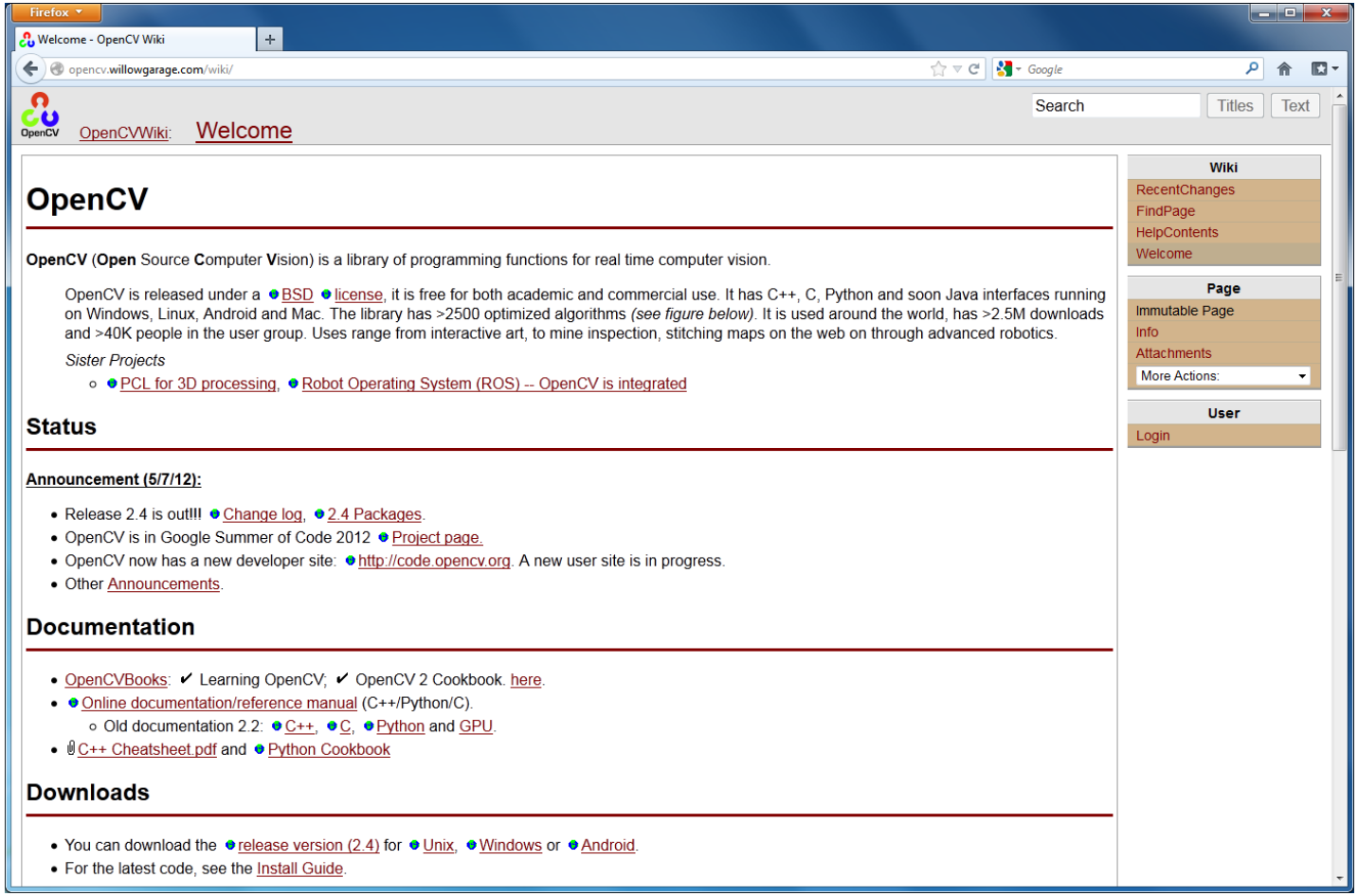
Introduction to OpenCV

Stefan Holzer, David Joseph Tan

Chair for Computer Aided Medical Procedures
Technische Universität München
Germany

Introduction to OpenCV

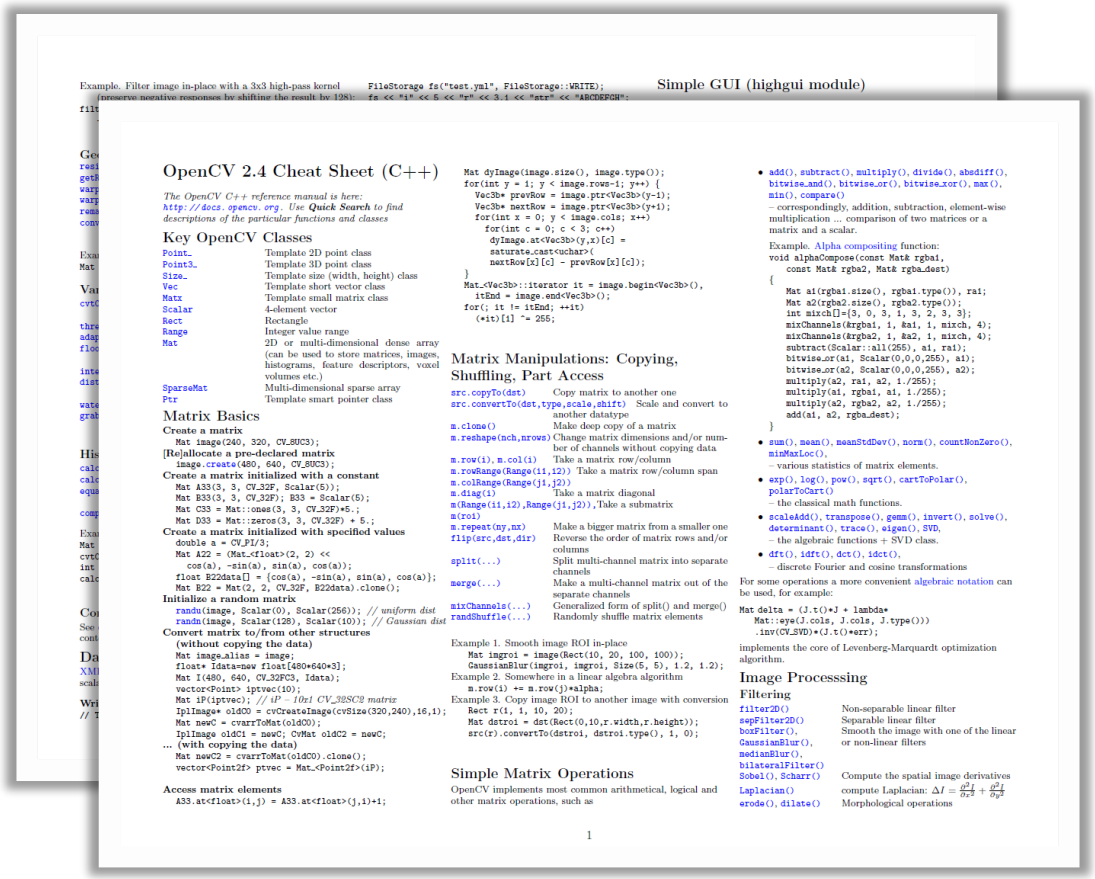
Where to get OpenCV?



<http://opencv.willowgarage.com/wiki/>

Introduction to OpenCV

Where to get OpenCV?



OpenCV 2.4 Cheat Sheet (C++)

The OpenCV C++ reference manual is here: http://docs.opencv.org. Use ***Quick Search*** *to find descriptions of the particular functions and classes*

Key OpenCV Classes

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height) class
Vec	Template short vector class
Matx	Template small matrix class
Scalar	4-element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specified values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) <<
    cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

Initialize a random matrix

```
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist
```

Convert matrix to/from other structures (without copying the data)

```
Mat image_alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3, Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP - 10x1 CV_32SC2 matrix
IplImage* oldC0 = cvCreateImage(cvSize(320,240),16,1);
Mat newC = cvarrToMat(oldC0);
IplImage oldC1 = newC; CvMat oldC2 = newC;
```

... (with copying the data)

```
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat_<Point2f>(iP);
```

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y < image.rows-1; y++) {
  Vec3b* prevRow = image.ptr<Vec3b>(y-1);
  Vec3b* nextRow = image.ptr<Vec3b>(y+1);
  for(int x = 0; x < image.cols; x++)
    for(int c = 0; c < 3; c++)
      dyImage.at<Vec3b>(y,x)[c] =
      saturate_cast<uchar>(
      nextRow[x][c] - prevRow[x][c]);
}
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
  itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
  (*it)[1] ^= 255;
```

Matrix Manipulations: Copying, Shuffling, Part Access

src.copyTo(dst)	Copy matrix to another one
src.convertTo(dst,type,scale,shift)	Scale and convert to another datatype
m.clone()	Make deep copy of a matrix
m.reshape(nch,nrows)	Change matrix dimensions and/or number of channels without copying data
m.row(i), m.col(i)	Take a matrix row/column
m.rowRange(Range(i1,i2))	Take a matrix row/column span
m.colRange(Range(j1,j2))	
m.diag(i)	Take a matrix diagonal
m(Range(i1,i2),Range(j1,j2)),	Take a submatrix
m(roi)	
m.repeat(ny,nx)	Make a bigger matrix from a smaller one
flip(src,dst,dir)	Reverse the order of matrix rows and/or columns
split(...)	Split multi-channel matrix into separate channels
merge(...)	Make a multi-channel matrix out of the separate channels
mixChannels(...)	Generalized form of split() and merge()
randShuffle(...)	Randomly shuffle matrix elements

Example 1. Smooth image ROI in-place

```
Mat imgroi = image(Rect(10, 20, 100, 100));
GaussianBlur(imgroi, imgroi, Size(5, 5), 1.2, 1.2);
```

Example 2. Somewhere in a linear algebra algorithm

```
m.row(i) += m.row(j)*alpha;
```

Example 3. Copy image ROI to another image with conversion

```
Rect r(1, 1, 10, 20);
Mat dstroi = dst(Rect(0,10,r.width,r.height));
src(r).convertTo(dstroi, dstroi.type(), 1, 0);
```

Simple Matrix Operations

OpenCV implements most common arithmetical, logical and other matrix operations, such as

- add(), subtract(), multiply(), divide(), absdiff(), bitwise_and(), bitwise_or(), bitwise_xor(), max(), min(), compare()
  – correspondingly, addition, subtraction, element-wise multiplication ... comparison of two matrices or a matrix and a scalar.

  Example. Alpha compositing function:

```
void alphaCompose(const Mat& rgba1,
    const Mat& rgba2, Mat& rgba_dest)
{
  Mat ai(rgba1.size(), rgba1.type()), rai;
  Mat a2(rgba2.size(), rgba2.type());
  int mixch[]={3, 0, 3, 1, 3, 2, 3, 3};
  mixChannels(&rgba1, 1, &ai, 1, mixch, 4);
  mixChannels(&rgba2, 1, &a2, 1, mixch, 4);
  subtract(Scalar::all(255), ai, rai);
  bitwise_or(ai, Scalar(0,0,0,255), ai);
  bitwise_or(a2, Scalar(0,0,0,255), a2);
  multiply(a2, rai, a2, 1./255);
  multiply(ai, rgba1, ai, 1./255);
  multiply(a2, rgba2, a2, 1./255);
  add(ai, a2, rgba_dest);
}
```

- sum(), mean(), meanStdDev(), norm(), countNonZero(), minMaxLoc(),
  – various statistics of matrix elements.
- exp(), log(), pow(), sqrt(), cartToPolar(), polarToCart()
  – the classical math functions.
- scaleAdd(), transpose(), gemm(), invert(), solve(), determinant(), trace(), eigen(), SVD,
  – the algebraic functions + SVD class.
- dft(), idft(), dct(), idct(),
  – discrete Fourier and cosine transformations

For some operations a more convenient algebraic notation can be used, for example:

```
Mat delta = (J.t()*J + lambda*
  Mat::eye(J.cols, J.cols, J.type()))
  .inv(CV_SVD)*(J.t()*err);
```

implements the core of Levenberg-Marquardt optimization algorithm.

Image Processsing

Filtering

filter2D()	Non-separable linear filter
sepFilter2D()	Separable linear filter
boxFilter(),	Smooth the image with one of the linear
GaussianBlur(),	or non-linear filters
medianBlur(),	
bilateralFilter()	
Sobel(), Scharr()	Compute the spatial image derivatives
Laplacian()	compute Laplacian: $\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$
erode(), dilate()	Morphological operations

1

<http://docs.opencv.org/trunk/opencv_cheatsheet.pdf>

Introduction to OpenCV

#include and namespace

Include

These are the most common include files are:

cv.h
highgui.h

```
#include <opencv/cv.h>
#include <opencv/highgui.h>
```

Namespace

All OpenCV functions use the namespace:

cv

```
using namespace cv;
cv::functionName();
```

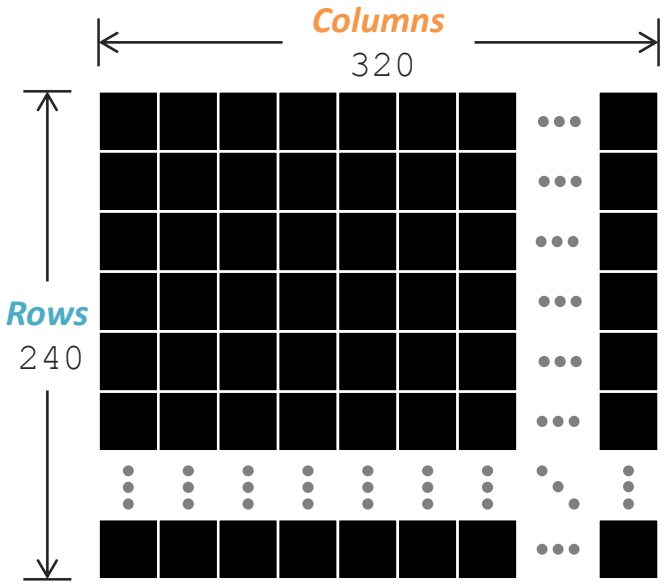
Images and Matrices

How to use them?

Images and Matrices are represented by the same type.

```
cv::Mat image(240, 320, CV_8UC3);
```

Rows *Columns* *Mat Type*



Mat Type

CV_8UC3

8U

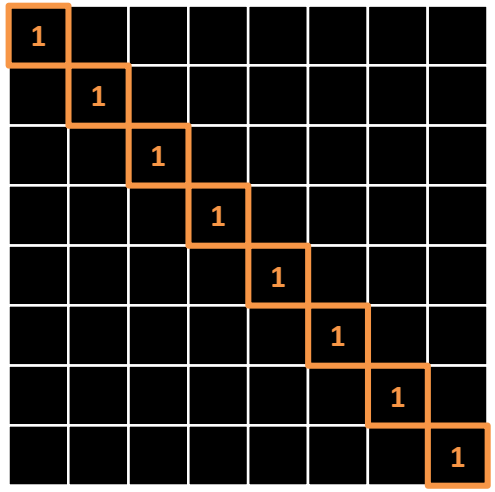
Datatype: 8U, 8S
16U, 16S, 32F, 64F

C3

Number of Channels:
C1, C2, C3, C4
Default: C1

Images and Matrices

How to initialize?

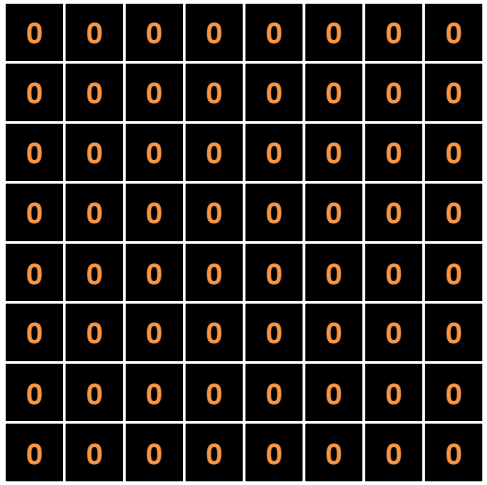


Matrix initialized as Identity

```
cv::Mat I88
= cv::Mat::eye(8, 8, CV_32F);
```

Images and Matrices

How to initialize?

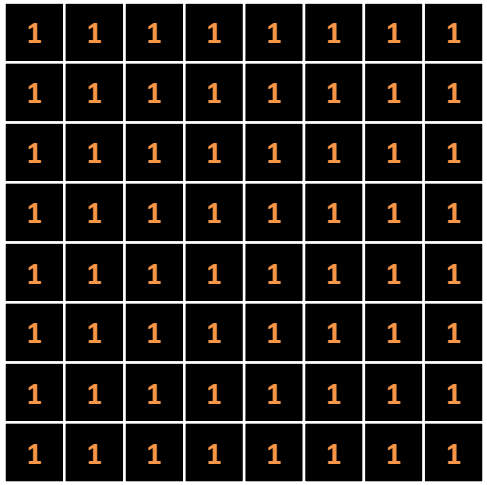


Matrix initialized with zeroes

```
cv::Mat D88
= cv::Mat::zeros(8, 8, CV_32F);
```

Images and Matrices

How to initialize?

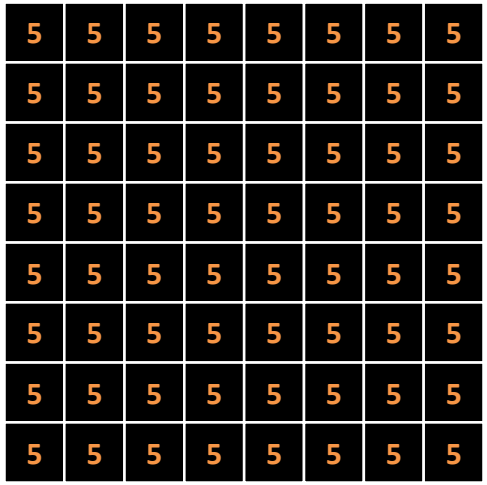


Matrix initialized with ones

```
cv::Mat D88
= cv::Mat::ones(8, 8, CV_32F);
```

Images and Matrices

How to initialize?



Matrix initialized with a constant

```
cv::Mat A88(8, 8, CV_32F);
A88 = cv::Scalar(5);

cv::Mat B88(8, 8, CV_32F, cv::Scalar(5));

cv::Mat C88
= cv::Mat::ones(8, 8, CV_32F) * 5.;

cv::Mat D88
= cv::Mat::zeros(8, 8, CV_32F) + 5.;
```

Images and Matrices

How to initialize?

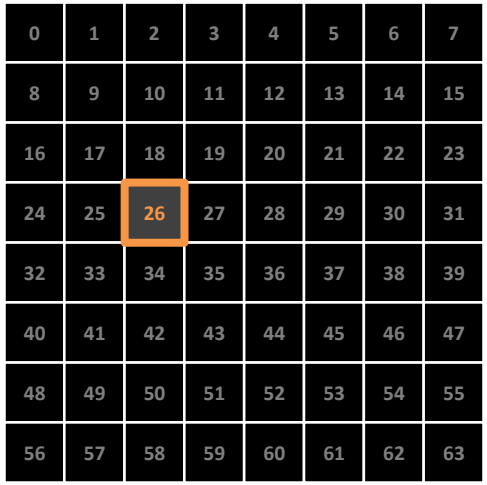
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Matrix initialized with specific values

```
float E88data[]
= {0, 1, 2, 3, ..., 63};
cv::Mat E88
= cv::Mat(8, 8, CV_32F, E88data).clone();
```

Images and Matrices

How to access the elements?



To access an element:

```
im.at<float>(idx);
im.at<float>(row,col);
```

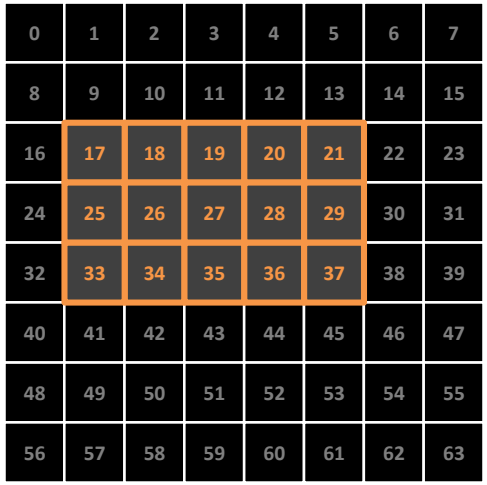
Depends on the Matrix Data Type

For example:

```
im.at<float>(26) = 1;
im.at<float>(3,2) = 1;
```

Images and Matrices

How to access the elements?



To access a submatrix:

```
im.row(i); im.col(i);
im.rowRange(cv::Range(r0,rn));
im.colRange(cv::Range(c0,cn));
im(cv::Range(r0,rn),cv::Range(c0,cn));
```

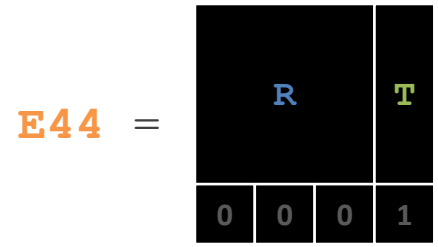
for rows $r_0, \ldots, r_{n-1}$ and *for columns* $c_0, \ldots, c_{n-1}$

For example:

```
im(cv::Range(2,5),cv::Range(1,6));
```

---

*How do you get the 3x3 rotation matrix **R** and 3x1 translation matrix **T** from a 4x4 extrinsic matrix **E44**?*




```
R = E44(cv::Range(0,3),cv::Range(0,3));
T = E44(cv::Range(0,3),cv::Range(3,4));
```

Copy by Address

How to copy by value?

`.clone()`

Images and Matrices

Scalar Operators and Matrix Operators

Scalar Operator

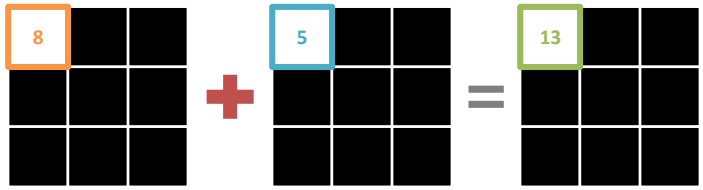
```
cv::Mat im2 = im + 5;
```


```
cv::Mat im2 = im - 5;
```

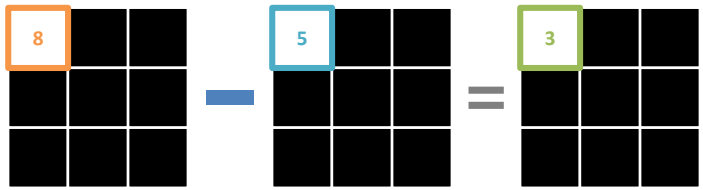

```
cv::Mat im2 = im * 5;
```

Matrix Operator

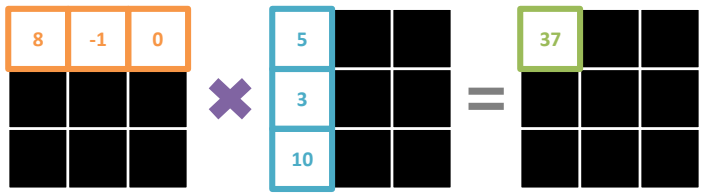
```
cv::Mat mat = mat1 + mat2;
```




```
cv::Mat mat = mat1 - mat2;
```




```
cv::Mat mat = mat1 * mat2;
```

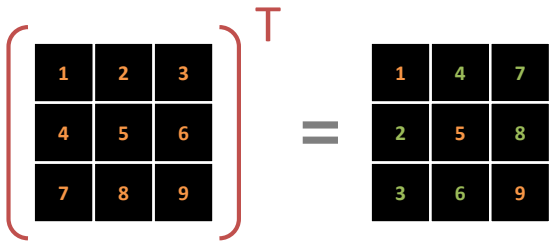


Matrices

Simple Math Operators

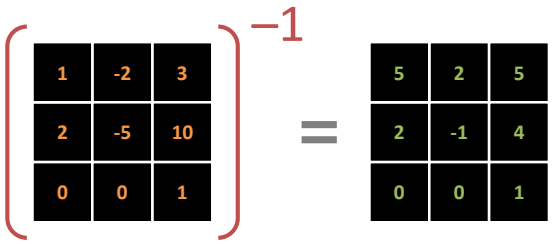
Matrix Transpose

`mat.t()`



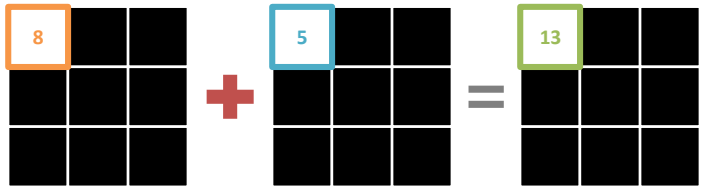
Matrix Inverse

`mat.inv()`

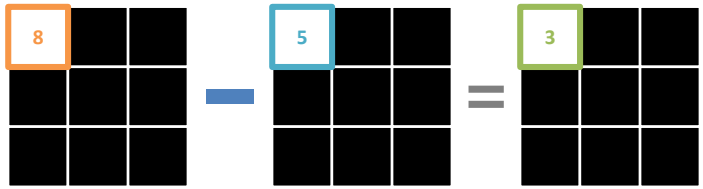


Matrix Operator

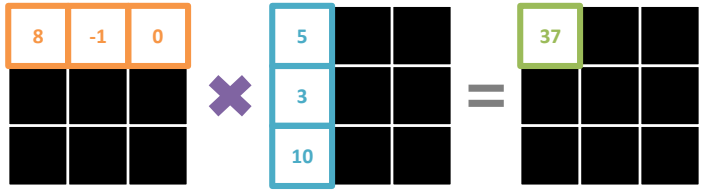
```
cv::Mat mat = mat1 + mat2;
```




```
cv::Mat mat = mat1 - mat2;
```




```
cv::Mat mat = mat1 * mat2;
```



Images

How to input or output an image?

Load Image

Read image from disk.

```
cv::imread(filename,0/1);
```

0: read as grayscale image
1: read as color image

Save Image

Write image to disk.

```
cv::imwrite(filename,im);
```

Visualize Image

Show image in a window.

```
cv::imshow(title,im);
```

Note: if CV_32FC1, the gray value range is 0 to 1. Everything above 1 is white and everything below 0 is black.

Waitkey

Waits n milliseconds for user input.

```
cv::waitkey(n);
```

If n == -1, it waits forever.
Note: There must be a waitkey to show the image.

Images

Simple Image Operations

Convert Color

```
cv::cvtColor(colorIm, grayIm, CV_BGR2GRAY);
```

Smoothing

```
cv::GaussianBlur(src, dst, ksize, sigma);
```

cv::Size(width, height)

Gradients

```
cv::Sobel(src, dst, ddepth, xorder, yorder);
```

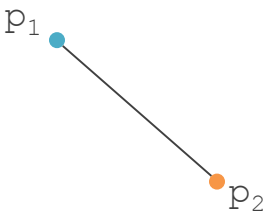
destination image depth e.g. CV_32FC1

Order of the derivative x

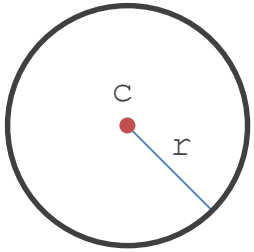
Order of the derivative y

Images

Drawing Primitives




```
cv::line(im,p1,p2,color,thickness);
         cv::Point(x,y)
```




```
cv::circle(im,c,r,color,thickness);
                  CV_RGB(r,g,b)
```

User Interface

How control a window using keyboard or mouse?

Window

Create and Destroy Window.

```
cv::namedWindow(windowName)
cv::destroyWindow(windowName)
```

Keyboard

Wait for user to press key.

```
const int key = cv::waitKey(-1);
If (key == 'a')
{
  // Do Something
}
```

Mouse

Setup a mouse handler.

```
cv::setMouseCallback(windowName,
mouseHandler, NULL );

void mouseHandler(int event, int x,
int y, int flags, void *param)
{
  if (event == CV_EVENT_LBUTTONDOWN)
    // Do Something
}
```

GUI Elements

Trackbar

Trackbar

```
int createTrackbar(const string& trackbarname,
const string& winname, int* value, int count,
TrackbarCallback onChange=0, void* userdata=0);
```

```
int sigma = 0;
cv::createTrackbar("sigma", "image", &sigma, 500);
while (true)
{
  cv::Mat smoothIm;
  cv::GaussianBlur(im, smoothIm, cv::Size(5,5), sigma/100.0f);
  cv::imshow("image", smoothIm);
  cv::waitKey(10);
}
```

Demo