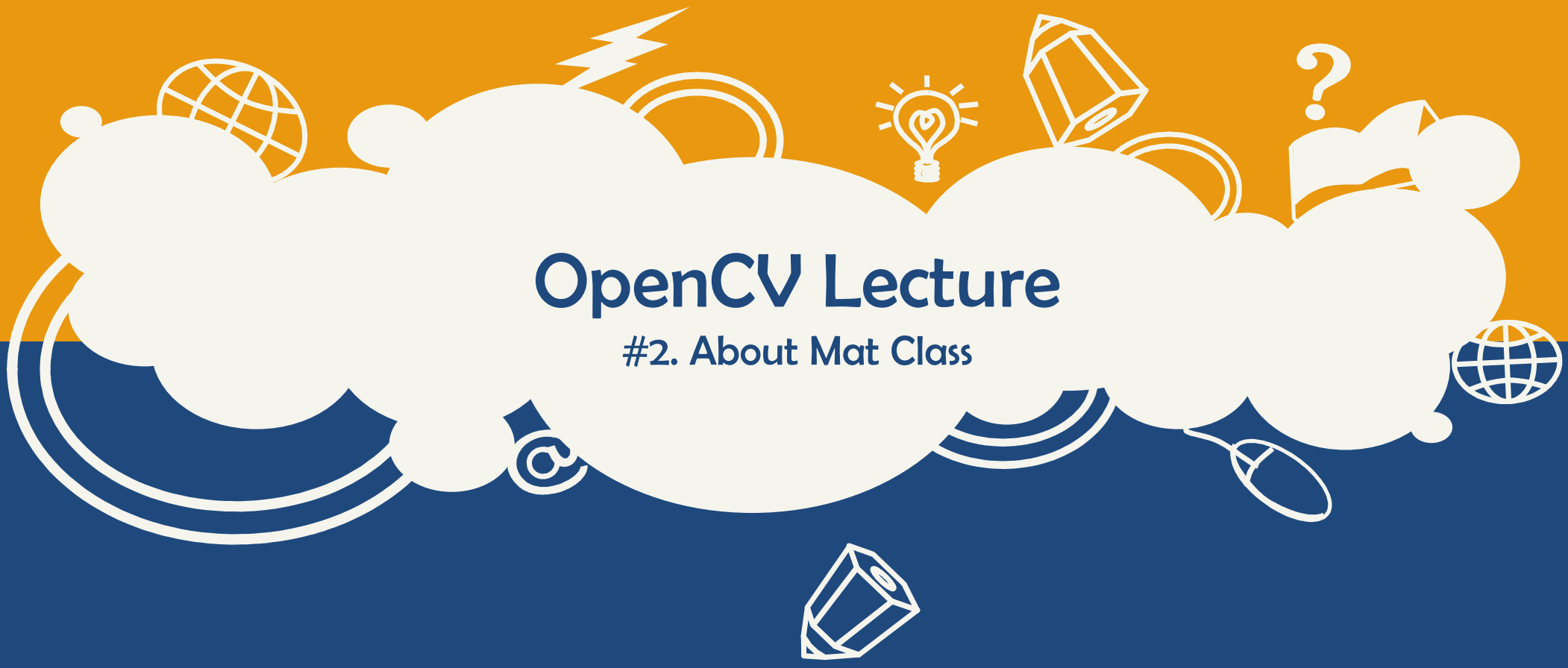


OpenCV Lecture

#2. About Mat Class

MareArts

Mat and Image, Video

Image Reading

```
//image reading
Mat img = imread("ss.jpg"); //imread("ss.jpg",0); -> grayscale
namedWindow("img", 0); //make window
imshow("img", img); //show
waitKey(0);
```

Video Reading

```
//video reading
//VideoCapture cap(0); // open the default camera
VideoCapture cap("rhinos.avi"); // open the video file
if (!cap.isOpened()) // check if we succeeded
    return -1;

namedWindow("video", 1);
for (;;)
{
    Mat frame;
    cap >> frame; // get a new frame from camera
    imshow("video", frame);
    if (waitKey(30) >= 0)
        break;
}
// the camera will be deinitialized automatically in VideoCapture destructor
return 0;
```

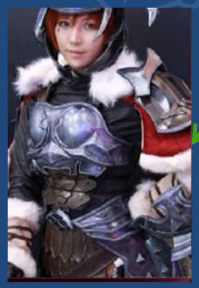
Image reading :
http://study.marearts.com/2016/06/opencv-mat-copyto-clone-roi-example-code.html

KeyCode :
http://study.marearts.com/2016/11/keycode-and-ascii-code.html

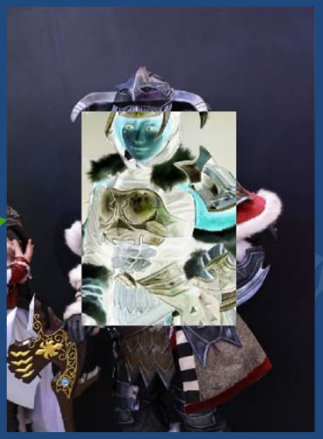
Video reading :
http://study.marearts.com/2013/09/opencv-video-file-load-and-display.html

Mat and Image

- Image related functions



```
Mat img = imread("ss.jpg");

Rect r(img.cols / 4, img.rows / 4, img.cols / 4 * 2, img.rows / 4 * 2);

//clone #1
Mat img2 = img.clone();
bitwise_not(img2, img2);

//clone #2
Mat img5 = img(r).clone();


//copyTo #1
Mat cimg;
img.copyTo(cimg);

//copyTo #2
Mat cimg2;
img(r).copyTo(cimg2);

//copyTo #3
Mat cimg3(Size(img.cols * 2, img.rows), img.type());
cimg3.setTo(255);
img.copyTo(cimg3(Range::all(), Range(0, img.cols)));
img2.copyTo(cimg3(Range::all(), Range(img2.cols, img2.cols * 2)));

//set roi
Mat roi(img, r);
//invert color
bitwise_not(roi, roi);
```



http://study.marearts.com/2016/06/opencv-mat-copyto-clone-roi-example-code.html

Mat and Pixel access

- Image Reading And Pixel Access in Mat
  - 1. using 'at'
    - Safety but most slow
  - 2. using 'ptr'
    - Faster than 'at'
  - 3. using 'data'
    - Fastest but unsafety

* Refer at, ptr, data, iteration example code to
http://study.marearts.com/2014/04/opencv-study-mat-point-access-method.html
http://study.marearts.com/2016/06/opencv-pixel-access-at-ptr-data.html

Mat and Pixel access

- at case

```
//using at
for (int i = img.rows / 10*3; i < img.rows / 10*4; ++i)
{
    for (int j = 0; j < img.cols; ++j)
    {
        //Vec3b means 'uchar 3ch'
        unsigned char b = img.at< cv::Vec3b>(i, j)[0];
        unsigned char g = img.at< cv::Vec3b>(i, j)[1];
        unsigned char r = img.at< cv::Vec3b>(i, j)[2];

        //printf("%d %d %d₩n", b, g, r);

        img.at< cv::Vec3b>(i, j)[0] = unsigned char(255 - b); //b
        img.at< cv::Vec3b>(i, j)[1] = unsigned char(255 - g); //g
        img.at< cv::Vec3b>(i, j)[2] = unsigned char(255 - r); //r
    }
}
```

Mat and Pixel access

- ptr case

```
//using ptr
for (int i = img.rows / 10 * 6; i < img.rows / 10 * 7; i++) {

    cv::Vec3b* ptr = img.ptr< cv::Vec3b>(i);

    for (int j = 0; j < img.cols; j++) {

        cv::Vec3b bgr = ptr[j];
        unsigned char b = (bgr[0]);
        unsigned char g = (ptr[j][1]); //note!!
        unsigned char r = (bgr[2]);

        ptr[j] = cv::Vec3b(255 - b, 255 - g, 255 - r);

    }
}
```

Mat and Pixel access

- data case

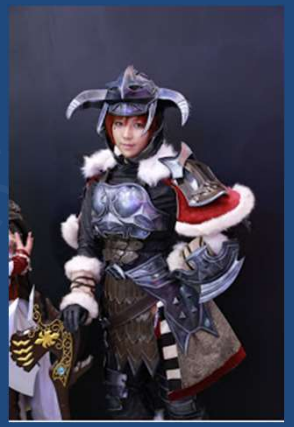
```
//using data
for (int i = img.rows / 10 * 8; i < img.rows / 10 * 9; i++) {
    for (int j = 0; j < img.cols; j++) {
        unsigned char r, g, b;
        b = img.data[i * img.step + j * img.elemSize() + 0];
        g = img.data[i * img.step + j * img.elemSize() + 1];
        r = img.data[i * img.step + j * img.elemSize() + 2];

        img.data[i * img.step + j * img.elemSize() + 0] = unsigned char(255 - b);
        img.data[i * img.step + j * img.elemSize() + 1] = unsigned char(255 - g);
        img.data[i * img.step + j * img.elemSize() + 2] = unsigned char(255 - r);

    }
}
```

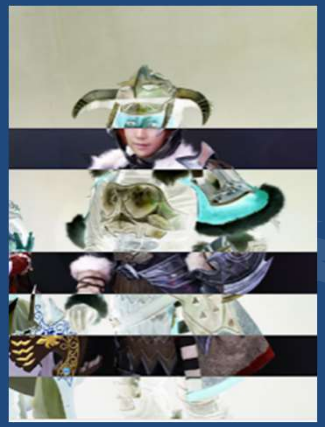
Mat and Pixel access

STL iteration case

```
//using iteration
cv::MatIterator_< cv::Vec3b> itd = img.begin< cv::Vec3b>(), itd_end = img.end< cv::Vec3b>();
for (int i = 0; itd != itd_end; ++itd, ++i) {
    cv::Vec3b bgr = (*itd);

    (*itd)[0] = 255 - bgr[0];
    (*itd)[1] = 255 - bgr[1];
    (*itd)[2] = 255 - bgr[2];
}
```

Input & output

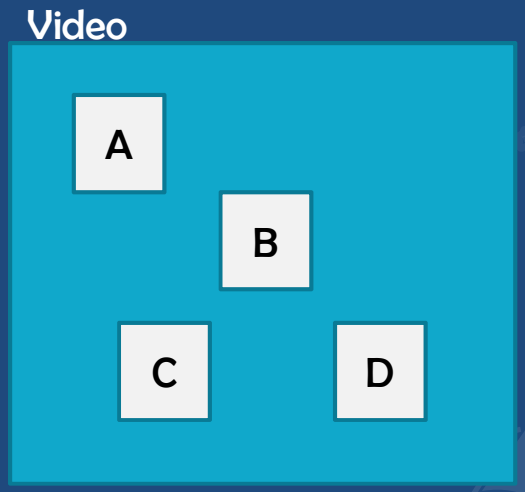




Mat and Pixel access

- Assignment #1
  - Make color invert in each region in video image
    - A : use 'at' operator
    - A : use 'ptr' operator
    - A : use 'iterator' operator
    - A : use 'data' operator

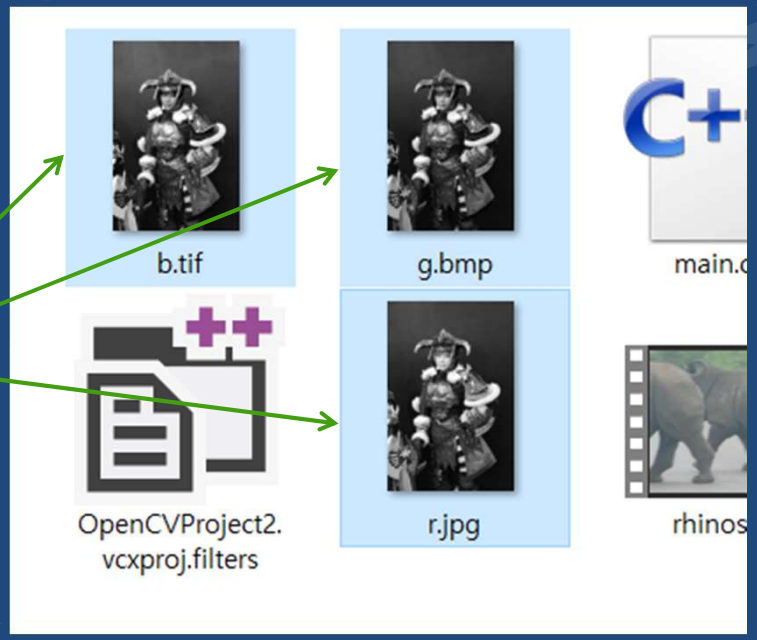
※ Region's position and scale is free.



Mat and Image, video write

Image write

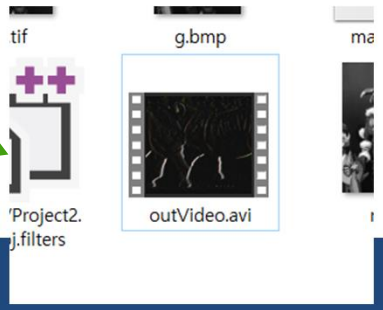
```
Mat img = imread("ss.jpg");
vector< Mat> rgbMat(3);
cv::split(img, rgbMat);

namedWindow("img", 0); //make window
imshow("img", rgbMat[2]); //show
waitKey(0);

imwrite("r.jpg", rgbMat[2]);
imwrite("g.bmp", rgbMat[1]);
imwrite("b.tif", rgbMat[0]);
```



Mat and Image, video write

- video write

```
VideoCapture capture("rhinos.avi");
Mat frame;

//Set properties
int askFileTypeBox = 0; //-1 is show box of codec
int Color = 1;
Size S = Size((int)capture.get(CV_CAP_PROP_FRAME_WIDTH), (int)capture.get(CV_CAP_PROP_FRAME_HEIGHT));

//make output video file
VideoWriter outVideo;
outVideo.open(".₩₩outVideo.avi", askFileTypeBox, capture.get(CV_CAP_PROP_FPS), S, Color);

//check
if (!capture.isOpened())
{
    printf("AVI file can not open.₩n");
    return 0;
}

//create window
namedWindow("w");
```




```
while (1)
{
    //grab frame from file & throw to Mat
    capture >> frame;
    if (frame.empty()) //Is video end?
        break;

    //processing example
    Sobel(frame, frame, frame.depth(), 1, 0);

    ////////////////////
    //outVideo.write(frame);
    outVideo << frame;
    //display and delay
    imshow("w", frame);

    if (waitKey(10) > 0)
        break;

}
```

http://study.marearts.com/2013/09/opencv-video-writer-example-source-code.html

Mat and Matrix

- We learned that in previous lesson
  - Mat creation and simple matrix operation
  - Let's look in more detail

```
Mat m = Mat::ones(3, 3, CV_64F);
m = m * 3;
cout << m << endl;

double dm[3][3] = { { 1, 2, 1 }, { 0, 1, 1 }, { 1, 0, 0 } };
Mat m2 = Mat(3, 3, CV_64F, dm);
cout << m2 << endl;
cout << m+m2 << endl;
cout << m-m2 << endl;
cout << m*m2 << endl;
cout << m/m2 << endl;
cout << m2.inv() << endl;
cout << m2.t() << endl;
```

```
int main(int, char)
{

    //Declaration and at the same time created
    Mat mtx(3, 3, CV_32F); // make a 3x3 floating-point matrix
    Mat cmtx(10, 1, CV_64FC2); // make a 10x1 2-channel floating-point
    // matrix (10-element complex vector)
    Mat img(Size(5, 3), CV_8UC3); // make a 3-channel (color) image
    // of 1920 columns and 1080 rows.

    //Created after the declaration
    Mat mtx2;
    mtx2 = Mat(3, 3, CV_32F);
    Mat cmtx2;
    cmtx2 = Mat(10, 1, CV_64FC1);

    //Create a point
    Mat* mtx3 = new Mat(3, 3, CV_32F);
    delete mtx3;

    //value set and print
    mtx.setTo(10);
    cout << mtx << endl;

    cmtx2.setTo(11);
    cout << cmtx2 << endl;


    return 0;
}
```

Mat and Matrix

- Matrix operations

```
//Matrix - matrix operations :
Mat Mc;
cv::add(Ma, Mb, Mc); // Ma+Mb    -> Mc
cout << Ma+Mb << endl;
cout << Mc << endl;
cv::subtract(Ma, Mb, Mc);       // Ma-Mb    -> Mc
cout << Ma - Mb << endl;
cout << Mc << endl;
Mc = Ma*Mb; //Ma*Mb;
cout << Mc << endl;
```

http://study.marearts.com/2016/06/opencv-mat-and-matrix-operation-examples.html

Mat and Matrix

- Elementwise matrix operations

```
//Elementwise matrix operations :
cv::multiply(Ma, Mb, Mc);   // Ma.*Mb   -> Mc
cout << Mc << endl;
Mc = Ma.mul(Mb);
cout << Mc << endl;
cv::divide(Ma, Mb, Mc);      // Ma./Mb  -> Mc
cout << Mc << endl;
Mc = Ma + 10; //Ma + 10 = Mc
cout << Mc << endl;
```

Mat and Matrix

- Vector products

```
//Vector products :
double va[] = { 1, 2, 3 };
double vb[] = { 0, 0, 1 };
double vc[3];

Mat Va(3, 1, CV_64FC1, va);
Mat Vb(3, 1, CV_64FC1, vb);
Mat Vc(3, 1, CV_64FC1, vc);

double res = Va.dot(Vb); // dot product:   Va . Vb -> res
Vc = Va.cross(Vb);    // cross product: Va x Vb -> Vc
cout << res << " " << Vc << endl;
```

Mat and Matrix

- Single matrix operations

```
//Single matrix operations :
Mc = Mb.t();      // transpose(Ma) -> Mb (cannot transpose onto self)
cout << Mc << endl;
cv::Scalar t = trace(Ma); // trace(Ma) -> t.val[0]
cout << t.val[0] << endl;
double d = determinant(Ma); // det(Ma) -> d
cout << d << endl;
Mc = Ma.inv();          // inv(Mb) -> Mc
invert(Ma, Mc);
cout << Mc << endl;
```

Mat and Matrix

- Inhomogeneous linear system solver

```
//Inhomogeneous linear system solver :
double dm2[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
Mat A(3, 3, CV_64FC1, dm2);
Mat x(3, 1, CV_64FC1);
double vvb[] = { 14, 32, 52 };
Mat b(3, 1, CV_64FC1, vvb);
cv::solve(A, b, x, DECOMP_SVD); //// solve (Ax=b) for x
cout << x << endl;
```

Mat and Matrix

- Eigen analysis

```
//Eigen analysis(of a symmetric matrix) :
float f11[] = { 1, 0.446, -0.56, 0.446, 1, -0.239, -0.56, 0.239, 1 };
Mat data(3, 3, CV_32F, f11);
Mat value, vector;
eigen(data, value, vector);
cout << "Eigenvalues" << value << endl;
cout << "Eigenvectors" << endl;
cout << vector << endl;
```

Mat and Matrix

- SVD example

```
//Singular value decomposition :
Mat w, u, v;
SVDecomp(data, w, u, v); // A = U W V^T
//The flags cause U and V to be returned transposed(does not work well without the transpose flags).
cout << w << endl;
cout << u << endl;
cout << v << endl;
```

Mat and gpuMat

- Cuda example

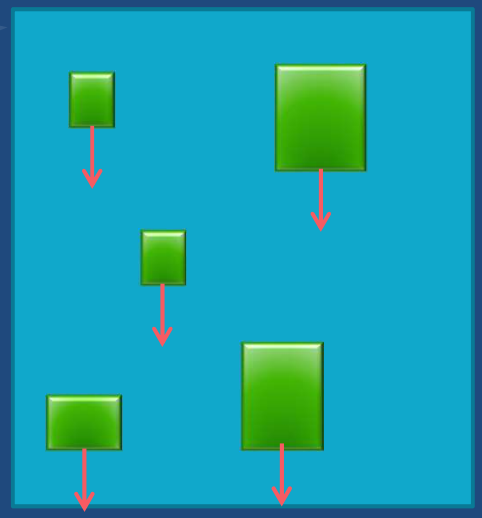
```
//#include "opencv2/cuda.hpp"
//#include "opencv2₩cudaarithm.hpp"

cuda::GpuMat gpuImg;
Mat img = imread("ss.jpg");

gpuImg.upload(img); //upload
vector< cuda::GpuMat> rgbGpuMat(3);
cuda::split(gpuImg, rgbGpuMat); //cuda processing

Mat r, g, b;
rgbGpuMat[0].download(b); //download
rgbGpuMat[1].download(g);
rgbGpuMat[2].download(r);


namedWindow("r", 0); //make window
imshow("r", r); //show
namedWindow("g", 0); //make window
imshow("g", g); //show
namedWindow("b", 0); //make window
imshow("b", b); //show
waitKey(0);
```

Mat and gpuMat

- Assignment #2
  - To use over 20 functions related to the GpuMat
  - Create example code

Bonus Assignment #1

- The bomb removing game
  - Square box comes down slowly from top to bottom.
  - Square box size and position is random.
  - When a user clicks on a box, box is eliminated.
  - And score is increase.
  - But if the box is touching the floor loses the score.



Thank you.

- See you later
  - Do not forget your assignment!!
  - I will miss you very much!!

Gyeongju, Anapji