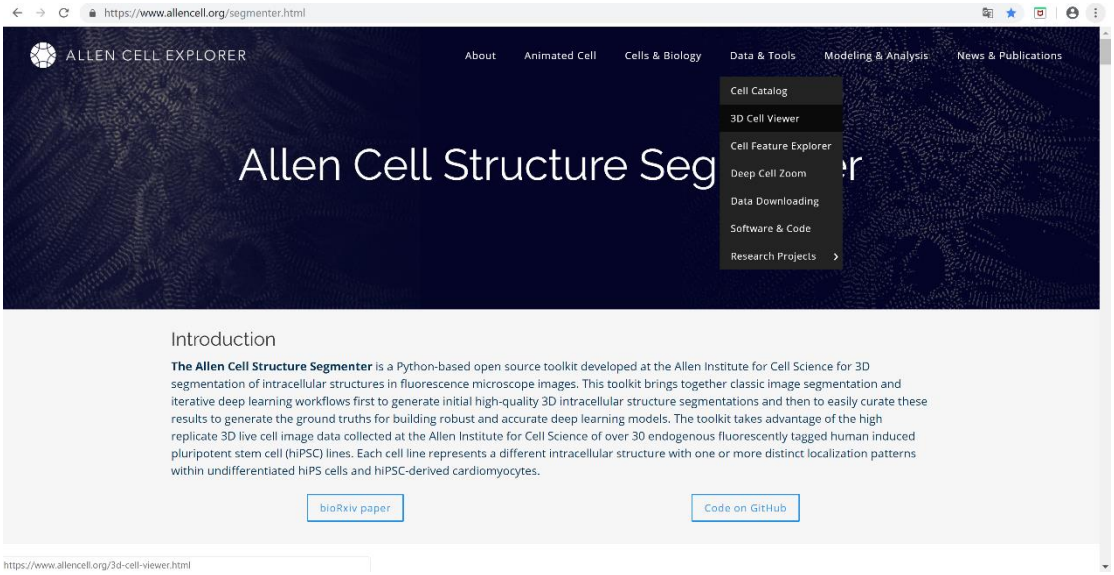
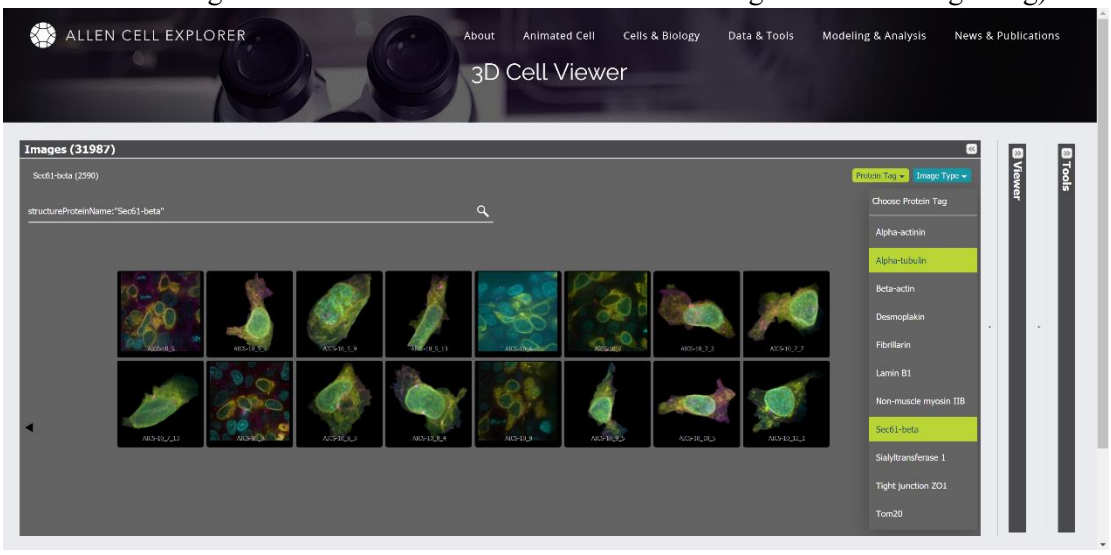
Point Cloud Dataset in Array Obtained from Allen Cell Explorer

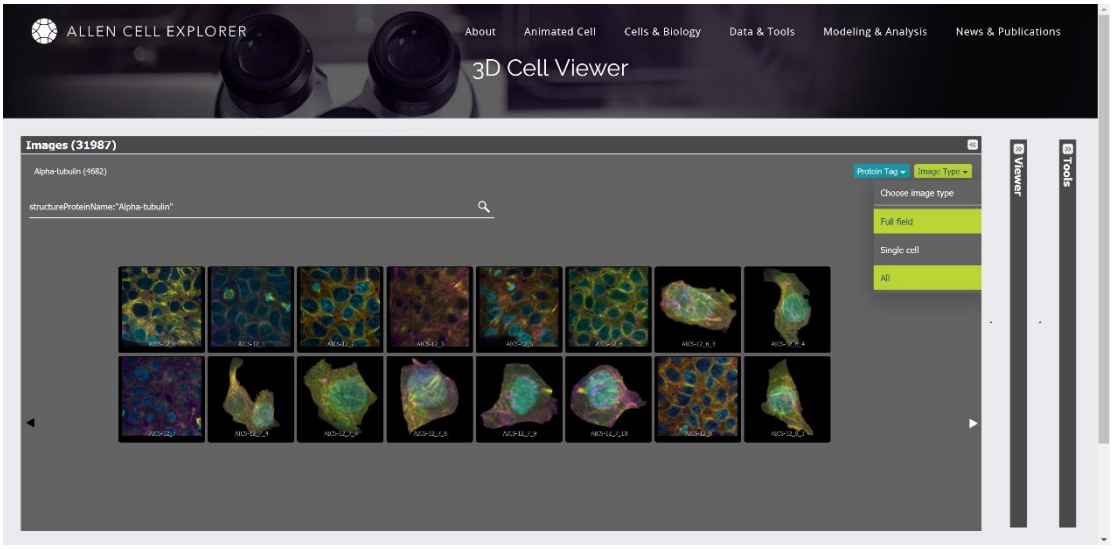
1. Point Cloud Cells Image

Basically the point cloud cell data used in our project is provide in Allen Cell Explorer (https://www.allencell.org/segmenter.html). We find the sample data through "3D Cell Viewer" under the tag "Data&Tools".

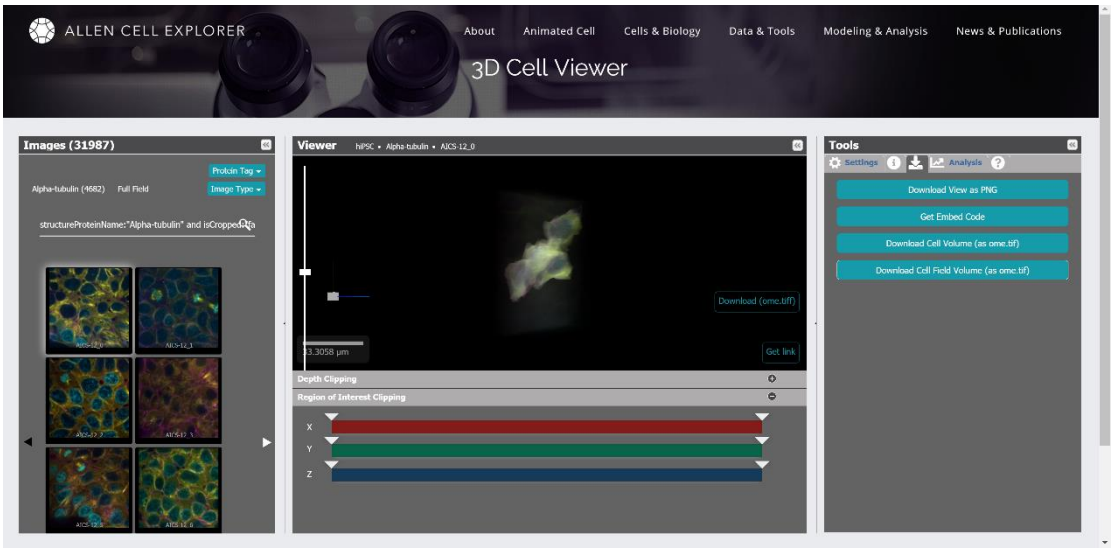


And to be more specific, we need cells image with clear membrane outline to identify every cell and calculate the accurate volume. Here we can choose "Alpha-tubulin" under the tag "Protein Tag" and then choose "Full field" under the tag "Image Type" (we may also use the "Single cell" here to test our volume calculation algorithm at the beginning).





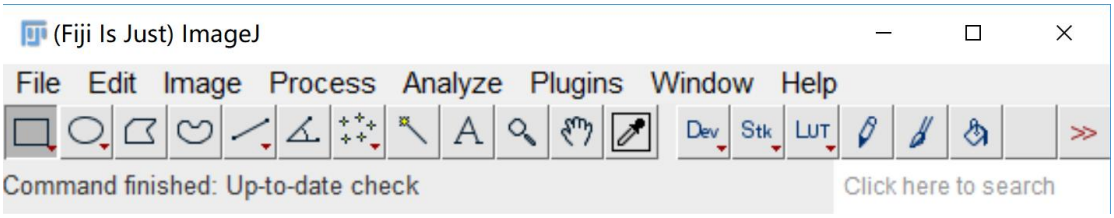
Then we can download the entire cell field point cloud dataset in tiff format. Simply click the “Download Cell Field Volume (as ome.tif)” in the “Tools” tag within a download icon.



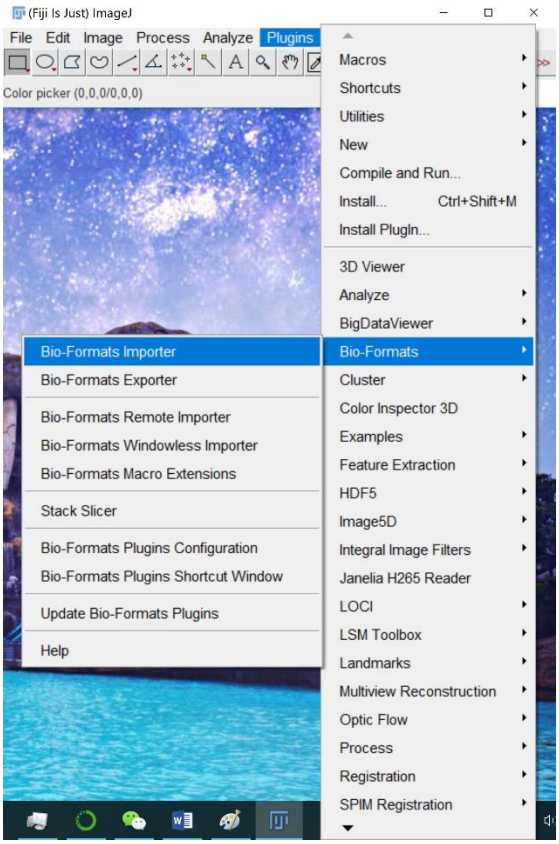
2. Duplicate Membrane Channel Dataset

We can use 3D Cell Viewer provided in Allen Cell Explorer to observe the point could image of cells, but we still need to use C code to read the data and calculate the volume for each cell. And we tried to duplicate the membrane channel directly from tiff format through python but failed. With the advise from Professor Tinggang Chew, we use a software ImageJ to adjust the provided point cloud image and duplicate the visible version of Membrane outline images (Only membrane can be seen on the black background in images).

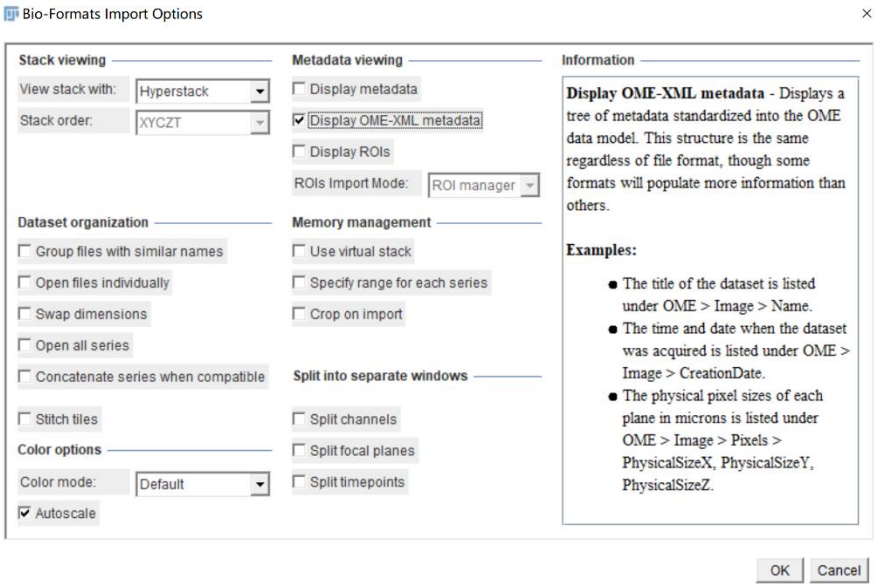
First download the Fiji version of ImageJ (which is different from download ImageJ directly). Open it and the appearance looks like following:



Then import the download file in tiff format.

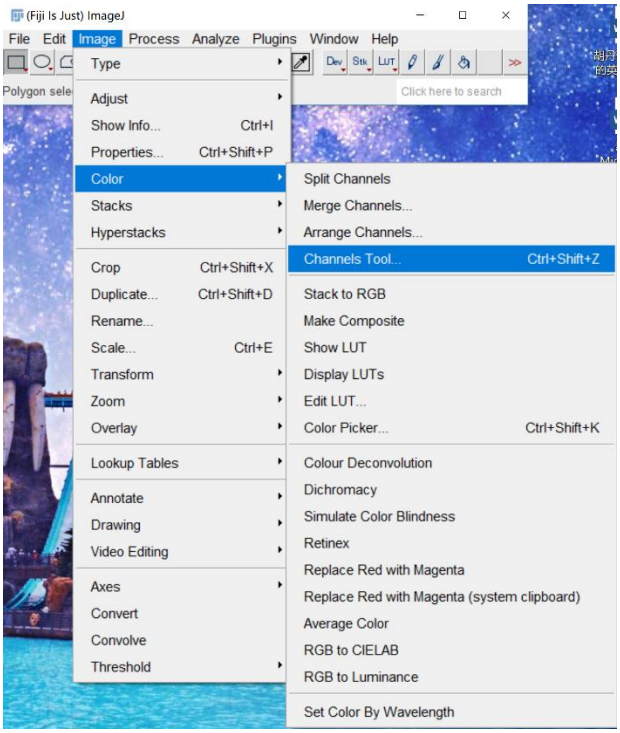


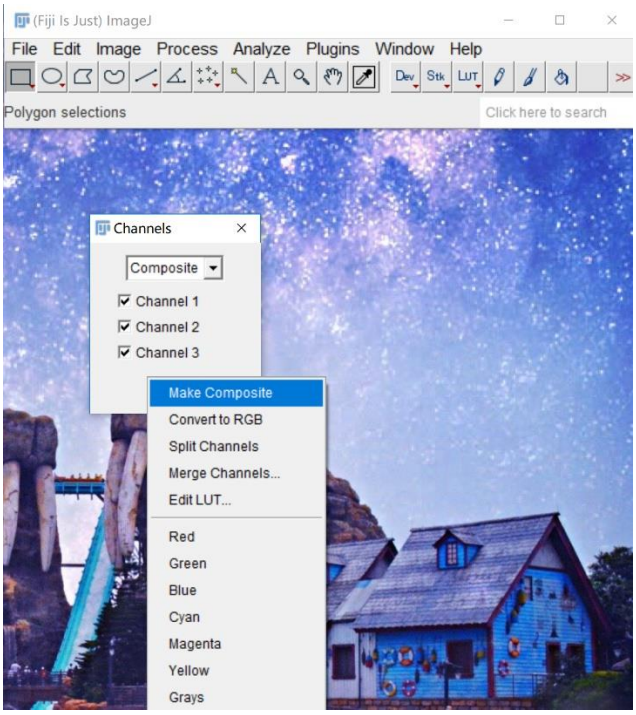
After importing it, a setting window will show up. To obtain the metadata we need to click the checkbox before “Display OME-XML metadata”. Then the information of tiff file will show up, which include the physical size of the picture or pixel with unit, the pixel size of images, the data type and so on, which is important for data read.



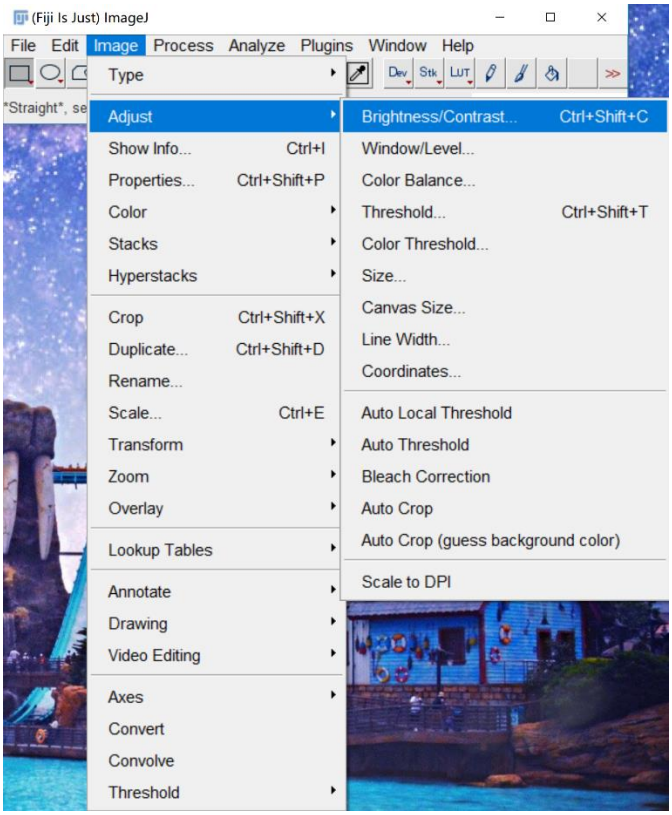
And we also obtain a window to view the point cloud image in X-Y plane, in which we can adjust the channel number and Z position to see different images. But currently there is no access to see the membrane outline images.

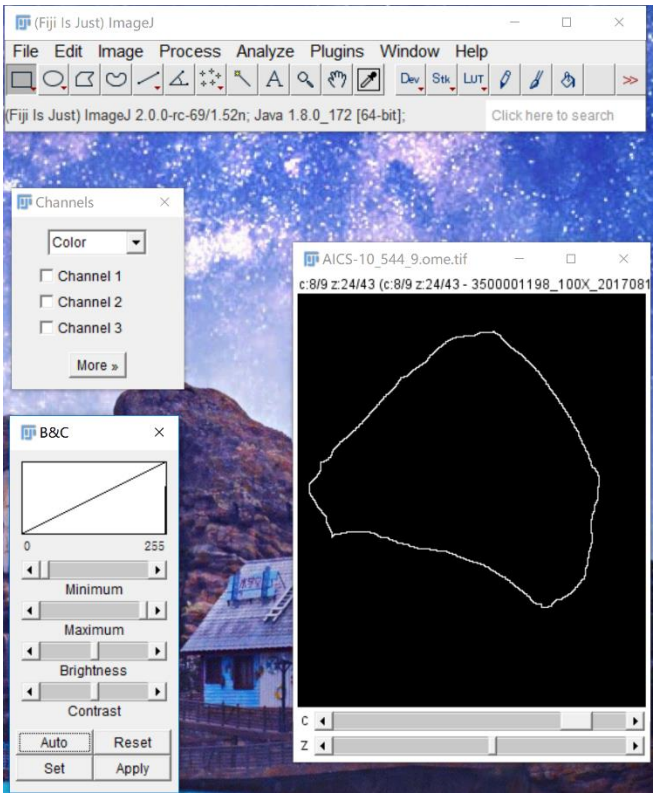
To adjust the view, there are two steps needed to be completed. First, we need to set the “Channels Tool” by click it under “Image”->”Color”. Then click “more” on the showing up window and choose “Make Composite”. Finish this step by clicking “OK”.



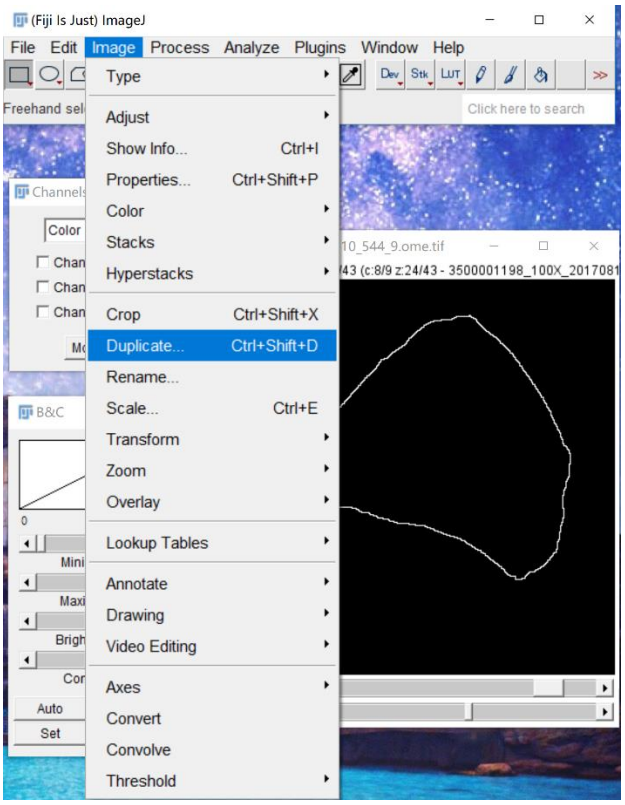


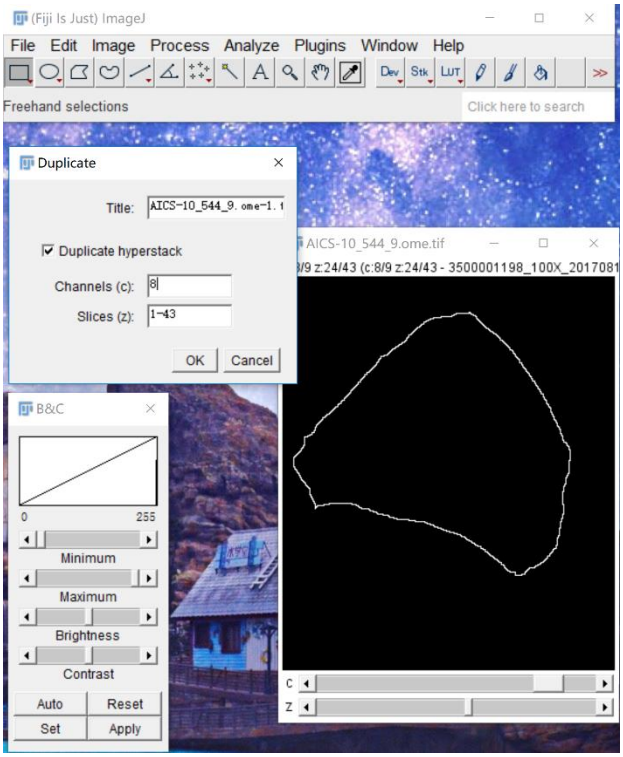
Second, we need to adjust the “Brightness/Contrast…” under “Image”-> “Adjust”. And then click “Auto” near the bottom of the showing up window. By finishing the above two steps, the layer images of membrane outline will appear clearly in channel 8. (It might be entirely dark for the beginning and ending few pictures.)





Finally, to store the membrane point cloud dataset, we can use “Duplicate” under “Image”, choosing only channel 8 for the showing up window and finish it by clicking “OK”. Then use “Save” under “File” to store the tiff format membrane point cloud dataset into the expected folder.





3. Store Membrane Channel into CSV and PNG for C Code Usage

After obtaining the membrane point cloud dataset, we need to transform it into accessible format by C code. There is available library for C code to read/write tiff file.

But here we choose to use python to read it which is more convenient. The sequential code looks as following:

In [2]:
```
import tifffile as tiff
import numpy as np
from PIL import Image

im = tiff.imread("AICS-10_544_9.ome-1.tif")

# tiff format information from allen cell data set
print ("dtype: ",im.dtype)
print ("image_size:",im.shape)
print ("image_x: ",im.shape[2])
print ("image_y: ",im.shape[1])
print ("image_z: ",im.shape[0])

# create csv file to store the 3-d array into 1-d
txt = np.arange(im.shape[0]*im.shape[1]*im.shape[2])
for i in range(im.shape[0]):
    for j in range(im.shape[1]):
        for k in range(im.shape[2]):
            txt[i*j*k] = im[i][j][k]
np.savetxt("im01.csv", txt, delimiter=",")

# create pngs for each layer of the cell
for i in range(im.shape[0]):
    image = Image.fromarray(im[i],'I;16B')
    image.convert('L').save("layer"+str(i)+".png")
```

```
dtype:  uint16
image_size: (43, 365, 307)
image_x:  307
image_y:  365
image_z:  43
```

In C, that is:


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 26,
   "metadata": {},
   "outputs": [],
   "source": [
    "import tifffile as tiff\n",
    "import numpy as np\n",
    "from PIL import Image\n",
    "import csv\n",
    "import cv2"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "metadata": {},
   "outputs": [
    {
```

```
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "dtype:    uint16\n",
       "image_size: (63, 437, 415)\n",
       "image_x:    437\n",
       "image_y:    415\n",
       "image_z:    63\n"
      ]
     }
    ],
    "source": [
     "im = tiff.imread(\"AICS-10_46_3.ome-c8.tif\")\n",
     "\n",
     "# tiff format information from allen cell data set\n",
     "print (\"dtype: \",im.dtype)\n",
     "print (\"image_size:\",im.shape)\n",
     "print (\"image_x: \",im.shape[1])\n",
     "print (\"image_y: \",im.shape[2])\n",
     "print (\"image_z: \",im.shape[0])\n",
     "\n",
     "# output csv file\n",
     "for k in range(im.shape[0]):\n",
     "        with open(\"csv/im_\"+ str(k)+ \".csv\",\"w\") as csvfile: \n",
     "                writer = csv.writer(csvfile)\n",
     "                txt = []\n",
     "                for i in range(im.shape[1]):\n",
     "                        txt.append(im[k][i])\n",
     "                writer.writerows(txt)"
    ]
   }
  ],
  "metadata": {
   "kernelspec": {
    "display_name": "Python 3",
    "language": "python",
    "name": "python3"
   },
   "language_info": {
    "codemirror_mode": {
     "name": "ipython",
     "version": 3
    },
    "file_extension": ".py",
```

```
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.7.1"
   }
  },
  "nbformat": 4,
  "nbformat_minor": 2
}
```


Here we read the 3-d array from tiff file. (It was 4-d with channels but we have choose the channel with only membrane outline). And we can output the 3-d array's shape and data type as printed above. To make it easier for C code, we can store 3-d array into 1-d array with CSV format and also store each X-Y layer into one image with PNG format. So it can be easier for C code to read the data with the above formats.