Instruction LSM, LIF, ND2, v3draw Formats

Last update: 2023-06-22

During the microscopy practicals you made 3D images.
Depending on the microscope used you will have either lif or lsm files.
These files cannot be used in diplib or Vaa3D as they are.
You will have to "unpack" them using readlif for lif files or tifffile for lsm files.

Both can be installed using pip.

readlif documentation:
https://readlif.readthedocs.io/en/latest/

tifffile documentation:
https://github.com/cgohlke/tifffile
(Scroll down for examples)

For ND2 files we will use pims_nd2 (or pims-nd2) which uses pims to index the image after reading.

Also available using pip.

pims_nd2 documentation:
https://github.com/soft-matter/pims_nd2

pims documentation:
http://soft-matter.github.io/pims/v0.6.1/index.html

There is no package for v3draw files.
Use the functions in the section below to read and write to and from a numpy array.

Examples:

The following example scripts will take the individual layers of the 3D images and write them as separate tiff files.

For LIF files:

```
from readlif.reader import LifFile
f=LifFile("your_file.lif")

#dimension check, change loop below based on output
for i in f.get_iter_image():
     print(i.dims)

imgl=[] #image list
#LIF files can contain multiple images
for image in f.get_iter_image():
     i=[] #frame list
     #frames in z axis
     for frame in image.get_iter_z():
          i.append(frame)
     imgl.append(i)

#Now you can save individual images from your list of images
as tif:
imgl[0][0].save("multilayer1.tiff", format="TIFF",
append_images=imgl[0][1:],save_all=True)
#This creates a single multilayer tiff file.
imgl[1][0].save("multilayer2.tiff", format="TIFF",
append_images=imgl[1][1:],save_all=True)

#If you want all frames separated:
fnames=["A.tiff","B.tiff","C.tiff","D.tiff","E.tiff","F.tiff",
"G.tiff","H.tiff","I.tiff","J.tiff","K.tiff","L.tiff"]
#fnames is a list of filenames to use when writing the images.
#Make sure this list is >= than the amount of layers in the
image.
for i ,j in zip(imgl[0], fnames):
    i.save(j,"TIFF")
```

For LSM files:

```
import tifffile
f=tifffile.imread("your_file.lsm")
tifffile.imwrite("multilayer.tiff",f)
#This creates a single multilayer tiff file.

#If you want all frames separated:
#tifffile creates a numpy array with all the images.
#import numpy as np
#You can inspect the numpy array with np.shape(f) to get the
amount of images in the object.
fnames=["A.tiff","B.tiff","C.tiff","D.tiff","E.tiff","F.tiff",
"G.tiff","H.tiff","I.tiff","J.tiff","K.tiff","L.tiff"]
#fnames is a list of filenames to use when writing the images.
#Make sure this list is >= than the amount of layers in the
image.
for i,j in zip(f[0],fnames):
    tifffile.imwrite(j,i)
```

For ND2 files:

```
from pims import ND2_Reader
f=ND2_Reader('your_file.nd2')

#Use print(f) to figure out your image dimensions and set your
bundle_axes and iter_axes accordingly.
f.bundle_axes="cyx"
f.iter_axes="z"
#This way it will iterate over the stack (z) of images (cyx).
```

For v3draw files:

```
#TODO: cleanup code

def Readv3draw(fname):
     import numpy as np
     import struct
     d = open(fname, "rb")
     img_str = d.read()
     d.close()
     header = img_str[0:43]#header is 43 bytes
     t = header[0:24]# file format info
     endian = header[24:25] # L for Little or B for Big
     if endian == b'L':
          order = '<'
     elif endian == b'B':
          order = '>'
     else:
          print('endian byte error')
     bpp = header[25:27] # bytes per pixel
     form = order + 'H'
     bppi = struct.unpack(form, bpp)[0] # 2 for uint16 1 for
uint8
     dim = header[27:] # dimensions 4 ints XYZC
     form = order + 'IIII'
     dimi = tuple(reversed(struct.unpack(form, dim)))
#dimensions for data are actually CZYX
     data = img_str[43:]
     if bppi == 2:
          img_arr = np.fromstring(data, np.uint16)
     else:
          img_arr = np.fromstring(data, np.uint8)
     img_arr = np.reshape(img_arr, dimi)
     return img_arr
```

```
def Writev3draw(data, fname):
     import numpy as np
     import struct
     if not isinstance(data, np.ndarray):
          raise TypeError
     import struct
     import sys
     t = b'raw_image_stack_by_hpeng'
     if sys.byteorder=='little':
          endian = b'L'
          order = '<'
     else:
          endian = b'B'
          order = '>'
     dimi = data.shape
     if data.dtype==np.uint8:
          bppi=1
     if data.dtype==np.uint16:
          bppi=2
     form = order + 'H'
     bpp = struct.pack(form,bppi)
     form = order
     for i in range(len(dimi)):
          form = form + 'I'
     dim = struct.pack(form,*tuple(reversed(dimi)))
     ds = data.data
     ws = t + endian + bpp + dim + ds
     wt = open(fname, 'wb')
     wt.write(ws)
     wt.close()
```

Extra notes:

If tifffile reads the image into a numpy array, can't we put that array into diplib using dip.Image()?
No, diplib wants the RGB values per pixel, but tifffile creates separate RGB layers per image. (You can maybe reshape the array if you are familiar with numpy though...)

Will you release the v3draw functions as a python package?
Probably not, unless someone else wants to maintain it.