nature portfolio

https://doi.org/10.1038/s41586-021-04050-9

Supplementary information

Few-fs resolution of a photoactive protein traversing a conical intersection

In the format provided by the authors and unedited

Supplementary Information

Few-fs resolution of a photoactive protein traversing a conical intersection

A. Hosseinizadeh[1], N. Breckwoldt[2,3,4], R. Fung[1], R. Sepehr[1], M. Schmidt[1], P. Schwander[1],
R. Santra[2,3,4], A. Ourmazd[1]*

[1] University of Wisconsin Milwaukee, 3135 N. Maryland Ave, Milwaukee WI 53211, USA
[2] Center for Free-Electron Laser Science, Deutsches Elektronen-Synchrotron DESY,
Notkestrasse 85, 22607 Hamburg, Germany
[3] Department of Physics, Universität Hamburg, Notkestr. 9-11, 22607 Hamburg, Germany
[4] The Hamburg Centre for Ultrafast Imaging, Luruper Chaussee 149, 22761 Hamburg, Germany

* Corresponding author: Ourmazd@uwm.edu

List of Contents

1. Data-analytical approach

Our approach is based on manifold-based machine learning, including Nonlinear Laplacian Spectral Analysis [19]. In this approach, data vectors are ordered based on their known timestamps, and concatenated to form the supervector matrix X. The supervectors are then projected onto their manifold,

viz. $A = X\mu\Phi$. [1]

Here, μ and Φ are respectively the Riemannian measure and the Diffusion Map empirical orthogonal functions (EOF).

Singular Value Decomposition: $A = USV^T$ [2]

and back projection: $\tilde{X} = A\Phi^T = USV^T\Phi^T$ [3]

are applied to yield the reconstruction matrix $\tilde{X}$, which must be unwrapped to give individual reconstructed data vectors [16,19].

Independent orthogonal dynamical modes can be studied by reconstructing with specific SVD modes: $\tilde{X}_1 = U_1S_1V_1^T\Phi^T$, $\tilde{X}_2 = U_2S_2V_2^T\Phi^T$, …, $\tilde{X}_k = U_kS_kV_k^T\Phi^T$. [4]

2. Computing Euclidean distances and dot products

For N data vectors with D pixels each, and concatenation parameter c, the supervector matrix X (dimensions $cD \times (N - c + 1)$) can be huge, even for modest values of N and D. It is, however, not necessary to explicitly store or manipulate X. For instance, the SVD step above (Equation [2]) can be more efficiently carried out by using the following steps:

(i) calculate the dot products amongst the supervectors, i.e. X^TX, in blocks (more details below)

(ii) form the A^TA matrix, i.e. $A^TA = (\mu\Phi)^TX^TX(\mu\Phi)$; [5]

(iii) solve for the eigenvalues and eigenvectors of the $A^T A$ matrix;

(iv) the right singular vectors (V) of A are the eigenvectors of $A^T A$; and the singular values (S) of A are the square roots of the eigenvalues of $A^T A$, or in other words,

$$A^T A = (\mu\Phi)^T (X^T X)\mu\Phi = VS^2V^T \; ; \qquad [6]$$

(v) the left singular vectors (U) of A are obtained from $U = X\mu\Phi\mathrm{VS}^{-1}$. [7]

Note that if we are using a small number ℓ_{max} of Diffusion Map EOFs, say $\ell_{max} = 100$, the matrix $A^T A$, of dimensions $\ell_{max} \times \ell_{max} = 100 \times 100$, is rather small, and can be accumulated using a double loop through the block structure of $X^T X$. Also, since a full reconstruction results in up to c copies of each individual snapshot, which might be too many, it is not necessary to calculate the full U matrix. Equation [7] can thus be used to compute U in a row-wise/ block-wise fashion to only generate enough copies of each individual snapshot for our reconstruction.

Squared Euclidean distances and dot products amongst supervectors are calculated in Nonlinear Laplacian Spectral Analysis (NLSA). For N data vectors with D pixels each, and concatenation parameter c, runtimes for these steps scale as $\sim N^2 cD$. Calculations with N, D, and c in the tens or hundreds of thousands can, literally, take years on a desktop machine.

For this paper, we have developed a so-called Shift-and-Add algorithm, which reduces the runtime scaling to $\sim N^2 D + N^2 log_2(c)$. Calculations with N, D, and c in the tens or hundreds of thousands now take only days on a desktop machine, and only hours on computer clusters with fairly modest resources. To describe this algorithm in more detail, we define:

$\vec{x}_j$ = data vector j ,

$\vec{x}_j^c$ = supervector j with concatenation parameter c , and

$s_{i,j}^c$ = the squared Euclidean distance between supervectors i and j . [8]

By definition, we have:

$$s_{i,j}^c = \left|\vec{x}_i^c - \vec{x}_j^c\right|^2 . \qquad [9]$$

Writing out the constituent data vectors of the supervectors explicitly, Equation [9] becomes:

$$s_{i,j}^c = \sum_{p=0}^{p=c-1} \left|\vec{x}_{i+p} - \vec{x}_{j+p}\right|^2 . \qquad [10]$$

For concatenation parameter a, where $a < c$, we break up the sum in Equation [10] to give:

$$s_{i,j}^c = \sum_{p=0}^{p=a-1} \left|\vec{x}_{i+p} - \vec{x}_{j+p}\right|^2 + \sum_{p=a}^{p=c-1} \left|\vec{x}_{i+p} - \vec{x}_{j+p}\right|^2 . \qquad [11]$$

Substituting $p = q + a$ in the second sum above yields:

$$s_{i,j}^c = \sum_{p=0}^{p=a-1} \left|\vec{x}_{i+p} - \vec{x}_{j+p}\right|^2 + \sum_{q=0}^{q=c-a-1} \left|\vec{x}_{i+a+q} - \vec{x}_{j+a+q}\right|^2 = s_{i,j}^a + s_{i+a,j+a}^{c-a} . \qquad [12]$$

Using Equation [12], the matrix of squared Euclidean distances amongst supervectors for any concatenation parameter can be built from the matrices with lower concatenation parameters. For example, starting with the matrix of squared Euclidean distances amongst data vectors, the matrices of squared Euclidean distances between supervectors with concatenation parameters c=2 and c=4 can be successively assembled as:

$$s_{i,j}^{c=1} = \left|\vec{x}_i - \vec{x}_j\right|^2 ,$$

$$s_{i,j}^{c=2} = s_{i,j}^{c=1} + s_{i+1,j+1}^{c=1} ,$$

$$s_{i,j}^{c=4} = s_{i,j}^{c=2} + s_{i+2,j+2}^{c=2} . \qquad [13]$$

After the calculation of $s_{i,j}^{c=1}$, it takes $log_2(c)$ steps of “doubling” ($\sim N^2$ additions each) to reach the concatenation parameter c. Runtime thus scales as $\sim N^2 D + N^2 log_2(c)$. [14]

Matrices of squared Euclidean distances amongst supervectors for arbitrary concatenation parameters can be assembled, for instance

$$s_{i,j}^{c=3} = s_{i,j}^{c=1} + s_{i+1,j+1}^{c=2} \text{ ,}$$

$$s_{i,j}^{c=5} = s_{i,j}^{c=2} + s_{i+2,j+2}^{c=3} \text{ ,}$$

$$s_{i,j}^{c=6} = s_{i,j}^{c=3} + s_{i+3,j+3}^{c=3} \text{ .} \qquad [15]$$

Starting with the elements of the matrix of squared Euclidean distances between data vectors in files (blocks), the results for successively higher concatenation parameters can be obtained as follows:

(i) Read files two at a time;

(ii) Shift the content of one with respect to the other; and

(iii) Add and save the results in files.

The above algorithm is named “Shift-and-Add”.

By replacing $|x - y|^2$ with $(x \cdot y)$ in the discussion above, it is obvious that Shift-and-Add can be used to calculate the matrix of dot products amongst supervectors with arbitrary concatenation parameters.

3. Modifications needed to handle sparse data matrices

Since our data matrix initially contains undefined elements (see Methods section entitled "Data preprocessing"), we must adjust the way we calculate squared distances and projection in NLSA. This adjustment is based on the number of times each unique reflection has been measured (across the dataset), and by pre-normalizing (dividing) each row of the data matrix by the number of times the corresponding Bragg reflection has been measured.

For squared distances:

i. Squared distance between two data vectors is calculated using only pixels defined in both vectors;

ii. Squared distance between two data vectors with no common pixels is set to infinity, and any supervector squared distance they contribute to will also be infinity (see section above for the "Shift-and-Add" algorithm);

iii. Infinities in the squared distance matrix are removed/ignored in Diffusion Map where only a small number of nearest-neighbor squared distances are kept.

To project on to the manifold ($X\mu\Phi = USV^T$) in NLSA:

i. Undefined pixels in the data matrix are set to 0;

ii. The dot-product X^TX is calculated using Shift-and-Add (see section above for the "Shift-and-Add" algorithm);

iii. V and S are obtained by solving for the eigenvectors/ eigenvalues of the matrix $(\mu\Phi)^T(X^TX)\mu\Phi$, i.e. $(\mu\Phi)^T(X^TX)\mu\Phi = VS^2V^T$;

iv. U is obtained from $U = X\mu\Phi\mathrm{VS}^{-1}$.

4. Time-labeling of reconstructed videos

In the time-lagged embedding used in this paper, the data vectors are ordered based on their known timestamps, and concatenated to form the supervector matrix X. This matrix is then projected onto its manifold Φ, and singular value decomposition and back projection are applied to obtain the reconstructed matrix $\tilde{X}$ in the data space, which must be unwrapped to give the individual (reconstructed) data vectors [16,19].

When applied to data with inaccurately known timestamps, our data-analytical pipeline has been shown to recover the dynamics on a uniform grid of timepoints with negligible timing error [16].

Defining the timestamp of a supervector as the average of the timestamps of its constituent data vectors, the concatenation parameter c is chosen so that:

i. The set of time steps Δt between consecutive supervectors in X becomes more or less uniform; and
ii. The time step Δt between consecutive supervectors remain more or less constant as the concatenation parameter is further increased.

For the present study, $c = 32768$, and $\Delta t = 7.35as$.

The columns of the reconstruction matrix $\tilde{X}$ have the same supervector timestamps as the matrix X, the individual constituent data vectors in $\tilde{X}$ are, however, uniformly spaced with time step Δt.

The start time (t_{start}) of a reconstructed movie is determined by

i. Knowing the timestamp of the first supervector: $\tilde{t}_1$;

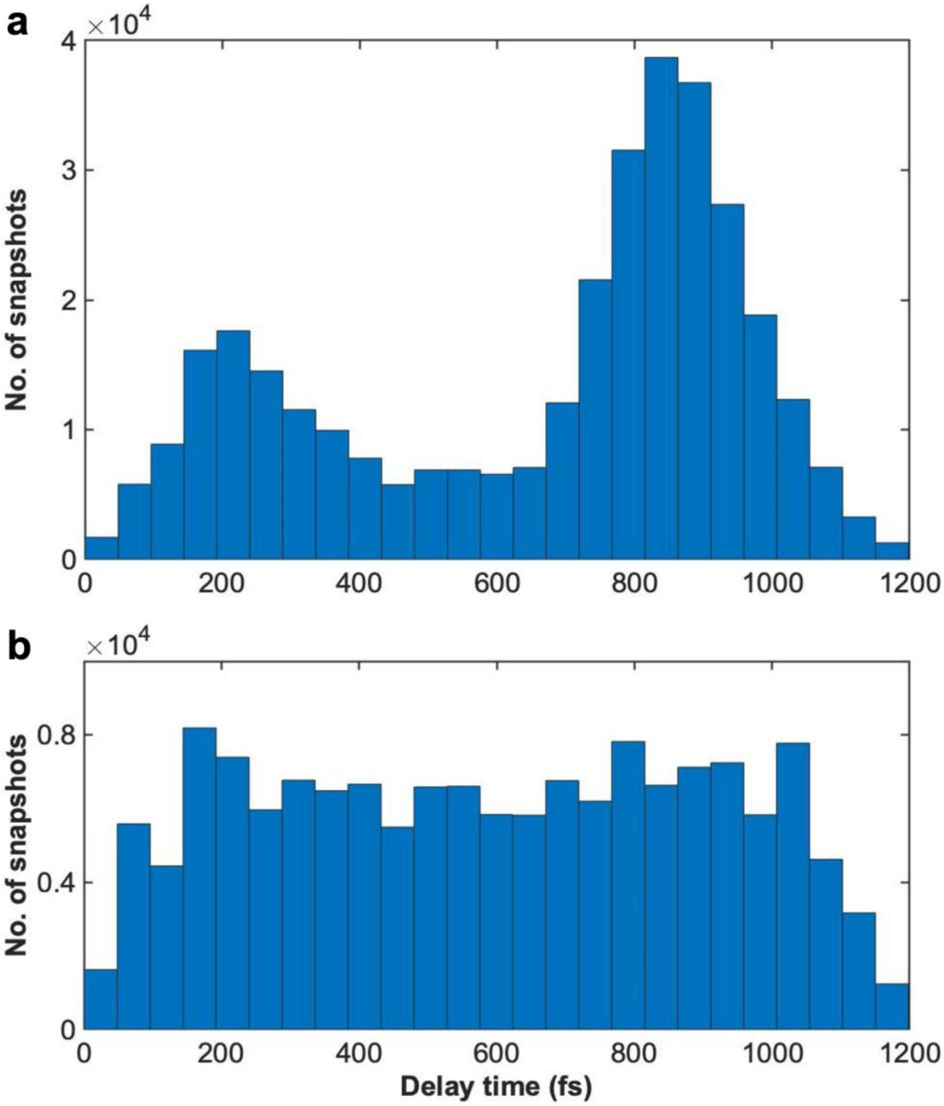
ii. Noting that the first data vector is half a concatenation window behind the supervector to which it belongs: $-\frac{c-1}{2}\Delta t$;

iii. Knowing the number (p) of early data vectors that have been dropped, because they have too few copies in the reconstruction: $+p\Delta t$.

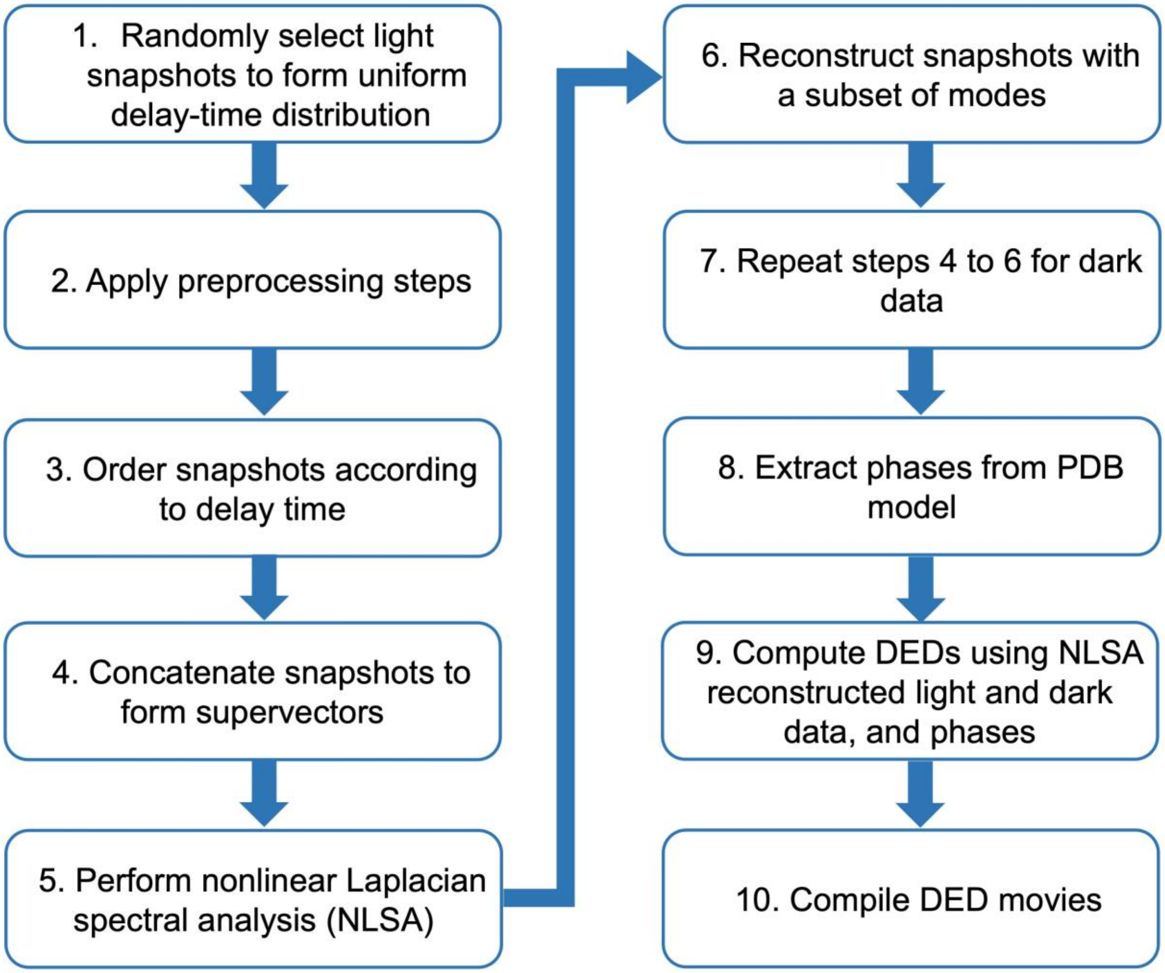
Finally, $t_{start} = \tilde{t}_1 + \left(p - \frac{c-1}{2}\right)\Delta t$. [16]

For the results presented in this paper, $p = c = 32768$, and $\tilde{t}_1 = 164.24fs$. The start time of our reconstructed movies is therefore:

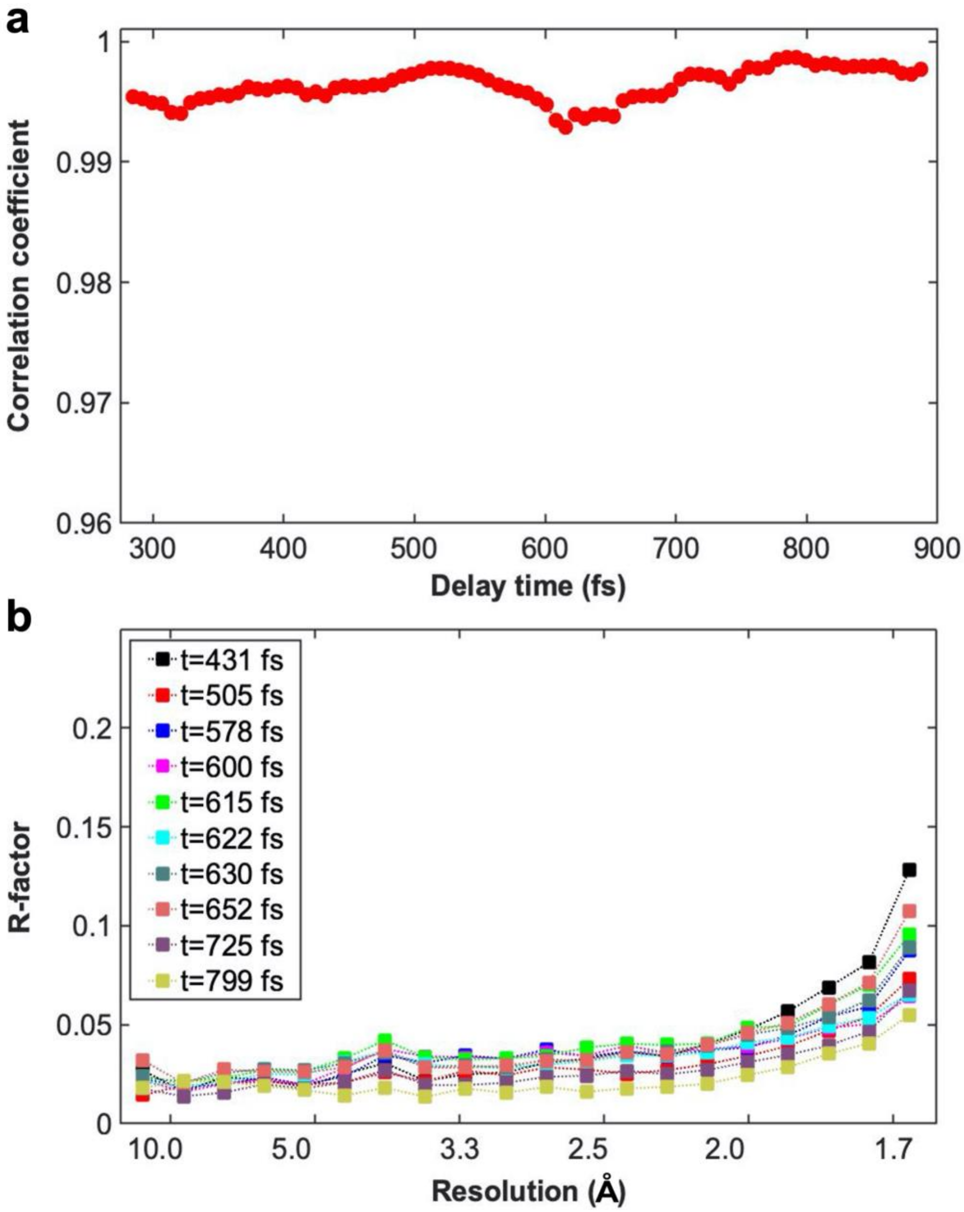
$$t_{start} = 164.24fs + \left(32768 - \frac{32768-1}{2}\right) 7.35as = 284.67fs\ . \quad [17]$$



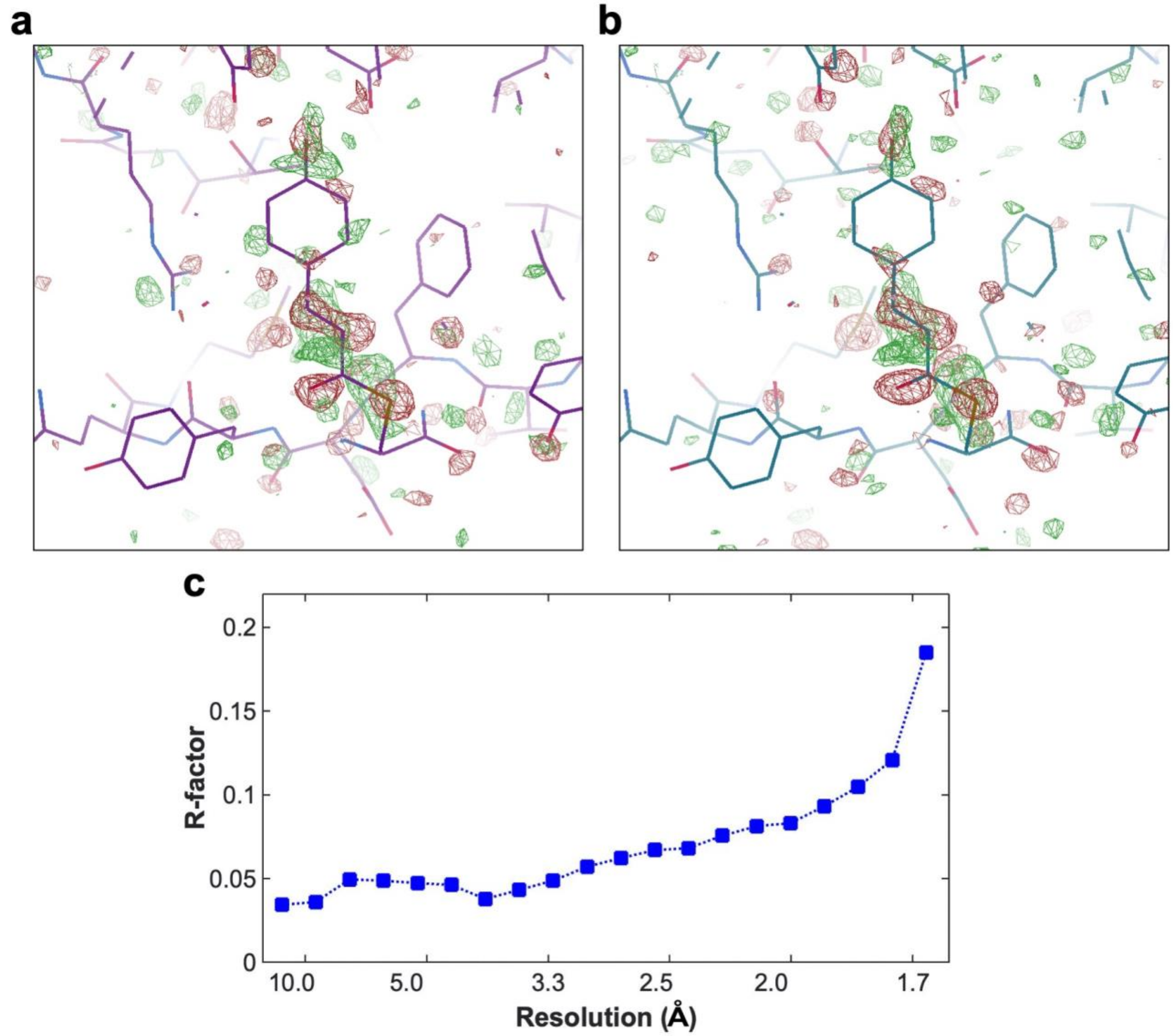
Supplementary Fig. 1 | Histograms of the snapshot delay times. a, Outcome of experiment. **b,** After random subsampling of the experimental data to obtain a statistically uniform distribution in delay time.


1. Randomly select light snapshots to form uniform delay-time distribution
2. Apply preprocessing steps
3. Order snapshots according to delay time
4. Concatenate snapshots to form supervectors
5. Perform nonlinear Laplacian spectral analysis (NLSA)
6. Reconstruct snapshots with a subset of modes
7. Repeat steps 4 to 6 for dark data
8. Extract phases from PDB model
9. Compute DEDs using NLSA reconstructed light and dark data, and phases
10. Compile DED movies

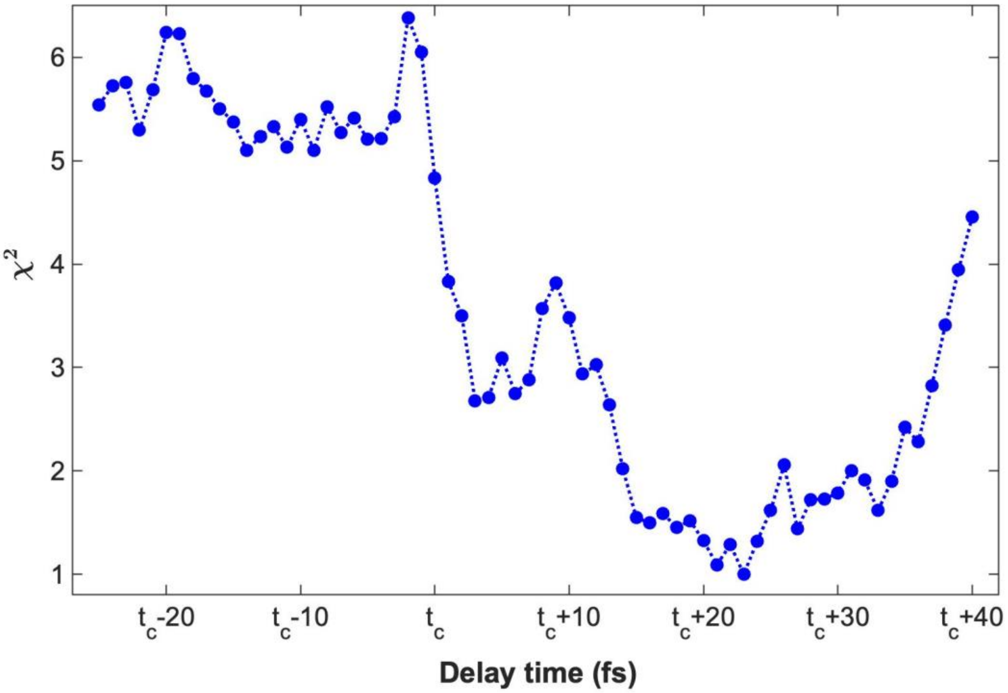
Supplementary Fig. 2 | Flowchart of the analytical pipeline.



Supplementary Fig. 3 | Pearson correlation and R-factor between synthetic (input) and output diffraction volumes obtained from step 7 of Supplementary Fig. 2. **a,** Correlation. The average of correlation coefficients is 0.996. **b,** R- factor. Diffraction volumes in both cases were reconstructed using all non-noise NLSA modes.



Supplementary Fig. 4 | Comparing difference electron density maps at 3 ps delay obtained by: **a**, Standard time-resolved crystallographic analysis; **b**, Machine learning algorithm used in this paper. Contour level for both maps: 3σ. **c**, R-factor between the diffraction volumes at 3ps obtained by standard crystallographic approaches and that obtained by the analytical pipeline in this paper.



Supplementary Fig. 5 | χ^2 landscape of a typical best-fit, in this case for the mode2-mode5 combination, for different trajectory segments. The index t_C refers to the center of the 100-fs timespan, which corresponds to the turning point in chronos.