

temDM

temDM MSA scripting examples

Pavel Potapov

January 18, 2022

You might design your own automatic treatment flow if you know the basics of DigitalMicrograph scripting. The package **temDM MSA** extends the library of DigitalMicrograph scripting language by adding the specific MSA functions. The file **temDM MSA scripting.chm** included in the distribution package consists of description of the available commands for MSA.

There are a couple of scripting examples in the distribution package. Note that the **temDM MSA** commands consist of “MSA_112” in their name. The rest commands are the part of the standard DigitalMicrograph scripting interface.

1 SIMULATE EFTEM SERIES

This following example simulates an “EFTEM series” from a STEM datacube.

```
//This script simulates an "EFTEM
series" from a STEM datacube

//Energy limits for EFTEM series
number E_Start_eV = 1230
number E_End_eV = 1380
number E_Step_eV = 50
number E_Window_eV = 40

//get DataCube
image DataCube := getFrontImage()

//read its dimensions
number width,height,depth
DataCube.get3Dsize(width,height,depth)

//read energy calibration
```

```
number dispersion = DataCube.
    ImageGetDimensionScale(2)
number origin = DataCube.
    ImageGetDimensionOrigin(2)
result("\ndispersion_"+dispersion +"_
origin_"+Origin)

//iterate through energy and integrate
within an energy window
for (number E_eV = E_Start_eV; E_eV <=
    E_End_eV;E_eV += E_Step_eV)
{
    //convert from eV to channels
    number Start = MSA112_eVtoCh(
        E_eV-E_Window_eV/2,origin,
        dispersion)
    number End = MSA112_eVtoCh(E_eV
        +E_Window_eV/2,origin,
        dispersion)
    result("\n"+E_eV)

    //select a section from the
    DataCube
    image CubeSection = DataCube
        [0,0,Start,width,height,End]

    //render along energy axia
    image Eftem =
        MSA112_Render3DtoFront(
            CubeSection)

    //set name and display
    Eftem.setName("EFTEM_"+E_eV+"eV
        ")
    Eftem.showimage()
}
```

2 EIGENVALUES VS NUMBER OF PIXELS

This script investigates the dependence of the eigenvalues on the number of pixels in the dataset.

```
//This function generates a random mask
image randomCluster(number N, number
    fract)
{
    image Cluster:= realImage("
        cluster",4,1,N)
    //mask image of total N pixels

    number range = 1/(1-fract)

    Cluster = random()*range
    Cluster = tert(cluster>1,1,0)
    //fract*N of pixels are 1, (1-
        fract)*N pixels are 0
    //choice is random

    return cluster
}

//parameters for convergence
number deviaSpec = 0.02
number maxItt = 50

//get an MSA image where the
    dataMatrix is kept as an attachment
image MSAimage := getFrontImage()
image DataMatrix = MSA112_GetMSAMatrix(
    MSAimage)

//get the number of pixels

number m = DataMatrix.
    ImageGetDimensionSize(1)

//define range and steps for
    fractioning the dataMatrix
number fractIni = 0.1
number fractFin = 1
number fractStep = 0.1

//fractioning
for(number fract = fractIni;fract <=
    fractFin;fract = fract + fractStep)
{
    //make a random mask to select
        a certain fraction from
        DataMatrix
    image Cluster:= integerImage("
        cluster",2,0,1,m)
    Cluster = randomCluster(m,fract
        )
    number actualFract = mean(
        Cluster)
```

```
//actual fraction can slightly
    differ from the nominal one
result("\nfraction_" +
    actualFract)

//apply the mask to DataMatrix
image DataMatrixSmaller =
    MSA112_TakeCluster(
        dataMatrix,1,cluster,1)
    //1: do
        centering;
    0: do not
    //mask with
        index 1 is
        chosen

//declare values and images
number EigenValue //
    eigenvalue of a component
image Pvec,Tvec //loading and
    score of a component
number Nitt //
    number of itterations
    required for convergence

//extract 1st PCA component
    Nitt=MSA112_FindNIPALS(
        DataMatrixSmaller,Pvec,
        Tvec,EigenValue,0,
        deviaSpec,maxItt)

EigenValue /= (m*actualFract)
//now the eigenvalue is the
    variance within a component
result("_eigenValue1_" +
    EigenValue)

//reduce DataMatrix by the 1
    stPCA component
DataMatrixSmaller =
    MSA112_ReduceDataMatrix(
        DataMatrixSmaller,Pvec,Tvec)

//extract 2nd PCA component
    Nitt = MSA112_FindNIPALS(
        DataMatrixSmaller,Pvec,Tvec,
        EigenValue,0,deviaSpec,
        maxItt)

EigenValue /= (m*actualFract)

//now the eigenvalue is the
    variance within a component
result("_eigenValue2_" +
    EigenValue)
}
```

3 BUILD ELEMENTAL MAPS FROM ENDMEMBERS

This script offers an alternative to the standard quantification of EELS spectrum-images. You can extract pixel-by-pixel elemental maps from a given spectrum-image using for instance the commercial Gatan quantification software. Alternatively you might first try to find few endmember spectra in your dataset and quantify them very carefully by using any available quantification program. There are only few spectra, thus you can treat them very carefully with paying a lot of attention to all details. Then, with the knowledge of the endmember distributions, you can retrieve the elemental maps over the entire your spectrum-image.

The script below is written for the simulated EELS spectrum-image available in <http://temdm.com/web/msa/>. The contents of all three endmembers are listed at the head of the script. You can easily recast it for your own dataset if you know the compositions of your endmembers.

```
number Alin1 =1 ,Mgin1 =0 ,Oin1 =0
number Alin2 =0 ,Mgin2 =0.5 ,Oin2 =0.5
number Alin3 =0.5 ,Mgin3 =0 ,Oin3 =0.5

number clust =0 //this should be 0 (no clustering) unless you wish to access a specific cluster

image MSAimage :=getFrontImage()
TagGroup MatrixTags =
    MSA112_GetMatrixTags(MSAimage,0)

string MSALabel = MSA112_Label()
string clusterLabel =
    MSA112_GetClusterLabel(clust)

number width,height
MatrixTags.MSA112_GetSIWidthHeight(
    width,height)
number scaleX,originX,scaleY,originY
string unitsX,unitsY
MatrixTags.MSA112_GetCalibration(scaleX
    ,originX,unitsX,"X")
MatrixTags.MSA112_GetCalibration(scaleY
    ,originY,unitsY,"Y")

image Abundancies :=MatrixTags.
    MSA112_getImagefromTags(clusterLabel
    +":EndMembering:Abundancies")
number EndmembersNumber,Points
Abundancies.getSize(EndmembersNumber,
    Points)

image MembersMapCube := realimage("
    EndMembers_abundancies",4,width,
```

```
height,endMembersNumber)
for (number j=0;j<height;j++)
{
    MembersMapCube[irow,j,
        icol] = Abundancies[
        j*width,0,(j+1)*
        width,
        endMembersNumber]
}

MembersMapCube.
    ImageSetDimensionCalibration(0,
    originX,scaleX,unitsX,1)
MembersMapCube.
    ImageSetDimensionCalibration(1,
    originY,scaleY,unitsY,1)

//MembersMapCube.showimage()

image ElementalMapCube :=MembersMapCube
    .imageClone()
ElementalMapCube =0

image recastOneMap(image MembersMapCube
    ,number ELin1,number ELin2,number
    ELin3)
{
    number width =MembersMapCube.
        ImageGetDimensionSize(0)
    number height =MembersMapCube.
        ImageGetDimensionSize(1)

    image ElementMap :=realimage("
        ,4,width,height)

    ElementMap +=MembersMapCube
        [0,0,0,width,height,1]*ELin1
    ElementMap +=MembersMapCube
        [0,0,1,width,height,2]*ELin2
    ElementMap +=MembersMapCube
        [0,0,2,width,height,3]*ELin3

    return ElementMap
}

ElementalMapCube[0,0,0,width,height,1]
    =recastOneMap(MembersMapCube,Alin1,
    Alin2,Alin3)
ElementalMapCube[0,0,1,width,height,2]
    =recastOneMap(MembersMapCube,Mgin1,
    Mgin2,Mgin3)
ElementalMapCube[0,0,2,width,height,3]
    =recastOneMap(MembersMapCube,Oin1,
    Oin2,Oin3)

ElementalMapCube.showimage()
```

4 SHOW ALL FOUND SPIKES

*Attention: this script functions only with the installed advanced version of **temDM MSA**.*

This script extracts all detected spikes from an **XXX MSA** image and plot it as a 2D map. The values assigned to the map reflect the strength of spikes. You can redefine it yourself assigning, for instance, the energy channel where a spike is observed.

```
//take the MSA image
image MSAimage := getFrontImage()

//get its size, this corresponds to the
    spatial dimensions of a spectrum
    image
number width,height
MSAimage.getSize(width,height)

//extract image consisting of all
    information about the detected
    spikes
number NSpikes //number of spikes
image outliers =
    MSA112_GetSpikesCoordinates(MSAimage
    ,0,NSpikes)
//first column : spike's coordinate in
    plain pixel's No
//second column : spike's energy
    channel
//third column : spike' strength
//forth column : index of component
    where a spike was found
result("\nNumber_of_spikes_" + NSpikes)

//break it on subimages
image Pixels = outliers[0,0,NSpikes,1]
    //spike's pixelNo
image Values = outliers[0,2,NSpikes,3]
    //spike's strength

//build the map of spikes
image Coord := realimage("map_of_spikes
    ",4,width,height)
Coord[mod(Pixels,width), trunc(Pixels/
    width)] = Values/256
//the pixelNo is converted into X,Y
    coordinates of a map, that are
    assigned to spike's strength

Coord.showimage()
```

This example performs the truncated SVD of a given data set. The weighted data matrix should be preliminary prepared in an **XXX MSA** input file.

```
//get MSA image, get its size
image MSAimage := getFrontImage()
number width,height
MSAimage.getSize(width,height)

//get datamatrix
image matrix = MSAimage.
    MSA112_GetMSAMatrix()

//center datamatrix
matrix = MSA112_ReduceAverageSpectrum(
    matrix)

//perform truncated SVD
image VL,VR,SVs
MSA112_truncSVD(matrix,VL,VR,SVs
    ,50,0.02,50,5)

//convert singular values into
    eigenvalues
image EVs = SVs**2/width/hight
EVs = log(EVs)
EVs.showimage()
```

5 MAKE SVD DECOMPOSITION

*Attention: this script functions only with the installed advanced version of **temDM MSA**.*