UNIVERSITÀ DI PARMA

OpenCV
a brief introduction (for C++)

Summary

UNIVERSITÀ DI PARMA

- OpenCV
- Installation
- Modules
- C pointers
- cv::Mat class + companions
- Few examples and simple.cpp skeleton

OpenCV

UNIVERSITÀ DI PARMA

- OpenCV (Open Source Computer Vision Library) is an Open Source library for computer vision and machine learning
- BSD License (also commercial use!)
- Thousands of algorithms
- Tenth of thousands of users
- Millions of downloads
- **C++**, Python, JAVA, MATLAB support

- Main functionalities
  - Read/write images, sequences of images, or videos
  - Process images
    - Many off the shelf libraries
  - Graphic output

Installation

UNIVERSITÀ DI PARMA

- Linux/gcc
- Two possibilities
  - Package manager
  - Download and compile sources
- Remember to install both core and contribs

Installation

UNIVERSITÀ DI PARMA

- Git repo
- git clone …
  - For both core and contribs
- When creating compile config, please include contribs:
  - cmake -DOPENCV_EXTRA_MODULES_PATH=/path/to/contribs/modules/ /path/to/core

Modules

UNIVERSITÀ DI PARMA

- OpenCV main modules are:
  - Core, basic data structures:
    - Mat, Scalar, Point, Range…
  - Image processing, we will use some just to match our results
  - Video, motion estimation, tracking, background subraction…
  - Calib3d, camera calibration
  - Features2d, features extraction and matching
  - …

C pointers

UNIVERSITÀ DI PARMA

- It is an OpenCV slide presentation, isn't it?
- Yes but we need some recap about how to access memory...
- What is a C pointer?
  - Kind of data to store memory addresses
  - 32 bits/64 bits

C pointers

- Address is simply a number
- Anyway C pointers feature a data type:
  - char *c ⟶ pointer to a char data
  - float *f ⟶ pointer to a float data
  - …
  - void *v ⟶ pointer to something…

C pointers

UNIVERSITÀ DI PARMA

- Why we need a data type for pointers?
- Basically for pointer arithmetics
- f=f+1 ⟶ what is the result?
  - It depends on which kind of data is expected to be found at address f
  - If f is a char*, f=f+1 ⟶ address f is increased by 1
  - If f is a uint32_t, f=f+1 ⟶ address f is increased by 4

C pointers

UNIVERSITÀ DI PARMA

- How to access to the pointed data?
- When f is a pointer we can use *f
  - Both read/write
- Anyway usually we deal with large chunks of data ⟶ arrays
- To access the n^{th} element we can use:
  - *(f+n) ⟶ old fashion, avoid…
  - f[n]

C pointers

UNIVERSITÀ DI PARMA

- We already know that images are (at least) 2D structures
- We can use pointers for that?
- Yes we can use pointers to other pointers
  - char **c;
- If we consider other dimensions things get even creepier…
- Hint: do not do that!

C pointers

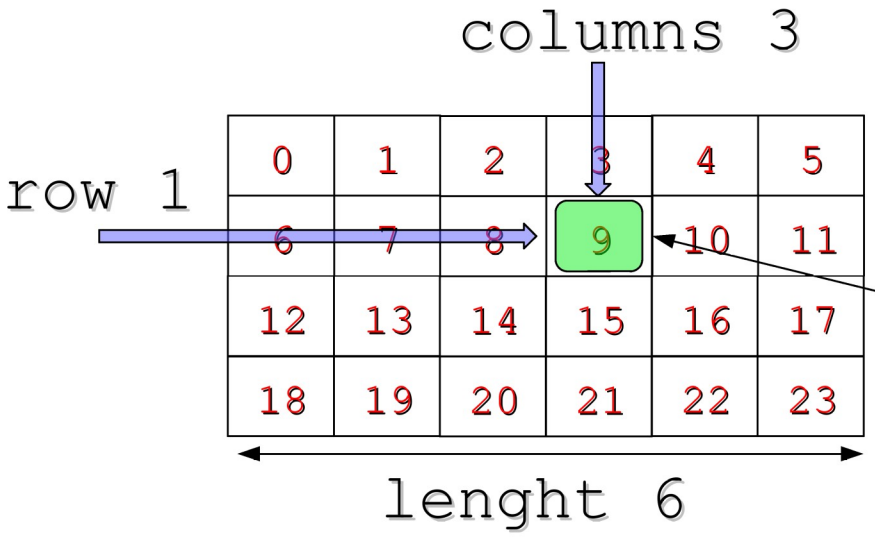
- Use simple array to deal with multidimensional matrices
- If we need to store n × m values:
  - data_type data[n*m];
- Access element at coordinates x,y
  - data[x+y*m]
- Logical representation vs Physical one

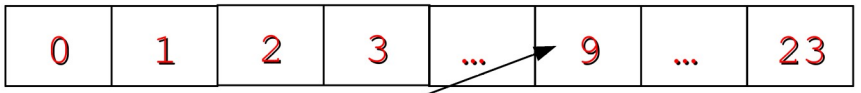
C pointers

UNIVERSITÀ DI PARMA

Logical layout

Physical layout





Array index: Row index*lenght + Column index

cv::Mat

UNIVERSITÀ DI PARMA

- Basic Image Container
- Two main elements:
  - Handler
    - Description of data
  - Shared pointer for data
    - Actual data pointer
    - Be careful! clone() and copyTo() methods
    - a=b (!)

cv::Mat constructors

UNIVERSITÀ DI PARMA

- cv::Mat()
- cv::Mat(int rows, int cols, int type)
- cv::Mat(int rows, int cols, int type, cv::Scalar s)
- cv::Mat(cv::Size size, int type)
- cv::Mat(cv::Size size, int type, cv::Scalar s)
- cv::Mat(const cv::Mat &m)
- cv::Mat(const cv::Mat &m, cv::Range rowRange)
- cv::Mat(const cv::Mat &m, cv::Range rowRange, cv::Range colRange)
- cv::Mat(const cv::Mat &m, cv::Rect roi)
- ...

OpenCV types

UNIVERSITÀ DI PARMA

CV_	C1	C2	C3	C4
8U	0	8	16	24
8S	1	9	17	25
16U	2	10	18	26
16S	3	11	19	27
32S	4	12	20	28
32F	5	13	21	29
64F	6	14	22	30

Some othe cv:: classes

UNIVERSITÀ DI PARMA

- cv::Scalar
  - Basically a short vector (up to 4) template
- cv::Rect
  - Template class for 2D rectangles
- cv::Range
  - Template class for a continuous subsequence

cv::Mat contruction examples

UNIVERSITÀ DI PARMA

- `cv::Mat A, B;` `// empty images`
- `cv::Mat C(A);` `// copy (!)`
- `cv::Mat D(1024, 900, CV_8UC3)` `// set size/type`
- `cv::Mat E(A, Rect(10, 10, 100, 100));` `// only part of A`
- `cv::Mat M(2,2, CV_8UC3, Scalar(0,0,255));` `// also set pixel initial value`
- `cv::Mat F = A.clone();`
- `cv::Mat G;`
- `A.copyTo(G);`

cv::Mat M

UNIVERSITÀ DI PARMA

- M.rows rows
- M.cols columns
- M.channels() channels
- M.type() image type (OpenCV type!)
- M.elemSize() pixel size (bytes)
- M.elemSize1() channel size (bytes)

- i.e. RGB8
- M.channels() == 3
- M.elemSize() == 3
- M.elemSize1() == 1
- M.type() == CV_8UC3 3 channels, 1 byte/channel

cv::Mat M

UNIVERSITÀ DI PARMA

- Where is my image?
- **uchar *cv::Mat::data** can be used
- M.data ⟶ address of image buffer
- M.data ⟶ points to first image byte
- It does not depend on pixel type

cv::Mat other access methods

UNIVERSITÀ DI PARMA

- To access specific row:
  - uchar * cv::Mat::ptr(int i)
  - Allows to access buffer at row i
- Actually a template
  - T * cv::Mat::ptr<T>(int i)
- Also single pixel can be referenced:
  - T cv::Mat::at<T>(row=0,col=0)[channel]
  - Allows to access to value/address
  - Do not use it before first homework

Example #1

UNIVERSITÀ DI PARMA

- Bare image access

```
cv::Mat M;
...
for(int i =0;i<M.rows*M.cols*M.elemSize();++i)
    M.data[i] = i;
```

Example #2

UNIVERSITÀ DI PARMA

- Single channel access

```
cv::Mat M;
...
for(int i =0;i<M.rows*M.cols;i+=M.elemSize())
    {
     M.data[i] = i;                                    //B
     M.data[i+M.elemSize1()] = i + 1;                  //G
     M.data[i+M.elemSize1()+M.elemSize1()] = i + 2; //R
}
```

Example #3

UNIVERSITÀ DI PARMA

- Row/Column access

```
cv::Mat M;
...
for(int v =0;v<M.rows;++v)
   {
    for(int u=0;u<M.cols;++u)
       {
        M.data[ (u + v*M.cols)*3] = u;         //B
        M.data[ (u + v*M.cols)*3 + 1] = u+1; //G
        M.data[ (u + v*M.cols)*3 + 2] = u+2; //R
       }
   }
```

Example #3

UNIVERSITÀ DI PARMA

- Row/Column/Channel (1 byte) access

```
cv::Mat M;
...
for(int v =0;v<M.rows;++v)
     {
      for(int u=0;u<M.cols;++u)
         {
           for(int k=0;k<M.channels();++k)
             M.data[ (u + v*M.cols)*M.channels() + k] = u + k;
         }
     }
```

simple.cpp

UNIVERSITÀ DI PARMA

- Skeleton for… everything?
- Prerequisites:
  - OpenCV
  - g++
  - cmake + make
- Build:

```
mkdir build; cd build
cmake ..
make
```

- Enjoy!