Changes to ossimTileToIplImage

Disclaimer

- Though SSC Pacific makes every effort to perform quality assurance on its training materials, the material in this presentation may inadvertently include technical inaccuracies or other errors. We would be grateful if users notify us of any errors or inaccuracies they may find.
- The presentation contains references to links and to third-party websites. These are provided for the convenience and interest of users and this implies neither responsibility for, nor approval of, information contained in these websites on the part of the U.S. Government. The USG makes no warranty, either express or implied, as to the accuracy, availability or content of information, text, graphics in the links/third party websites. The USG has not tested any software located at these sites and does not make any representation as to the quality, safety, reliability or suitability of such software, nor does this presentation serve to endorse the use of such sites.

Overview of Talk

- Introduction
- Making the image a color image instead of grayscale
  - Accessing OpenCV image data
  - Accessing ossimImageData data
  - Allocating an image
  - Displaying images
- Adding a threshold to the image
  - Using what we learned from previous examples
- Blob process the image
  - Allocating an image
  - Converting a color image to grayscale
  - Extracting the individual blobs and printing their information to the screen
- Conclusions

Accessing OpenCV image data

- Image data is stored in an array
- To access all data in an image you will need to use a loop similar to:

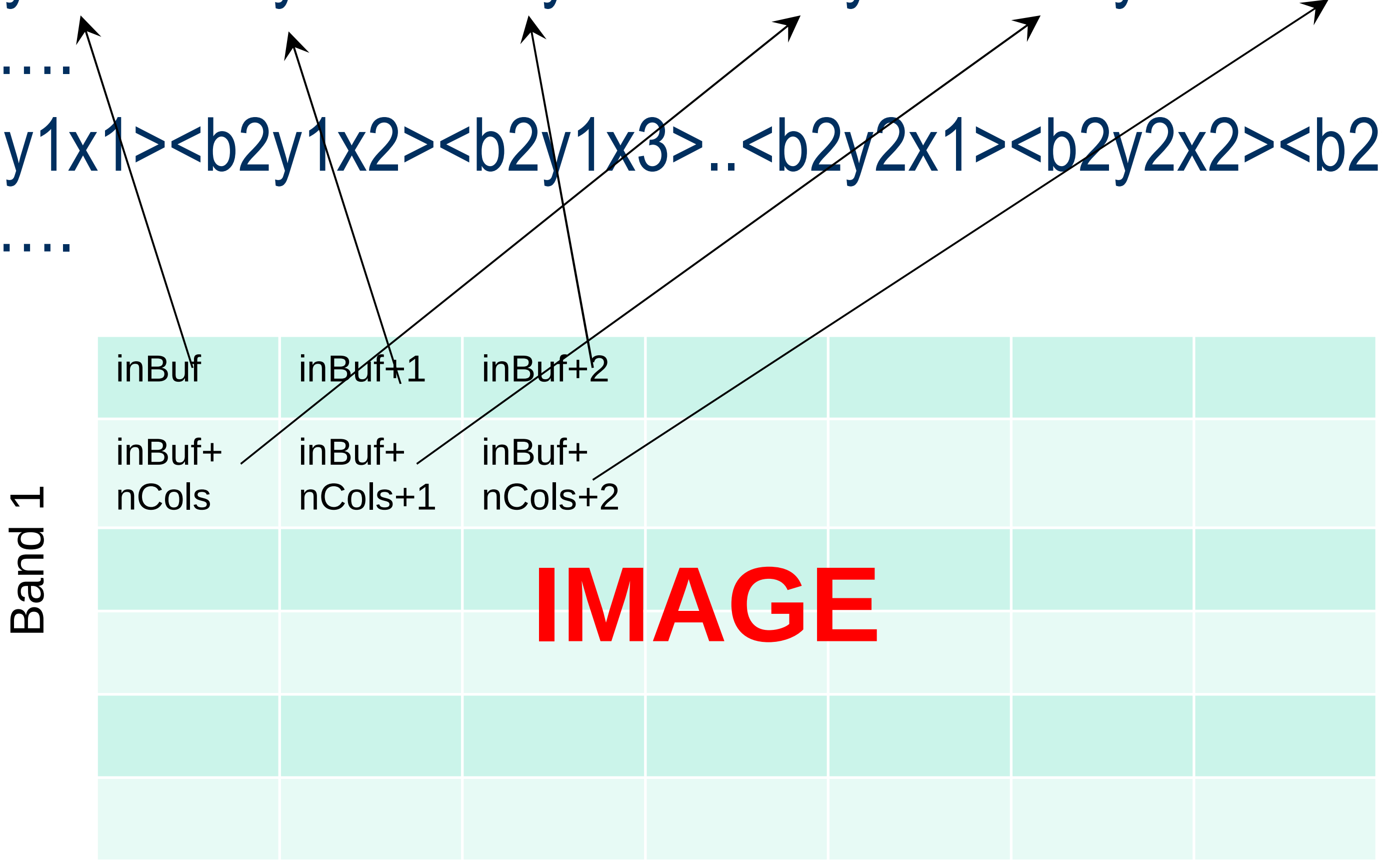
```
for(i=0;i<height;i++){
    for(j=0;j<width;j++){
                    for(k=0;k<channels;k++){
                              data[i*step+j*channels+k]=0;
                    }
    }
}
RGBRGBRGBRGBRGB…
```

Accessing ossimImageData data

- For images with bit depth == 8
- unsigned char* inBuf = static_cast<unsigned char*>(inputTile->getBuf(band));
- unsigned char pixVal = (unsigned char)(*inBuf);
- The ossim image data is organized by:
    - R(1,1)R(1,2)R(1,3)…G(1,1)G(1,2)G(1,3)...B(1,1)B(1,2)B(1,3)
- That is the fastest changing data is the data along the rows of the image, then the columns, then the bands

Accessing ossimImageData data

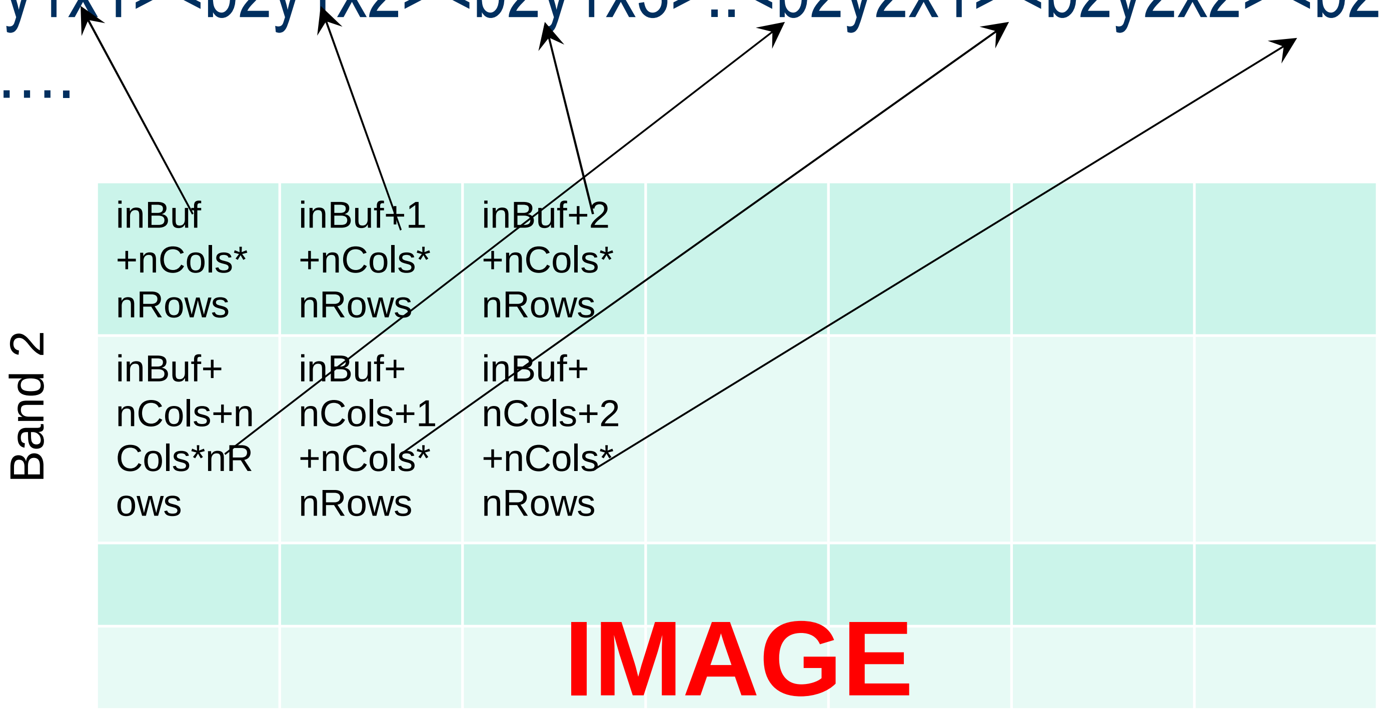
- <b1y1x1><b1y1x2><b1y1x3>..<b1y2x1><b1y2x2><b1y2x3>.......
- <b2y1x1><b2y1x2><b2y1x3>..<b2y2x1><b2y2x2><b2y2x3>.......



inBuf	inBuf+1	inBuf+2				
inBuf+ nCols	inBuf+ nCols+1	inBuf+ nCols+2				

Accessing ossimImageData data

- <b1y1x1><b1y1x2><b1y1x3>..<b1y2x1><b1y2x2><b1y2x3>.......
- <b2y1x1><b2y1x2><b2y1x3>..<b2y2x1><b2y2x2><b2y2x3>.......

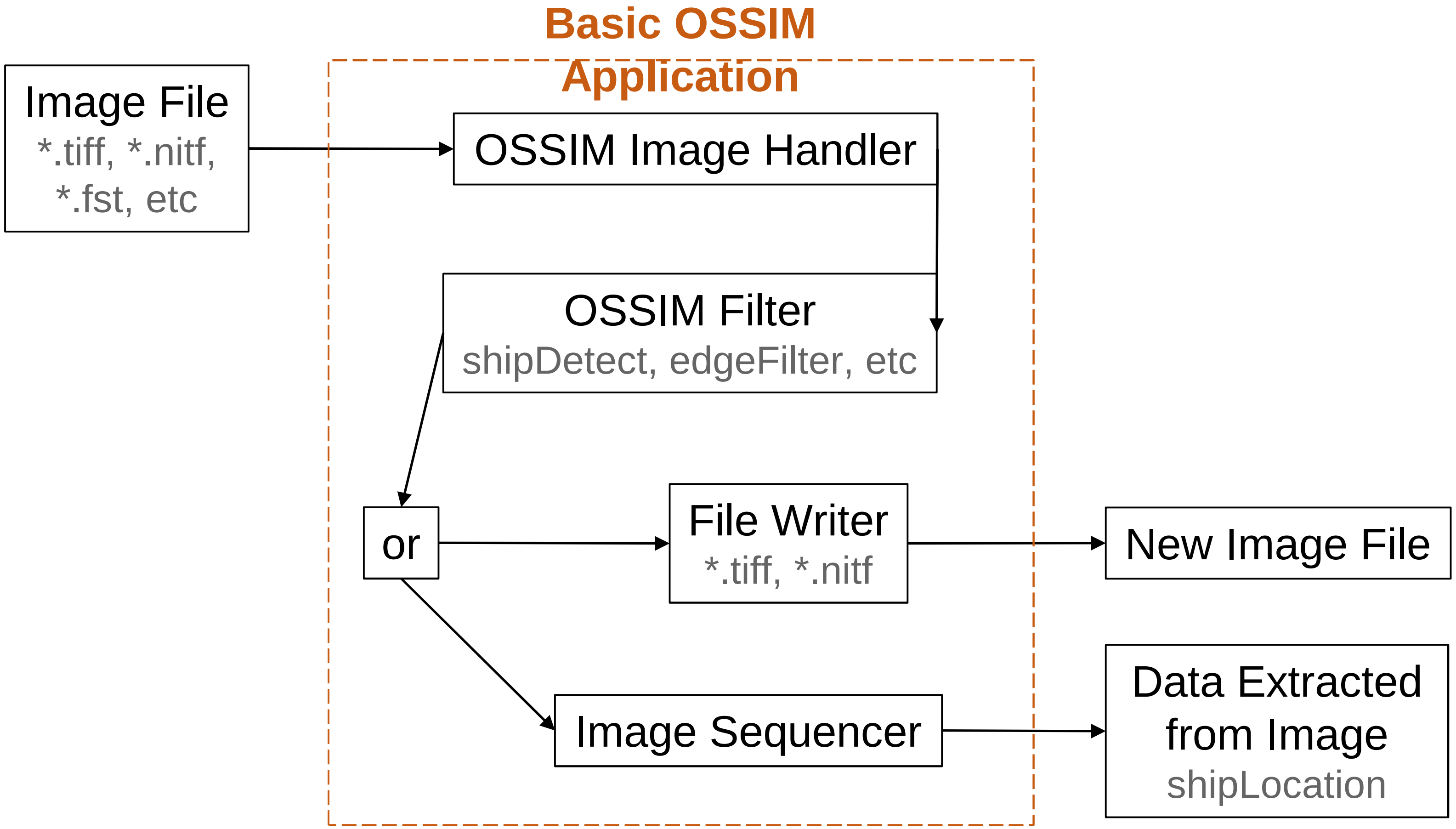


inBuf +nCols* nRows	inBuf+1 +nCols* nRows	inBuf+2 +nCols* nRows				
inBuf+ nCols+nCols*nRows	inBuf+ nCols+1 +nCols* nRows	inBuf+ nCols+2 +nCols* nRows				

Allocating an Image

- IplImage* image = cvCreateImage(size,depth,numChannels);
- size = cvSize(width, height)
- depth =
  - IPL_DEPTH_8U – 8 bit unsigned
  - IPL_DEPTH_16U – 16 bit unsigned
  - IPL_DEPTH_16S - 16 bit signed
  - IPL_DEPTH_32F – 32 bit floating point
- numChannels = number of channels in the input image

OSSIM Basics



OSSIM Basics

```
ossimInit::instance()->initialize();
```

```
ossimRefPtr<ossimImageHandler> ih = ossimImageHandlerRegistry::instance()->open(image_file);
```

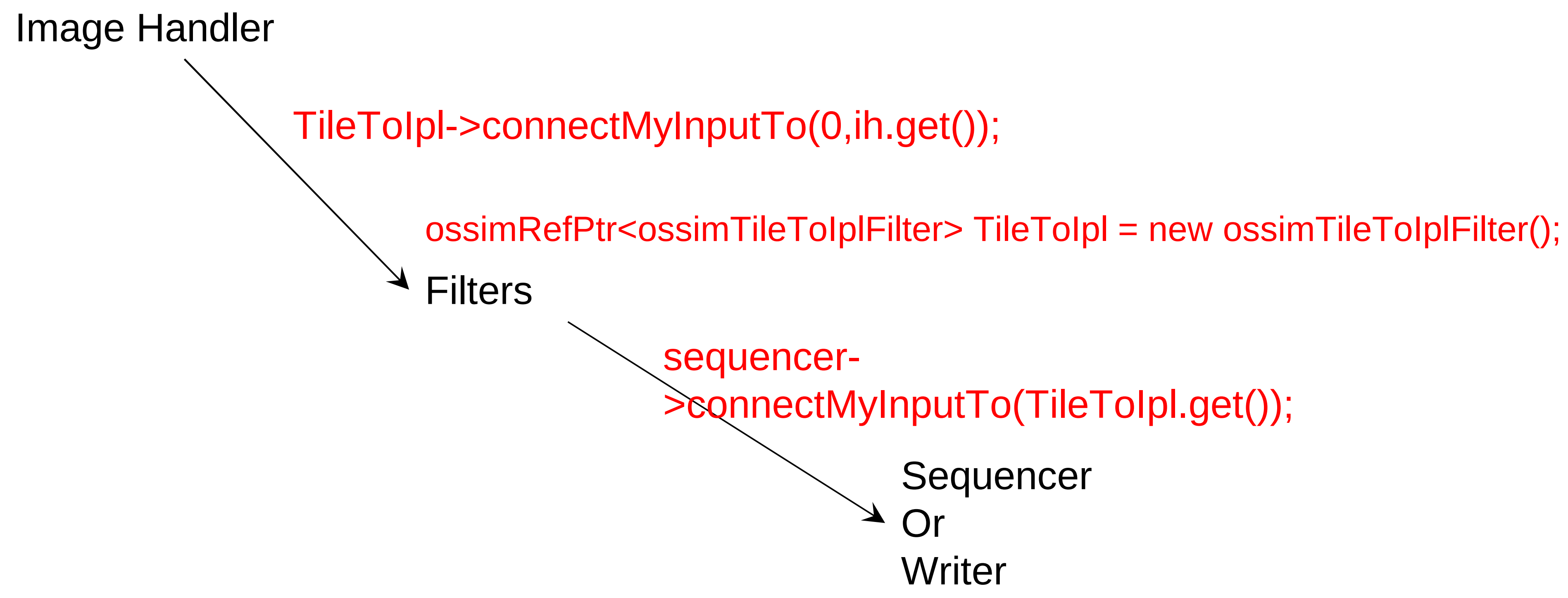


Image Handler

```
TileToIpl->connectMyInputTo(0,ih.get());
```

```
ossimRefPtr<ossimTileToIplFilter> TileToIpl = new ossimTileToIplFilter();
```

Filters

```
sequencer->connectMyInputTo(TileToIpl.get());
```

Sequencer
Or
Writer

```
ossimRefPtr<ossimImageSourceSequencer> sequencer = new ossimImageSourceSequencer();
```

RUN THE CHAIN

```
while( (dataObject=sequencer->getNextTile()).valid() );
```