Ways of accessing pixel information from Image in OPENCV

3 methods

- **Using ‘at’**
- **Using ‘direct address calculation’**
- **Using ‘ptr’**

1) Using Mat::at<_p>(for grayscale image)

```
Mat img = imread(“image.jpg”,0);      //0 is for grayscale image reading
for(int i=0;i< img.rows;i++)            //run through out the rows
       {
              for(int j=0;j< img.cols;j++)//run through out the cols
              {
                     cout<<(int)img.at<uchar>(i,j)<<” ”;//for getting
                     img.at<uchar>(i,j) = 0;     //for setting
              }
       }
```

2)

for grayscale image: by directly going to the address

```
Mat img = imread(“image.jpg”,0);      //0 is for grayscale image reading
for(int i=0;i< img.rows;i++)            //run through out the rows
       {
              for(int j=0;j< img.cols;j++)//run through out the cols
              {
                     cout<<(int)*(Img.data + Img.step[0]*i + j*Img.step[1]);
//for getting
                     *(Img.data + Img.step[0]*i + j*Img.step[1]) = 0;
//for setting

              }
       }
```

```
3)using pointer concept for grayscale image
       for(int i=0;i< img.rows;i++)            //run through out the rows
       {
              uchar *rowPtr = img.ptr<uchar>(i);  //rowPtr holds the pointer to
ith row
                     //so changes to rowPtr intern changes the grayImg

              for(int j=0;j< img.cols;j++)//run through out the cols
              {
                cout<<(int)rowPtr[j]<<" ";   //for getting
                rowPtr[j] = 0;     //for setting
            }
        }
```

```
    4)
    For IplImage(RGB access)

    for(i=0;i<image->height;i++)
    {
        for(j=0;j<image->width;j++)
        {
            CvScalar s;
            s=cvGet2D(image,i,j);   //for getting
            s.val[0]=0;//B
            s.val[2]=0;//R
            cvSet2D(image,i,j,s);   //for setting
        }
    }
```

5)

for color image: by directly going to the address

```
    Mat img = imread("image.jpg",1);      //1 is for color image reading
    for(int k =0;k<img.step[1];k++)
    {
        for(int i=0;i< img.rows;i++)           //run through out the rows
        {
            for(int j=0;j< img.cols;j++)//run through out the cols
            {
                cout<<(int)*(Img.data + k + Img.step[0]*i +
j*Img.step[1]);  //for getting
                *(Img.data + k + Img.step[0]*i + j*Img.step[1]) = 0;
//for setting

            }
        }
    }
```

```
    6)
    For Mat(RGB pixel access)using 'ptr'


    Mat img=imread("image.jpg",1);  //1 is for rgb image read

    vector<Mat> planes;  //planes variable is vector type with Mat as individual
element
    split(img,planes);  //splitting img(which is rgb) into 3 planes and stored in
different Mat of planes variable

    for(int i=0;i< img.rows;i++)           //run through out the rows
        {
            uchar *rowPtr0 = planes[0].ptr<uchar>(i);  //rowPtr holds the
pointer to ith row
                //so changes to rowPtr intern changes the grayImg

            for(int j=0;j< img.cols;j++)//run through out the cols
            {
                cout<<(int)rowPtr[j]<<” ”;   //for getting
                rowPtr[j] = 0;     //for setting
            }
        }
    merge(planes,img);    //opposite of split which merges arrayes of Mat to single multidimensional Mat
```

7)

For Mat(RGB pixel access)using ‘at’

```
    Mat img=imread(“image.jpg”,1);  //1 is for rgb image read

    vector<Mat> planes;  //planes variable is vector type with Mat as individual element
    split(img,planes);  //splitting img(which is rgb) into 3 planes and stored in different Mat of planes variable

    for(int i=0;i< img.rows;i++)           //run through out the rows
        {
            for(int j=0;j< img.cols;j++)//run through out the cols
            {
                cout<<(int)planes[0].at<uchar>(i,j)<<” ”;   //for getting
                planes[0].at<uchar>(i,j) = 0;     //for setting
            }
        }
    merge(planes,img);    //opposite of split which merges arrayes of Mat to single multidimensional Mat
```

-Shridhar Kini