UNIVERSITÀ DEGLI STUDI DI PADOVA

OpenCV – Basic data structures and data manipulation

Stefano Ghidoni

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

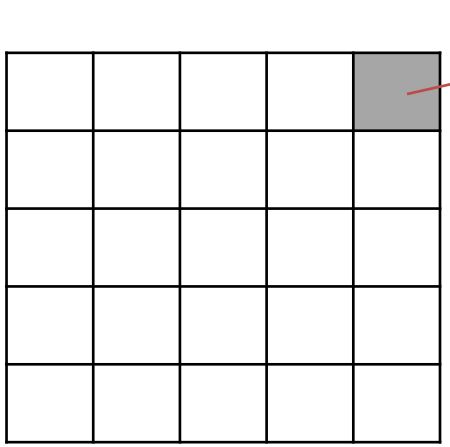
INTELLIGENT AUTONOMOUS SYSTEMS LAB

Agenda

- The structure of an image
- The Mat and Vec classes
- Pixel representations
- Accessing the pixels

Storing images

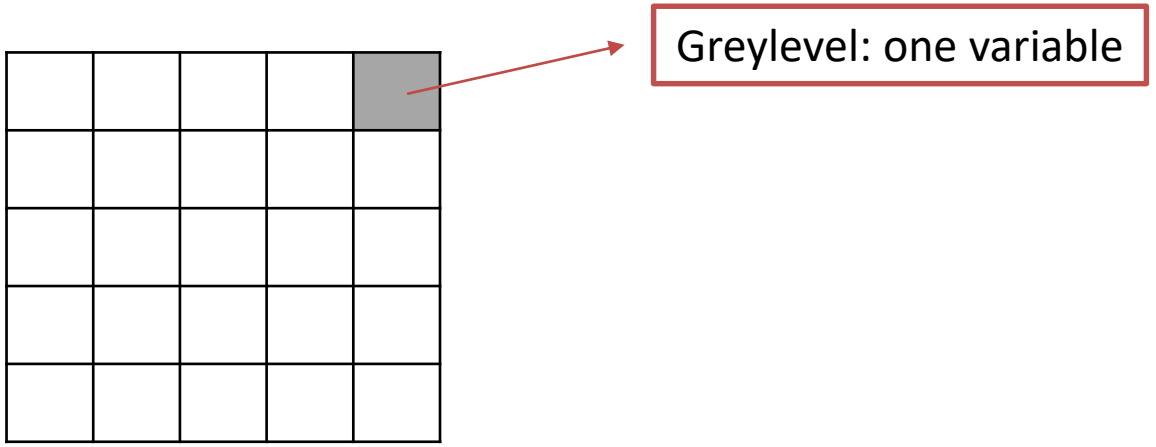
- An image is a set of pixels
- The matrix is a natural shape for an image



- How can we represent a pixel?
  - Which data type? How many variables?

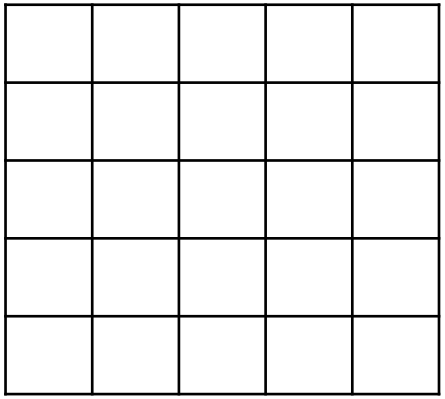
Grayscale images

- Which data type?
- Possible values taken by a pixel:
  - unsigned char: [0, 255]
  - int 16 bits: [0, 65535]
  - float: [0, 1]
- **Depth**: # of bits for each pixel



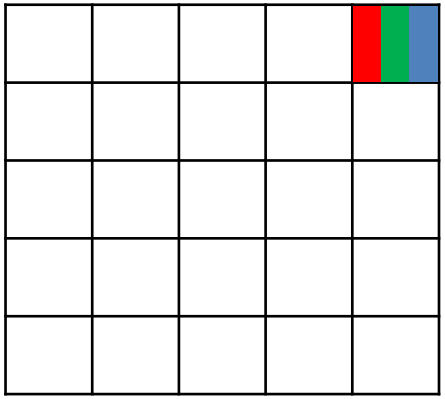
Color images

- What if the image has color pixels?



Color images

- What if the image has color pixels?
- Color representation: RGB (many others are available)
  - Additive color



Color images

- What if the image has color pixels?
- Color representation: RGB (many others are available)
  - Additive color
  - Each color has the same representation of a grayscale pixel
- Each color is called channel
- Depth applies to each channel

The cv::Mat class

```
class Mat{
public:
  // bit-fields: depth, #channels, ...
  int flags;
  // array dimensionality >= 2
  int dims;

  int rows, cols;

  // pointer to the data
  uchar *data;

  // ...
};
```

Constructors

```
cv::Mat(nrows, ncols, type[, fillValue])

ex: cv::Mat(10, 10, CV_8U);
```

- Type is: CV_<depth>[C<# channels>]
  - CV_8U, CV_8S
  - CV_16U, CV_16S
  - CV_32S, CV_32F
  - CV_64F

Inspectors

```
int Mat::channels() const;    // # channels

int Mat::depth() const;       // matrix element depth for
                              // each channel

int Mat::type() const;        // matrix element type
```

Inspectors

```
int Mat::channels() const;    // # channels

int Mat::depth() const;       // matrix element depth for
                              // each channel

int Mat::type() const;        // matrix element type
```

- What does const mean in this context?

Accessing the image pixels

- cv::Mat
- cv::Mat::at function

```
template <typename T>
T& Mat::at(int i, int j);
```

Example

```
#include <opencv2/highgui>

using namespace cv;

int main(void)
{
  Mat img(200, 200, CV_8U);
  for (int i = 0; i < 200; ++i)
    for (int j = 0; j < 200; ++j)
      img.at<unsigned char> (i, j) = std::min(i+j, 255);

  namedWindow("Example 2");
  imshow ("Example 2", img);

  waitKey(0);

  return 0;
}
```

Accessing the image pixels

- What if we need to work on color images?

- No spoiler ☺

The Vec class

- Vec is an OpenCV class used for dealing with tuples
- Elements accessed using operator[]

```
template <typename T, int cn>
class Vec

// examples

Vec<uchar, 2> v;
v[0] = 2;
v[1] = 4;   // interpreted as: v.operator[](1) -- overloading
```

Vec shortcut

```
typedef Vec<uchar, 2> Vec2b;
typedef Vec<uchar, 3> Vec3b;
typedef Vec<uchar, 4> Vec4b;

typedef Vec<short, 2> Vec2s;
typedef Vec<int, 2> Vec2i;
typedef Vec<float, 2> Vec2f;
typedef Vec<double, 2> Vec2d;
```

Accessing color images

- Vec is used to describe the triplet of color pixels

Accessing the image pixels

- cv::Mat
- cv::Mat::at function
- Need to exploit the cv::Vec2b class for color images
  - `cv::Vec3b current_color = image_out.at<cv::Vec3b>(r, c);`
  - `image.at<cv::Vec3b>(r,c) = cv::Vec3b(37,201,92);`
- Accessing the Vec2b elements
  - `current_color[0] = 22;`

Example

```
// ...

int main(void)
{
  Mat img(200, 200, CV_8UC3);
  for (int i = 0; i < img.rows; ++i)
  {
    for (int j = 0; j < img.cols; ++j)
    {
      img.at<Vec3b> (i, j)[0] = 0;
      img.at<Vec3b> (i, j)[1] = 0;
      img.at<Vec3b> (i, j)[2] = 255;
    }
  }
  // visualization
  return 0;
}
```

OpenCV tutorials

- A good number of tutorials is available at https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- Your homework for this week:
  - Have a look
  - Read the tutorial about Mat https://docs.opencv.org/4.x/d6/d6d/tutorial_mat_the_basic_image_container.html
  - Even if you don't get 100%, please get in touch

UNIVERSITÀ DEGLI STUDI DI PADOVA

OpenCV – Basic data structures and data manipulation

Stefano Ghidoni

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB