OAK RIDGE National Laboratory

Image Analysis and Segmentation

Rama Vasudevan
Center for Nanophase Materials Sciences,

AI for atoms: How to machine learn STEM
December 7th, 2020

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Some notes on workshop structure

- We will be utilizing GoogleColab for all of our work
- You can find our repository at https://github.com/pycroscopy/AISTEM_WORKSHOP_2020
- If there is any issue with Colab, you can also download the repository and run it locally. Run individual code cells by pressing the play button, or shift+enter
- Note that the order of cell execution matters
- Some packages may require restarts of the kernel.

Basic Image processing in Python

- There are generic image processing methods that are not specific to STEM images that deal with aspects such as object detection, edge finding, and image cleaning, registration, and so forth
- Two major packages that offer useful tools for generic image processing are OpenCV and scikit-image
- Here we will begin with a quick notebook on scikit-image

Basic Image processing in scikit-image

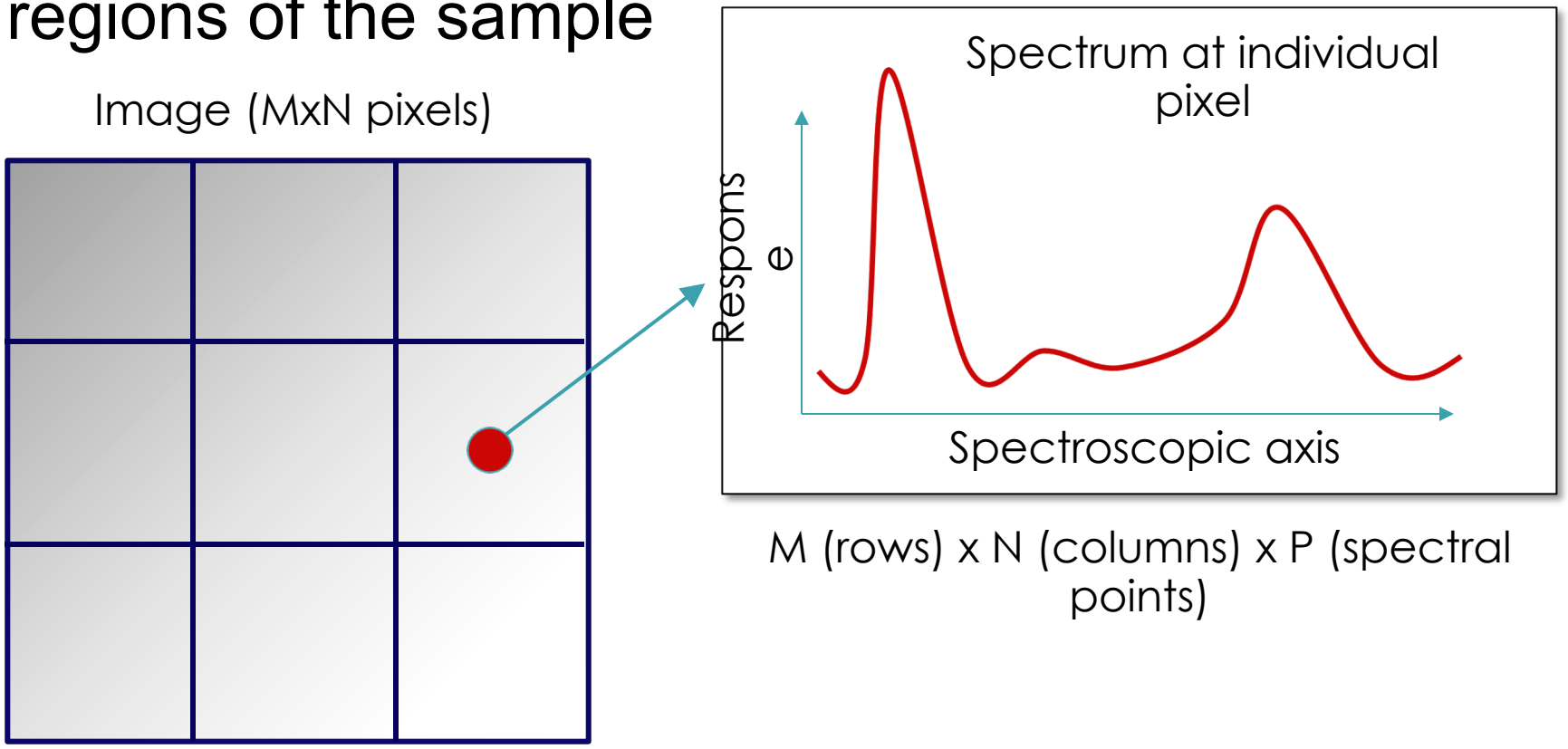
Open

Day00/Intro_to_Image_Analysis/01_Image_Processing.ipynb

in Google colab to continue

Spectral Data

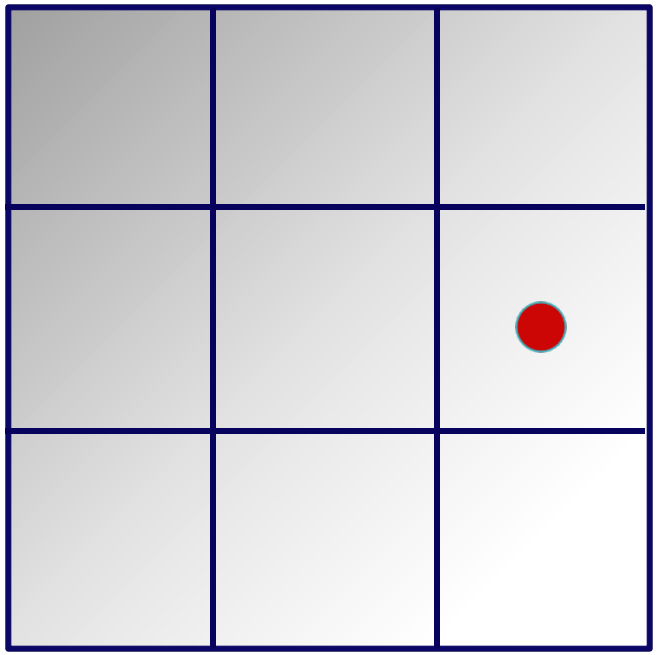
- Many imaging modalities have spectroscopy components
- By capturing spectra across an image, it allows us to correlate features of the spectra with specific regions of the sample

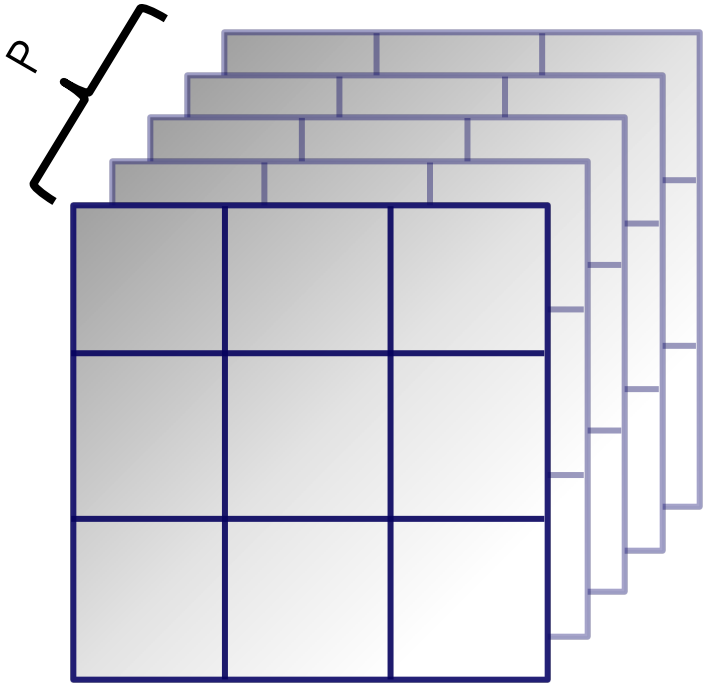


Spectral Data

- Many imaging modalities have spectroscopy components
- By capturing spectra across an image, it allows us to correlate features of the spectra with specific regions of the sample

Image (MxN pixels)





(Repeat MxN matrix P times to make 3D cube)

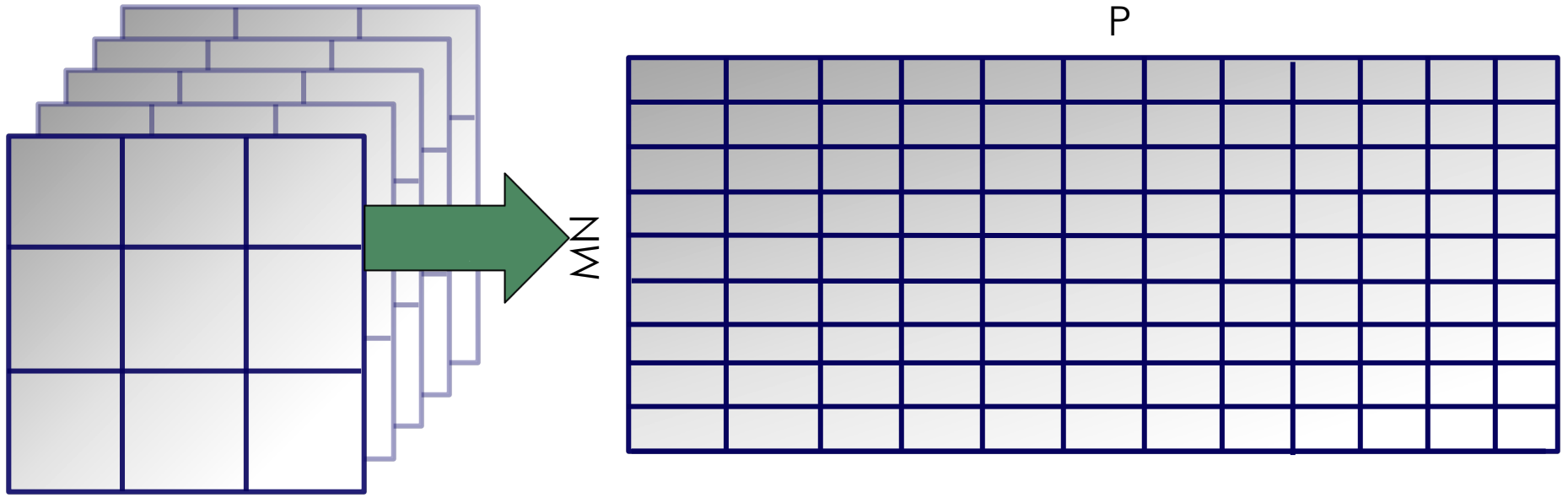
Spectral Data

- Examples include EELS, micro Raman imaging, force-distance curve maps in AFM, Scanning tunneling spectroscopy, etc.
- Usually, the 3D matrix is converted to a 2D matrix

M (rows) x N (columns) x P (spectral points)

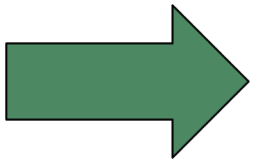
= (MN) x P

Pixels x spectral points



Matrix Factorization

- We now have a 2D matrix (call it **X**) of size (MN,P)
- If **X** is 'small' (e.g., (10,10)) we can comprehend it easily. But if the size of X is (2500, 32), then this becomes more difficult
- The number of spectral points **P** is often termed the "number of dimensions".
- To see trends in our data, we want to visualize them in some manner. But the 3D data cube is big, so what can we do?



Matrix Factorization

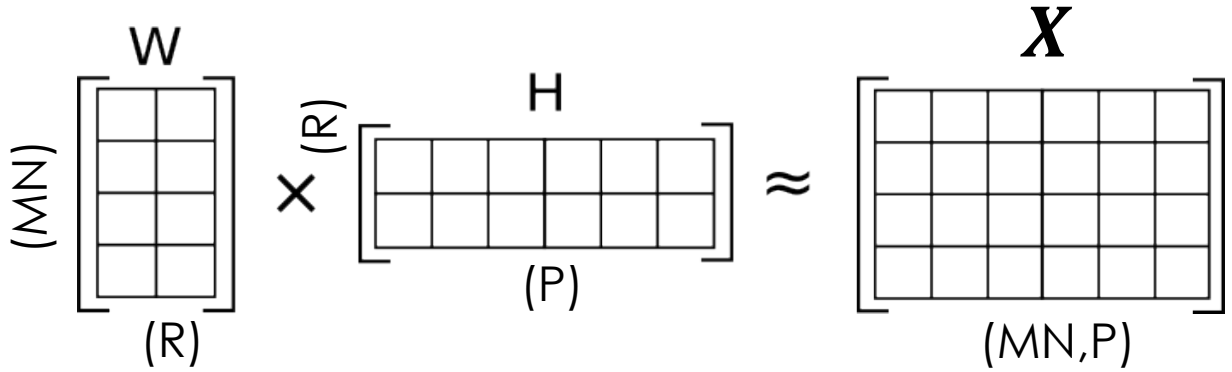
Dimensionality Reduction

- The reason it is difficult to visualize is because there are too many spectral points *P* per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- What if we could write our matrix **X** (size MN,P) as

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$

Where W is size (MN,R)
H is size (R,P)

R <P



Dimensionality Reduction

- The reason it is difficult to visualize is because there are too many spectral points *P* per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- Alternative viewpoint: Represent the spectrum at pixel z by a linear expansion

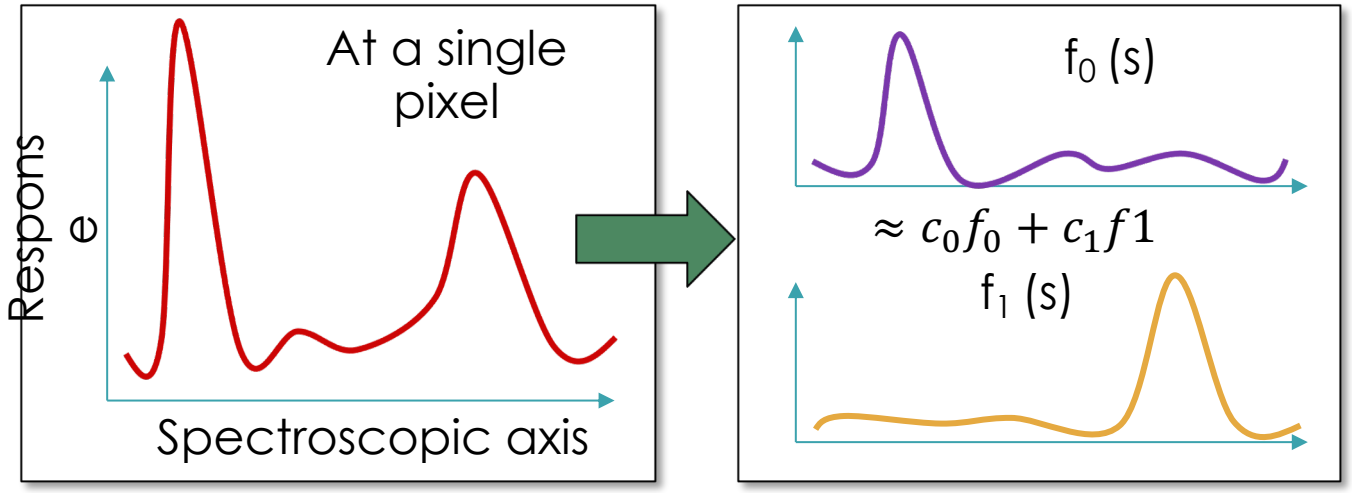
$$f(z_i, s) = c_0(z_i)f_0(s) + c_1(z_i)f_1(s) + \ \ldots + c_n(z)f_n(s)$$

$$f(z, s) = \sum_{i=1}^{MN} c_i(z_i)f_i(s)$$

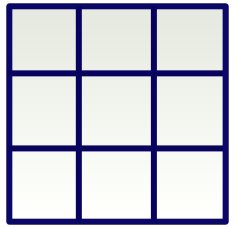
The task boils down to determining the functions $f_n(s)$, and determining the coefficients c_n

Dimensionality Reduction

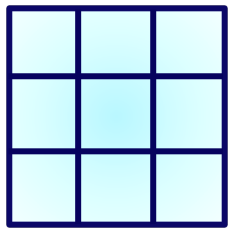
- The reason it is difficult to visualize is because there are too many spectral points P per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- Alternative viewpoint: Represent the spectrum at pixel z by a linear expansion



$c_0(x, y)$



$c_1(x, y)$



(1) Singular Value Decomposition

$$\boldsymbol{X = USV^T}$$

Note: In the full decomposition, r = n. Practically, r<<n

X is of size (m,n)
U is of size (m,r)
S is of size (r,r)
V is of size (n,r)

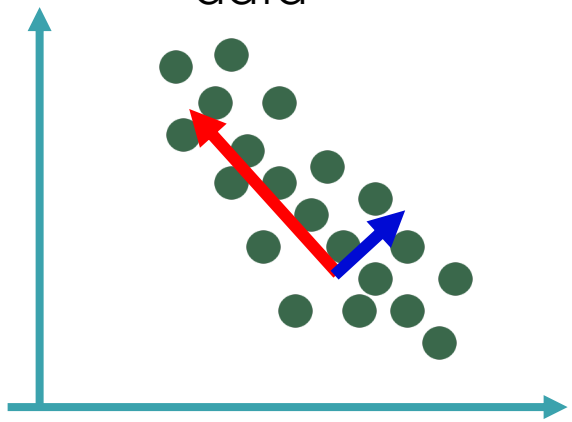
The columns of U,V form the basis
S scales them, in descending order

By the SVD theorem, it is always possible to decompose real matrix **X** into USV^T where

U,S,V: unique
U,V: column orthonormal
- $U^TU = I$, $V^TV = I$
- Columns are orthogonal unit vectors

S: diagonal
- Positive, sorted in descending order

Finding the main 'directions' in the data



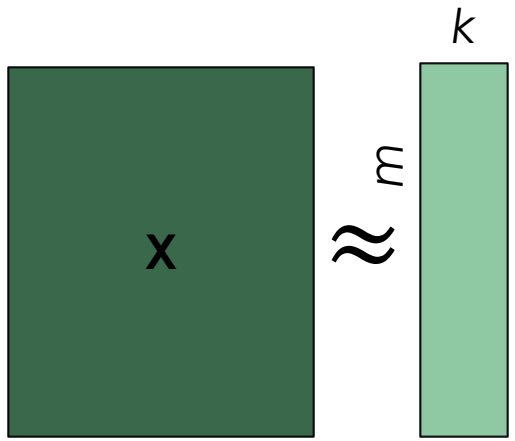
Non-negative Matrix Factorization

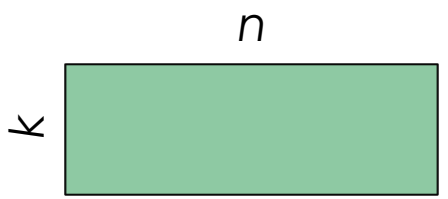
- SVD is an extremely powerful tool to visualize data and observe trends, but it has weaknesses. Mainly, it often produces highly unphysical behavior, for instance negative intensity values in spectra where only positive values are physically realizable
- Therefore, we can turn to other methods, such as non-negative matrix factorization approaches

Matrix Form

$$\boldsymbol{X} \approx \boldsymbol{WH}$$

Here, we enforce that X, W and H are all non-negative matrices





k is called 'low-rank'. The positivity constraint often helps to make the factor matrices more interpretable.

Spectral Unmixing: N-FINDR

- Spectra for a given pixel is assumed to be a linear combination of the end-member spectra (+ Gaussian noise). The mixing proportions sum to 1

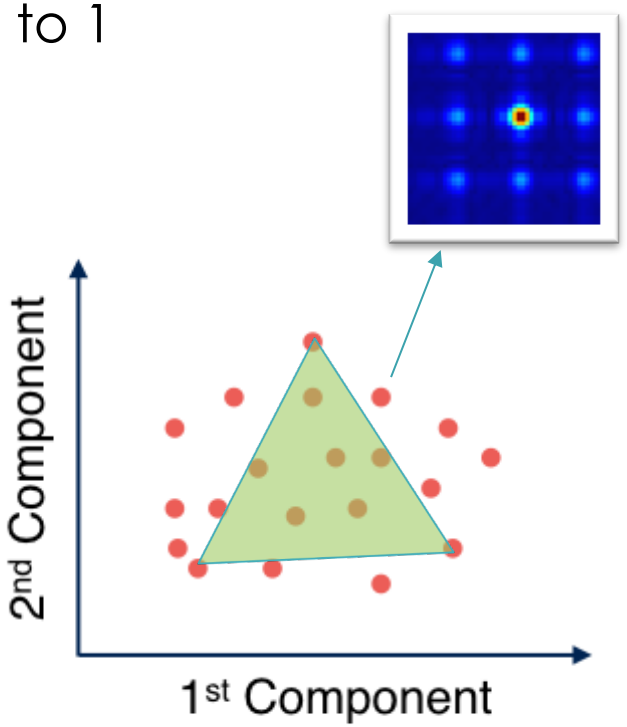
$$p_{ij} = \sum_k e_{ik} c_{kj} + \varepsilon$$

Physics constraint

$$\sum_k c_{kj} = 1$$

- Let E be the matrix of end-members (here, 3).

$$\boldsymbol{E} = \left[\overrightarrow{e_1} \quad \overrightarrow{e_2} \quad \overrightarrow{e_3}\right] \qquad V\left(\frac{\mathbf{1}}{(\boldsymbol{l}-1)!}\right)|\det(E)|$$



- Iteratively select endmembers, accepting the new selection if the volume increases

Segmenting based on phases

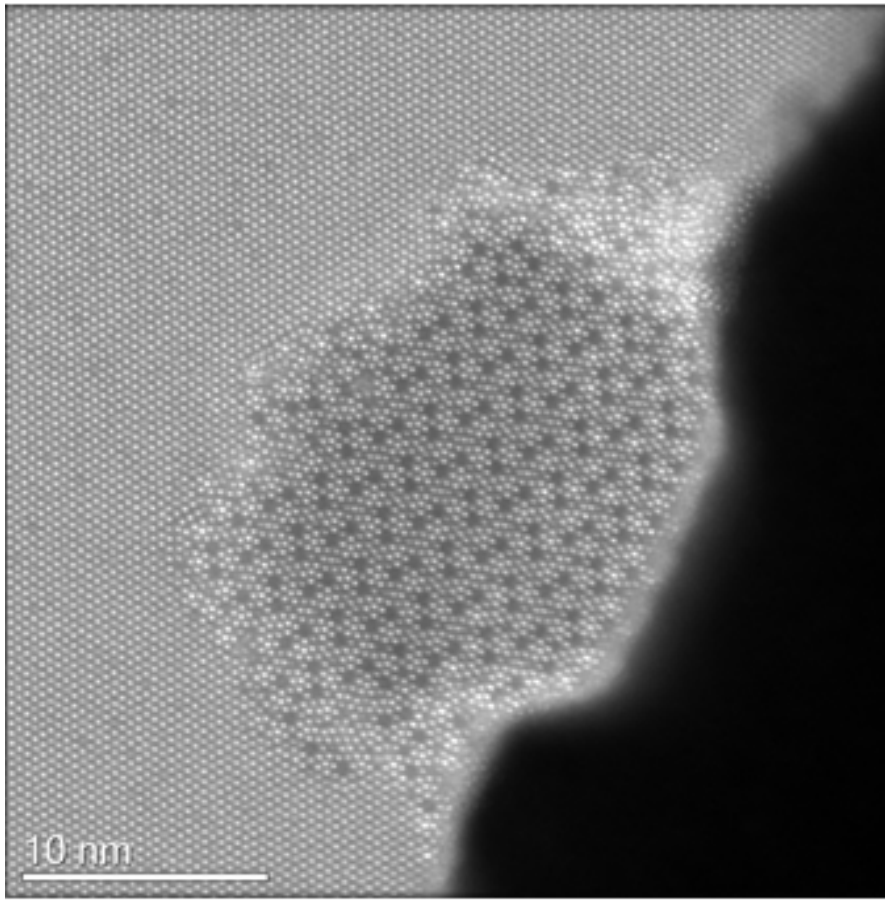
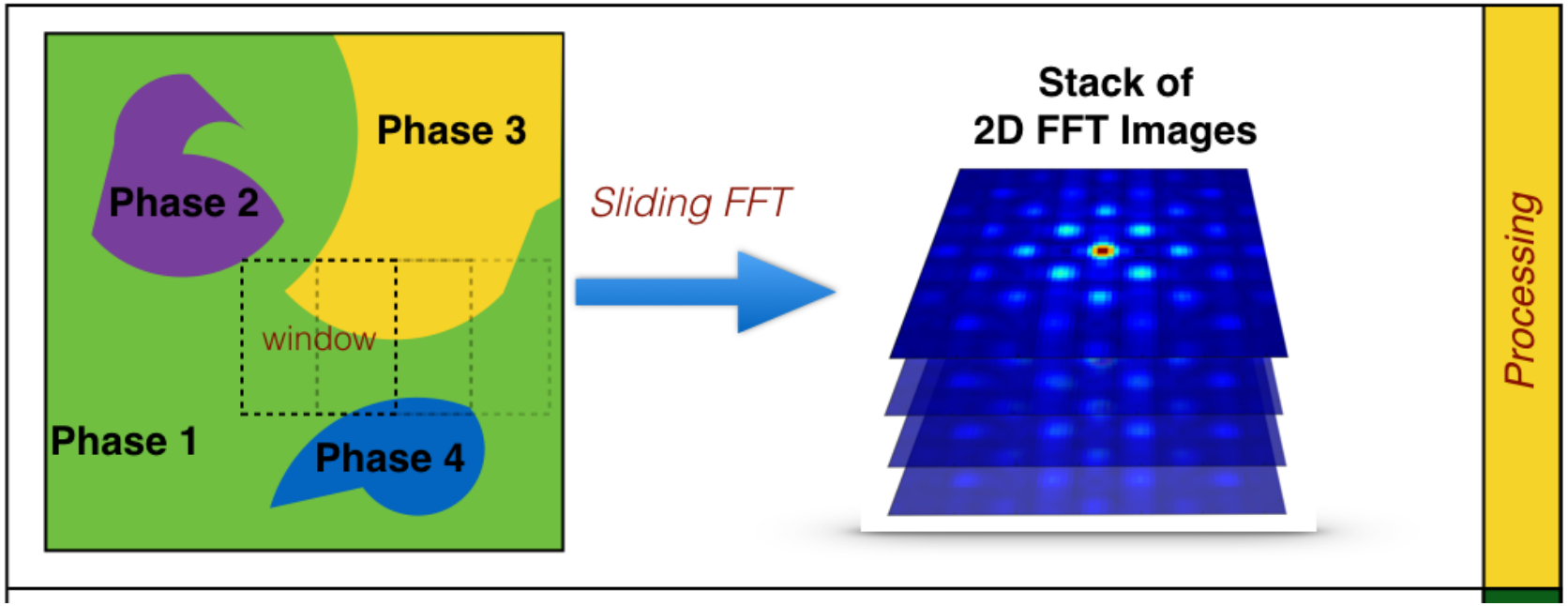


Image courtesy of
A. Borisevich (ORNL)
and
Q. He
(ORNL/Manchester)

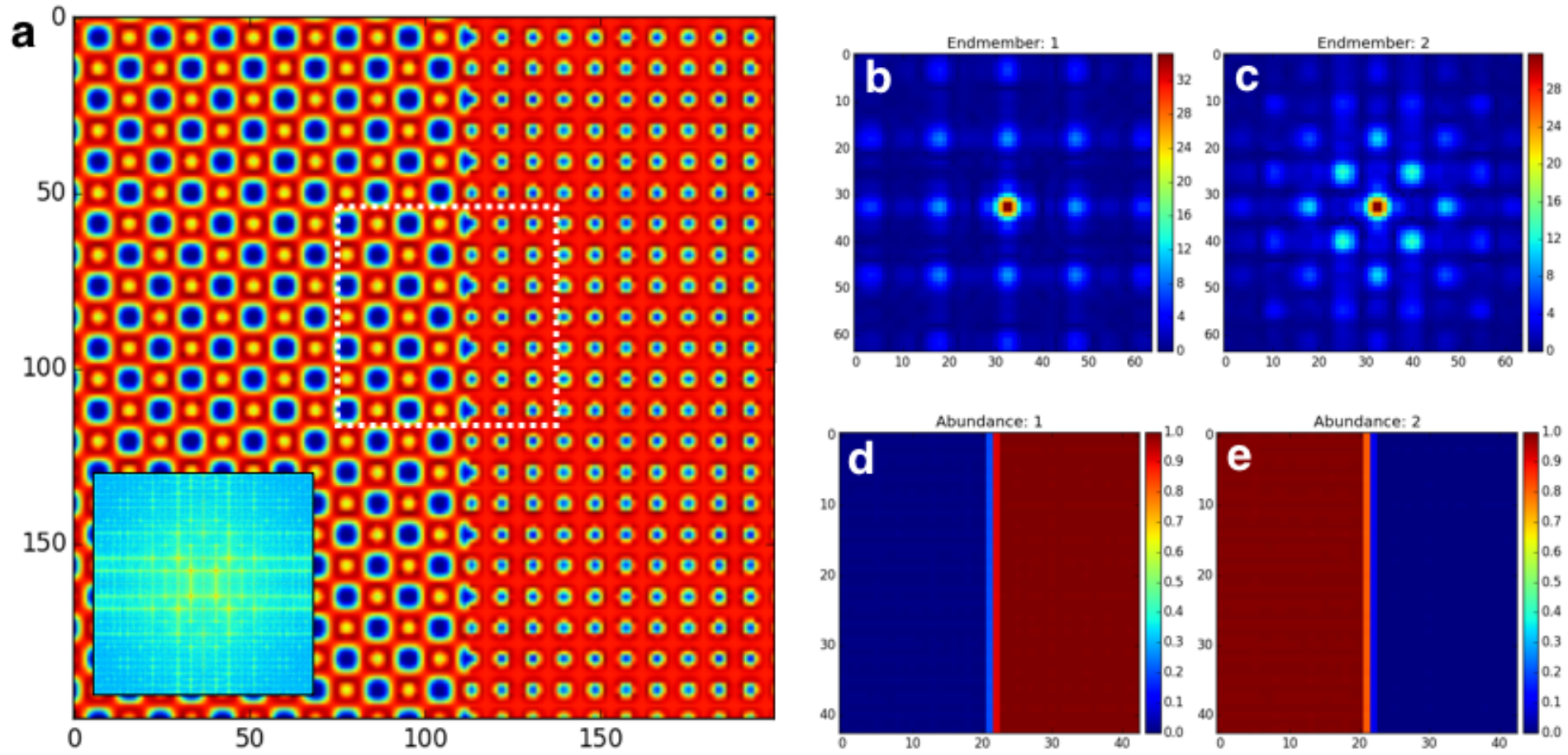
How can we segment this image based on the phases present?

Sliding Window Method



Now, we can try to do spectral unmixing on the 2D stack of FFT images

Ideal test case



Spectral Unmixing: Notebook

Let's try out various unmixing methods including N-FINDR on the STEM image shown below

Open the notebook to try

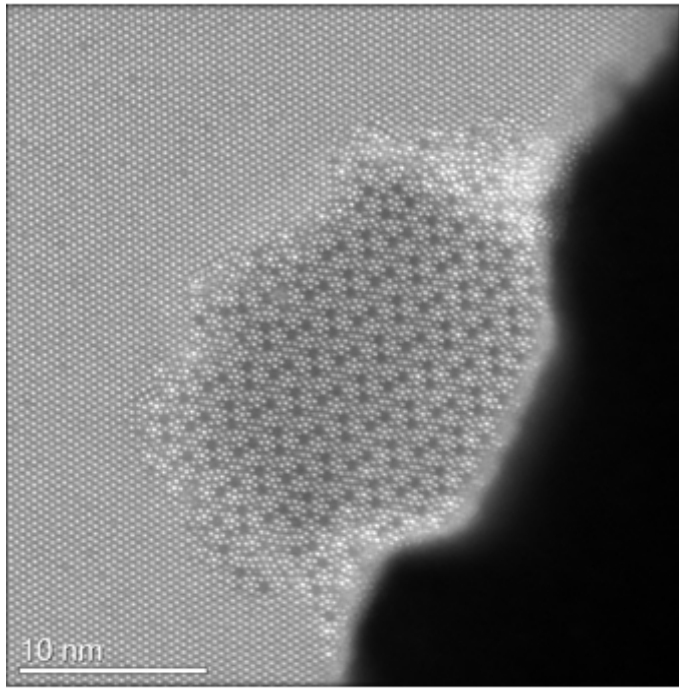


Image courtesy of
A. Borisevich (ORNL)
and
Q. He
(ORNL/Manchester)

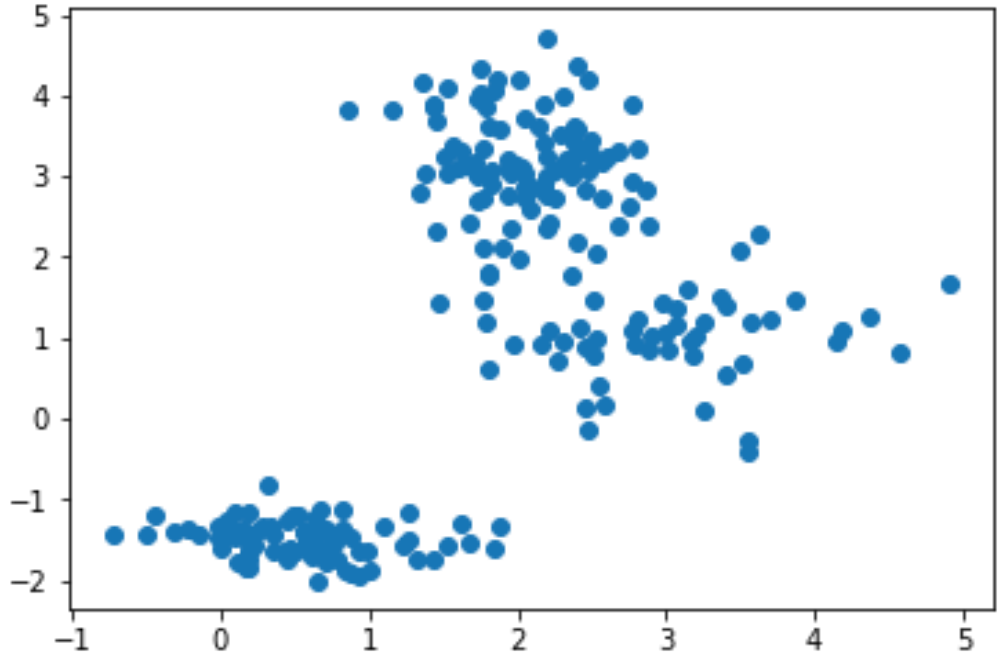
Linear Unmixing as a means for segmentation

Open

Day00/Intro_to_Image_Analysis/
02_Spectral_Unmixing in Google colab to continue

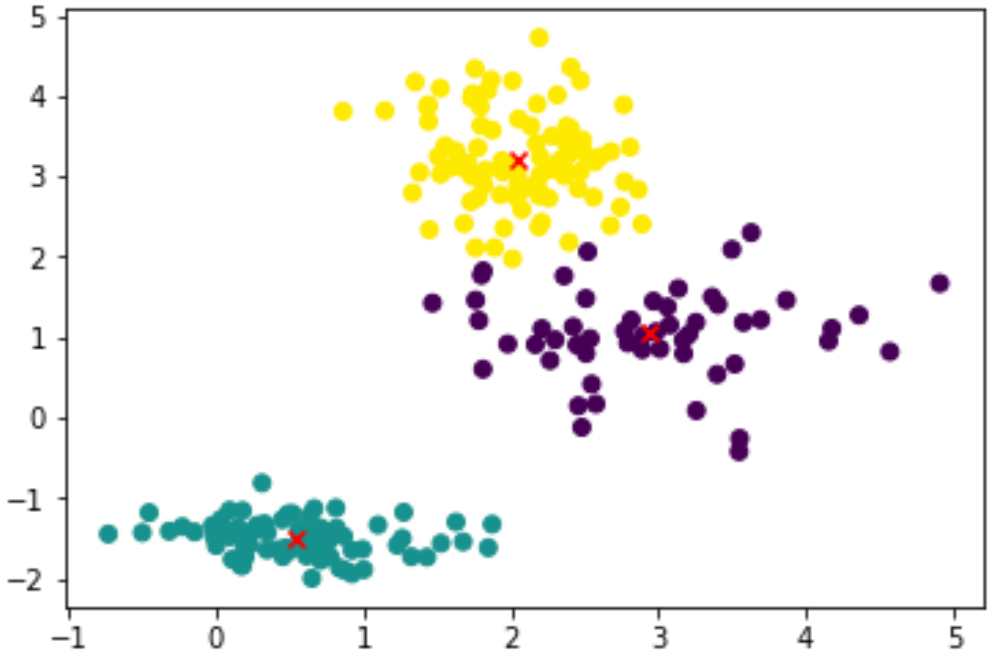
K-Means Clustering

- SVD, NMF and ICA are decompositions. But sometimes, we don't want to decompose our signal, but just group them into 'alike' sets.
- This is termed 'clustering'. The easiest and most widely used method is the k-means algorithm



K-Means Clustering

- SVD, NMF and ICA are decompositions. But sometimes, we don't want to decompose our signal, but just group them into 'alike' sets.
- This is termed 'clustering'. The easiest and most widely used method is the k-means algorithm



K-Means Clustering

- SVD, NMF and ICA are decompositions. But sometimes, we don't want to decompose our signal, but just group them into 'alike' sets.
- This is termed 'clustering'. The easiest and most widely used method is the k-means algorithm

K-means Clustering algorithm, to separate data ($x_1, x_2, \ldots, x_n$) into *k* clusters

$$\arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left|\left|\mathbf{x} - \boldsymbol{\mu}_i\right|\right|^2$$ where μ_i is the mean of points in S_i

(Determine **S** = $\{S_1, S_2, \ldots S_k\}$, such that within cluster sum of squares is minimized)