Ex1: How does a program read the cvMat object, in particular, what is the order of the pixel structure?

The class Mat represents an n-dimensional dense numerical single-channel or multi-channel array. It can be used to store real or complex-valued vectors and matrices, grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms (though, very high-dimensional histograms may be better stored in a SparseMat ). The data layout of the array M is defined by the array M.step[], so that the address of element (i0,...,iM.dims−1), where 0≤ik<M.size[k], is computed as:

*addr(Mi0,...,iM.dims−1)=M.data+M.step[0]∗i0+M.step[1]∗i1+...+M.step[M.dims−1]∗iM.dims−1*

In case of a 2-dimensional array, the above formula is reduced to:

*addr(Mi,j)=M.data+M.step[0]∗i+M.step[1]∗j*

Note that M.step[i] >= M.step[i+1] (in fact, M.step[i] >= M.step[i+1]\*M.size[i+1] ). This means that 2-dimensional matrices are stored row-by-row, 3-dimensional matrices are stored plane-by-plane, and so on. M.step[M.dims-1] is minimal and always equal to the element size M.elemSize() .

So, the data layout in Mat is fully compatible with CvMat, IplImage, and CvMatND types from OpenCV 1.x. It is also compatible with the majority of dense array types from the standard toolkits and SDKs, such as Numpy (ndarray), Win32 (independent device bitmaps), and others, that is, with any array that uses steps (or strides) to compute the position of a pixel. Due to this compatibility, it is possible to make a Mat header for user-allocated data and process it in-place using OpenCV functions.

So, the cvMat object is also a matrix (2D array), we can access the pixels in cvMat by using cvMat.at(x,y),each of whose indice represents each pixel in an image. If the pixel has multiple color channels, we could access a pixel by using cvMat.at(x,y)[index], where index indicates which channel we want to access the pixel.