Exercise

This exercise will give you some experience in using OpenCV to read images and to perform basic color space conversions.

1. Download OpenCV and compile the helloWorldOpenCV.zip project found on black board
2. Load and display the legoHouse.jpg image (in images.zip) using OpenCV. The following functions are usefull: cv::imread, cv::imshow, cv::namedWindow and cv::waitKey.
3. Astronaut1.jpg needs to be rotated 90 degrees. Implement a rotation algorithm using single pixel access.
4. Try to segment the red spoons. First: convert to a suitable color space. Next: apply a threshold. Display the segmented spoon(s) using cv::imshow.

Hints and information regarding the image container cv::Mat

OpenCv tutorials

- Check out OpenCV tutorials

  opencv.org → tutorials → core module

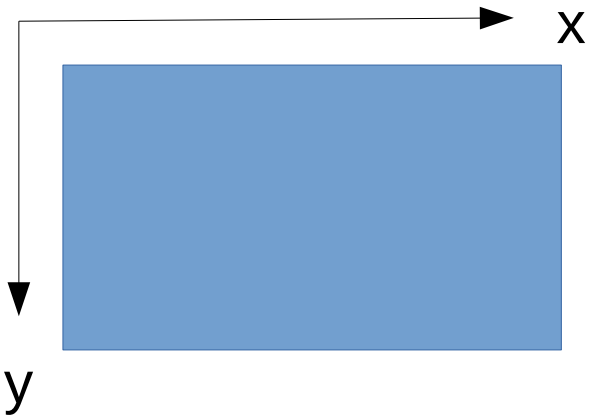
  - Mat – The Basic Image Container
  - How to scan images, ...

https://docs.opencv.org/master/d9/df8/tutorial_root.html

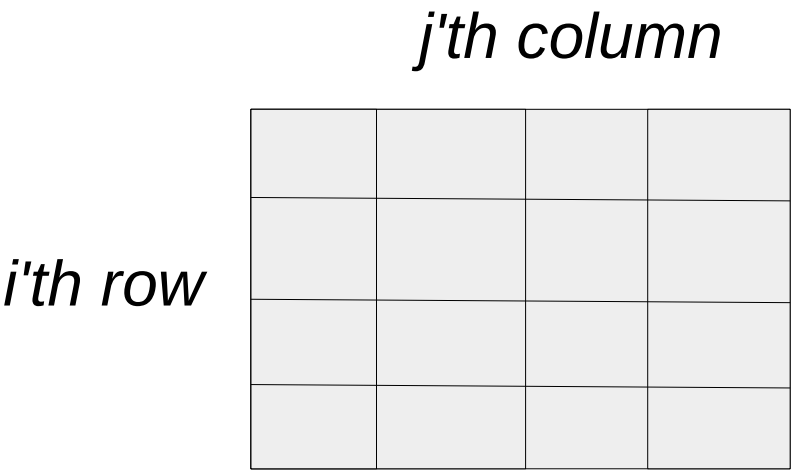
Image coordinates

Two conventions

- xy-coordinates



- ij-matrix indices
  (sometimes called yx)



The cv::Mat

An image is stored as a matrix containing one or more **channels**, with pixel intensities stored as defined by a color **depth**

Type of matrix

```
cv::Mat m(50,50, CV_8UC3 , cv::Scalar(0,0,255));
```

- Depth
- Channels

Mat::depth

Returns the depth of a matrix element.

C++: int Mat::**depth**() const

The method returns the identifier of the matrix element depth (the type of each individual channel). For example, for a 16-bit signed element array, the method returns CV_16S . A complete list of matrix types contains the following values:

- CV_8U - 8-bit unsigned integers (0..255)
- CV_8S - 8-bit signed integers (-128..127)
- CV_16U - 16-bit unsigned integers (0..65535)
- CV_16S - 16-bit signed integers (-32768..32767)
- CV_32S - 32-bit signed integers (-2147483648..2147483647)
- CV_32F - 32-bit floating-point numbers (-FLT_MAX..FLT_MAX, INF, NAN)
- CV_64F - 64-bit floating-point numbers (-DBL_MAX..DBL_MAX, INF, NAN)

Mat::channels

Returns the number of matrix channels.

C++: int Mat::**channels**() const

The method returns the number of matrix channels.

http://docs.opencv.org/modules/core/doc/basic_structures.html#mat-depth

Type of matrix

If unsure which type your matrix is → check depth + channels or type:

Mat::type

Returns the type of a matrix element.

C++: int Mat::**type**() const

The method returns a matrix element type. This is an identifier compatible with the CvMat type system, like CV_16SC3 or 16-bit signed 3-channel array, and so on.

http://docs.opencv.org/modules/core/doc/basic_structures.html#mat-depth

The easy way: Consult table

A Mapping of Type to Numbers in OpenCV

	C1	C2	C3	C4
CV_8U	0	8	16	24
CV_8S	1	9	17	25
CV_16U	2	10	18	26
CV_16S	3	11	19	27
CV_32S	4	12	20	28
CV_32F	5	13	21	29
CV_64F	6	14	22	30

http://ninghang.blogspot.dk/2012/11/list-of-mat-type-in-opencv.html

Two ways to access a cv::Mat

1) Pointer to row i + access column index j

```
cv::Vec3b* data = img.ptr<cv::Vec3b>(i);
data[j][0] = 0; //set blue to 0
```

2) Use the “at” method specifying type

```
img.at<cv::Vec3b>(i,j)[0] = 0; //set blue to 0
```

Matrix type

- Note the image is stored as a matrix storing elements of type type cv::Vec3b
  - Blue, green, red ordering with values between 0 and 255 (uchar)
  - (see http://docs.opencv.org/doc/user_guide/ug_mat.html for alternatives)

```
cv::Vec3b intensity = img.at<cv::Vec3b>(y, x);
uchar blue = intensity.val[0];
uchar green = intensity.val[1];
uchar red = intensity.val[2];
```