Undersampling Raster Scans in Spectromicroscopy for reduced dose and faster measurements: supplemental document

This document describes in further detail some of the processes discussed in the main article.

1. PCA AND RTE

We begin by describing the Principal Component Analysis (PCA) and cluster analysis of spectromicroscopy data in more detail. Additionally, we explain how Reduced Thickness Effects (RTE) are implemented. This is based on the methods outlined in [1].

Given our flattened x-ray spectromicroscopy dataset $A \in \mathbb{R}^{n_E \times N}$, we wish to separate the data into its principal components by imposing the following decomposition,

$$A = CR, \tag{S1}$$

with $C \in \mathbb{R}^{n_E \times n_E}$ and $R \in \mathbb{R}^{n_E \times N}$. This can be done either by computing the covariance matrix of A, or using the SVD (singular value decomposition). Both methods are equivalent; we shall describe the covariance approach here.

First, we compute the spectral covariance matrix of A, $B \in \mathbb{R}^{n_E \times n_E}$, then perform an eigendecomposition on B to get,

$$B = AA^T, \tag{S2}$$

$$BC = C\Lambda, \tag{S3}$$

where $\Lambda \in \mathbb{R}^{n_E \times n_E}$ is a diagonal matrix of the eigenvalues of B, and the columns of $C \in \mathbb{R}^{n_E \times n_E}$ are the corresponding eigenvectors. Since the columns of C are orthogonal, its left inverse is equal to C^T. Note that we compute the spectral covariance since typically $n_E << N$.

To complete the C-R decomposition of A in Eq. (S1), we compute

$$R = C^{-1}A = C^T A, \tag{S4}$$

The columns of C and rows of R are the principal components of A, and the corresponding eigenvalues in Λ describe their significance in the dataset A. Additionally, the columns of C can be thought of as abstract absorption spectra and the rows of R as abstract spatial maps - linear combinations of true spectra and spatial maps, without any physical interpretation.

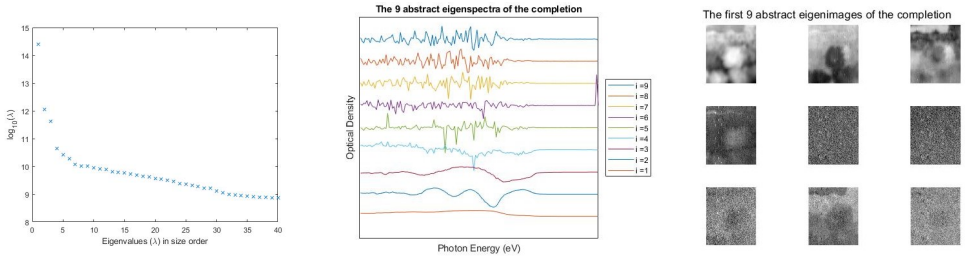
To compute the PCA's low rank approximation, we simply take the first L columns of C and the first L rows of R, and discard the remaining data. Thus we get,

$$A' = C'R', \tag{S5}$$

where $A' \in \mathbb{R}^{n_E \times N}$, $C' = C(:, 1:L) \in \mathbb{R}^{n_E \times L}$ and $R' = R(1:L, :) \in \mathbb{R}^{L \times N}$. Note, we have used MATLAB-like notation to partition the rows and columns of the matrices. To maximise the variation of the approximation while minimising the rank, L is chosen to be at the elbow (point of maximum curvature) of the eigenvalues, Λ. Again, this is equivalent to setting L to be the elbow of the singular values of A.

In Figure S1 we illustrate the eigenvalues of DS2, alongside the first 9 eigenspectra and eigenimages (columns of C and rows of R respectively). KNEEDLE, [2], was used and identified the elbow as $L = 4$. In Figures S1b & S1c, we see that $L = 4$ correctly determines the number of components that show significant variation and little noise: the first four spectra are much more smooth than the rest, and all eigenimages after the 4^{th} show the 'salt and pepper' pattern that is indicative of noise. In practice we use $L = 5$ to ensure we capture all the variation in the data.

To identify the different materials in the specimen, we now use cluster analysis on the columns of R' to collect together pixels with similar combinations of abstract absorption spectra. The number of cluster centres must be set, and should be greater than, or equal to, the number of



(a) Eigenvalues (diag. entries of Λ) **(b)** Eigenspectra (columns of C) **(c)** Eigenimages (rows of R)

Fig. S1. PCA results of DS2. KNEEDLE identified the elbow of the eigen values at 4. Notice the first 4 eigenspectra and eigenimage contain useful information, after which we see the components are corrupted by excessive noise.

materials within the specimen. Several different cluster algorithms may be used, but we generally use a standard implementation of kmeans. Averaging the x-ray absorption spectrum over all the pixels in each cluster provides the corresponding absorption spectra of that material, with a much higher signal to noise ratio.

As was explained in the main text, it can be seen that the first component is simply averaged over all pixels and is associated more with the overall thickness of the specimen [1]. Occasionally, we wish to avoid the thickness of the specimen skewing the clustering results by reducing the effect of the first principal component. To implement reduced thickness effect (RTE), we simply ignore the most significant component and set $C' = C(:\,,\,2:L)$ and $R' = R(2:L\,,\,:)$ before applying the cluster analysis as usual. Note that other approaches are available, such as scaling the principal components.

2. ROBUST RASTER SAMPLING

Given a dense dataset A, we model the sparse scan $\mathcal{P}_\Omega(A)$ by computing the following projection,

$$\mathcal{P}_\Omega(A) = \Omega \circ A, \tag{S6}$$

where $\circ$ is the Hadamard product, and $\Omega \in \{0,1\}^{n_E \times N}$ is the sampling pattern indicating the location of the known entries. We now discuss different methods of producing the randomised sampling patterns to be used in sparse scans.

The most common sampling model used is Bernoulli sampling, where each data-point is sampled i.i.d (independent and identically distributed) with probability p. To improve the efficiency of sparse spectromicroscopy, we instead use Raster sampling, which scans physical rows of the specimen together i.i.d. with probability p. This way, rows that aren't sampled can be passed over quickly.

One disadvantage is that, for low undersampling ratios, the probability that a row is not sampled at all increases sharply.

It is clear that it is impossible to recover pixels that are not sampled for any energy level using Low Rank Matrix Completion. In fact, it seems the more frequently a point is sampled, the easier it is to recover the surrounding missing entries (e.g. one pixel at different energy levels, or different pixels at one energy level). To ensure all rows of the specimen are scanned, and to promote a more even sampling pattern, we use a robust variation of Raster Sampling called *Robust Raster Sampling*.

Robust Raster Sampling can be implemented either by varying the probability of each row, or by discretely restricting which rows can be scanned. Here we describe the discrete method.

Begin with a set $v = [n_1] = \{1, \ldots, n_1\}$ that indexes the rows that have yet to be scanned (recall that n_1 is the number of rows on the sample). Next, compute the expected number of rows to be sampled at each energy, pn_1. For each energy level, sample pn_1 rows uniformly at random from v, removing the sampled rows from v. If pn_1 is not an integer, simply take the ceiling or floor at ranodm using the decimal as the probability. Repeat this process for each energy level E_i, $i = 1, ..., n_E$. For energy levels where $pn_1 > |v|$ (i.e. there are fewer than pn_1 rows remaining in v), we must first record the deficit $m = pn_1 - |v|$ before sampling the last rows in v. We now repopulate v, withholding only the rows that have already been sampled on this energy level (to

avoid sampling them twice). Finally, we sample the remaining m rows from the repopulated set v before moving to the next energy level.

In short, Robust Raster sampling ensures that every row is sampled once before any can be sampled twice. Additionally, be ensuring pn_1 rows are sampled for each energy level, the known entries are more evenly distributed across $\mathcal{P}_\Omega(A)$.

3. DETAILS ON ASD

To recover the missing entries of $\mathcal{P}_\Omega(A)$, ASD [3] imposes the decomposition $A = XY^T, X \in \mathbb{R}^{n_E \times r}, Y \in \mathbb{R}^{N \times r}$ and seeks to minimise the objective function:

$$f(X,Y) = \frac{1}{2}||\mathcal{P}_\Omega(A - XY^T)||_F^2. \tag{S7}$$

This is done by alternately fixing one component and minimising the other using gradient descent with exact step sizes. For this simple procedure, the directions $d_X,\ d_Y$ and step sizes $t_X,\ t_Y$ are computed to give the next iterates:

$$X_{i+1} = X_i + d_X t_X \qquad \text{and} \qquad Y_{i+1} = Y_i + d_Y t_Y. \tag{S8}$$

We write $f(X,Y)$ as $f_Y(X)$ when Y is fixed, and $f_X(Y)$ when X is fixed. Thus, we have gradients,

$$\nabla f_Y(X) = -(\mathcal{P}_\Omega(A) - \mathcal{P}_\Omega(XY^T))Y \qquad \text{and} \qquad \nabla f_X(Y) = -X^T(\mathcal{P}_\Omega(A) - \mathcal{P}_\Omega(XY^T)). \tag{S9}$$

For gradient descent, we use the step directions,

$$d_X = -\nabla f_Y(X) \qquad \text{and} \qquad d_Y = -\nabla f_X(Y). \tag{S10}$$

Finally, we can compute the exact step sizes required for steepest descent. This is done by minimising,

$$g_X(t) = f_Y(X + td_X), \qquad \text{and} \qquad g_Y(t) = f_X(Y + td_Y), \tag{S11}$$

$$= \frac{1}{2}||\mathcal{P}_\Omega(A) - \mathcal{P}_\Omega((X + td_X)Y^T)||_F^2, \qquad = \frac{1}{2}||\mathcal{P}_\Omega(A) - \mathcal{P}_\Omega(X(Y + td_Y)^T)||_F^2. \tag{S12}$$

Indeed, we can compute the step sizes exactly, giving

$$t_X = \operatorname{argmin} g_X(t) \qquad \text{and} \qquad t_Y = \operatorname{argmin} g_Y(t) \tag{S13}$$

$$= \frac{||\nabla f_Y(X)||_F^2}{||\mathcal{P}_\Omega(\nabla f_Y(X)Y^T)||_F^2} \qquad = \frac{||\nabla f_X(Y)||_F^2}{||\mathcal{P}_\Omega(X[\nabla f_X(Y)]^T)||_F^2} \tag{S14}$$

It can be seen that, following the updates to X_{i+1}, Y_{i+1}, the residuals can be written as

$$\text{res}_{X_{i+1}} = \mathcal{P}_\Omega(A - X_{i+1}Y_i^T) \qquad \text{res}_{Y_{i+1}} = \mathcal{P}_\Omega(A - X_{i+1}Y_{i+1}^T) \tag{S15}$$

$$= \mathcal{P}_\Omega(A - (X_i + t_{X_i}d_{X_i})Y_i^T) \qquad = \mathcal{P}_\Omega(A - X_{i+1}(Y_i + t_{Y_i}d_{Y_i})^T) \tag{S16}$$

$$= \mathcal{P}_\Omega(A - X_iY_i^T) - t_{X_i}\mathcal{P}_\Omega(d_{X_i}Y_i^T) \qquad = \mathcal{P}_\Omega(A - X_{i+1}Y_i^T) - t_{Y_i}\mathcal{P}_\Omega(X_id_{Y_i}^T) \tag{S17}$$

$$= \text{res}_{Y_i} - t_{X_i}\mathcal{P}_\Omega(d_{X_i}Y_i^T) \qquad = \text{res}_{Y_i} - t_{Y_i}\mathcal{P}_\Omega(X_id_{Y_i}^T) \tag{S18}$$

This allows for an efficient implementation. Since $\mathcal{P}_\Omega(d_{X_i}Y_i^T)$ is already formed when computing t_{X_i}, we can avoid the matrix-matrix multiplication required to calculate the residual, and simply update it for each iteration. Using this update, the per iteration cost of this algorithm is $8\Omega r$ [3], making it very efficient. The method is summarised in Algorithm S1.

4. CHOOSING THE COMPLETION RANK

It has been noted before that the completion rank of the dataset must be set before implementing ASD or LoopedASD. In Section 2 of the main document, we derived the low rank nature of spectromicroscopy datasets, but correctly identifying the optimal rank will determine the success of the completion algorithm used. We have noted in Section 4 of the main document that datasets with lower approximate ranks require fewer known entries to produce good reconstructions, making the sparse experiments more efficient. It is therefore crucial that robust and reliable

Algorithm S1. Alternating Steepest Decent (ASD)

Input: $\mathcal{P}_\Omega(A)$, $X_0 \in \mathbb{R}^{n_E \times r}$, $Y_0 \in \mathbb{R}^{r \times N}$, $kmax$

for $i = 1$ **to** kmax **do**

$$\begin{aligned}
\nabla f_{Y_i}(X_i) &= -res_{Y_i} Y_i^T \\
d_{X_i} &= -\nabla f_{Y_i}(X_i) \\
t_{X_i} &= ||\nabla f_{Y_i}(X_i)||_F^2 \;/\; ||P_\Omega(\nabla f_{Y_i}(X_i) Y_i)||_F^2 \\
X_{i+1} &= X_i + t_{X_i} d_{X_i} \\
\text{res}_{X_{i+1}} &= \text{res}_{Y_i} - t_{X_i} \mathcal{P}_\Omega(\nabla f_{Y_i}(X_i) Y_i^T)
\end{aligned}$$

$$\begin{aligned}
\nabla f_{X_{i+1}}(Y_i) &= -X_{i+1}^T (P_\Omega(A) - P_\Omega(X_{i+1} Y_i)) \\
d_{Y_i} &= -\nabla f_{X_{i+1}}(Y_i) \\
t_{Y_i} &= ||\nabla f_{X_{i+1}}(Y_i)||_F^2 \;/\; ||P_\Omega(X_{i+1} \nabla f_{X_{i+1}}(Y_i))||_F^2 \\
Y_{i+1} &= Y_i + t_{Y_i} d_{Y_i} \\
\text{res}_{Y_{i+1}} &= \text{res}_{X_i} - t_{Y_i} \mathcal{P}_\Omega(X_i^T \nabla f_{X_i}(Y_i))
\end{aligned}$$

if Stopping Conditions Reached **then**
break

methods for estimating the rank of a sampled dataset are developed. By accurately estimating this value, the reconstruction should capture the full variation of the data, while filtering out as much of the noise as possible; this will ensure the sparse clustering results are similar to the full-data case.

The first step is to identify the most appropriate completion rank using the full datasets. We begin by considering the Eckart-Young-Mirsky (EYM) Theorem [4], which provides the **minimum approximation error** in the frobenius norm for a rank-k approximation of A, denoted $A^{(k)}$:

$$\min_{A^{(k)}} ||A - A^{(k)}||_F^2 = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + ... + \sigma_n^2}, \tag{S19}$$

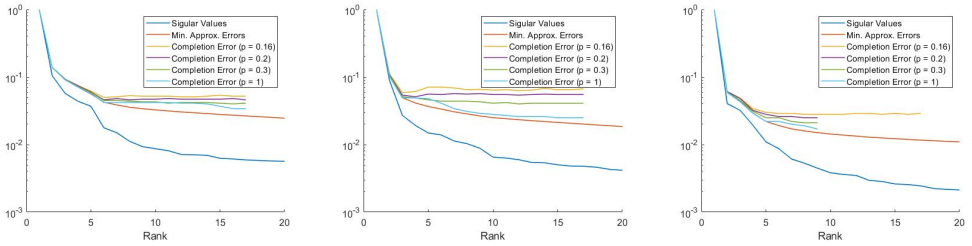
where $\sigma_1, \ldots, \sigma_n$ are the singular values (SVs) of A. It should be noted that this approach cannot be used on sparse data.

By repeatedly sampling and completing full datasets for ranks $r = 1, ..., 25$ and a range of undersampling ratios, we can determine how the **completion results** vary with the completion rank. In Figure S2, we have plotted the singular values (dark blue), the minimum completion errors (orange), and the completion results for each independent dataset available (DS1, DS3, DS5).

We can see that the curve of the minimum approximation errors produced by the EYM Theroem have similar curved shapes to the plot of SVs. This similarity is due to the dominance of the larger SVs in the estimate. When the SVs decay rapidly, $\sigma_{k+1}^2 >> \sigma_i^2$ for $i = (k+2), \ldots, n$; thus $||A - A^{(k)}||^2 \approx \sigma_{k+1}$ and also decays rapidly. Then, as the SV plot flattens, we have that $\sigma_{k+1} \approx \sigma_{k+2} \approx \sigma_{k+3} \approx ...$ and the difference between consecutive minimum approximation errors decreases, flattening this plot as well (orange plot in Figure S2).

Naturally, the minimum approximation error is a lower bound for any completion algorithm. As can be seen in Figure S2, the errors will easily achieve their optimal values when using lower ranks. For higher completion ranks, the flattening of the EYM plot means the completion errors must flatten as well. Despite the fact that the bound decrease slowly, it becomes increasingly difficult for algorithms to produce minimal errors due to slower convergence rates, so the flattening of the observed completion errors is more severe. Overall, we see that the average completion errors plots a similar curve to the SVs and minimum approximation errors - initially decreasing rapidly before flattening and remaining relatively constant.

Consider now what are the properties of the optimal rank: it should minimise the completion error using the smallest completion rank. This is precisely the same problem as selecting the rank L for the PCA in the main document, and given the connection we have established between the SV plot and the completion error plot, we apply the same selection method here as well. Once again, we seek the elbow point (the point of maximum curvature) of the known completion errors, which we label r^*. We see in Figure S2 that for $r < r^*$, the both the average and minimum completion errors decrease rapidly, thus increasing the rank will significantly improve the reconstruction; for $r > r^*$, there is little difference in the mean completion error, and increasing the rank will not



(a) DS1 **(b)** DS3 **(c)** DS5

Fig. S2. Plots illustrating the Singular Values, the minimum approximation errors and mean ASD completion errors for each rank. Note the singular values and minimum approximation errors have been normalised to enable a qualitative comparison. The minimum approximation errors are computed using the Eckart-Young-Mirsky Theorem (Eq. S19). Completion errors, e_c, are computed as the relative Frobenius norm of the difference between the full data and the reconstruction (see Eq(16) in the main text). The values illustrated are means taking over 16 independent completions. We see the similar patterns indicate the optimal completion rank is the same as the optimal approximation rank for standard spectromicroscopy PCA.

yield better results. Thus, r^* reaches the optimal balance - it allows ASD to capture almost all the variation in the data, while remaining low to improve the probability of completion, especially with lower undersample ratios.

Now that we have a target, we can develop methods for identifying r^* from the sparse samples, $\mathcal{P}_\Omega(A)$. The general outline will be to compute short, approximate completions for each trial rank, compute and plot the residual-norm, then find the elbow point using KNEEDLE. Since the data is sparse, we can only evaluate a completion using the residual-norm and not the full completion error, i.e. we must compare back to the known entries used for the completion. To avoid overfitting, we take a cross validation approach.

Iterating over the trial completion rank, r, we partition the known entries into k subsets and select one at random to be the validation set - the remaining entries become the training data. From experience, it is very unlikely a dataset will have an approximate rank higher that 15, so iterating through 20 trial ranks is sufficient in general. This value is easily increased if needs be.

For each trial rank $r = 1, ..., 20$, the training data is completed using ASD, and the validation error is computed by taking the relative Frobenius norm over the validation set. We set ASD to run for n_it iterations; this parameter, alongside the number of partitions k, can be tuned to adjust the efficiency and accuracy of this process. Once all validation errors have been computed, we use KNEEDLE to compute the elbow.

This new procedure for identifying the completion rank is summarised below:

$$
\begin{cases}
\textbf{for } r = 1, ..., 20: \\
\quad \text{Partition the known entries to get } \{B_1, ...B_k\}, \text{ so that } \Omega = \bigcup_{j=1}^{k} B_k\,. \\
\quad \text{Choose } i \in [k] \text{ uniformly at random.} \\
\quad \text{Set } \hat{B} = B_i \text{ to be the validation set and } \Omega \backslash B_i \text{ to be the training set.} \\
\quad \text{Complete the training data using ASD with completion rank } r \text{ and } n\_it \text{ iterations.} \\
\quad \text{Compute the validation error } \frac{||\mathcal{P}_{\hat{B}}(A) - \mathcal{P}_{\hat{B}}(A^*)||_F^2}{||\mathcal{P}_{\hat{B}}(A)||_F^2}, \text{ where } A^* = (X^*)^{(r)}(Y^*)^{(r)}. \\
\quad \text{Record the validation error as the } r^{th} \text{ trial error.} \\
\textbf{end} \\
\text{Apply KNEEDLE to identify the elbow of the trial errors.}
\end{cases}
\tag{S20}
$$

Following some testing of the new method, it was quickly found that there was little change in accuracy for $k > 10$. For $k \leq 10$, the training sets were made too small by the partitioning to reliably reconstruct the datasets. From here on, we fix $k = 20$, and we found that $n_{it} = 300$ iterations of ASD were generally sufficient to ensure reliable results across all undersample ratios and completion ranks. These results averaged slightly below the previously computed optimal rank, with a difference of no more than 1. Since it is better to overestimate the completion rank

than under estimate it, we add 2 to the result to ensure we've captured the full variation within the data.

REFERENCES

1. M. lerotic, C. Jacobsen, T. Schafer, and S. Vogt, "Cluster analysis of soft x-ray spectromicroscopy data," Ultramicroscopy **100.1**, 35–57 (2004).
2. V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," (2011).
3. J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," Appl. Comput. Harmon. Analysis **40**, 417–429 (2016).
4. C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," Psychometrika **1**, 211 –218 (1936).