Phase retrieval for superposed signals from multiple binary objects

Andreas Alpers,[1,*] Gabor T. Herman,[2] Henning Friis Poulsen,[1] and Søren Schmidt[1]

[1]*Center for Fundamental Research: "Metal Structures in Four Dimensions," Risø-DTU, Technical University of Denmark, DK-4000 Roskilde, Denmark*
[2]*Department of Computer Science, The Graduate Center, City University of New York, New York, New York 10016, USA*
[*]*Corresponding author: awalpers@yahoo.de*

Received January 25, 2010; revised May 30, 2010; accepted July 7, 2010; posted July 9, 2010 (Doc. ID 123299); published August 10, 2010

We introduce the binary superposed phase retrieval problem that aims at reconstructing multiple 0/1-valued functions with nonoverlapping bounded supports from moduli of superpositions of several displaced copies of their individual Fourier transforms. We discuss an application in coherent diffraction imaging of crystalline objects, propose two algorithms, and evaluate their performance by means of simulations. © 2010 Optical Society of America

OCIS codes: 100.5070, 140.2600, 160.4236.

1. INTRODUCTION

Reconstructing a function from the modulus (i.e., absolute magnitudes) of its Fourier transform (FT) constitutes the well-known *phase retrieval* problem: the phase of the complex valued FT has to be retrieved before it can be inverted. This nonlinear inverse problem arises in optics, astronomy, crystallography, and numerous other areas of physics and engineering [1–3] and is known as *binary phase retrieval* problem [4] if the function is 0/1-valued.

A particular imaging application in which objects have successfully been modeled by 0/1-functions is in *incoherent* x-ray diffraction imaging of small crystals (called *grains*) inside polycrystalline materials [5–7]. *Coherent diffraction imaging* (CDI) of multiple grains however has not yet been accomplished, while non-crystalline nanostructures [8–10] and single nanoparticles [11,12] have successfully been reconstructed. In CDI of single grains, there is a direct relation between the diffraction pattern and the FT modulus of the grain's shape function [13]. Experience with incoherent x-ray diffraction from multiple grains suggests that diffracted signals from different grains superpose, and that the modulus of this superposition can be measured on a detector. While the development of a detailed imaging model is left for a future paper, we show that "standard" phase retrieval algorithms (such as Algorithm II introduced below) are not well suited for reconstructing objects from highly superposed data. Our main contribution is Algorithm I, which yields reconstructions from superposed data of superior quality compared to our standard phase retrieval algorithm. Our presentation is deliberately kept rather general, because we feel that our algorithms are of independent interest in other (yet unexplored) application areas in which phase retrieval problems of superposed signals from binary objects occur.

We define the following generalization of the binary phase retrieval problem; to our knowledge it has not been considered in the literature before: the reconstruction of multiple 0/1-valued functions with nonoverlapping bounded supports from moduli of superpositions of several displaced copies of their individual FTs. In other words, the two-dimensional (2D) version of the problem is the following (generalizations to higher dimensions are straightforward): for given G, $H \in \mathbb{N}$ (throughout this paper $\mathbb{N}$ denotes the set of positive integers) and *displacement vectors* $\mathbf{t}_{gh} \in \mathbb{R}^2$ (with $g=1,\ldots,G$ and $h=1,\ldots,H$) the aim is to reconstruct the functions $f_1:\mathbb{R}^2 \to \{0,1\},\ldots,f_G:\mathbb{R}^2 \to \{0,1\}$ with nonoverlapping bounded supports from the knowledge of $|\Sigma_{g=1}^{G}\Sigma_{h=1}^{H}\widehat{f_g}(\cdot-\mathbf{t}_{gh})|$, where $\widehat{f_1},\ldots,\widehat{f_G}$ denote the FTs of $f_1,\ldots,f_G$, respectively. We call this variant the *binary superposed phase retrieval* problem. For $G=H=1$, we obtain as special case the binary phase retrieval problem for a single object. However, we do not place any restrictions on the displacement vectors other than that they should be known, and thus the problem usually does not reduce to the binary phase retrieval problem.

Binary superposed phase retrieval poses major algorithmic challenges. Standard phase retrieval algorithms, in fact, aim at reconstructing a single function from its FT modulus. Thus only a *superposed* function would be returned as the output from such an algorithm that is applied to superposed FT modulus data. We aim, however, at reconstructing the individual binary functions—and this is the major algorithmic challenge.

Algorithms I and II introduced in this paper are the first algorithms for binary superposed phase retrieval. Algorithm I is a Monte Carlo method, while Algorithm II is based on the *hybrid-input-output* (HIO) algorithm [14]. Not surprisingly, Algorithm I is far superior than Algorithm II in terms of reconstruction quality (see Section 5). While Algorithm I aims at reconstructing the individual

functions simultaneously, Algorithm II proceeds sequentially. We remark that Algorithm I is more time efficient than other comparable Monte Carlo methods, because we make use of the fact that the functions to be reconstructed are binary.

Details of Algorithm I are discussed in Sections 2 and 3. Here we remark only that Monte Carlo methods in imaging applications follow the paradigm of simulating Markov chains; usually only the type of prior information and the forward operator relating the image to the measurements differ. One ends up with an algorithm that is useless for practical applications if the priors and operators cannot be implemented effectively. Our particular forward operator, however, can be implemented efficiently, as we discuss in Section 3. For the (different) forward operator used for phase retrieval of non-crystalline objects, Monte Carlo methods were developed in the 1980s [15–17]. With the exception of [18], they seem to have been abandoned in favor of more effective methods, such as the HIO algorithm.

In the application area of CDI of multiple grains, several experimental imaging methods—such as holography or *ab initio* phase retrieval for the complete oversampled diffraction pattern [19]—are possible. However, experimental concerns (signal/noise and sampling issues) imply that robust reconstructions could be considerably facilitated by incorporating additional prior information. The approach that we propose incorporates diffraction data from both incoherent and coherent x-rays. Additional information in this case is that (1) the grains are binary objects (we consider the general grain shape and not the individual atom positions, as we discuss in Subsection 2.B) and (2) from existing x-ray methodology for incoherent beams we can obtain (partial) low-resolution maps [6,7,20–22], which we propose to use as prior knowledge for algorithms that aim at reconstructing objects from coherent x-ray diffraction data (see Fig. 1). Our algorithms are capable of including such prior knowledge.

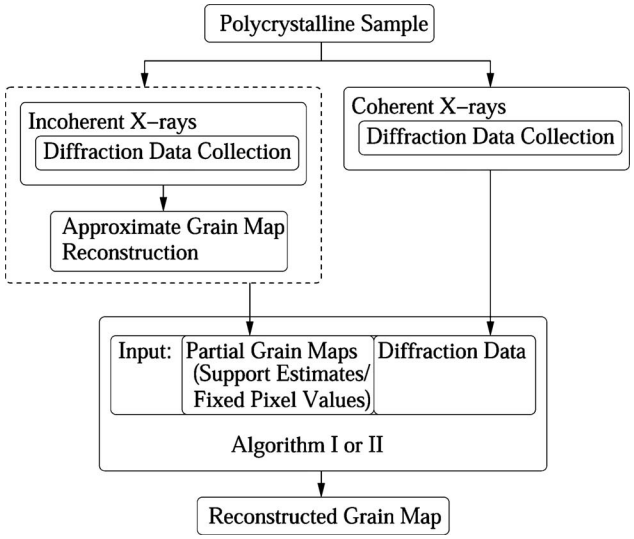

Fig. 1. Schematic of grain map reconstruction from CDI data. In addition to CDI data, we propose that incoherent x-ray diffraction images are acquired to provide partial grain maps as input for Algorithms I and II. Note, however, that Algorithms I and II do not strictly require such input data.

We remark that grains are of central interest in many branches of materials science, because polycrystalline materials (such as metals, alloys, and ceramics) are composed of multiple grains, which in turn determine the material's physical, chemical, and mechanical properties. Our results show that, under simplifying assumptions, CDI of multiple grains is possible with Algorithm I. Thus, CDI is potentially a vital imaging tool for materials science.

2. BINARY SUPERPOSED PHASE RETRIEVAL

A. Our General Inverse Problem

We define the binary superposed phase retrieval problem and discuss a relevant application in simplified terms. We repeat from the introduction the definition of the 2D *binary superposed phase retrieval problem*: For given G, $H \in \mathbb{N}$ and *displacement vectors* $\mathbf{t}_{gh} \in \mathbb{R}^2$ (with $g = 1,\dots,G$ and $h = 1,\dots,H$), the aim is to reconstruct the functions $f_1:\mathbb{R}^2 \to \{0,1\},\dots,f_G:\mathbb{R}^2 \to \{0,1\}$ with nonoverlapping bounded supports from the knowledge of $|\Sigma_{g=1}^{G}\Sigma_{h=1}^{H}\widehat{f_g}(\cdot - \mathbf{t}_{gh})|$. This is usually an ill-posed inverse problem since the binary phase retrieval subcase is already ill-posed. It is well known [23] that there are trivial ambiguities in the phase retrieval problem; for instance, it is impossible to determine the absolute position of f_g, because $f_g(\cdot - \mathbf{t})$ with an arbitrary $\mathbf{t}$ has the same FT modulus as f_g. However, such ambiguities can often be resolved if prior information is available (see, e.g., [24]). The main prior information utilized in our case is provided by the availability of partial reconstructions obtained by other imaging methods and the knowledge that the function values are 0/1 (the effectiveness of such binary constraints has been demonstrated in [4]). In the next subsection we discuss an application for which our assumptions are largely justified.

We define the *continuous color map* $f:\mathbb{R}^2 \to \{0,1,\dots,G\}$ with

$$f(\mathbf{x}) = \begin{cases} g, & \text{if there is a } g \in \{1,\dots,G\} \text{ with } f_g(\mathbf{x}) = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Obviously, we can reverse this procedure by decomposing f into $f_1,\dots,f_G$; thus reconstructing the individual functions is equivalent to reconstructing the continuous color map.

In order to deal with the reconstruction problem on a computer, we need to discretize it. Following [[25], Section 2] we define, for $N \in \mathbb{N}$ and positive *sampling distance* $d \in \mathbb{R}$, the discretization $f^{N,d}:[0,N-1]^2 \to \{0,1,\dots,G\}$ of f by

$$f^{N,d}(y_1,y_2) = f\left(-\frac{N-2y_2-1}{2}d, \frac{N-2y_1-1}{2}d\right). \tag{2}$$

In this definition $[0,N-1]^2$ denotes the set of all *pixels* $\mathbf{y}^T = (y_1,y_2)$, where the nonnegative integers y_1 and y_2 are less than N. Typically N and d would be chosen large enough so that $f(\mathbf{x}) = 0$ if $\mathbf{x}^T = (x_1,x_2)$ and $\max\{|x_1|,|x_2|\} \geq Nd/2$. We refer to such an $f^{N,d}$ in Eq. (2) as an $N \times N$

color image. Our algorithms aim at reconstructing such color images.

Since usually knowledge of N and d can be assumed or is not needed, we sometimes simplify the notation by using $\bar{f}$ for a discretized version of a function f. Motivated by the application, in the next subsection, we refer to discretizations of the data $|\Sigma_{g=1}^{G}\Sigma_{h=1}^{H}\widehat{f_g}(\cdot-\mathbf{t}_{gh})|$ as *detector data*.

To help the reader distinguish between continuous and discrete concepts, we use $\mathbf{x}$ (position in real space), $\mathbf{u}$ (vector in Fourier space), and $\mathbf{p}$ (position on a detector) throughout this paper to denote real vectors. The corresponding discrete counterparts are denoted by $\mathbf{y}$ (pixel in real space), $\mathbf{v}$ (vector in discretized Fourier space), and $\mathbf{q}$ (pixel on a detector). We note that, for any $f_g:\mathbb{R}^2\to\{0,1\}$, the discrete Fourier transform (DFT) $\widehat{f_g^{N,d}}$ is an approximation of the discretization of the FT $\widehat{f_g}$ with the same N but with a sampling distance $1/Nd$. Hence, for any fixed d, we can make the sampling distance in Fourier space as small as we wish by choosing N sufficiently large (this is usually referred to as *zero-padding*).

B. Application: Coherent Diffraction Imaging of Multiple Undeformed Grains

The aim in this application is to reconstruct collections of grains (small crystals) inside bulk materials from (monochromatic) coherent x-ray diffraction images. The term "collection" means in our context that the relative positions of the grains are relevant. Typical collections are space-filling—for example, if the collection represents a sample from a *polycrystalline material*—but generally voids might be present, too. Grains are naturally three-dimensional (3D) objects, but by considering 2D slices through the sample we restrict our attention to the 2D case.

Initially, two aspects of a grain are relevant: (1) the positions in space occupied by the grain and (2) the internal lattice structure specifying the positions of the grain's unit cells. Signal/noise ratios and detector technology, however, imply that one cannot aim at atomic resolution for grain imaging of bulk materials (see, e.g., [26] for an unsuccessful attempt in electron microscopy). One thus needs to make assumptions about the imaged material. In our case, we assume that the grain lattice is perfect, which means that the grains are undeformed (see [6,7] for an imaging application in incoherent x-ray diffraction); deformed grains are outside the scope of this paper.

To date, collections of grains have been reconstructed only from incoherent x-ray diffraction data, but higher-resolution images are expected to be obtained from coherent x-ray diffraction data. To facilitate robust reconstructions from coherent x-ray diffraction data, we propose incorporating additional diffraction images from incoherent x-ray diffraction data into the reconstruction process (see Fig. 1). In fact, the lattice of undeformed grains can be inferred already from incoherent x-ray diffraction data [27,28], and the reconstruction task from coherent x-ray diffraction data thus reduces to the task of determining for each grain the positions it occupies in space. Figure 4(e) shows a 2D collection of grains; the internal lattice structure is hidden.

Referring to the global sample coordinate system, we can describe each grain, labeled by a number $g\in\{1,\dots G\}$, by two parameters f_g and $\mathcal{B}_g$, where $f_g:\mathbb{R}^2\to\{0,1\}$, the so-called *shape function*, is a function whose (necessarily bounded) support in the sample is occupied by the grain, and $\mathcal{B}_g\subseteq\mathbb{R}^2$ is a set of two basis vectors defining the internal 2D lattice structure. A collection of G grains is thus specified by $f_1,\dots,f_G$ and $\mathcal{B}_1,\dots,\mathcal{B}_G$. Positions in real-world samples are occupied by at most one grain, and thus f in Eq. (1) and its discretized version $f^{N,d}$ in Eq. (2), called the *grain map* in this application, are well-defined. For short, we refer to a grain labeled g as grain g; note that this is also the color of the pixels occupied by that grain.

1. Forward Model

In the following we briefly describe our forward model, used to obtain from G, H, and f_g, $\mathcal{B}_g$, $\mathbf{t}_{gh}$, for $g=1,\dots,G$, $h=1,\dots,H$, the corresponding coherent x-ray diffraction image measured on a detector. For a single grain g, it is well known [13] that measured detector intensities are proportional to the squared modulus of superpositions of FTs of shape functions f_g, and displacement vectors $\mathbf{t}_{gh}$ correspond to the reciprocal lattice positions (Bragg peaks) of the grain. The theory of coherent x-ray diffraction from multiple grains has not yet been developed in detail, but it is plausible that the signals (for Bragg diffraction) from multiple grains superpose with a subsequent phase loss owing to the detector reading. It is well known that there is a one-to-one correspondence between grain lattices and their reciprocal lattices. The one-to-one correspondence is given—if we model both lattices by sums of delta distributions—by the FT [29,30]. Thus from $\mathcal{B}_g$ we can compute $\mathbf{t}_{gh}$, for $h=1,\dots,H$.

Fourier space can be probed by a properly aligned detector. If the detector had an arbitrarily fine resolution, we could measure values proportional to $|\Sigma_{g=1}^{G}\Sigma_{h=1}^{H}\widehat{f_g}(\cdot-\mathbf{t}_{gh})|^2$. We are free to set the proportionality factor involved to 1, because in practice one observes that the technical detector implementation introduces yet another factor. Taking square roots shows that the measured data are exactly of the form stated in the binary superposed phase retrieval problem. The detector, however, has a finite resolution. Thus an additional discretization step is involved, which is discussed in Subsection 3.B.

2. Inverse Problem

Assuming, as previously discussed, that G, H, and $\mathcal{B}_g$, $\mathbf{t}_{gh}$, for $g=1,\dots,G$, $h=1,\dots,H$, are known, the inverse problem in this application is to reconstruct the grain map $\bar{f}$ ($=f^{N,d}$, for suitable selected N and d). We have discussed in the previous section that this is a binary superposed phase retrieval problem.

3. ALGORITHM I: MONTE CARLO APPROACH

Our first method is a Monte Carlo method. Similar algorithms have been developed for the reconstruction of grain maps from tomographic diffraction data that have been acquired by *incoherent* x-rays [6,7,20–22]. The main

difference from the algorithms presented in those papers is that here we have to use a totally dissimilar forward operator.

Let $\mathcal{I}$ denote the set of all possible $N\times N$ color images with at most $G+1$ colors, and let $\mathcal{D}$ denote the set of all possible detector data. Elements of these sets can be rearranged into vectors from finite-dimensional real vector spaces, and we thus have norms, such as the ℓ_1-norm $\|\ \|_1$ at our disposal. For $\mathcal{L}\subseteq[0,N-1]^2$ and $f_{init}\in\mathcal{I}$, we define $\mathcal{I}(f_{init},\mathcal{L})$ to be the set of color images $\bar{f}\in\mathcal{I}$ whose pixel values are equal to those of f_{init} at each pixel in $[0,N-1]^2\setminus\mathcal{L}$. The idea behind this notation is that we can think of f_{init} as a partial image reconstruction obtained by other methods and of $\mathcal{L}$ as being the set of the so-called *ambiguous pixels*, whose values are to be reconstructed with our method. We explicitly allow $\mathcal{L}=[0,N-1]^2$, which amounts to $\mathcal{I}(f_{init},[0,N-1]^2)=\mathcal{I}$. Moreover, we assume that $\mathcal{L}$ is a subset of the union of the supports of $f_1,\ldots,f_G$. It is straightforward to adapt our algorithm to the situation in which this assumption is violated.

Let $\mathbf{P}:\mathcal{I}\to\mathcal{D}$ denote the forward operator mapping each color image $\bar{f}$ to its detector data, and let $\mathbf{P}_0\in\mathcal{D}$ denote the measured detector data. As for many inverse problems, the general idea is to find

$$f^* := \arg\min_{\bar{f}\in\mathcal{I}(f_{init},\mathcal{L})}\|\mathbf{P}(\bar{f})-\mathbf{P}_0\|_1. \tag{3}$$

A technique borrowed from statistical mechanics is to introduce the family of probability distributions $\gamma_\beta:\mathcal{I}(f_{init},\mathcal{L})\to\mathbb{R}_+$ (where $\beta\in\mathbb{R}_+$), with

$$\gamma_\beta(\bar{f})=\frac{1}{Z_\beta}\exp(-\beta\|\mathbf{P}(\bar{f})-\mathbf{P}_0\|_1), \tag{4}$$

where Z_β is a (usually unknown) normalizing constant. Distributions of the form in Eq. (4) are known in the literature as *Gibbs distributions* [31]. We "solve" Eq. (3) by sampling from the distribution γ_β. If $\mathcal{F}$ denotes the set of minimizers in Eq. (3), then one can prove (Proposition 5.2.1 in [31]) that

$$\gamma(\bar{f}) := \lim_{\beta\to\infty}\gamma_\beta(\bar{f})=\begin{cases}1/|\mathcal{F}|, & \text{if } \bar{f}\in\mathcal{F}\\ 0, & \text{otherwise.}\end{cases} \tag{5}$$

In other words, sampling from γ_β at high β yields the minimizers with a high probability. Estimates for β are often hard to obtain and often too weak for practical purposes. In practice, however, it often turns out that one can sample efficaciously if β is chosen reasonably large. We remark in passing that a common technique is to vary β according to some "cooling" scheme, and the resulting algorithm becomes a *simulated annealing algorithm* [32]. We do not follow this approach—in this paper β is fixed.

We have not yet discussed how to sample from γ_β. The *Metropolis algorithm* introduced in [33] achieves exactly this. It simulates a Markov chain and, after a suitably long "burn-in phase," it samples from γ_β. Our Algorithm I, based on the Metropolis algorithm, is outlined in Fig. 2.

We highlight four aspects of Algorithm I:

- only ambiguous image pixels as specified in the input are processed;
- in theory, the initial assignment of colors to the ambiguous pixels in f_{init} is irrelevant—in our implementation we choose a random assignment;
- the algorithm reduces to the original Metropolis algorithm if $\bar{f}$ instead of f^* is returned at termination; and
- the terms $\gamma_\beta(\bar{f}')/\gamma_\beta(\bar{f})$ and $\gamma_\beta(f^*)/\gamma_\beta(\bar{f})$ can be evaluated without knowledge of Z_β.

In summary, the core of Algorithm I is the Metropolis-based algorithm (Fig. 2) applied to γ_β as defined in Eq. (4). In Subsections 3.A and 3.B we discuss how $\mathbf{P}$ is implemented.

A. Fast Updates of the Discrete Fourier Transforms

The key step of the Metropolis algorithm is the proposal of changing the current color assignment to a pixel $\mathbf{y}^*\in[0,N-1]^2$. This affects only two discretized functions, i.e., the value of an $\bar{f}_a(\mathbf{y}^*)$ changes from 0 to 1, while the value of an $\bar{f}_b(\mathbf{y}^*)$ changes from 1 to 0. For the forward

Input: A color image f_{init} with at most $G+1$ colors and a set $\mathcal{L}$ of n ambiguous pixels defining the space $\mathcal{I}(f_{init},\mathcal{L})$ of reconstructible color images.
Output: A color image $f^*\in\mathcal{I}(f_{init},\mathcal{L})$ for which the value of γ is nearly maximal.

```
Choose β and mc_cycles sufficiently large (as discussed in the text);
f* ← f_init and f̄ ← f_init;
for i = 1 to n·mc_cycles do
    Select randomly a candidate pixel q ∈ L and color c ∈ {1,...,G};
    Calculate r = γ_β(f̄')/γ_β(f̄), the ratio of the probabilities of the new color image f̄'
    (which we would obtain if we assigned color c to q) and the old color image f̄;
    Set f̄ ← f̄' with probability min(1, r);
    if γ_β(f*)/γ_β(f̄) < 1 then
        f* ← f̄;
    end if
end for
return f*;
```

Fig. 2. Our general Metropolis-based algorithm for "solving" Eq. (3).

projection we need to update their 2D DFTs. However, we do not have to resort to a full recomputation using a fast Fourier transform (FFT) algorithm. To see this let $\bar{f}_{a,1}$ (respectively, $\bar{f}_{b,0}$) denote the function resulting from $\bar{f}_a$ (respectively, $\bar{f}_b$) after the change, and let $\widehat{f_{a,1}^{N,d}}$ (respectively, $\widehat{f_{b,0}^{N,d}}$) and $\widehat{f_a^{N,d}}$ (respectively, $\widehat{f_b^{N,d}}$) denote the corresponding DFTs, respectively. We immediately obtain

$$\begin{aligned}\widehat{f_{a,1}^{N,d}}(\mathbf{v}) &= \sum_{\mathbf{y}\in[0,N-1]^2} \bar{f}_{a,1}(\mathbf{y})\exp(-2\pi i\mathbf{v}^T\mathbf{y}/N)\\ &= \bar{f}_{a,1}(\mathbf{y}^*)\exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N)\\ &\quad + \sum_{\mathbf{y}\in[0,N-1]^2\setminus\{\mathbf{y}^*\}} \bar{f}_{a,1}(\mathbf{y})\exp(-2\pi i\mathbf{v}^T\mathbf{y}/N)\\ &= \exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N) + \sum_{\mathbf{y}\in[0,N-1]^2} \bar{f}_a(\mathbf{y})\\ &\quad \times\exp(-2\pi i\mathbf{v}^T\mathbf{y}/N)\\ &= \exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N) + \widehat{f_a^{N,d}}(\mathbf{v}),\end{aligned} \tag{6}$$

for all $\mathbf{v}\in[0,N-1]^2$. In the same way, we obtain for the other change that $\widehat{f_{b,0}^{N,d}}(\mathbf{v})=\widehat{f_b^{N,d}}(\mathbf{v})-\exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N)$. We update the DFTs according to these formulas, i.e., we add (or subtract, respectively) the term $\exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N)$ from the previous DFT. Computing the DFTs in this way gives an $O(N^2)$ algorithm—FFT algorithms [34], on the other hand, would require a computation time of the order $O(N^2\log(N^2))$.

To achieve an additional speed-up, one could compute and store the values of $\exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N)$ prior to the execution of the Metropolis algorithm. For memory reasons we decided against storing the values for all N^4 possible $(\mathbf{v}^T,\mathbf{y}^{*T})=(v_1,v_2,y_1^*,y_2^*)$-tuples. In our implementation we store the values of $\exp(-2\pi i v_1 y_1^*/N)$ for all (v_1,y_1^*)-pairs. The evaluation of $\exp(-2\pi i\mathbf{v}^T\mathbf{y}^*/N)$ requires then only one complex multiplication. A further speed-up is discussed in Subsection 3.C.

B. Computation of Simulated Detector Images

In order to calculate the ℓ_1-norm in Eq. (4) we need to estimate $\mathbf{P}(\bar{f})$ for each pixel for which we have a detector reading in $\mathbf{P}_0$. Let us denote the spacing between detector pixel centers (both horizontally and vertically) by δ; this is the sampling distance of the detector data. As explained in the last paragraph of Subsection 2.A, for any $f_g:\mathbb{R}^2\to\{0,1\}$, the discretization of the FT $\widehat{f_g}$ with a sampling distance $\delta'=1/Nd$ can be estimated from $f_g^{N,d}$.

During the execution of the Metropolis algorithm (Fig. 2), we need to calculate $\mathbf{P}(\bar{f})$ for various color images $\bar{f}$. Let us first look at the special simple case when $H=1$ and $\mathbf{t}_{g1}$ is the zero vector, for $g=1,\ldots,G$. Then the detector data are a discretization of $|\Sigma_{g=1}^G\Sigma_{h=1}^H\widehat{f_g}(\cdot-\mathbf{t}_{gh})|=|\Sigma_{g=1}^G\widehat{f_g}(\cdot)|$. We see that, by an appropriate selection of N and d, we can make sure that $\delta/\delta'\in\mathbb{N}$. This means that in this case the values of $\mathbf{P}(\bar{f})$ at the detector pixels can be directly estimated by decomposing $\bar{f}$ and taking DFTs.

Consider now the general case. What we need is a method for estimating at the detector pixels the values of $\widehat{f_g}(\cdot-\mathbf{p})$ for an arbitrary $\mathbf{p}\in\mathbb{R}^2$. As in the previous paragraph, we generate an approximation to a discretization of $\widehat{f_g}(\cdot)$ with sampling distance δ'. By shifting the points at which $\widehat{f_g}(\cdot)$ has been estimated by $\mathbf{p}$, we get estimates at certain points on the detector plane of $\widehat{f_g}(\cdot-\mathbf{p})$. This situation is depicted in Fig. 3: the shaded dots indicate locations of the detector readings and the white dots indicate the places where we have estimates of $\widehat{f_g}(\cdot-\mathbf{p})$. By selecting $\delta/\delta'\in\mathbb{N}$ large enough, we can ensure that for every shaded dot $\mathbf{p}'$ there will be an empty dot $\mathbf{p}^*$ that is near enough to it so that $\widehat{f_g}(\mathbf{p}'-\mathbf{p})$ can be approximated by the known $\widehat{f_g}(\mathbf{p}^*-\mathbf{p})$. Furthermore, and this is important for the computational efficiency, due to the fact that $\delta/\delta'\in\mathbb{N}$, for any $\mathbf{p}$ we need to find only one such pair of points $\mathbf{p}'$ and $\mathbf{p}^*$; for all other detector pixels the contribution of $\widehat{f_g}(\cdot-\mathbf{p})$ can be identified by a simple subsampling (using sampling distance δ) of the discretization of $\widehat{f_g}(\cdot)$ with sampling distance δ'; see Fig. 3. (This technique is similar to the so-called footprint-based texture mapping in computer graphics.) We remark that the process is repeated for several f_g's and $\mathbf{p}$'s. The detector values are not overwritten: all complex values at each detector pixel are summed up. The very last step to obtain the final detector image $\mathbf{P}(\bar{f})$ is to replace each complex valued detector pixel entry with its absolute value.

C. Additional Code Optimizations

Several other speed-ups have been implemented. The following list gives a short overview:

- *Updating of the simulated detector image:* We have discussed in Subsection 3.A how we update the DFTs of a single $\bar{f}_g$. In a similar way we update the whole simulated detector image. We are not only keeping the simulated detector image, i.e., the modulus of the superposed DFTs, but we are also storing the complex values in a different array. Now, if two DFTs are changed in a Metropolis proposal step, then we first subtract both unchanged DFTs (placed at positions as described in Subsection 3.B) from the complex valued array. In a second step, we add the DFTs of the changed DFTs to the complex valued array. Finally, we obtain the simulated detector image by taking

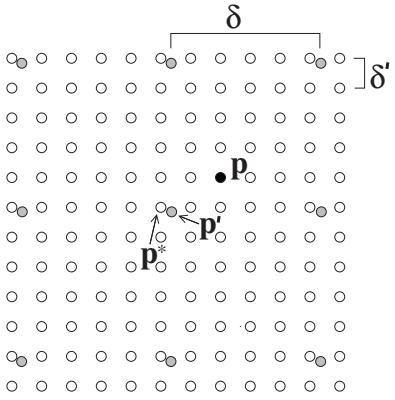

Fig. 3. Computation of $\mathbf{P}(\bar{f})$ (see Subsection 3.B). Detector pixels are depicted as shaded dots. Locations at which estimates, provided by a shifted DFT, are available are depicted as white dots.

the absolute values of these complex numbers. The described updating technique has the advantage that only two DFTs are involved when the simulated detector image is recalculated. The drawback, however, is that rounding errors might accumulate.

- *Avoiding unnecessary reassignments of large data arrays:* At each step of the Metropolis algorithm one needs to compute the simulated detector image for the proposed pixel change. We store these values in a different array, because this permits a quick recovery of the old detector values in case the proposal is rejected. We only need to keep a pointer telling us which of the two arrays contains the current simulated detector image.
- *Look-up tables for frequent computations:* The idea of using look-up tables for frequently occurring computations has already been discussed for the DFT updates. We also use look-up tables in another situation: when the DFTs are copied onto the various places on the detector one needs to ensure that no pixels outside the detector are considered. Instead of performing range checks in each iteration, we store the permissible ranges for all DFTs and displacement vectors in a look-up table. Thus, we need no range checks; we use only loops with variables in the range indicated by the look-up table.

4. ALGORITHM II: ADAPTATION OF PHASE RETRIEVAL ALGORITHMS

In the following we introduce a second method for binary superposed phase retrieval, called *Algorithm II*, that uses as a subroutine a standard phase retrieval method. In the next subsection we discuss the general framework of Algorithm II; details of the subroutine are given in Subsection 4.B.

A. Adaptation

Phase retrieval methods reconstruct single functions from single FTs. In certain cases we can reconstruct $f_1,\dots,f_G$ independently. For example, if for each $g'\in\{1,\dots,G\}$ there is an $h'\in\{1,\dots,H\}$ such that, for all $\mathbf{u}$ in a neighborhood of $\mathbf{t}_{g'h'}$, we have

$$\left|\sum_{g=1}^{G}\sum_{h=1}^{H}\widehat{f_g}(\mathbf{u}-\mathbf{t}_{gh})\right|\approx|\widehat{f_{g'}}(\mathbf{u}-\mathbf{t}_{g'h'})|, \tag{7}$$

then we could try to reconstruct $f_{g'}$ from the data using a phase retrieval method. This motivates the following algorithm, which is our Algorithm II (with an integer parameter pcycles): select for each $g'\in\{1,\dots,G\}$ the $h'\in\{1,\dots,H\}$ whose minimum distance to any other displacement vector $\mathbf{t}_{gh}$ is maximal and reconstruct $f_{g'}$ with pcycle iterations of a phase retrieval algorithm using the data $|\Sigma_{g=1}^{G}\Sigma_{h=1}^{H}\widehat{f_g}(\mathbf{u}-\mathbf{t}_{gh})|$, with $\mathbf{u}$ being in the neighborhood of $\mathbf{t}_{g'h'}$.

Equation (7) is a valid approximation only in special scattering situations in which superposition effects from different FTs are small. Our simulations in Section 5 include several scattering situations, and the results can thus be interpreted as testing whether Algorithm II gives useful results in situations in which Eq. (7) does not strictly hold.

B. HIO

The phase retrieval algorithm that we use as a subroutine in Algorithm II is basically the HIO algorithm introduced in [14]. Our exposition closely follows [35]. We remark that one could replace HIO with any other phase retrieval algorithm, e.g., with the *difference map* [36].

HIO follows the Gerchberg–Saxton [37] scheme for reconstructing a function f from its FT modulus $|\hat{f}|$ by generating a sequence $(s_k)_{k\in\mathbb{N}}$ of functions that approximate f. Starting with an initial guess s_0 (usually randomly chosen; in our case function values are randomly chosen from {0,1}), the iteration step from s_k to s_{k+1} is as follows: (1) the amplitude of the FT of s_k is replaced with $|\hat{f}|$, (2) the inverse FT of this function yields a function s_k', and finally (3) we obtain s_{k+1} from s_k, s_k', and prior information.

Algorithms following the Gerchberg-Saxton scheme differ in the implementation of Step (3). For real nonnegative valued functions f—as in our case—one typically assumes that an estimate $\mathcal{S}_{object}$ of the support of f is available (details for our choice in the simulations are given in Section 5). Let c_k be defined as

$$c_k(\mathbf{x})=\begin{cases}\Re(s_k'(\mathbf{x})), & \text{if } \mathbf{x}\in\mathcal{S}_{object} \text{ and } \Re(s_k'(\mathbf{x}))>0\\ 0, & \text{otherwise,}\end{cases} \tag{8}$$

where $\Re(s_k'(\mathbf{x}))$ denotes the real part of the complex value $s_k'(\mathbf{x})$. The idea is that $s_k'(\mathbf{x})$ violates prior knowledge if $|c_k(\mathbf{x})-s_k'(\mathbf{x})|>\varepsilon$, where ε denotes a prescribed tolerance level (see Subsection 5.B). In HIO we have

$$s_{k+1}(\mathbf{x})=\begin{cases}s_k'(\mathbf{x}), & \text{if } |c_k(\mathbf{x})-s_k'(\mathbf{x})|\le\varepsilon\\ s_k(\mathbf{x})+\beta_{\mathrm{HIO}}(c_k(\mathbf{x})-s_k'(\mathbf{x})), & \text{otherwise,}\end{cases} \tag{9}$$

with a prescribed system parameter β_{HIO} (see Subsection 5.B). Algorithm II thus performs pcycle iterations of HIO; thereafter the reconstructed function values are rounded to 0 or 1 depending on whether or not the real part is smaller than 0.5.

5. SIMULATIONS

In the following we describe five sets of simulations that we performed on a personal computer with a 2.66 GHz Intel Core2Duo processor and 3.23 Gbytes random access memory. The algorithms have been implemented as C programs running under Windows XP. Details of the five simulations are discussed below; a summary is given in Table 1.

A. Data Generation

For our tests we selected three grain maps of aluminum samples that were acquired by electron microscopy, more specifically, by electron backscattered diffraction (EBSD). In what follows we refer to them as Maps I–III. While we aim (in the next subsection) at reconstructing maps $f^{N,d}$, we remark that each EBSD map is a higher-resolution discretization $f^{KN,d/K}$ of a continuous color map f. By introducing K, we ensure that the discretization for the

Table 1. Overview of the Data Generation in the Simulations (See Section 5)

Simulation	Input Map	Size of Input Map ($KN\times KN$)	Detector Dimension ($N_{det}\times N_{det}$)	K	# Spots	D_{best}
1	I	128×128	200×200	2	14	35
2	I	128×128	200×200	2	16	25
3	II	128×128	200×200	2	14	35
4	II	128×128	200×200	2	16	25
5	III	192×192	500×500	3	69	21

simulated data generation is somewhat independent from and is more accurate than the discretization used for the reconstruction. This is a way of avoiding the commitment of the "inverse crime" of using the same model for generating the data for testing an algorithm as the one used in the design of the algorithm. These $f^{KN,d/K}$'s are shown in the left column of Fig. 4. For Maps I and II (containing $G=2$ grains) we have $N=64$ and $K=2$. For Map III (containing $G=11$ grains) we have $N=64$ and $K=3$.

We generate the data (i.e., the moduli of the superposed FTs on the detector) from these maps at the sampling distance $1/Nd$, which is the same as the sampling distance

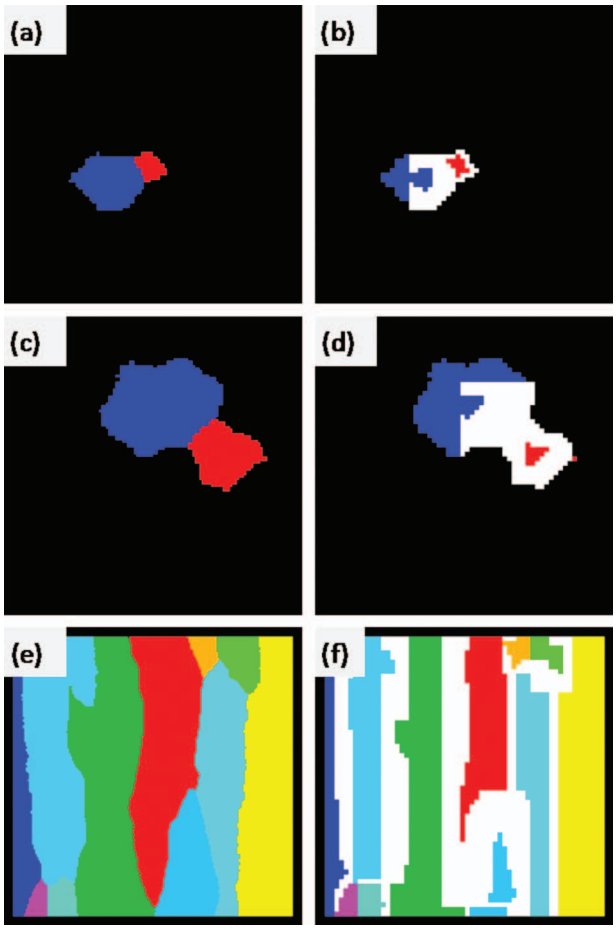

Fig. 4. (a) Map I comprising two grains, (b) corresponding $f^{N,d}$ with $N=64$ and ambiguous pixels (white) for Simulations 1 and 2, (c) Map II comprising two grains, (d) corresponding $f^{N,d}$ with $N=64$ and ambiguous pixels (white) for Simulations 3 and 4, (e) Map III comprising 11 grains, (f) corresponding $f^{N,d}$ with $N=64$ and with ambiguous pixels (white) for Simulation 5.

when we approximate the FT $\widehat{f_g}$ by the DFT $\widehat{f_g^{KN,d/K}}$ (see the last paragraph of Subsection 2.A). However, we make the detector dimension $N_{det}\times N_{det}$ larger than $KN\times KN$ since the detector has to accommodate also the shifted versions of $\widehat{f_g}$. The actual values used in our five simulations are shown in Columns 2–5 of Table 1.

For each grain g we specify a lattice basis $\{\mathbf{a}_g,\mathbf{b}_g\}$ and define the set of $\mathbf{t}_{gh}$'s as $\{\lambda\mathbf{a}_g+\mu\mathbf{b}_g:\lambda,\mu\in\{-1,0,1\}\}$. We assume that $\mathbf{t}_{gh}=0$ is in the center of the detector. Simulation 1 differs from Simulation 2 exactly in the choice of $\mathbf{t}_{gh}$'s, while $\mathbf{t}_{gh}$'s are equal in Simulation 1 (respectively, Simulation 2) and Simulation 3 (respectively, Simulation 4). Instead of listing $\mathbf{t}_{gh}$ for each simulation, we choose to report two numbers "# Spots" and D_{best}; the former gives the total number of $\mathbf{t}_{gh}$'s whose discretized version $\bar{\mathbf{t}}_{gh}$ hits the detector, while the latter gives information about the general FT "overlap." Since we choose for HIO the FT that is furthest apart from any other FT (see Subsection 4.A), it is natural to define D_{best} as $\min\{D_1,\ldots D_G\}$, where $D_g=\max_h \min_{(g',h')\neq(g,h)}\|\bar{\mathbf{t}}_{gh}-\bar{\mathbf{t}}_{g'h'}\|_\infty$ with $\bar{\mathbf{t}}_{gh}$ and $\bar{\mathbf{t}}_{g'h'}$ denoting the discretized versions of $\mathbf{t}_{gh}$ and $\mathbf{t}_{g'h'}$, respectively, that appear on the detector. The values of # Spots and D_{best} in our simulations are given in Columns 6 and 7 of Table 1, respectively.

The detector data in our simulations are subjected to noise. The exact nature of the noise needs to be determined in future experiments, but at present it is clear that one needs to distinguish between random noise introduced by the detector reading and geometric noise that is due to uncertainties in the experimental geometry. To define these types of noise in precise terms, we need to introduce some notations. Let `det_value`($\mathbf{q}$) denote the value of the detector pixel $\mathbf{q}$ of the $N_{det}\times N_{det}$ detector image. A detector shift by, say s pixels, is obtained by replacing `det_value`($\mathbf{q}$) with `det_value`($\mathbf{q}+(s,0)^T$), with `det_value`($\mathbf{q}+(s,0)^T$)$=0$ if $\mathbf{q}+(s,0)^T\notin[0,N_{det}-1]^2$. In our simulations we distinguish four different noise scenarios:

- No noise: In this case no detector pixel value is altered.
- Random noise: The no noise detector image is altered by setting `det_value`($\mathbf{q}$)$\leftarrow\max\{0,$`det_value`($\mathbf{q}$)$+r\}$, where r is chosen uniformly at random from the interval $[-100$ `det_value`($\mathbf{q}$), 100 `det_value`($\mathbf{q}$)$]$ for $\mathbf{q}\in[0,N_{det}-1]^2$.
- 4 pixel shift: The no noise detector image is shifted by 4 pixels.
- 8 pixel shift: The no noise detector image is shifted by 8 pixels.

Detector data of all five no noise simulations are depicted in Fig. 5.

B. Recovery

As mentioned above the aim is to reconstruct the color image $f^{N,d}$ for Maps I–III. Since we assume that prior information about approximate grain positions is available, we aim at reconstructing only "ambiguous pixels" and not the whole color image. We assume that the approximate grain position is given by means of the grain's ($=f_g^{N,d}$) minimal bounding box that is extended by a number of N_{margin} pixels at each border and by some partial reconstructions that might have been obtained by other methods (e.g., by incoherent diffraction imaging). We simulate the partial reconstruction by eroding the grain boundaries. Those removed pixels that fall into the extended bounding boxes of at least two grains are defined to be the ambiguous pixels in our simulations. The right column of Fig. 4 shows the $f^{N,d}$'s together with the ambiguous pixels (colored white), and the total number of ambiguous pixels in the $f^{N,d}$'s is given in the second column of Table 2. Note that low-accuracy position information and lattice structure are implicitly incorporated into Algorithm II via the support constraints and the FT data that require that the $\mathbf{t}_{gh}$'s (Bragg peak positions) are known.

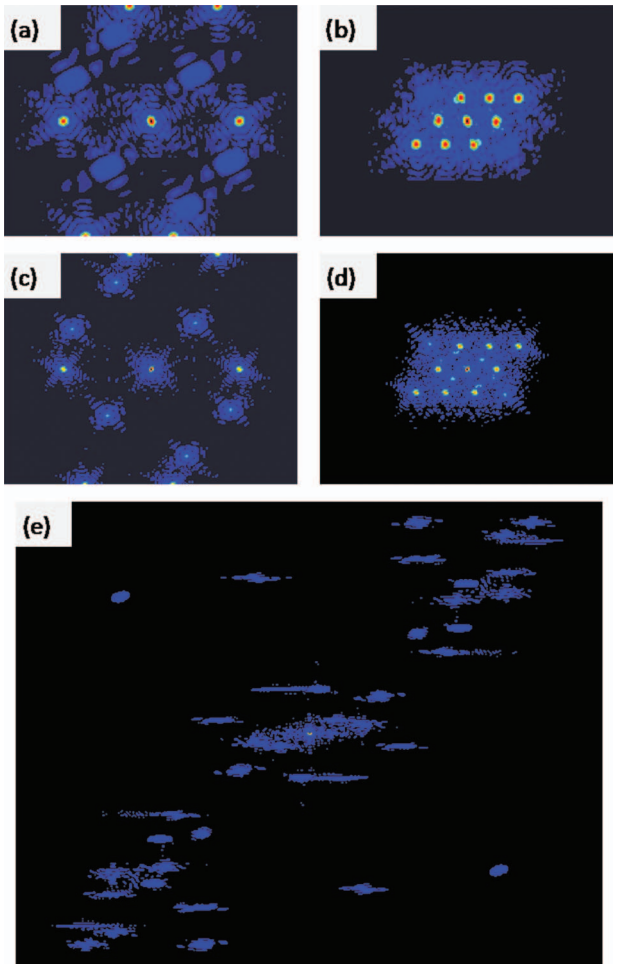

Fig. 5. Detector data for Simulations (a) 1, (b) 2, (c) 3, (d) 4, and (e) 5.

We set up five simulations as summarized in Table 1. The parameter N_{margin} is set to 8; only in Simulation 5 we choose $N_{margin}=0$. The only difference between Simulations 1 and 2 and Simulations 3 and 4 is that a different input map is used, while from Simulation 1 to Simulation 2 we only vary the set of $\mathbf{t}_{gh}$'s. Simulation 5 tests the reconstruction of a more complex grain map.

A common technique adopted in many stochastic algorithms (and the HIO algorithm [35]) is to repeat reconstructions from random initial guesses and to return the solution with the best data fit (or objective function value) that occurred over these independent runs. We follow this approach and use for Algorithm I two independent reconstructions, while for Algorithm II we use five. Details about the initial guesses are discussed below. For each simulation 50 reconstruction results are obtained using Algorithms I and II.

The number of iterations for all algorithms is `mc_cycles=HIO_cycles=200`. Moreover, the following specific parameters are used:

Algorithm I: The parameter β in Eq. (4) is set to $\beta=20$. The candidate colors for each ambiguous pixel are restricted to be the colors of the grains into whose extended bounding box it falls. Premature termination occurs if no change in the objective function γ_β is observed over 10 of the `mc_cycles=200` iterations.

Algorithm II: In agreement with [14,35] we choose $\varepsilon=0.01$ and $\beta_{\mathrm{HIO}}=0.9$. The support $\mathcal{S}_{object}$ of each grain is chosen as tight as possible, i.e., as the set of those pixels of the extended minimal bounding box that are either pixels in that grain or ambiguous. In each HIO iteration we enforce prescribed pixel values as additional object domain constraints. As initial guesses we randomly assign the value of 0/1 to each ambiguous pixel in $\mathcal{S}_{object}$ (0 corresponding to the background and 1 corresponding to the grain pixel). Grains are processed in a random order.

C. Results and Discussion

In Table 2 we report the results of the simulations. The reconstruction error is defined as the number of pixels in the reconstruction that differ from the pixels of the true color image. Since each simulation is based on 50 results obtained by each algorithm, we report mean reconstruction errors, median reconstruction errors, and standard deviations (the latter two are given in parentheses). Additionally, we report mean running times (the standard deviation was comparatively small in all cases and is thus not reported). Each simulation is performed for each of the noise scenarios discussed above.

Several facts from Table 2 are worth noting:

- Algorithm II is usually dramatically faster than Algorithm I, but the quality of the reconstruction is always better with Algorithm I. This can be explained by the fact that Algorithm I tries to fit simultaneously the complete data, while Algorithm II uses only one FT at a time for the reconstruction.
- The random noise does not have an appreciable effect on the reconstruction quality of Algorithms I and II.

In Fig. 6 we show a typical difference image (black pixels correspond to pixels in the reconstruction that differ from the original image) obtained by Algorithm I in Simulation

Table 2. Results of the Simulations. Each Entry is Based on 50 Simulations from Random Initial Guesses. Errors (Defined as Number of Pixels That Differ from the Corresponding Pixels in the Original) are Reported as Mean Pixel Errors, Followed by the Median and the Standard Deviation in Parentheses. Median Reconstruction Times are Given in the Corresponding Columns.

	No. of Ambig. Pixels		Algorithm I		Algorithm II	
			Error	Time (s)	Error	Time (s)
Simul. 1:	103					
		No noise	0.1 (0.0/0.4)	21	5.5 (5.5/2.5)	4
		Random noise	0.0 (0.0/0.3)	21	6.4 (6.5/3.1)	4
		4 pixel shift	11.1 (8.0/7.8)	24	48.8 (49.0/5.5)	4
		8 pixel shift	17.3 (17.0/6.6)	24	45.4 (50.5/13.2)	4
Simul. 2:	103					
		No noise	4.2 (4.0/0.7)	23	24.1 (18.0/14.1)	4
		Random noise	4.4 (4.0/1.1)	24	23.9 (18.0/13.2)	4
		4 pixel shift	28.8 (29.0/6.4)	24	47.6 (54.0/17.5)	4
		8 pixel shift	44.4 (45.5/7.1)	23	54.0 (52.0/22.3)	4
Simul. 3:	311					
		No noise	0.0 (0.0/0.0)	61	8.4 (8.0/2.6)	4
		Random noise	1.4 (0.0/7.5)	66	9.6 (7.5/11.7)	4
		4 pixel shift	83.8 (81.0/19.9)	76	147.2 (144.5/10.2)	4
		8 pixel shift	112.4 (113.0/18.4)	81	146.8 (144.0/10.0)	4
Simul. 4:	311					
		No noise	2.9 (3.0/1.0)	68	135.5 (123.0/22.0)	4
		Random noise	2.8 (3.0/1.0)	68	133.3 (120.5/22.9)	4
		4 pixel shift	90.8 (89.5/20.2)	92	151.4 (142.5/15.0)	4
		8 pixel shift	128.7 (128.0/15.1)	93	153.0 (142.0/16.6)	4
Simul. 5:	1318					
		No noise	124.7 (124.5/17.3)	2265	582.0 (588.0/43.8)	106
		Random noise	123.3 (121.0/18.6)	2124	584.8 (583.0/45.5)	106
		4 pixel shift	478.6 (482.0/31.3)	2288	682.0 (682.5/58.5)	106
		8 pixel shift	574.5 (577.5/22.2)	2398	682.8 (679.5/61.6)	106

5 (no noise). Reconstruction errors seemingly appear only at the grain boundaries, which is not surprising if one takes into account that data generation might introduce some uncertainties.

As in [6,7,20–22], one could think of improving Algorithm I by incorporating statistical information about grain shapes (via Gibbs priors) into the reconstruction process. However, we implemented this approach and did not find an appreciable improvement in our experiments.

One may argue that in Algorithm II the tight support constraints may not be available in practice. In this case one can use loose supports in the initial reconstructions, which could be tightened in subsequent reconstructions. Algorithm I, in contrast to Algorithm II, does not rely on support constraints, and thus its (already superior) performance will not be negatively affected by the weakening of support constraints.

Algorithms I and II, as presented, yield 2D images as outputs. Generalizations of the methods to three dimensions are straightforward. Algorithm I is particularly well suited to be implemented in a parallel computing environment. Reconstructions in three dimensions can be obtained from 2D reconstructions if the data are acquired slice-by-slice. We remark, however, that CDI for multiple grains in three dimensions does not necessarily correspond to the 3D binary superposed phase retrieval problem as introduced in this paper if the data are acquired truly in three dimensions; in that case, the data would involve superposed FTs of *projections* of $f_1, \ldots, f_G$.

6. CONCLUSIONS

We have introduced the *binary superposed phase retrieval* problem, which generalizes the binary phase retrieval problem. As a relevant application we discussed, under simplifying assumptions, the reconstruction of grain maps from coherent diffraction data. We proposed a Monte Carlo based algorithm and another algorithm based on standard phase retrieval methods. It appears that the Monte Carlo algorithm gives far superior results in cases where the superposition effects on the FTs cannot be ignored.

ACKNOWLEDGMENTS

The authors would like to thank Ivan Vartanyants for helpful discussions and Fengxiang Lin for providing us with the EBSD maps. This work was partially supported by the European Union (EU) program TotalCryst, the Danish National Research Foundation, the Danish XFEL Initiative, and Award Number R01HL070472 from the

Fig. 6. Typical reconstruction error of Algorithm I in Simulations 5 (no noise). Pixels of the reconstruction that differ from the original are depicted in black.

National Heart, Lung, and Blood Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Heart, Lung, and Blood Institute or the National Institutes of Health.

REFERENCES

1. R. P. Millane, "Phase retrieval in crystallography and optics," J. Opt. Soc. Am. A **7**, 394–411 (1990).
2. J. C. Dainty and J. R. Fienup, "Phase retrieval and image reconstruction for astronomy," in *Image Recovery: Theory and Application*, H. Stark, ed. (Academic, 1987), pp. 231–275.
3. P. Jaming, "Phase retrieval techniques for radar ambiguity problems," J. Fourier Anal. Appl. **5**, 309–329 (1999).
4. V. L. Lo and R. P. Millane, "Reconstruction of compact binary images from limited Fourier amplitude data," J. Opt. Soc. Am. A **25**, 2600–2607 (2008).
5. H. F. Poulsen, *Three-Dimensional X-Ray Diffraction Microscopy* (Springer-Verlag, 2004).
6. A. Alpers, H. F. Poulsen, E. Knudsen, and G. T. Herman, "A discrete tomography algorithm for improving the quality of 3DXRD grain maps," J. Appl. Crystallogr. **39**, 582–588 (2006).
7. A. Alpers, L. Rodek, H. F. Poulsen, E. Knudsen, and G. T. Herman, "Discrete tomography algorithms for grain map reconstructions," in *Advances in Discrete Tomography and Its Applications*, G. T. Herman and A. Kuba, eds. (Birkhäuser, 2007), pp. 271–301.
8. J. W. Miao, P. Charalambous, J. Kirz, and D. Sayre, "Extending the methodology of x-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens," Nature **400**, 342–344 (1999).
9. J. Miao, T. Ishikawa, Q. Shen, and T. Earnest, "Extending x-ray crystallography to allow the imaging of noncrystalline materials, cells, and single protein complexes," Annu. Rev. Phys. Chem. **59**, 387–410 (2008).
10. J. M. Zuo, I. Vartanyants, M. Gao, R. Zhang, and L. A. Nagahara, "Atomic resolution imaging of a carbon nanotube from diffraction intensities," Science **300**, 1419–1421 (2003).
11. I. A. Vartanyants, I. K. Robinson, J. D. Onken, M. A. Pfeifer, G. J. Williams, F. Pfeiffer, H. Metzger, Z. Zhong, and G. Bauer, "Coherent x-ray diffraction from quantum dots," Phys. Rev. B **71**, 245302 (2005).
12. M. A. Pfeifer, G. J. Williams, I. A. Vartanyants, R. Harder, and I. K. Robinson, "Three-dimensional mapping of a deformation field inside a nanocrystal," Nature **442**, 63–66 (2006).
13. P. P. Ewald, "X-ray diffraction by finite and imperfect crystal lattices," Proc. Phys. Soc. **52**, 167–174 (1940).
14. J. R. Fienup, "Phase retrieval algorithms: A comparison," J. Opt. Soc. Am. A **21**, 2758–2769 (1982).
15. M. Nieto-vesperinas and J. A. Mendez, "Phase retrieval by Monte-Carlo methods," Opt. Commun. **59**, 249–254 (1986).
16. M. Nieto-Vesperinas, R. Navarro, and F. J. Fuentes, "Performance of a simulated-annealing algorithm for phase retrieval," J. Opt. Soc. Am. A **5**, 30–38 (1988).
17. R. Navarro, F. J. Fuentes, and M. Nieto-vesperinas, "Simulated annealing image-reconstruction in photon-limited stellar speckle interferometry," Astron. Astrophys. **208**, 374–380 (1989).
18. D. Morris, "Simulated annealing applied to the Misell algorithm for phase retrieval," IEE Proc., Part H **143**, 298–303 (1996).
19. J. Miao and D. Sayre, "On possible extensions of x-ray crystallography through diffraction-pattern oversampling," Acta Crystallogr. A **56**, 596–605 (2000).
20. A. Alpers, E. Knudsen, H. F. Poulsen, and G. T. Herman, "Resolving ambiguities in reconstructed grain maps using discrete tomography," Electron. Notes Discrete Math. **20**, 419–437 (2005).
21. L. Rodek, H. F. Poulsen, E. Knudsen, and G. T. Herman, "A stochastic algorithm for reconstruction of grain maps of moderately deformed specimens based on x-ray diffraction," J. Appl. Crystallogr. **40**, 313–321 (2007).
22. A. K. Kulshreshth, A. Alpers, G. T. Herman, E. Knudsen, L. Rodek, and H. F. Poulsen, "A greedy method for reconstructing polycrystals from three-dimensional x-ray diffraction data," Inverse Probl. Imaging **3**, 69–85 (2009).
23. R. H. T. Bates, "Uniqueness of solutions to two-dimensional Fourier phase problems for localized and positive images," Comput. Vis. Graph. Image Process. **25**, 205–217 (1984).
24. J. Miao, D. Sayre, and H. N. Chapman, "Phase retrieval from the magnitude of the Fourier transforms of nonperiodic objects," J. Opt. Soc. Am. A **15**, 1662–1669 (1998).
25. G. T. Herman and R. Davidi, "Image reconstruction from a small number of projections," Inverse Probl. **24**, 045011 (2008).
26. J. Wu, U. Weierstall, and J. C. H. Spence, "Diffractive electron imaging of nanoparticles on a substrate," Nature Mater. **4**, 912–916 (2005).
27. E. M. Lauridsen, S. Schmidt, R. M. Suter, and H. F. Poulsen, "Tracking: A method for structural characterization of grains in powders or polycrystals," J. Appl. Crystallogr. **34**, 744–750 (2001).
28. M. Moscicki, P. Kenesei, J. Wright, H. Pinto, T. Lippmann, A. Borbely, and A. R. Pyzalla, "Friedel-pair based indexing method for characterization of single grains with hard x-rays," Mater. Sci. Eng., A **524**, 64–68 (2009).
29. N. W. Ashcroft and D. N. Mermin, *Solid State Physics* (Thomson Learning, 1976).
30. B. E. Warren, *X-Ray Diffraction* (Dover, 1990).
31. G. Winkler, *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods* (Springer-Verlag, 2003).
32. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science **220**, 671–680 (1983).
33. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast

computing machines," J. Chem. Phys. **21**, 1087–1092 (1953).
34. J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput. **19**, 297–301 (1965).
35. I. A. Vartanyants and I. K. Robinson, "Partial coherence effects on the imaging of small crystals using coherent x-ray diffraction," J. Phys. Condens. Matter **13**, 10593–10611 (2001).
36. V. Elser, I. Rankenburg, and P. Thibault, "Searching with iterated maps," Proc. Natl. Acad. Sci. U.S.A. **104**, 418–423 (2007).
37. R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane pictures," Optik (Stuttgart) **35**, 237–246 (1972).