Date: June 7, 2020

Please, read this lightly before you watch video.

We only have used Mat class in the previous video. However, there are more useful classes we are going to use from OpenCV.

Basic Object Types

1.Mat Class

This class is for representing a matrix in Algebra. It includes matrix-related methods and member variables that will be useful in image processing and computer vision tasks.

How to declare Mat Object:

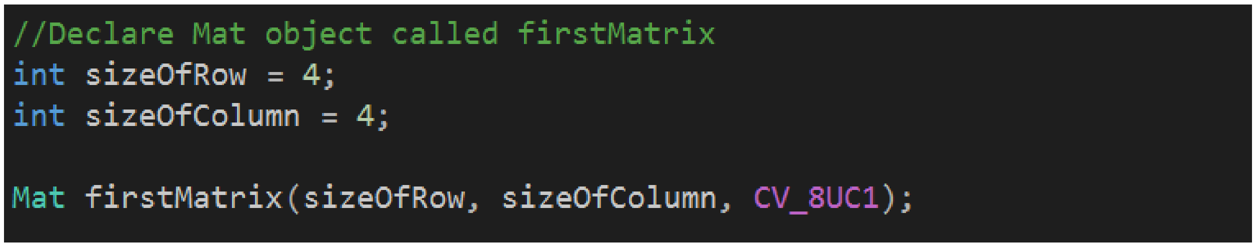
There are many ways to declare a Mat Object. We are going to take a look at only 2 ways here.

Method 1.

cv::Mat nameOfObject (sizeOfRow, sizeOfColumn, type(it indicates number of channels and data type stroed in a Mat object)

Note: if you use namespace cv, you don't have to type cv::.

e.g.



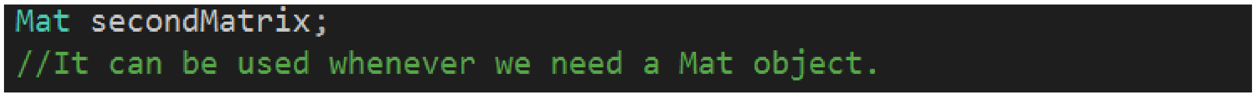
```
//Declare Mat object called firstMatrix
int sizeOfRow = 4;
int sizeOfColumn = 4;

Mat firstMatrix(sizeOfRow, sizeOfColumn, CV_8UC1);
```

//This creates 4x4 matrix with 1 channel (1 in CV_8UC1) and it will store 8-bits (8 in CV_8UC1) Unsigned type (UC in CV_8UC1).

Method 2.

Mat nameOfObject

e.g.

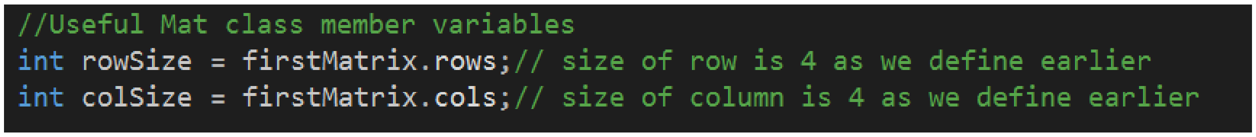


```
Mat secondMatrix;
//It can be used whenever we need a Mat object.
```

Date: June 7, 2020

Mat class also has many methods (a.k.a class functions) and member variables that can be used while image processing or computer vision tasks.

e.g.



```
//Useful Mat class member variables
int rowSize = firstMatrix.rows;// size of row is 4 as we define earlier
int colSize = firstMatrix.cols;// size of column is 4 as we define earlier
```

We will explore more details as we do computer vision projects with OpenCV

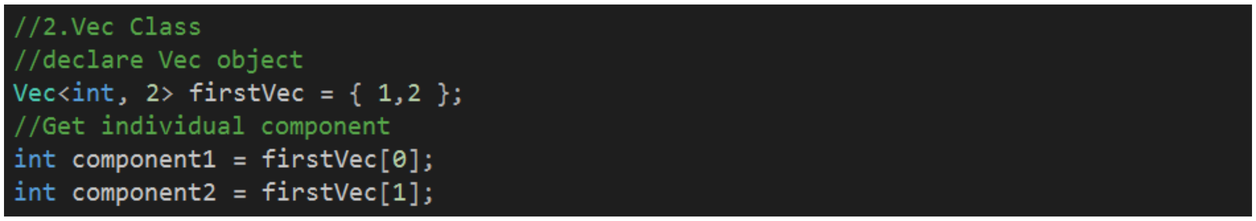
OpenCV Reference:

[1] https://docs.opencv.org/master/d3/d63/classcv_1_1Mat.html#details

2.Vec Class

This class is for representing a vector in Algebra. It includes vector-related methods and member variables that will be useful in image processing and computer vision tasks.

e.g.



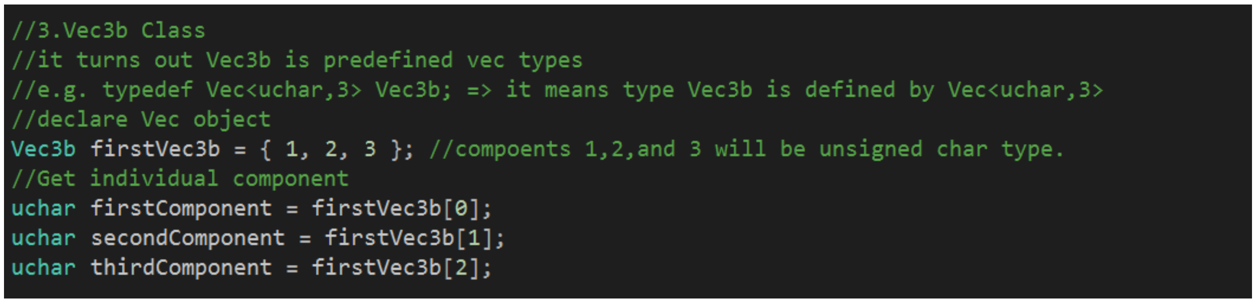
```
//2.Vec Class
//declare Vec object
Vec<int, 2> firstVec = { 1,2 };
//Get individual component
int component1 = firstVec[0];
int component2 = firstVec[1];
```

Date: June 7, 2020

3.Vec3b Class

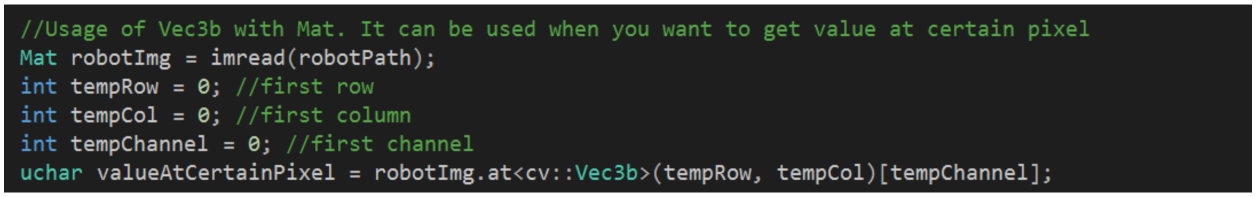
=>This class is nothing but predefined Vec type. i.e., typedef Vec<uchar, 2> Vec3b.

There are many predefined Vec types.



```
//3.Vec3b Class
//it turns out Vec3b is predefined vec types
//e.g. typedef Vec<uchar,3> Vec3b; => it means type Vec3b is defined by Vec<uchar,3>
//declare Vec object
Vec3b firstVec3b = { 1, 2, 3 }; //compoents 1,2,and 3 will be unsigned char type.
//Get individual component
uchar firstComponent = firstVec3b[0];
uchar secondComponent = firstVec3b[1];
uchar thirdComponent = firstVec3b[2];
```

Also, it can be used with Mat because in Algebra, matrix can be considered as a form of vector.

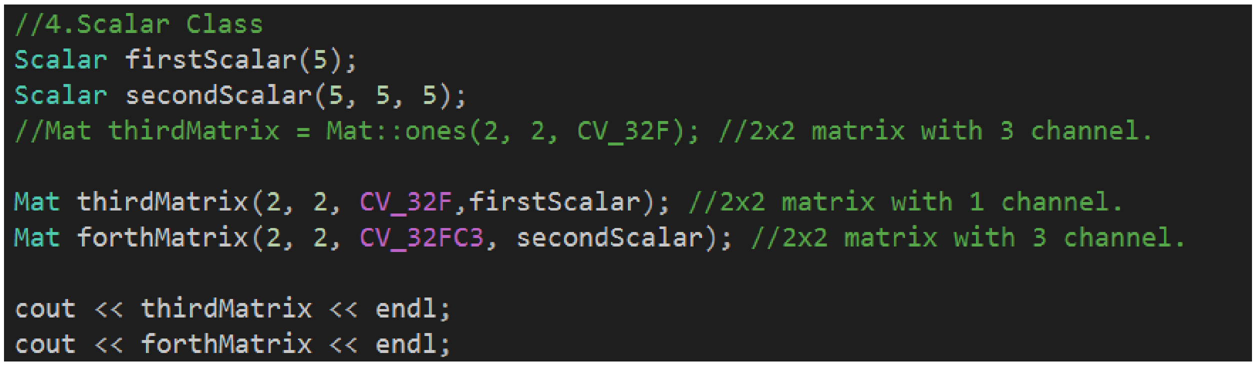


```
//Usage of Vec3b with Mat. It can be used when you want to get value at certain pixel
Mat robotImg = imread(robotPath);
int tempRow = 0; //first row
int tempCol = 0; //first column
int tempChannel = 0; //first channel
uchar valueAtCertainPixel = robotImg.at<cv::Vec3b>(tempRow, tempCol)[tempChannel];
```

4. Scalar

Scalar Class is often used to initialize value with certain value. It can save at most 4 values meaning it can be passed as a parameter when you initialize a matrix with 4 channels.

e.g.



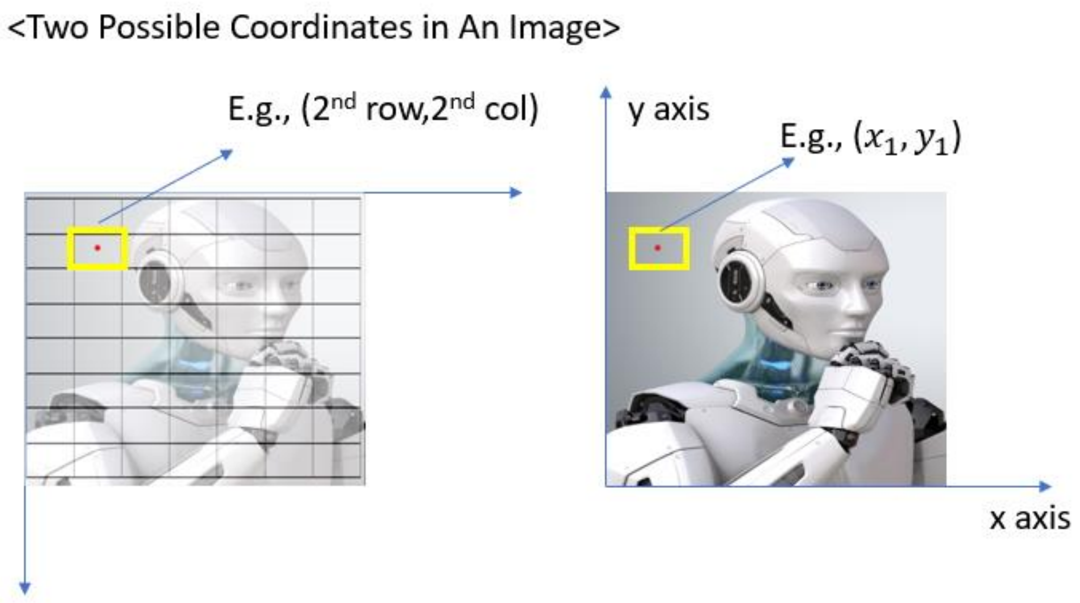
```
//4.Scalar Class
Scalar firstScalar(5);
Scalar secondScalar(5, 5, 5);
//Mat thirdMatrix = Mat::ones(2, 2, CV_32F); //2x2 matrix with 3 channel.

Mat thirdMatrix(2, 2, CV_32F,firstScalar); //2x2 matrix with 1 channel.
Mat forthMatrix(2, 2, CV_32FC3, secondScalar); //2x2 matrix with 3 channel.

cout << thirdMatrix << endl;
cout << forthMatrix << endl;
```

Date: June 7, 2020

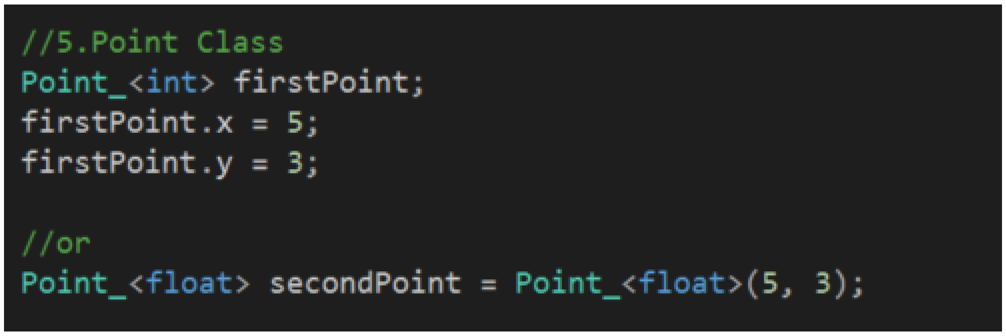
5.Point

Point class is often used to represent literally 2D points such as x, and y of image coordinates. This allows you to consider a pixel location as x, and y coordinates. For instance, pixel locations in an image can have different axis. We will see the usage of it later.

e.g.



2 Possible ways to declare a Point Object and its data type in it. Int type and Float type respectively.



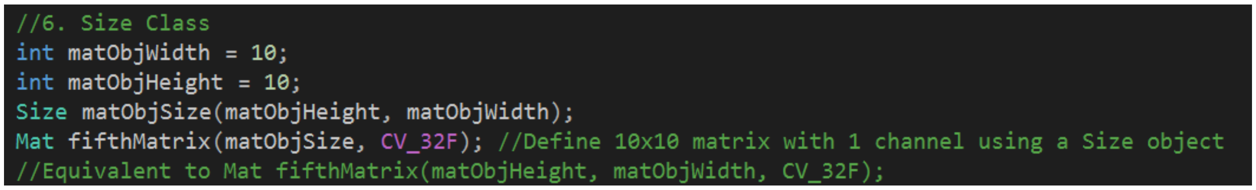
```
//5.Point Class
Point_<int> firstPoint;
firstPoint.x = 5;
firstPoint.y = 3;

//or
Point_<float> secondPoint = Point_<float>(5, 3);
```

Date: June 7, 2020

6.Size

Size class is to represent shape of a rectangle. Therefore, we can define Mat object with help of Size object.

e.g.



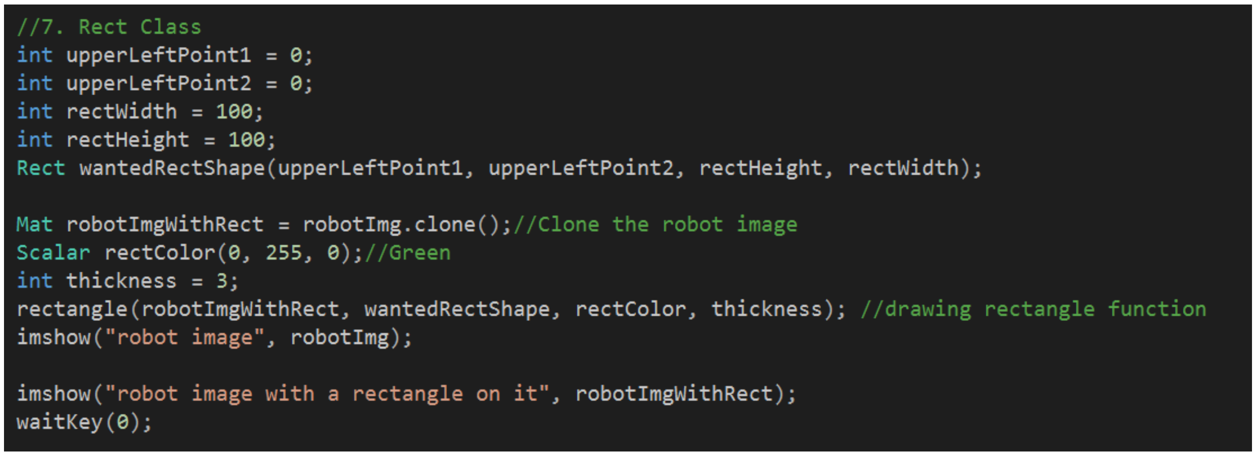
```
//6. Size Class
int matObjWidth = 10;
int matObjHeight = 10;
Size matObjSize(matObjHeight, matObjWidth);
Mat fifthMatrix(matObjSize, CV_32F); //Define 10x10 matrix with 1 channel using a Size object
//Equivalent to Mat fifthMatrix(matObjHeight, matObjWidth, CV_32F);
```

7.Rect

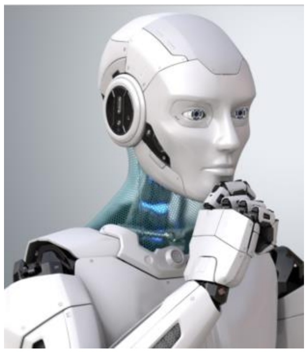
Rect class is very useful whenever you want to draw rectangle shape on an image.

A Rect object can be created with 2 parameters, one is coordinates of the upper-left corner and the other is width and height of a rectangle.

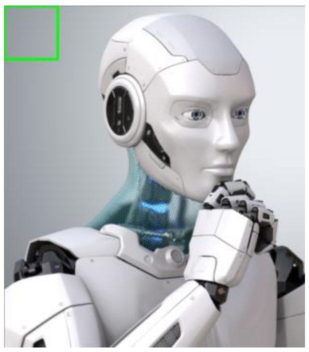
e.g.



```
//7. Rect Class
int upperLeftPoint1 = 0;
int upperLeftPoint2 = 0;
int rectWidth = 100;
int rectHeight = 100;
Rect wantedRectShape(upperLeftPoint1, upperLeftPoint2, rectHeight, rectWidth);

Mat robotImgWithRect = robotImg.clone();//Clone the robot image
Scalar rectColor(0, 255, 0);//Green
int thickness = 3;
rectangle(robotImgWithRect, wantedRectShape, rectColor, thickness); //drawing rectangle function
imshow("robot image", robotImg);

imshow("robot image with a rectangle on it", robotImgWithRect);
waitKey(0);
```



<Original Image>



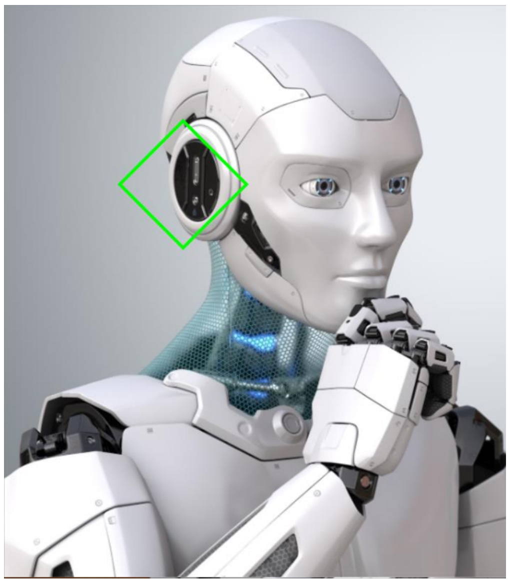
<A rectangle on the robot image>

8.RotatedRect

Sometimes, you want to define a rotated rectangle on your image. This is when RotatedRect Class becomes handy. It needs 3 parameters to be defined. (1) center point, (2) size of rotated rectangle (3) angle of rotation.

e.g.

```
//8. RotatedRect Class
Mat robotImgWithRotatedRect = robotImg.clone();
Point_<int> center = Point_<int>(200, 200);
Size rotatedRectSize(rectHeight, rectWidth);
float angle = 45;//45 degree
RotatedRect wantedRotatedRect(center, rotatedRectSize, angle);

const int numOfVertices = 4;
Point2f vertices[numOfVertices];
wantedRotatedRect.points(vertices); //get vertices of the rotated rectangle
//draw the rotated rectagle with drawing line function
for (int i = 0; i < numOfVertices; i++){
    Point_<float> currentVertix1 = Point_<float>(vertices[i]);
    Point_<float> currentVertix2 = Point_<float>(vertices[(i + 1) % 4]);//need 2 vertix to draw a line
    line(robotImgWithRotatedRect, currentVertix1, currentVertix2, rectColor, thickness);
}

imshow("robot image with a rotated rect", robotImgWithRotatedRect);
waitKey(0);
```



Reference

[1] robot image:

https://blogs.3ds.com/northamerica/future-robots-and-ensuring-human-safety/