIDTH+2;ii++){
      for(jj=0;jj<HEIGHT+2;jj++){
         if(ii==0 || jj==0 || ii==WIDTH+1 || jj==HEIGHT+1){
            rbuffer[ii][jj] = 0;
            gbuffer[ii][jj] = 0;
            bbuffer[ii][jj] = 0;
         }
```

gcc

- the GNU Compiler Collection
- Tutorial : http://pages.cs.wisc.edu/~beechung/ref/gcc-intro.html
- gcc : GNU C compiler
- g++ : GNU C++ compiler

Example 1: Compiling a simple program

- Consider the following example:
- Let "hello.C" be a file that contains the following C++ code.

```
#include "iostream.h"
int main() {
        cout << "Hello₩n";
}
```

- The standard way to compile this program is with the command

```
g++ hello.C -o hello
```

- This command compiles hello.C into an executable program named "hello" that you run by typing 'hello' at the command line. It does nothing more than print the word "hello" on the screen.
- Alternatively, the above program could be compiled using the following two commands.

```
g++ -c hello.C
g++ hello.o -o hello
```

- The end result is the same, but this two-step method first compiles hello.C into a machine code file named "hello.o" and then links hello.o with some system libraries to produce the final program "hello". In fact the first method also does this two-stage process of compiling and linking, but the stages are done transparently, and the intermediate file "hello.o" is deleted in the process.

Frequently used compilation options

- C and C++ compilers allow for many options for how to compile a program, and the examples below demonstrate how to use many of the more commonly used options. In each example, "myprog.C" contains C++ source code for the executable "myprog". In most cases options can be combined, although it is generally not useful to use "debugging" and "optimization" options together.
- Compile myprog.C so that myprog contains symbolic information that enables it to be debugged with the gdb debugger.

```
g++ -g myprog.C -o myprog
```

  - This debugging information is stored in the object file; it describes the data type of each variable or function and the correspondence between source line numbers and addresses in the executable code

- Have the compiler generate many warnings about syntactically correct but questionable looking code. It is good practice to always use this option with gcc and g++.

```
g++ -Wall myprog.C -o myprog
```

- Generate symbolic information for gdb and many warning messages.

```
g++ -g -Wall myprog.C -o myprog
```

Frequently used compilation options

- Generate optimized code on a Linux machine.

```
g++ -O myprog.C -o myprog
```

- If "myprog.c" is a C program, then the above commands will all work by replacing g++ with gcc and "myprog.C" with "myprog.c". Below are a few examples that apply only to C programs.

- Compile a C program that uses math functions such as "sqrt".

```
gcc myprog.C -o myprog –lm
```

- Compile a C program with the "electric fence" library. This library, available on all the Linux machines, causes many incorrectly written programs to crash as soon as an error occurs. It is useful for debugging as the error location can be quickly determined using gdb. However, it should only be used for debugging as the executable myprog will be much slower and use much more memory than usual.

```
gcc -g myprog.C -o myprog -lefence
```

Capture Image

- C example (camera.c)

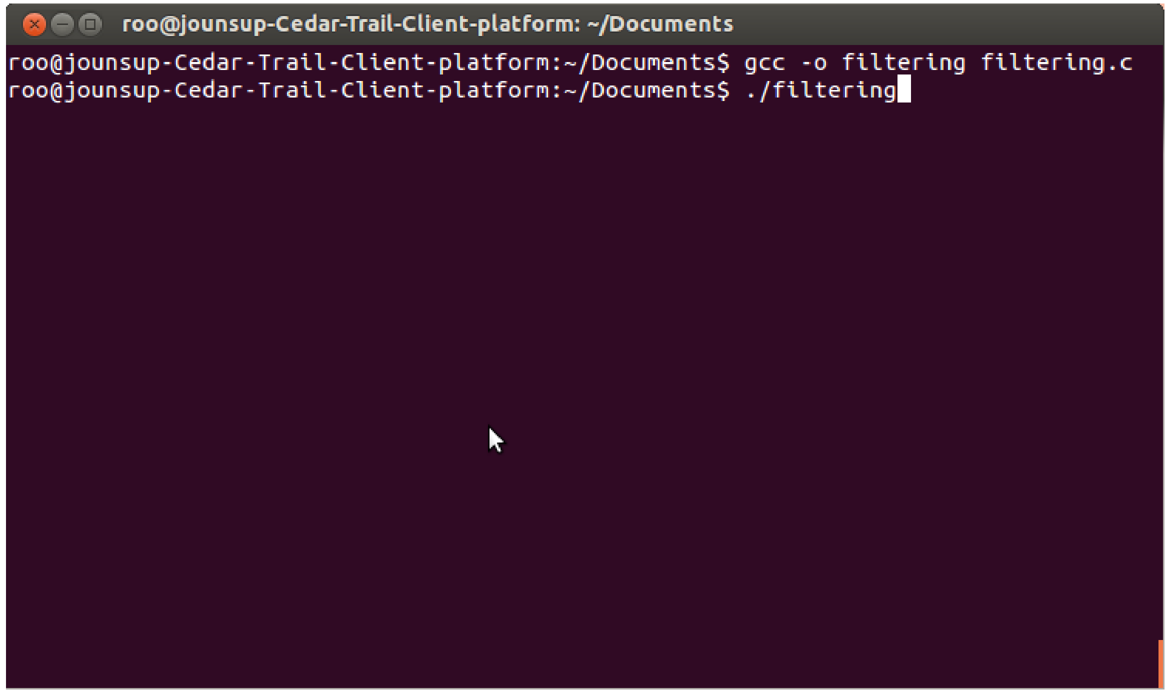
```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 307200 // number of pixels (640x480 for my webcam)
int main() {
         FILE *camera, *grab;
         camera=fopen("/dev/video0", "rb");
         grab=fopen("grab.jpg", "wb");
         unsigned char data[SIZE]; fread(data, sizeof(data[0]), SIZE, camera);
         fwrite(data, sizeof(data[0]), SIZE, grab);
         fclose(camera);
         fclose(grab);
         return 0;
}
```

Compile

- In the terminal
  - sudo apt-get update
  - sudo apt-get install libjpeg-dev
  - gcc –o original.jpg filtered.bmp -ljpeg

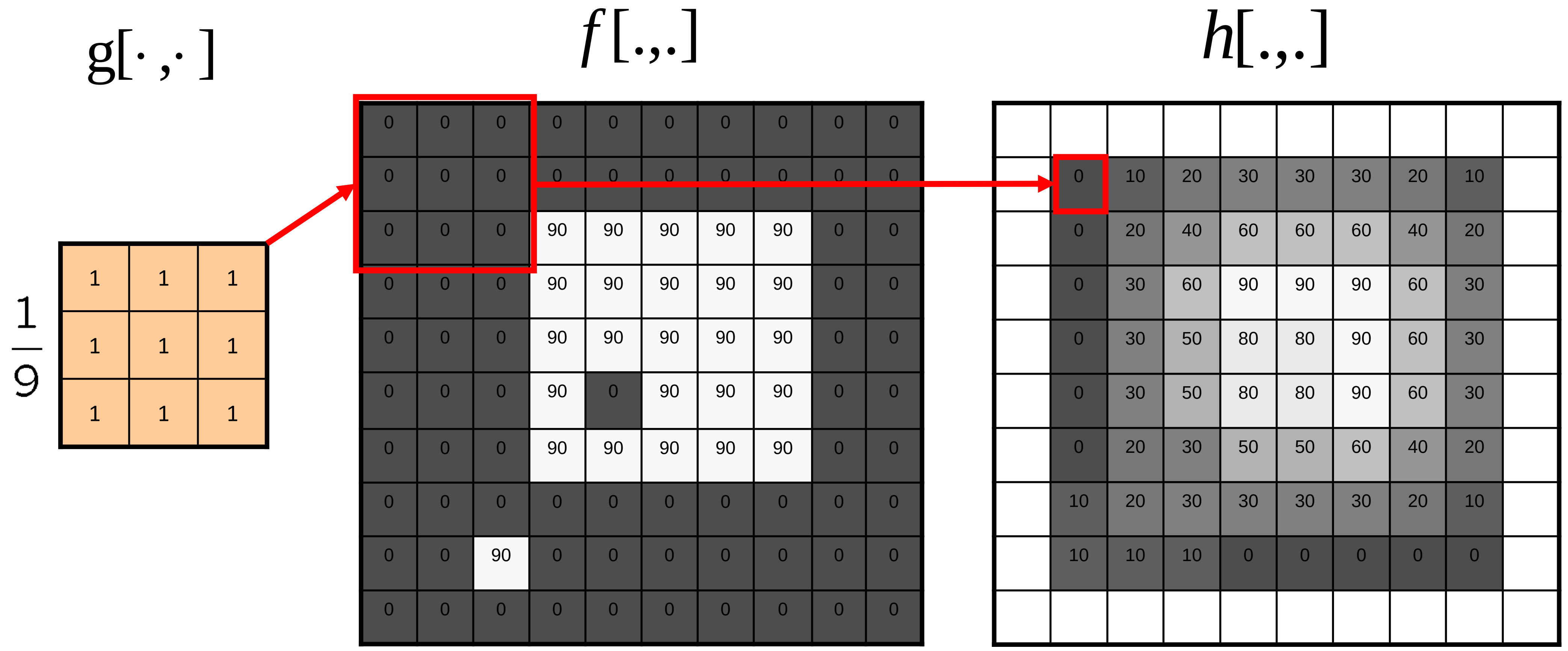
Compile

- Compile
  - ➢ gcc –o filtering filtering.c
- Execute the program
  - ➢ ./filtering



```
roo@jounsup-Cedar-Trail-Client-platform:~/Documents$ gcc -o filtering filtering.c
roo@jounsup-Cedar-Trail-Client-platform:~/Documents$ ./filtering
```

Image filtering



1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Filter design

```
int filter(float* h, unsigned char* input, unsigned char* output) {
   int ii, jj, kk, ll;
   int ibuffer[642][482];
   int data2d[640][480];
   float output2d[640][480];
```

```
for(ii=0;ii<640;ii++){
          for(jj=0;jj<480;jj++){
                       data2d[ii][jj] = (int)input[ii*480+jj];
          }
}
```

Rearrange Pixels into 2D array

```
for(ii=0;ii<642;ii++){
   for(jj=0;jj<482;jj++){
      if(ii==0 || jj==0 || ii==641 || jj==481)
          ibuffer[ii][jj] = 0;
      else
          ibuffer[ii][jj] = data2d[ii-1][jj-1];
   }
}
```

Using buffer to dealing with edges

```
for(ii=0;ii<640;ii++){
   for(jj=0;jj<480;jj++){
      output2d[ii][jj] = 0;
      for(kk=0;kk<3;kk++){
         for(ll=0;ll<3;ll++){
            output2d[ii][jj] += (float)ibuffer[ii+kk][jj+ll]*h[kk*3+ll];
         }
      }
   }
}
```

2D-convolution for filtering image

```
for(ii=0;ii<640;ii++){
          for(jj=0;jj<480;jj++){
                       output[ii*480+jj] = (unsigned char)output2d[ii][jj];
          }
}
```

Rearrange 2D array into original form

```
   return 0;
}
```

```
int norm(float* input, float* output){
             float power=0;
             int ii;
             for(ii=0;ii<9;ii++){
                          power += input[ii];
             }
             for(ii=0;ii<9;ii++){
                          output[ii] = input[ii]/power;
             }
             return 0;
}
```

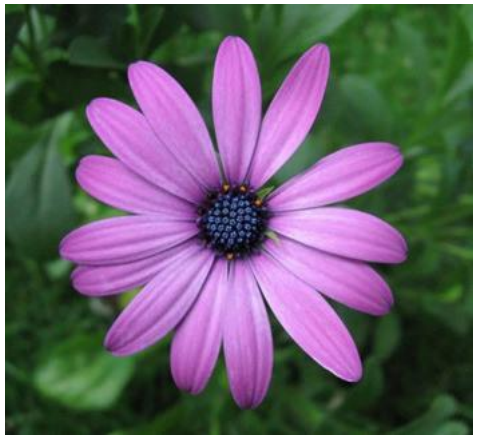
Normalize the filter coefficient

Processing the Sequence of Frames (Video)

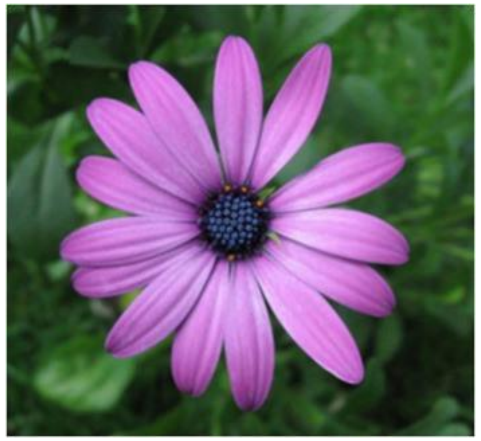
```
int main(int argc,char **argv)
{
        unsigned char *raw_image;
        unsigned char *filtered_image;

        raw_image = (unsigned char*)malloc(WIDTH*HEIGHT*BYTES_PER_PIXEL);
        filtered_image = (unsigned char*)malloc(WIDTH*HEIGHT*BYTES_PER_PIXEL);

        /* dimensions of the image we want to write */
        int width;
        int height;
        int bytes_per_pixel;   /* or 1 for GRACYSCALE images */
        int color_space; /* or JCS_GRAYSCALE for grayscale images */

        if(argc != 3){
                printf("Usage: %s original.jpg filtered.bmp\n",argv[0]);
                return -1;
        }

        FILE *camera, *grab;
        int numframe = 10;
        char nof[50], nof2[50];
        int ii;
        int data[307200];

        for(ii=0; ii<numframe; ii++){
                camera=fopen("/dev/video0", "rb");
                fread(data, sizeof(data[0]), 307200, camera);
                fclose(camera);
                sprintf(nof,"original%d.jpg",ii);
                grab=fopen(nof, "wb");
                fwrite(data, sizeof(data[0]), 307200, grab);
                fclose(grab);
                sprintf(nof2,"filtered%d.bmp",ii);
                /* Try opening a jpeg*/
                if(read_jpeg_file(nof, raw_image) > 0 ){
                  filter(raw_image, filtered_image);
                  write_bmp_file(nof2, filtered_image);
                }else return -1;
        }

        free(raw_image);
        free(filtered_image);

        return 0;
}
```

Lowpass Filters

Original

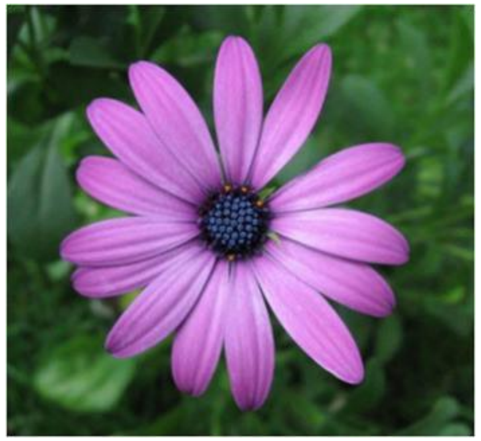


Mean 5x5



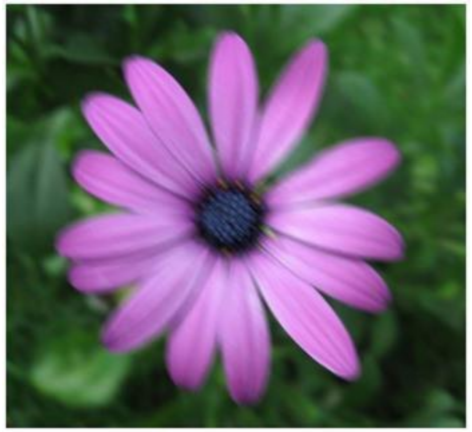
$$\begin{pmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{pmatrix}\cdot\left(\frac{1}{25}\right)$$

Gaussian 5x5



$$\begin{pmatrix}2&4&5&4&2\\4&9&12&9&4\\5&12&15&12&5\\4&9&12&9&4\\2&4&5&4&2\end{pmatrix}\left(\frac{1}{159}\right)$$

Motion blur 9x9 135 degree

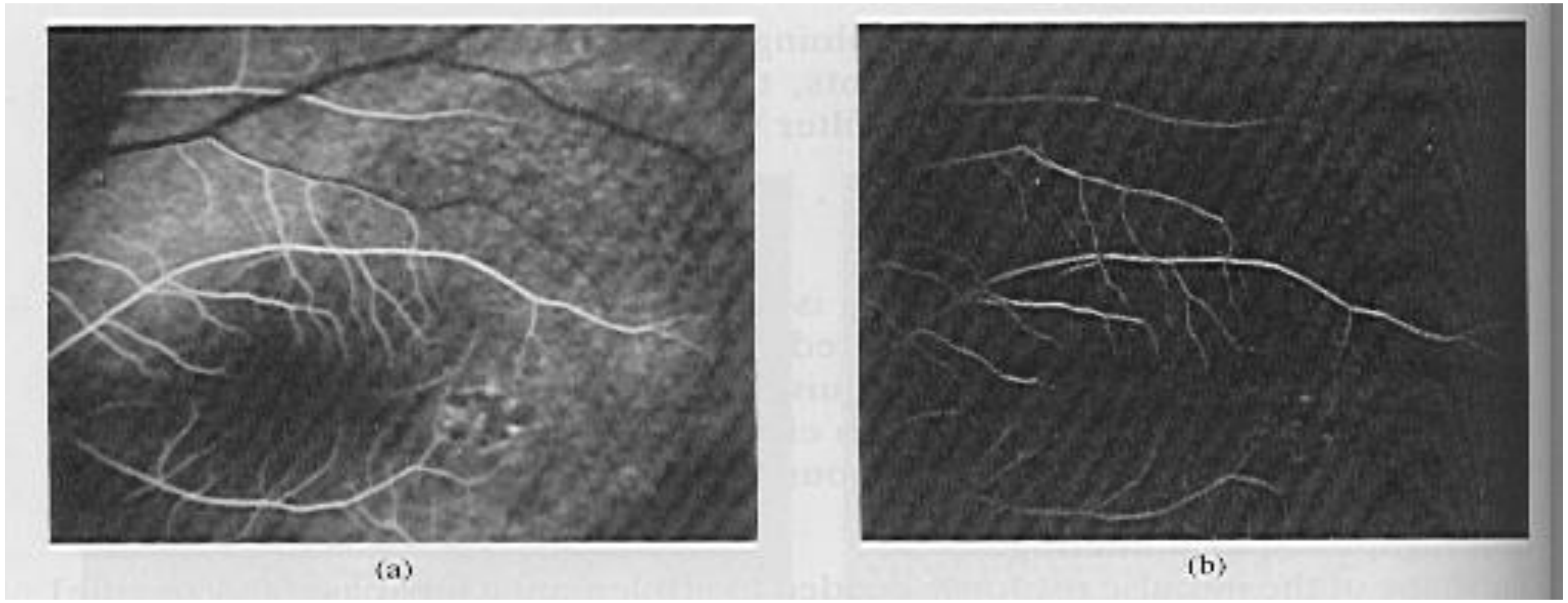


$$\begin{pmatrix}1&0&0&0&1\\0&1&0&1&0\\0&0&1&0&0\\0&1&0&1&0\\1&0&0&0&1\end{pmatrix}\left(\frac{1}{9}\right)$$

Highpass filters

Original

Highpass 5x5



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

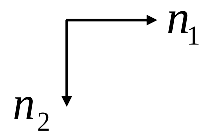
Derivative Operators

- An **averaging** is analogous to **“integration”**, **“differentiation”** can be expected to have the opposite effect: **sharpening**.
- **Differentiation** (gradient) of $f(x,y)$

$$\nabla f = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \qquad mag(\nabla f) = [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$$

$$or \quad mag(\nabla f) \approx |\frac{\partial f}{\partial x}| + |\frac{\partial f}{\partial y}|$$

- In **discrete** domain: $\frac{\partial f(n_1,n_2)}{\partial n_1} \approx f(n_1,n_2) - f(n_1-1,n_2)$



$$\approx f(n_1+1,n_2) - f(n_1,n_2)$$

$$\approx \frac{1}{2}\left[f(n_1+1,n_2) - f(n_1-1,n_2)\right]$$

Derivative Operators

- Typically, we want to average over several neighboring rows, i.e.,

$$\frac{\partial f(n_1,n_2)}{\partial n_1} \approx \left[f(n_1+1,n_2-1) - f(n_1-1,n_2-1) \right]$$
$$+\left[f(n_1+1,n_2) - f(n_1-1,n_2) \right]$$
$$+\left[f(n_1+1,n_2+1) - f(n_1-1,n_2+1) \right]$$

- These result in **Prewitt** operators: $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Derivative Operators

- How about emphasize the center row more:

$$\frac{\partial f(n_1,n_2)}{\partial n_1} \approx \left[f(n_1+1,n_2-1) - f(n_1-1,n_2-1) \right]$$
$$+2\left[f(n_1+1,n_2) - f(n_1-1,n_2) \right]$$
$$+\left[f(n_1+1,n_2+1) - f(n_1-1,n_2+1) \right]$$

- These result in **Sobel** operators: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

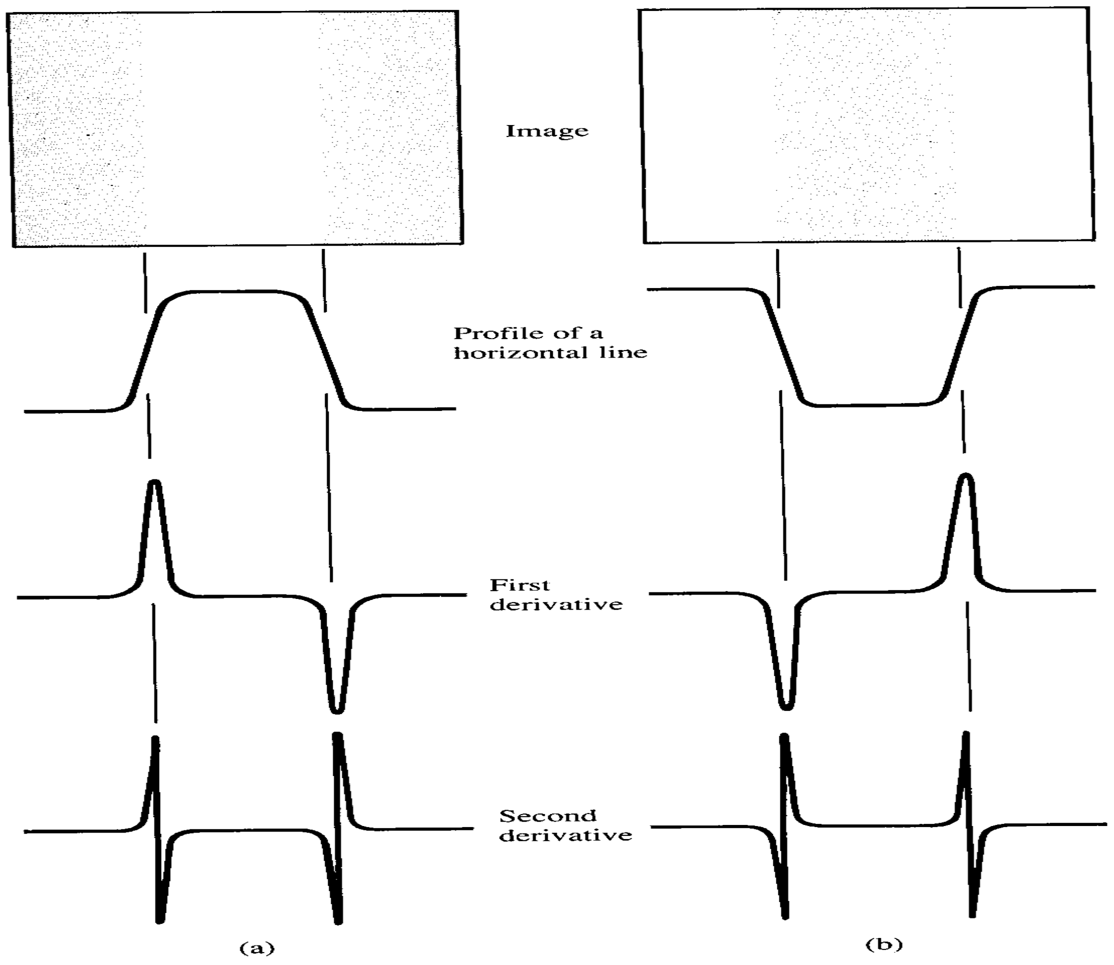
Diagonal Derivative Operators

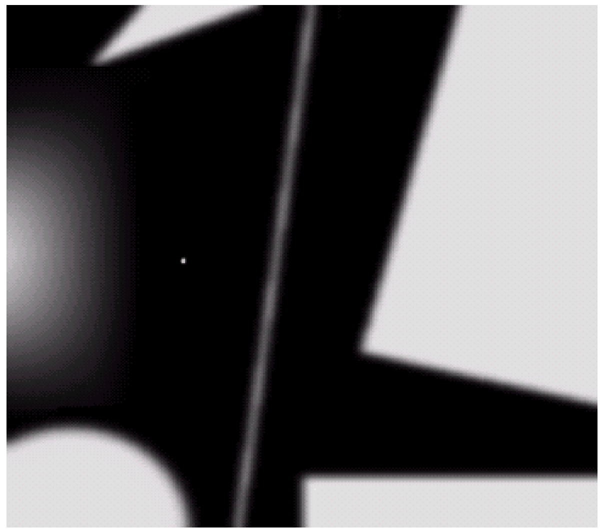
- We can also deal with both derivatives simultaneously:

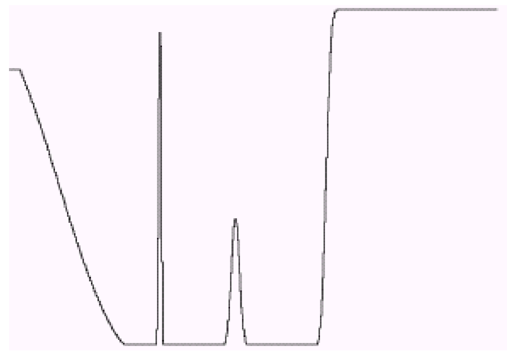
$$\frac{\partial f(n_1,n_2)}{\partial n_1}+\frac{\partial f(n_1,n_2)}{\partial n_2}\approx\left[f(n_1,n_2)-f(n_1+1,n_2+1)\right]$$

- This results in **Robert** operators: $\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ $\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$

Effects of 1st Derivatives







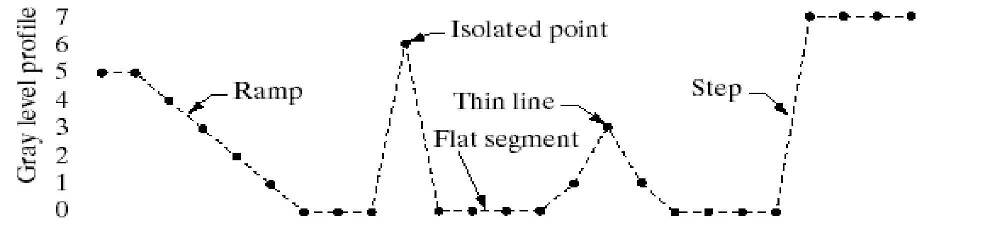


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
First Derivative		−1	−1	−1	−1	−1	0	0	6	−6	0	0	0	1	2	−2	−1	0	0	0	7	0	0	0		
Second Derivative		−1	0	0	0	0	1	0	6	−12	6	0	0	1	1	−4	1	1	0	0	7	−7	0	0		

Laplacian Operators : 2nd Derivative

$$\nabla^2 f(n_1,n_2) \cong \frac{\partial^2 f(n_1,n_2)}{\partial n_1^2} + \frac{\partial^2 f(n_1,n_2)}{\partial n_2^2}$$

- Difference Eq. approximation of Laplacian:

$$\frac{\partial f(n_1,n_2)}{\partial n_1} \approx f(n_1+1,n_2) - f(n_1,n_2)$$

$$\frac{\partial^2 f(n_1,n_2)}{\partial n_1^2} \approx \left[f(n_1+1,n_2) - f(n_1,n_2)\right] - \left[f(n_1,n_2) - f(n_1-1,n_2)\right]$$

$$= f(n_1+1,n_2) - 2f(n_1,n_2) + f(n_1-1,n_2)$$

$$\nabla^2 f(n_1,n_2) \cong f(n_1+1,n_2) + f(n_1-1,n_2) +$$

$$f(n_1,n_2+1) + f(n_1,n_2-1) - 4f(n_1,n_2)$$

Laplacian Operators

- The edges are indicated by **zero crossings** (which offer more reliable edge location), instead of **large gradients**.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

Effect of Laplacian Operator

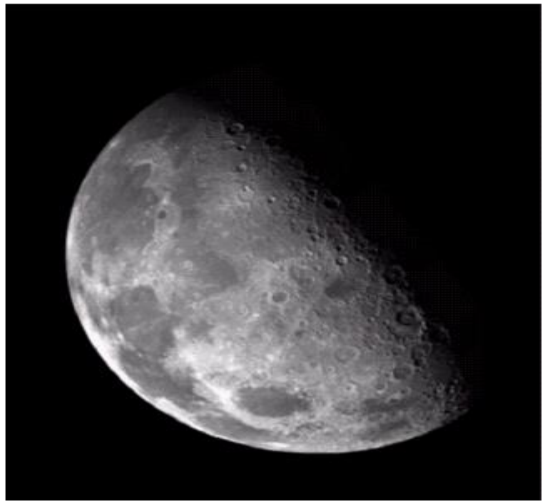
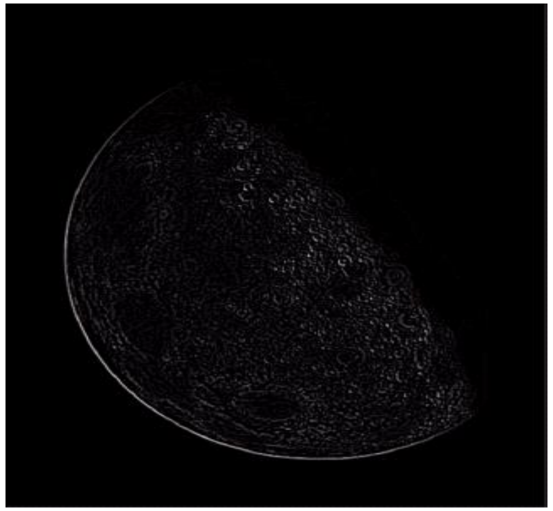
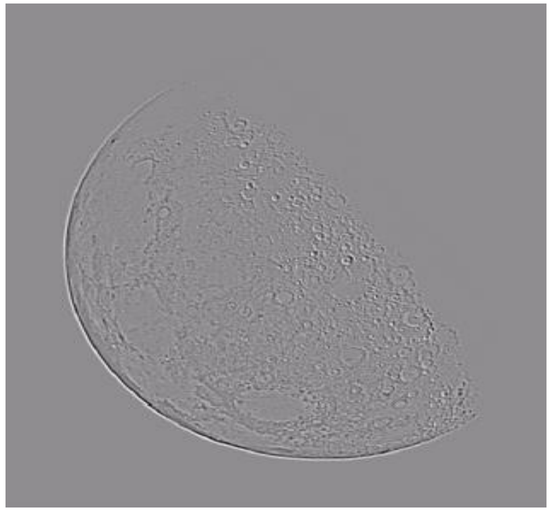


Image of the
North Pole of the moon


Laplacian filtered image


Laplacian image scaled
for display purposes

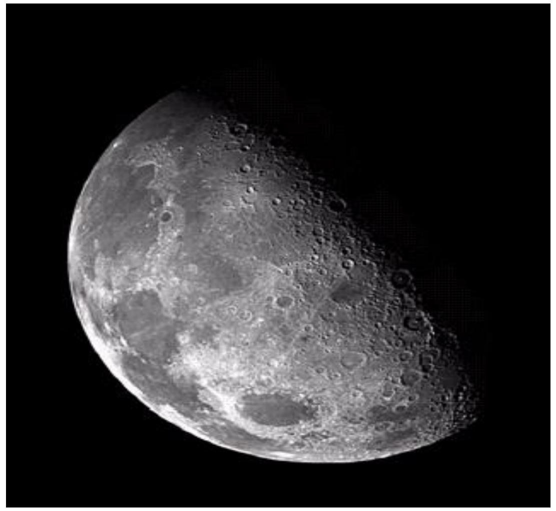


Image enhanced
By following equations

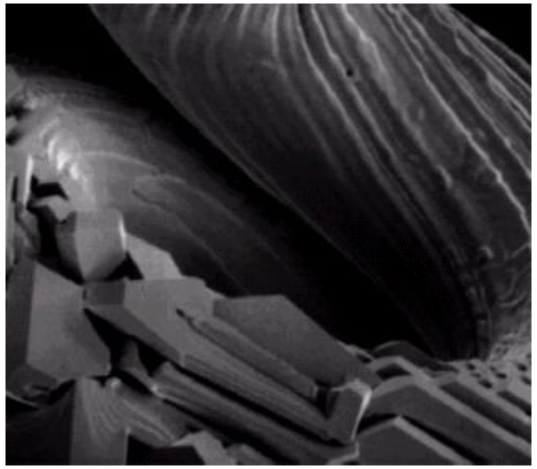
Enhancement Using
Laplacian Operators

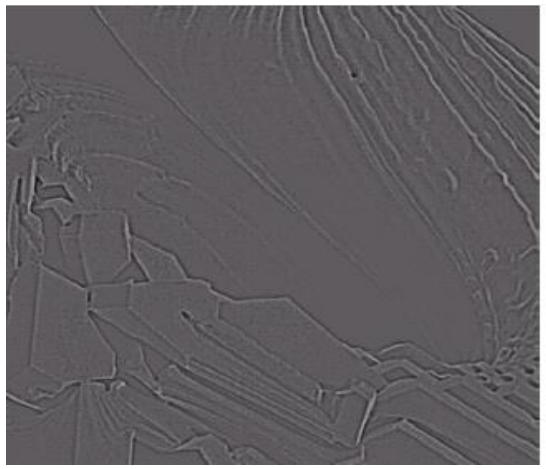
$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y), \text{if center Laplacian coeff. is negative} \\ f(x,y) + \nabla^2 f(x,y), \text{if center Laplacian coeff. is positive} \end{cases}$$

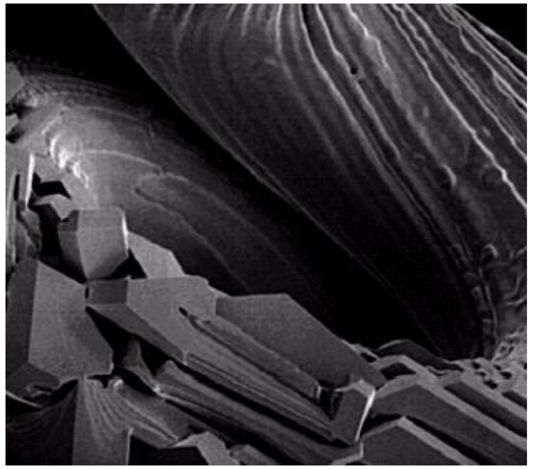
$$g(x,y) \cong f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)$$
$$-4f(x,y)] = 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$$

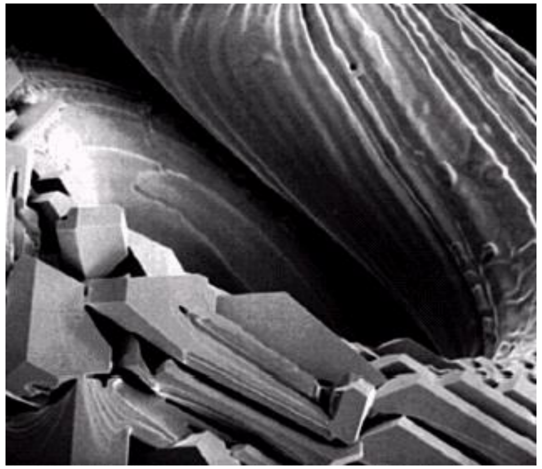
High-Boost Filtering

- With $A \geq 1$, $\begin{bmatrix} -1 & -1 & -1 \\ -1 & A+8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & 0 \end{bmatrix}$


Original


A=0


A=1

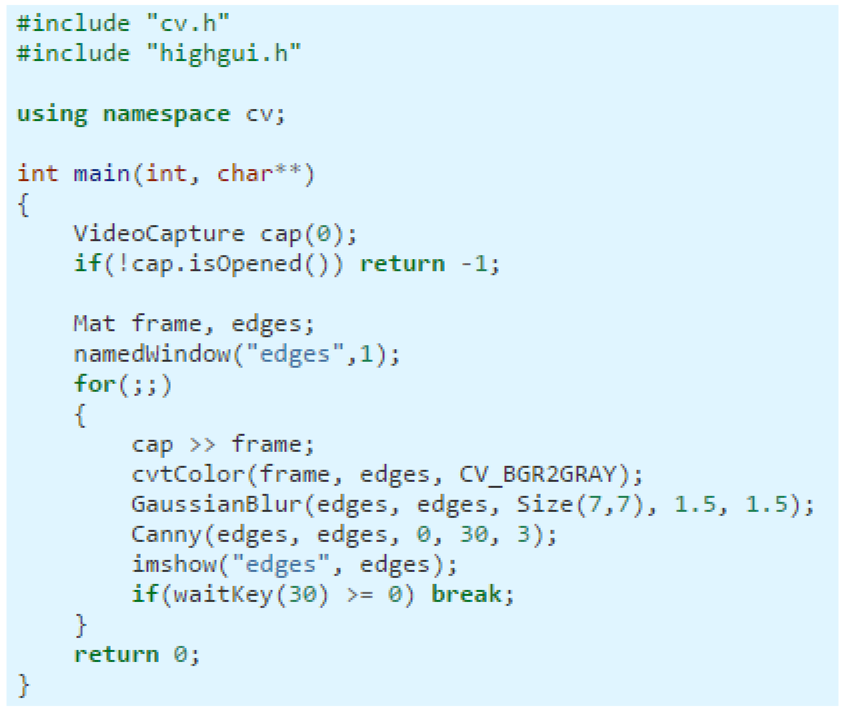

A=1.7

Video processing

- OpenCV (Open Source Computer Vision Library) : an open-source library that includes several hundreds of computer vision algorithms
- OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:
  - **core** - a compact module defining basic data structures, including the dense multi-dimensional array **Mat** and basic functions used by all other modules.
  - **imgproc** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
  - **video** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
  - **calib3d** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
  - **features2d** - salient feature detectors, descriptors, and descriptor matchers.
  - **objdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
  - **highgui** - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
  - **gpu** - GPU-accelerated algorithms from different OpenCV modules.
  - ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others

OpenCV

- Automatic Allocation of the Output Data
- OpenCV deallocates the memory automatically, as well as automatically allocates the memory for output function parameters most of the time.
- So, if a function has one or more input arrays (cv::Mat instances) and some output arrays, the output arrays are automatically allocated or reallocated.
- The size and type of the output arrays are determined from the size and type of input arrays. If needed, the functions take extra parameters that help to figure out the output array properties



```
#include "cv.h"
#include "highgui.h"

using namespace cv;

int main(int, char**)
{
    VideoCapture cap(0);
    if(!cap.isOpened()) return -1;

    Mat frame, edges;
    namedWindow("edges",1);
    for(;;)
    {
        cap >> frame;
        cvtColor(frame, edges, CV_BGR2GRAY);
        GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);
        Canny(edges, edges, 0, 30, 3);
        imshow("edges", edges);
        if(waitKey(30) >= 0) break;
    }
    return 0;
}
```

Automatic Allocation

- The array frame is automatically allocated by the **>>** operator since the video frame resolution and the bit-depth is known to the video capturing module.
- The array edges is automatically allocated by the cvtColor function. It has the same size and the bit-depth as the input array. The number of channels is 1 because the color conversion code CV_BGR2GRAY is passed, which means a color to grayscale conversion. Note that frame and edges are allocated only once during the first execution of the loop body since all the next video frames have the same resolution. **If you somehow change the video resolution, the arrays are automatically reallocated**.

- The key component of this technology is the Mat::create method. It takes the desired array size and type. If the array already has the specified size and type, the method does nothing. Otherwise, it releases the previously allocated data, if any (this part involves decrementing the reference counter and comparing it with zero), and then allocates a new buffer of the required size. Most functions call the Mat::create method for each output array, and so the automatic output data allocation is implemented.

- Some notable exceptions from this scheme are cv::mixChannels, cv::RNG::fill, and a few other functions and methods. They are not able to allocate the output array, so you have to do this in advance

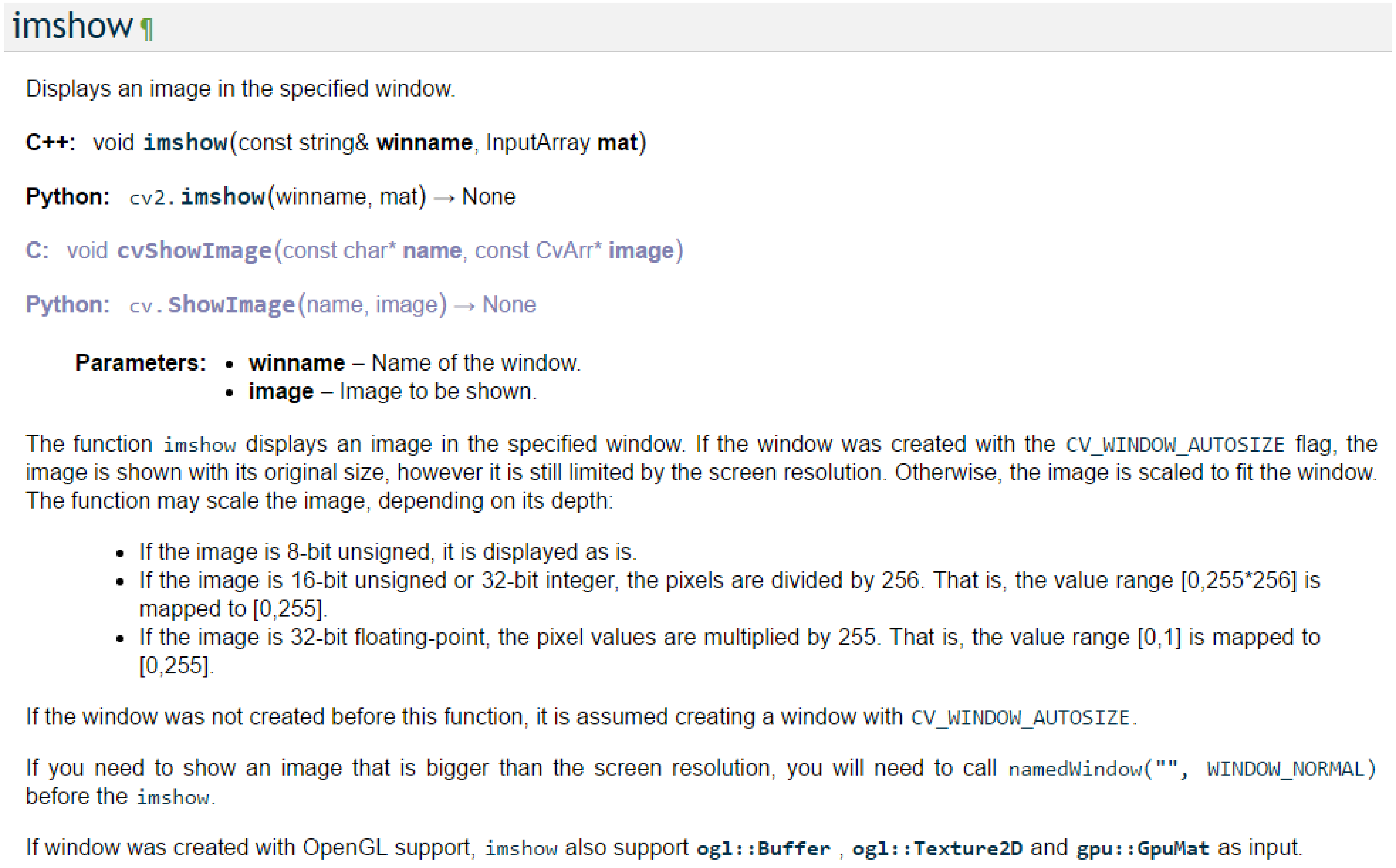
Saturation Arithmetic

- As a computer vision library, OpenCV deals a lot with image pixels that are often encoded in a compact, 8- or 16-bit per channel, form and thus have a limited value range.
- Furthermore, certain operations on images, like color space conversions, brightness/contrast adjustments, sharpening, complex interpolation (bi-cubic, Lanczos) can produce values out of the available range.
- If you just store the lowest 8 (16) bits of the result, this results in visual artifacts and may affect a further image analysis.
- To solve this problem, the so-called saturation arithmetics is used. For example, to store r, the result of an operation, to an 8-bit image, you find the nearest value within the 0..255 range

highgui. High-level GUI and Media I/O

- While OpenCV was designed for use in full-scale applications and can be used within functionally rich UI frameworks (such as Qt*, WinForms*, or Cocoa*) or without any UI at all, sometimes there it is required to try functionality quickly and visualize the results. This is what the HighGUI module has been designed for.

- It provides easy interface to:
    - Create and manipulate windows that can display images and "remember" their content (no need to handle repaint events from OS).
    - Add trackbars to the windows, handle simple mouse events as well as keyboard commands.
    - Read and write images to/from disk or memory.
    - Read video from camera or file and write video to a file



imshow ¶

Displays an image in the specified window.

C++: void **imshow**(const string& **winname**, InputArray **mat**)

Python: cv2.**imshow**(winname, mat) → None

C: void **cvShowImage**(const char* **name**, const CvArr* **image**)

Python: cv.**ShowImage**(name, image) → None

Parameters:
- **winname** – Name of the window.
- **image** – Image to be shown.

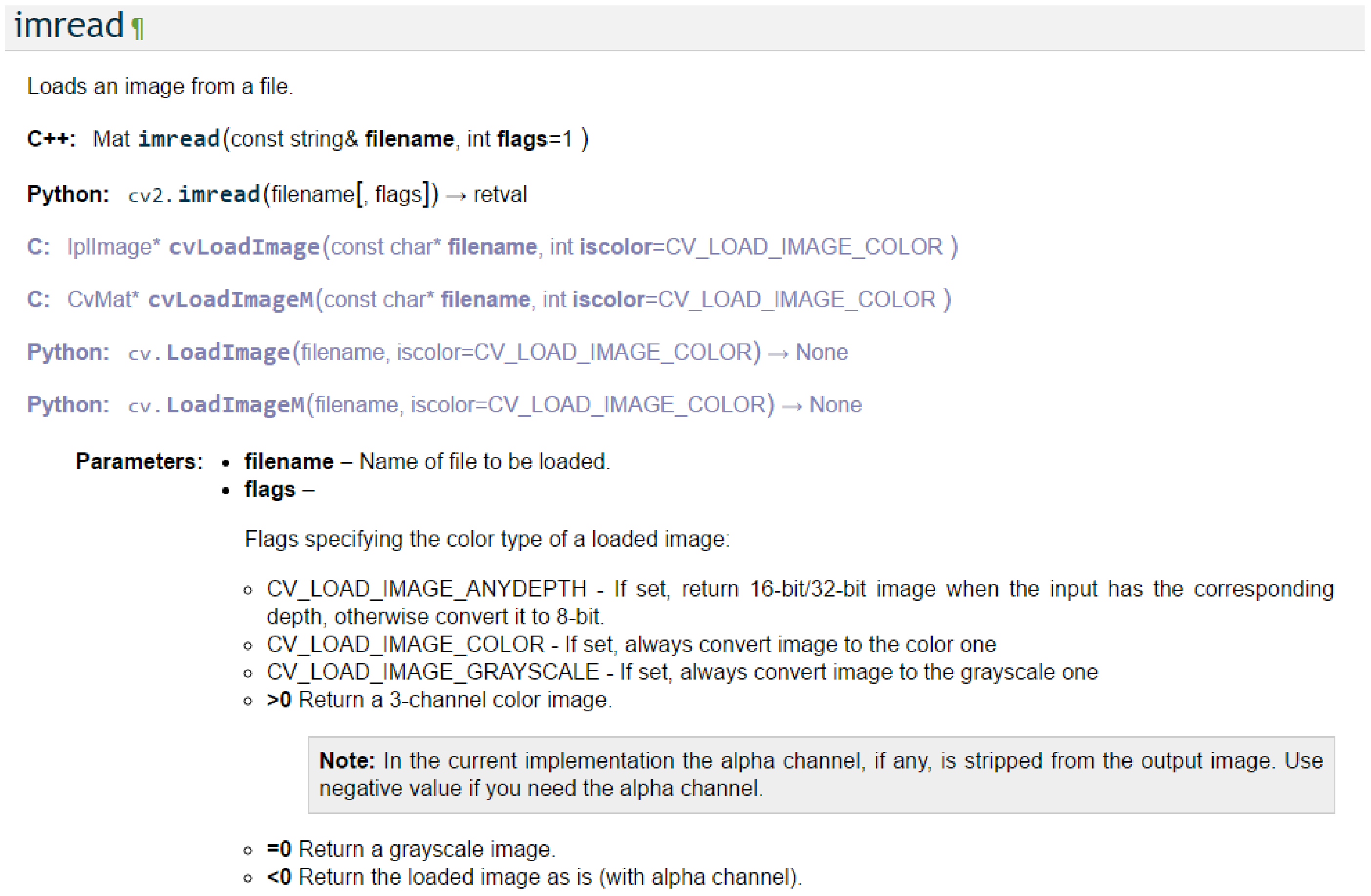
The function `imshow` displays an image in the specified window. If the window was created with the `CV_WINDOW_AUTOSIZE` flag, the image is shown with its original size, however it is still limited by the screen resolution. Otherwise, the image is scaled to fit the window. The function may scale the image, depending on its depth:

- If the image is 8-bit unsigned, it is displayed as is.
- If the image is 16-bit unsigned or 32-bit integer, the pixels are divided by 256. That is, the value range [0,255*256] is mapped to [0,255].
- If the image is 32-bit floating-point, the pixel values are multiplied by 255. That is, the value range [0,1] is mapped to [0,255].

If the window was not created before this function, it is assumed creating a window with `CV_WINDOW_AUTOSIZE`.

If you need to show an image that is bigger than the screen resolution, you will need to call `namedWindow("", WINDOW_NORMAL)` before the `imshow`.

If window was created with OpenGL support, `imshow` also support **`ogl::Buffer`** , **`ogl::Texture2D`** and **`gpu::GpuMat`** as input.



imread ¶

Loads an image from a file.

C++: Mat **imread**(const string& **filename**, int **flags**=1)

Python: `cv2.`**`imread`**(filename[, flags]) → retval

C: IplImage* **cvLoadImage**(const char* **filename**, int **iscolor**=CV_LOAD_IMAGE_COLOR)

C: CvMat* **cvLoadImageM**(const char* **filename**, int **iscolor**=CV_LOAD_IMAGE_COLOR)

Python: `cv.`**`LoadImage`**(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None

Python: `cv.`**`LoadImageM`**(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None

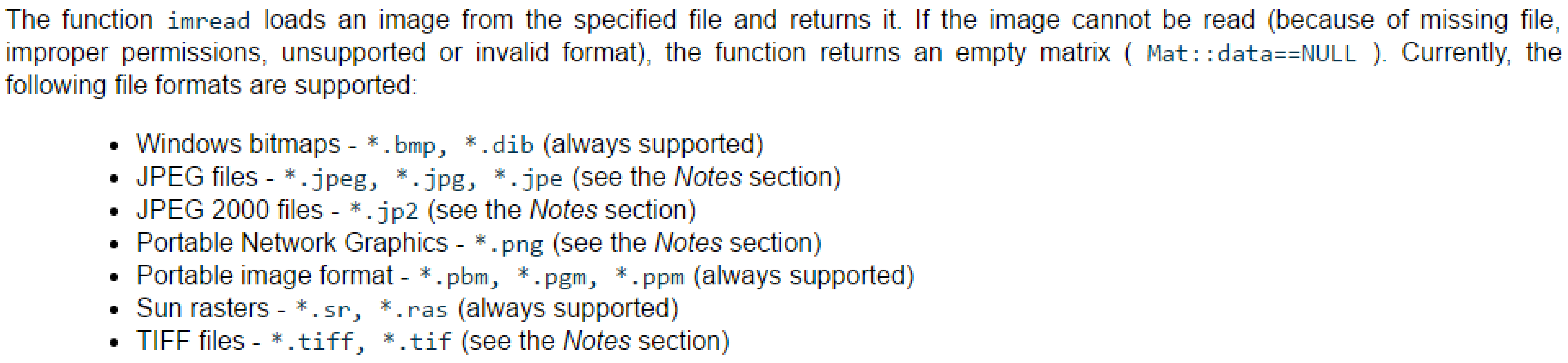
Parameters:
- **filename** – Name of file to be loaded.
- **flags** –

  Flags specifying the color type of a loaded image:

  - CV_LOAD_IMAGE_ANYDEPTH - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.
  - CV_LOAD_IMAGE_COLOR - If set, always convert image to the color one
  - CV_LOAD_IMAGE_GRAYSCALE - If set, always convert image to the grayscale one
  - **>0** Return a 3-channel color image.

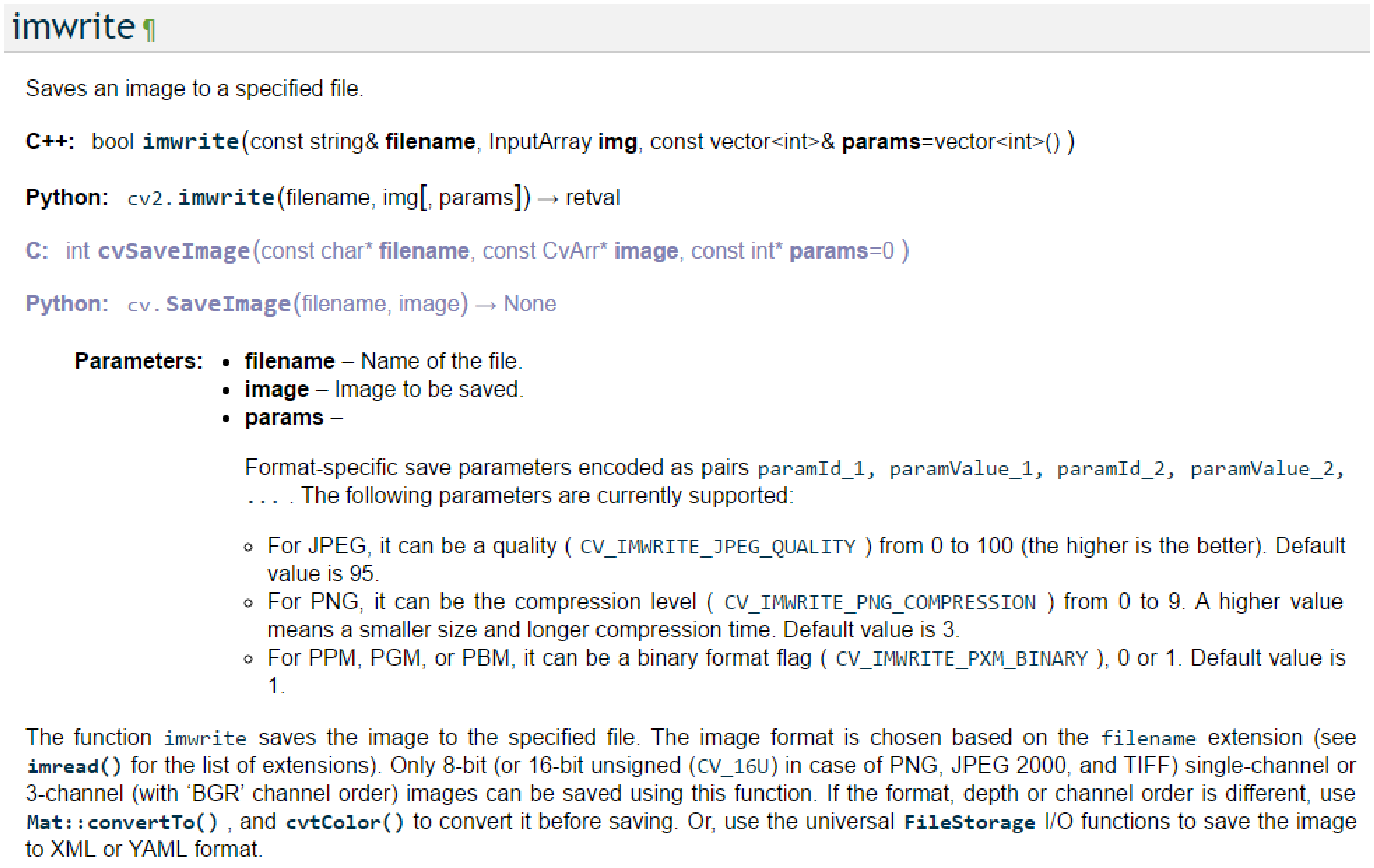
    Note: In the current implementation the alpha channel, if any, is stripped from the output image. Use negative value if you need the alpha channel.

  - **=0** Return a grayscale image.
  - **<0** Return the loaded image as is (with alpha channel).



The function `imread` loads an image from the specified file and returns it. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format), the function returns an empty matrix (`Mat::data==NULL`). Currently, the following file formats are supported:

- Windows bitmaps - `*.bmp, *.dib` (always supported)
- JPEG files - `*.jpeg, *.jpg, *.jpe` (see the *Notes* section)
- JPEG 2000 files - `*.jp2` (see the *Notes* section)
- Portable Network Graphics - `*.png` (see the *Notes* section)
- Portable image format - `*.pbm, *.pgm, *.ppm` (always supported)
- Sun rasters - `*.sr, *.ras` (always supported)
- TIFF files - `*.tiff, *.tif` (see the *Notes* section)



imwrite ¶

Saves an image to a specified file.

C++: bool **`imwrite`**(const string& **filename**, InputArray **img**, const vector<int>& **params**=vector<int>())

Python: `cv2.`**`imwrite`**(filename, img[, params]) → retval

C: int **`cvSaveImage`**(const char* **filename**, const CvArr* **image**, const int* **params**=0)

Python: `cv.`**`SaveImage`**(filename, image) → None

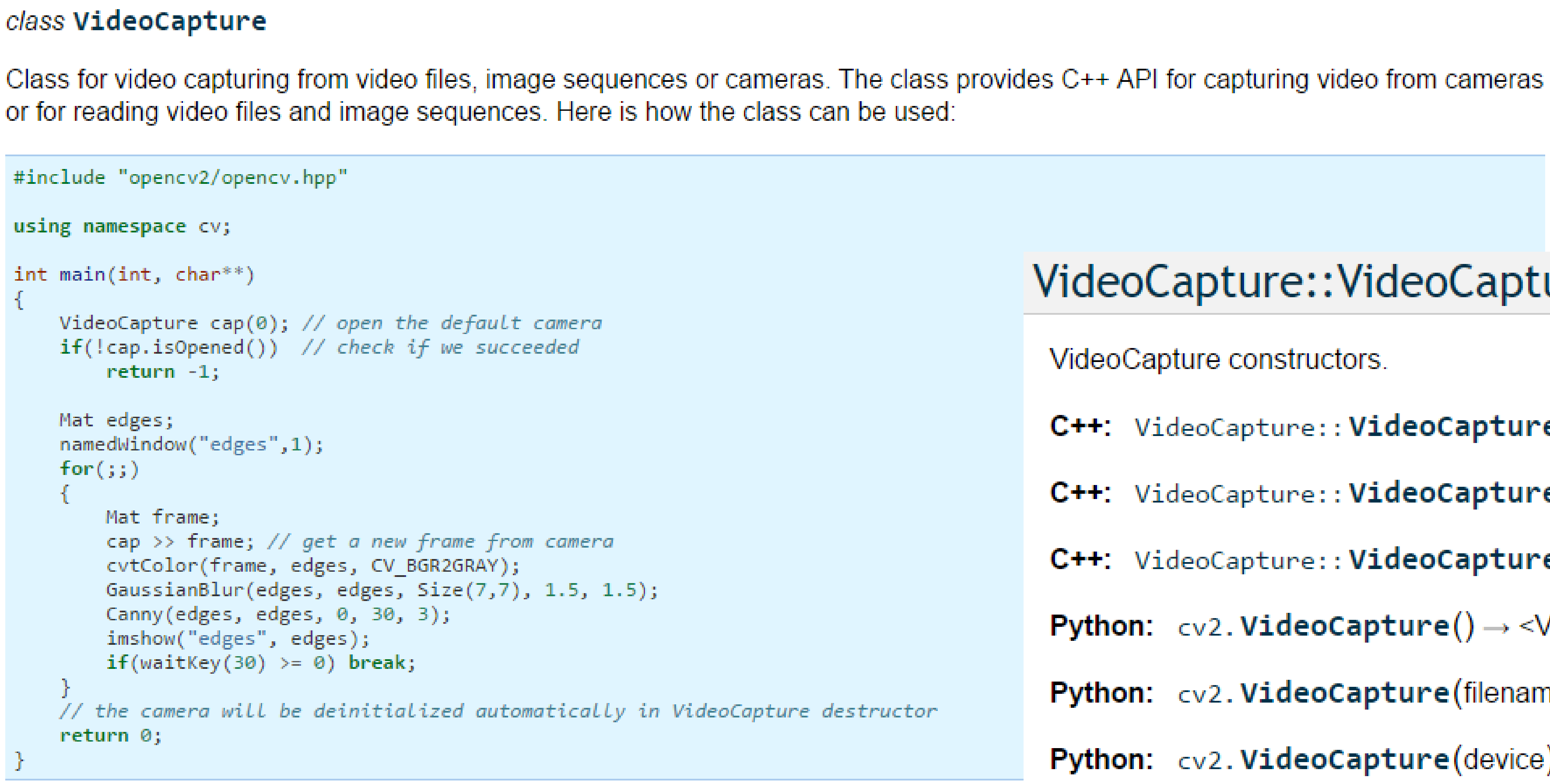
Parameters:
- **filename** – Name of the file.
- **image** – Image to be saved.
- **params** –

  Format-specific save parameters encoded as pairs `paramId_1, paramValue_1, paramId_2, paramValue_2, ...` . The following parameters are currently supported:

  - For JPEG, it can be a quality (`CV_IMWRITE_JPEG_QUALITY`) from 0 to 100 (the higher is the better). Default value is 95.
  - For PNG, it can be the compression level (`CV_IMWRITE_PNG_COMPRESSION`) from 0 to 9. A higher value means a smaller size and longer compression time. Default value is 3.
  - For PPM, PGM, or PBM, it can be a binary format flag (`CV_IMWRITE_PXM_BINARY`), 0 or 1. Default value is 1.

The function `imwrite` saves the image to the specified file. The image format is chosen based on the `filename` extension (see **`imread()`** for the list of extensions). Only 8-bit (or 16-bit unsigned (`CV_16U`) in case of PNG, JPEG 2000, and TIFF) single-channel or 3-channel (with 'BGR' channel order) images can be saved using this function. If the format, depth or channel order is different, use **`Mat::convertTo()`** , and **`cvtColor()`** to convert it before saving. Or, use the universal **`FileStorage`** I/O functions to save the image to XML or YAML format.

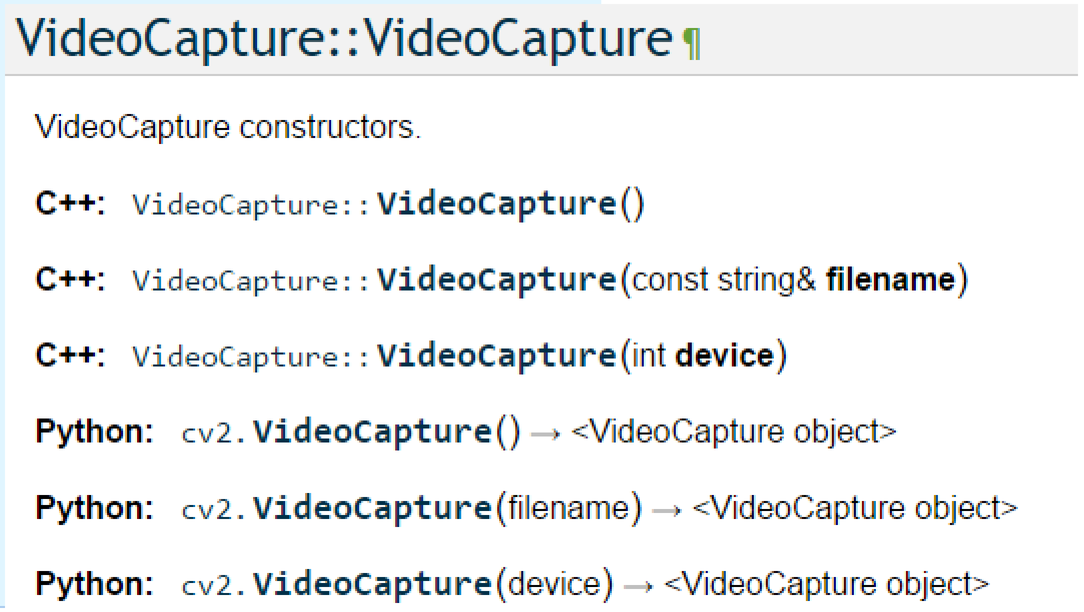
Video Capture



class **VideoCapture**

Class for video capturing from video files, image sequences or cameras. The class provides C++ API for capturing video from cameras or for reading video files and image sequences. Here is how the class can be used:

```
#include "opencv2/opencv.hpp"

using namespace cv;

int main(int, char**)
{
    VideoCapture cap(0); // open the default camera
    if(!cap.isOpened())  // check if we succeeded
        return -1;

    Mat edges;
    namedWindow("edges",1);
    for(;;)
    {
        Mat frame;
        cap >> frame; // get a new frame from camera
        cvtColor(frame, edges, CV_BGR2GRAY);
        GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);
        Canny(edges, edges, 0, 30, 3);
        imshow("edges", edges);
        if(waitKey(30) >= 0) break;
    }
    // the camera will be deinitialized automatically in VideoCapture destructor
    return 0;
}
```



VideoCapture::VideoCapture ¶

VideoCapture constructors.

C++: `VideoCapture::`**`VideoCapture`**`()`

C++: `VideoCapture::`**`VideoCapture`**`(const string& `**`filename`**`)`

C++: `VideoCapture::`**`VideoCapture`**`(int `**`device`**`)`

Python: `cv2.`**`VideoCapture`**`()` → <VideoCapture object>

Python: `cv2.`**`VideoCapture`**`(filename)` → <VideoCapture object>

Python: `cv2.`**`VideoCapture`**`(device)` → <VideoCapture object>

Install the OpenCV in your board

- https://help.ubuntu.com/community/OpenCV
- Step 1.
  - Download the latest opencv.sh from https://github.com/jayrambhia/Install-OpenCV/blob/master/Ubuntu/ or Copy the script in the next page to gedit and save as opencv.sh
- Step 2.
  - Open the terminal

```
$ chmod +x opencv.sh

$ ./opencv.sh
```

Shell script for installing the OpenCV

```
version="$(wget -q -O - http://sourceforge.net/projects/opencvlibrary/files/opencv-unix | egrep -m1 -o '\"[0-9](\.[0-9]+)+' | cut -c2-)"
echo "Installing OpenCV" $version
mkdir OpenCV
cd OpenCV
echo "Removing any pre-installed ffmpeg and x264"
sudo apt-get -qq remove ffmpeg x264 libx264-dev
echo "Installing Dependenices"
sudo apt-get -qq install libopencv-dev build-essential checkinstall cmake pkg-config yasm libjpeg-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev
libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev libqt4-dev libgtk2.0-dev
libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev
libxvidcore-dev x264 v4l-utils ffmpeg cmake qt5-default checkinstall
echo "Downloading OpenCV" $version
wget -O OpenCV-$version.zip http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/$version/opencv-"$version".zip/download
echo "Installing OpenCV" $version unzip OpenCV-$version.zip
cd opencv-$version mkdir build
cd build cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D
BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
make -j2
sudo checkinstall
sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
echo "OpenCV" $version "ready to be used"
```

Running OpenCV

- Python : loading an image in Python

```
from cv2.cv import *

img = LoadImage("/home/USER/Pictures/python.jpg")
NamedWindow("opencv")
ShowImage("opencv",img)
WaitKey(0)
```

```
$ python filename.py
```

Loading an image file in C

```c
#include
#include<opencv2/highgui/highgui.hpp>
int main() {
		IplImage* img = cvLoadImage("/home/USER/Pictures/python.jpg",CV_LOAD_IMAGE_COLOR);
		cvNamedWindow("opencvtest",CV_WINDOW_AUTOSIZE);
		cvShowImage("opencvtest",img);
		cvWaitKey(0);
		cvReleaseImage(&img);
		return 0;

}
```

```
$ gcc -ggdb `pkg-config --cflags opencv` -o `basename opencvtest.c .c` opencvtest.c `pkg-config --libs opencv`
$ ./opencvtest
```

Loading an image file in C++

```
#include<opencv2/highgui/highgui.hpp>
using namespace cv;
int main() {
		Mat img = imread("/home/USER/Pictures/python.jpg",CV_LOAD_IMAGE_COLOR);
		Imshow("opencvtest",img);
		waitKey(0);
		return 0;
}
```

```
$ g++ -ggdb `pkg-config --cflags opencv` -o `basename opencvtest.cpp .cpp` opencvtest.cpp `pkg-config --libs opencv`
$ ./opencvtest
```