Supplemental Document

Optics EXPRESS

High-speed processing of X-ray wavefront marking data with the Unified Modulated Pattern Analysis (UMPA) model: supplement

FABIO DE MARCO,[1,2,*] SARA SAVATOVIĆ,[1,2] RONAN SMITH,[3] VITTORIO DI TRAPANI,[1,2] MARCO MARGINI,[1,2] GINEVRA LAUTIZI,[1,2] AND PIERRE THIBAULT[1,2]

[1] *Department of Physics, University of Trieste, Via Valerio 2, 34127 Trieste, Italy*
[2] *Elettra-Sincrotrone Trieste, Strada Statale 14 – km 163.5, 34149 Basovizza, Italy*
[3] *Department of Physics, University of Southampton, University Road, Southampton SO17 1BJ, UK*
[*] *fabiodomenico.demarco@units.it*

Supplement DOI: https://doi.org/10.6084/m9.figshare.21684920

Parent Article DOI: https://doi.org/10.1364/OE.474794

High-speed processing of X-ray wavefront marking data with the Unified Modulated Pattern Analysis (UMPA) model: supplemental document

1. PARTIAL OPTIMIZATION OF THE UMPA COST FUNCTION

This section contains the mathematical details for minimizing the two versions of the UMPA cost function (with and without dark-field) with respect to T, and both T and D, respectively. This is an intermediate step in the procedure of global cost function minimization.

A. Partial optimization for the model without dark-field

When setting $D = 1$ in Eq. (1), the cost function in Eq. (2) (in the main text) can be rewritten as

$$L^{(T)} = T^2 \cdot l_3 - 2T \cdot l_5 + l_1, \tag{S1}$$

where

$$\begin{aligned} l_1 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_m^2(\mathbf{r}+\mathbf{w}-\mathbf{S}_m), \\ l_3 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_{0,m}^2(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}), \\ l_5 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_{0,m}(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}) I_m(\mathbf{r}+\mathbf{w}-\mathbf{S}_m). \end{aligned} \tag{S2}$$

Since, for a local minimum, $\partial L_T / \partial T = 2T \cdot l_3 - 2l_5 = 0$,

$$\hat{T} = l_5 / l_3. \tag{S3}$$

Reinserting Eq. (S3) into Eq. (S1), this yields

$$\hat{L}^{(T)} = l_1 - l_5^2 / l_3. \tag{S4}$$

Thus, for each estimate of $\mathbf{u}$, the terms l_1, l_3, and l_5 are calculated, and from these, the cost and the transmittance estimates are calculated according to Eq. (S3), Eq. (S4). An equivalent calculation can be performed for the model including dark-field, as shown below.

B. Partial optimization for the model with dark-field

Here we show the equivalent of the calculation in subsection A for the model including dark-field. We substitute the fit variables T, D by the quantities $\alpha = TD$, $\beta = T(1-D)$ and different summation terms by $l_1, \ldots, l_6$.

$$\begin{aligned} L^{(T,D)}(\mathbf{r};\mathbf{u},T,D) &= l_1 + \beta^2 l_2 + \alpha^2 l_3 - 2\beta l_4 - 2\alpha l_5 + 2\alpha\beta l_6, \\ \text{where} \qquad l_1 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_m^2(\mathbf{r}+\mathbf{w}-\mathbf{S}_m), \\ l_2 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) \langle I_{0,m}\rangle^2(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}), \\ l_3 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_{0,m}^2(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}), \\ l_4 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) \langle I_{0,m}\rangle(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}) I_m(\mathbf{r}+\mathbf{w}-\mathbf{S}_m), \\ l_5 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) I_{0,m}(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}) I_m(\mathbf{r}+\mathbf{w}-\mathbf{S}_m), \\ l_6 &= \sum_{m,\mathbf{w}} \Gamma(\mathbf{w}) \langle I_{0,m}\rangle(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}) I_{0,m}(\mathbf{r}+\mathbf{w}-\mathbf{S}_m-\mathbf{u}). \end{aligned} \tag{S5}$$

The terms l_i depend on $\mathbf{r}$, as well as on $\mathbf{u}$. Note that l_1, l_3 and l_5 are identical to the ones used by the model without dark-field. These six terms are thus calculated for each new estimate of shift $\mathbf{u}$. As before, we know that for a local minimum of $L^{(T,D)}$, its first derivative with respect to T and D, and thus also with respect to α and β, is zero:

$$\left.\frac{\partial L^{(T,D)}}{\partial \alpha}\right|_{\substack{\alpha=\hat\alpha,\\ \beta=\hat\beta}} = 2\hat\alpha l_3 - 2l_5 + 2\hat\beta l_6 = 0, \quad \left.\frac{\partial L^{(T,D)}}{\partial \beta}\right|_{\substack{\alpha=\hat\alpha,\\ \beta=\hat\beta}} = 2\hat\beta l_2 - 2l_4 + 2\hat\alpha l_6 = 0. \tag{S6}$$

This can be expressed as a matrix multiplication, which also reveals the solution:

$$\begin{pmatrix} l_3 & l_6 \\ l_6 & l_2 \end{pmatrix} \begin{pmatrix} \hat\alpha \\ \hat\beta \end{pmatrix} = \begin{pmatrix} l_5 \\ l_4 \end{pmatrix} \Rightarrow \begin{pmatrix} \hat\alpha \\ \hat\beta \end{pmatrix} = \begin{pmatrix} l_3 & l_6 \\ l_6 & l_2 \end{pmatrix}^{-1} \begin{pmatrix} l_5 \\ l_4 \end{pmatrix} = \frac{1}{l_3 l_2 - l_6^2} \begin{pmatrix} l_2 l_5 - l_4 l_6 \\ l_3 l_4 - l_5 l_6 \end{pmatrix}. \tag{S7}$$

Finally, we can retrieve T and D from α and β via:

$$\hat T = \hat\alpha + \hat\beta, \; \hat D = \hat\alpha / (\hat\alpha + \hat\beta). \tag{S8}$$

Inserting these values for T and D in Eq. (S5) yields the cost function value as a function of only the differential shifts:

$$\hat L^{(T,D)}(\mathbf{r};\mathbf{u}) = L^{(T,D)}(\mathbf{r};\mathbf{u},\hat T,\hat D). \tag{S9}$$

2. UMPA CODE EXAMPLE

Below is a use example for the new UMPA implementation, followed by an explanation of individual steps:

```python
import numpy as np
import UMPA as u
sample = np.load("/path/to/sample.npy")
ref    = np.load("/path/to/ref.npy")
grid   = np.load("/path/to/grid.npy")
mask   = np.load("/path/to/mask.npy")
roi    = np.s_[50:-50:2,50:-50:2]
m = u.model.UMPAModelDF(sample, ref, pos_list=grid, mask_list=mask,
window_size=4, max_shift=5, ROI=roi)
m.assign_coordinates = "sam"
res = m.match(num_threads=16)
mb = u.model.UMPAModelDF(ref, ref, pos_list=grid, mask_list=mask,
window_size=4, max_shift=5, ROI=roi)
mb.assign_coordinates = "sam"
resb = mb.match(num_threads=16)
ux, uy = res["dx"] - resb["dx"], res["dy"] - resb["dy"]
```

The variable `sample` holds a stack of images (NumPy arrays) with diffuser and sample, while `ref` holds the image stack with only the diffuser in the beam. These image volumes should be contiguous in memory and in the double-precision floating-point (`numpy.float64`) data type so that they are correctly interpreted by the C++ routines. However, the module can also easily be compiled to use the single-precision floating point (`numpy.float32`) data type instead. The `grid` variable (used below for the parameter `pos_list`) is only required for sample-stepping measurements, where the sample is moved laterally instead of the diffuser (introduced in subsection 4.1 in the main text), and contains the (x, y) positions of the sample motor stage, in multiples of the effective pixel size. The `mask` variable (used as parameter `mask_list`) can optionally be used to selectively exclude data (e.g., bad pixels) for use with UMPA. It must be a NumPy array of the same shape as `sample` and `ref`. If it holds non-binary values, it is instead used as an additional per-pixel weighting factor for the UMPA cost function. The `ROI` parameter is also optional and can be used to restrict the range of processed pixel values. While the `mask_list` parameter is applied relative to detector coordinates, `ROI` is applied relative to sample coordinates, which is an important distinction when the sample-stepping mode is used. These two parameters are discussed in greater detail in subsection 4.5 in the main text.

Lines 8–9 create a model object which holds references to the image data, as well as processing parameters. Two models are available: `UMPAModelDF` includes dark-field, and `UMPAModelNoDF` does not. The parameter `window_size` represents the variable N from sections 2 and 3 from the main text, while `max_shift` is an upper threshold for the absolute value of the shifts (u_x, u_y) before the discrete minimization is interrupted.

Line 10 sets the value of the `assign_coordinates` parameter, which determines the position to which a determined match between sample and reference analysis window is assigned: For `assign_coordinates="ref"`, it is assigned to the center of the reference analysis window, while for `assign_coordinates="sam"`, it is assigned to the center of the sample analysis window. This is discussed in more detail in subsection 4.4 in the main text.

Finally, the actual minimization procedure is executed in line 11, with the number of threads being controlled by the `num_threads` parameter. Upon completion, the `res` variable holds a Python dictionary containing the four image modalities (attenuation T, horizontal and vertical analysis window shifts (u_x, u_y) in pixels, and dark-field D), as well as the minimized cost function value $[\hat{L}^{(T,D)}(\mathbf{r}; \mathbf{u})$ or $\hat{L}^{(T)}(\mathbf{r}; \mathbf{u})]$ for each pixel.

Lines 12–15 are essentially a repetition of lines 8–11, except that the reference image stack is now compared with itself. This step is beneficial for estimating a bias in the calculation of the shifts `ux`$=\hat{u}_x$, `uy`$=\hat{u}_y$ (line 16). This step is explained in more detail in subsection 4.3 in the main text.

3. DERIVATION OF THE INTERPOLATION KERNEL $B \star B$

It can be shown that all cost function terms can be expressed as sums of cross-correlations of two images.

$$\begin{aligned} C_{DE}(\mathbf{r}; \mathbf{u}) &= \sum_{m=1}^{M} \sum_{w_x=-N}^{N} \sum_{w_y=-N}^{N} \underbrace{\Gamma(\mathbf{w}) D_m(\mathbf{r}+\mathbf{w})}_{=\tilde{D}_m^{(\mathbf{r})}(\mathbf{r}+\mathbf{w})} E_m(\mathbf{r}+\mathbf{w}+\mathbf{u}) \\ &= \sum_{m=1}^{M} \sum_{w_x=-\infty}^{\infty} \sum_{w_y=-\infty}^{\infty} \tilde{D}_m^{(\mathbf{r})}(\mathbf{r}+\mathbf{w}) E_m(\mathbf{r}+\mathbf{w}+\mathbf{u}) = \sum_{m=1}^{M} \left[\tilde{D}_m^{(\mathbf{r})} \star E_m \right](\mathbf{u}). \end{aligned} \tag{S10}$$

The summation range can be formally extended to infinity because $\Gamma(\mathbf{w})$, and thus $\tilde{D}_m^{(\mathbf{r})}(x, y)$, is zero outside of the original summation range. To reproduce the cost function terms $l_1, \ldots, l_6$, D_m and E_m are set equal to either $I_{0,m}$, I_m, or $\langle I_{0,m} \rangle$.

The following calculation assumes that the function being interpolated is a summation of such terms. However, this is not the case for the cost functions already minimised for T (see Eq. S4), or T and D (Eq. S7, S8 inserted in Eq. S5). Since these equations are non-linear combinations of the cost function terms, convolution with the found kernel is not strictly equivalent to a cost function calculation of bi-linearly interpolated images. Furthermore, the following implies that not D and E are being interpolated, but $\tilde{D} = \Gamma D$ and E. This is not quite equivalent to the "natural" way of calculating cross-correlations of interpolated data, i.e., interpolating Γ, D, and E separately, and calculating the cost function terms from these. However, we have not been able to find an equivalent expression of this that uses only a single convolution.

C_{DE} is only defined for integer values of u_x and u_y, but a continuous form of this which is defined for non-integer shifts can be derived by bilinear interpolation. If we assume D and E to be defined on $\mathbb{R}^2$, e.g. as a grid of Dirac impulses:

$$D(\mathbf{r}) = \sum_{i,j} D_{ij}\delta(\mathbf{r} - [i,j]^T), \tag{S11}$$

we can express bilinear interpolation as a convolution of D with the bilinear interpolation kernel B:

$$\widehat{D}(\mathbf{r}) = (D \otimes B)(\mathbf{r}) = \iint d^2\mathbf{r}' D(\mathbf{r}')B(\mathbf{r} - \mathbf{r}'). \tag{S12}$$

By extension, the cross-correlation function of the two interpolated functions $\widehat{D}$, $\widehat{E}$ then is

$$\begin{aligned}(\widehat{D} \star \widehat{E})(\mathbf{r}) = [(D \otimes B) \star (E \otimes B)](\mathbf{r}) = (D \otimes B)(-\mathbf{r}) \otimes (E \otimes B)(\mathbf{r}) = \\ D(-\mathbf{r}) \otimes B(-\mathbf{r}) \otimes E(\mathbf{r}) \otimes B(\mathbf{r}) = D(-\mathbf{r}) \otimes E(\mathbf{r}) \otimes B(-\mathbf{r}) \otimes B(\mathbf{r}) = \\ [D \star E](\mathbf{r}) \otimes [B \star B](\mathbf{r}).\end{aligned} \tag{S13}$$

The above calculation uses the fact that $(f \star g)(\mathbf{r}) = f(-\mathbf{r}) \otimes g(\mathbf{r})$ (for real-valued f), that $(f \otimes g)(-\mathbf{r}) = f(-\mathbf{r}) \otimes g(-\mathbf{r})$, and that $f \otimes g = g \otimes f$. In short, the cross-correlation of the bilinear-interpolated versions of D and E can be derived from their discrete-domain cross-correlation by convolution with the kernel $B \star B$ (i.e., the autocorrelation of B). Since

$$B(\mathbf{r}) = \Lambda(r_x)\Lambda(r_y), \Lambda(x) = \begin{cases} 1 + x, & -1 \leq x \leq 0, \\ 1 - x, & 0 \leq x \leq 1, \\ 0 \text{ else.} \end{cases} \tag{S14}$$

it follows that

$$(B \star B)(\mathbf{r}) = (\Lambda \star \Lambda)(r_x) \cdot (\Lambda \star \Lambda)(r_y). \tag{S15}$$

It is thus sufficient to solve the one-dimensional cross-correlation problem:

$$(\Lambda \star \Lambda)(x) = \int_{-\infty}^{\infty} dx' \Lambda(x')\Lambda(x' + x). \tag{S16}$$

It is evident from Eq. (S14) that $(\Lambda \star \Lambda)(x) = 0$ for $|x| > 2$, and that, since Λ is symmetric, $\Lambda \star \Lambda$ is symmetric as well. For the remaining cases, Eq. (S16) can be solved by splitting it into intervals according to the cases in Eq. (S14). For $0 \leq x \leq 1$, these intervals are $[x-1, 0]$, $[0, x]$, and $[x, 1]$:

$$\begin{aligned}(\Lambda \star \Lambda)(x) &= \int_{x-1}^{0} (1 + x')(1 + x' - x)dx' + \int_{0}^{x} (1 - x')(1 + x' - x)dx' \\ &+ \int_{x}^{1} (1 - x')(1 - x' + x)dx' \\ &= \frac{1}{6}\left(3x^3 - 6x^2 + 4\right).\end{aligned} \tag{S17}$$

For $1 \leq x \leq 2$, the only nonzero part of the integral is

$$(\Lambda \star \Lambda)(x) = \int_{x-1}^{1} (1 - x')(1 + x' - x) = \frac{1}{6}\left(-x^3 + 6x^2 - 12x + 8\right) \tag{S18}$$

The cases $-2 \leq x \leq -1$ and $-1 \leq x \leq 0$ are easily derived using the symmetry of the problem, i.e. knowing that $(\Lambda \star \Lambda)(-x) = (\Lambda \star \Lambda)(x)$, we can substitute $x \to -x$, yielding the final result of

$$(\Lambda \star \Lambda)(x) = \begin{cases} \frac{1}{6}(x+2)^3, & -2 \leq x \leq -1, \\ \frac{1}{6}(-3x^3 - 6x^2 + 4), & -1 \leq x \leq 0, \\ \frac{1}{6}(3x^3 - 6x^2 + 4), & 0 \leq x \leq 1, \\ -\frac{1}{6}(x-2)^3, & 1 \leq x \leq 2, \\ 0 \text{ else.} \end{cases} \tag{S19}$$

The shape of this curve is illustrated in Fig. S1.

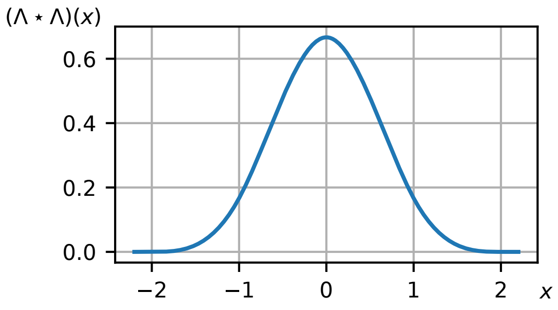

Fig. S1. Curve shape of $(\Lambda \star \Lambda)(x)$, the autocorrelation function of the linear interpolation kernel $\Lambda(x)$. For the calculation of the continuous "cost landscape", the grid of cost function values for discrete-valued shifts $\mathbf{u}$ is convolved with the function $(B \star B)(\mathbf{r}) = (\Lambda \star \Lambda)(r_x) \cdot (\Lambda \star \Lambda)(r_y)$.