Integrating Materials and Manufacturing Innovation (2023) 12:251–266
https://doi.org/10.1007/s40192-023-00304-8

TECHNICAL ARTICLE

Restoration of Noisy Orientation Maps from Electron Backscatter Diffraction Imaging

Emmanuel Atindama[1] · Peter Lef[1] · Günay Doğan[2] · Prashant Athavale[1]

Received: 9 April 2023 / Accepted: 22 July 2023 / Published online: 28 August 2023

Abstract

Crystallographic orientations can be measured using scanning electron microscope-based techniques, such as electron backscatter diffraction (EBSD). The orientation data thus obtained may contain noise and misindexed data. There are several methods to restore the orientation data. The restorations from these methods may have varying levels of quality. Moreover, many such methods are parameter-dependent. Therefore, finding suitable parameter settings for optimal restorations can take time and effort for users of such methods. In this paper, we propose an algorithm to obtain high-quality restorations of noisy orientation data and to circumvent the parameter selection problem by automating it. We estimate the noise variance in the data to determine the amount of denoising to apply. We then use this information to determine the stopping criteria for a vector-valued weighted total variation flow, a nonlinear diffusion applied to the noisy orientation map. We compare the results obtained by our approach with the results from other commonly used denoising filters. As benchmarks, we used simulated EBSD maps with varying noise levels. Our proposed method outperformed denoising methods, such as mean, median, spline, half-quadratic, and Kuwahara filters. The denoising results were statistically significantly better for higher levels of noise.

Keywords EBSD · Denoising · Orientation maps · Restoration · TV flow

Introduction

Polycrystalline materials consist of aggregates of crystal grains, regions with similar crystalline orientations. The grain orientations are obtained using various technologies such as electron-backscatter diffraction (EBSD), transmission electron microscopy (TEM) [1], automated crystal orientation mapping inside the transmission electron microscope (ACOM-TEM) [2], and the X-ray diffraction (XRD) [3]. In recent years, high-energy X-ray diffraction microscopy (HEDM, also known as 3D-XRD) [4] is also used to obtain orientation data. The EBSD technique obtains such data as Euler angles [5]. Orientation maps thus obtained can be used to calculate various microstructure statistics, e.g., average grain sizes [6], or to simulate microstructure physics [7].

However, the orientation data collected by such techniques are not perfect. The data collected often contain misorientation noise. Moreover, the range of the angle data can cause further misorientation errors [8]. Misindexing is another problem present in EBSD orientation data. Misindexing at a point is a random orientation change caused by various external factors [9]. Isolated points or clusters of points are categorized as misindexed if their corresponding angles differ significantly from their local neighbors. Misindexed data points can appear as impulse noise.

In this paper, we propose a new algorithm for addressing the issues of misorientation noise, jump discontinuities,

Günay Doğan and Prashant Athavale have contributed equally to this work.

✉ Prashant Athavale
pathaval@clarkson.edu

Emmanuel Atindama
atindaea@clarkson.edu

Peter Lef
lefp@clarkson.edu

Günay Doğan
gunay.dogan@nist.gov

[1] Department of Mathematics, Clarkson University, 8 Clarkson Avenue, Potsdam, NY 13699, USA

[2] National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899, USA

and misindexing errors in crystallographic orientation data. There are numerous methods to restore the orientation data, which include traditional filters such as the mean, median, and spline filters [10], as well as advanced techniques such as Kuwahara filter [11], half-quadratic filter [12] to name a few. These algorithms come with various levels of denoising and restoration performance, and obtaining their best performances can be tedious in practice as the user of these algorithms may need to tune algorithm parameters and execute multiple test runs with different parameters to compute the best result. We recognize this difficulty and propose a restoration algorithm that performs better than most widely used algorithms and does not require tuning denoising parameters. Our approach is first to remove the jump discontinuities from the data. We then correct the misindexed data points. Finally, we use vector-valued weighted total variation (TV) flow to denoise the data. To tune the amount of denoising, we estimate the noise in the data using a fast noise estimation technique [13], and use this information to determine the amount of denoising to perform, and thus obtain a restoration of the data using total variation flow [14, 15] to the Euler angles $\boldsymbol{f}$ representing the grain orientations. We compare our approach with established methods.

Current Approaches for Restoration of Orientation Data

Given noisy orientation data $\boldsymbol{f}$, we can use various existing denoising algorithms to obtain denoised data $\boldsymbol{u}$. We list a few of these algorithms here.

i. *Mean filter*: The restored data at a location $\boldsymbol{x} = (x, y)$ is a formal averaging of the orientations in the spatial neighborhood $\mathcal{N}(\boldsymbol{x})$ of $\boldsymbol{x}$.
ii. *Median filter*: The denoising is achieved by replacing the data at $\boldsymbol{x}$ with the median of the neighborhood orientations.
iii. *Spline filter*: We minimize the distance between the given orientation data $\boldsymbol{f}$ and the solution $\boldsymbol{u}$, using the square of the Euclidean norm of the Laplacian of the orientation data as the penalty term to enforce regularization.
iv. *Kuwahara filter*: We split the neighborhood $\mathcal{N}(\boldsymbol{x})$ into smaller subregions $\mathcal{N}_i(\boldsymbol{x})$ for $i = 1, 2, 3, 4$. Then the data at $\boldsymbol{x}$ is replaced by the mean in the subregion $\mathcal{N}_j(\boldsymbol{x})$ with the least variance [11].
v. *Half-quadratic filter*: The half-quadratic filter uses the total variation (TV) of $\boldsymbol{u}$ [14] as the penalty term [12] to induce regularization. It uses a cut-off at a certain threshold angle, such that the disorientation angles larger than the threshold, e.g., at grain boundaries, are not penalized.

Contributions of the Paper

One of the drawbacks of the denoising algorithms mentioned in Sect. 1.1 is that the algorithm-specific parameters need to be adjusted depending on the noise in the given orientation data. For example, one needs to set the parameters for the TV penalty, and the threshold angle in the half-quadratic filter [12]. How to set the correct values for these parameters is not obvious for non-expert users of these algorithms. This paper proposes a denoising methodology that does not depend on the user to set the denoising parameters. To further improve the quality of restorations, we propose a preprocessing algorithm to correct for angle jump discontinuity and misindexing. In Sect. 2, we discuss this preprocessing algorithm and denoising using the weighted total variation (TV) flow. Using the "weight" in the TV flow was shown to better preserve prominent edges in images compared to regular TV flow, and other standard denoising algorithms [15]. In EBSD maps, the grain boundaries are the prominent edges, e.g., see Fig. 2a. Hence, using a spatially-varying weight in our algorithm retains the grain boundaries as the weight inhibits data diffusion on the grain boundaries. In Sect. 4, we present the results of the proposed algorithm and its comparison with the contemporary methods. Tables 4 and 5 show that our proposed algorithm performs better than other contemporary methods, especially for high noise levels.

The Proposed Algorithm

We describe the details of our algorithms in this section. Before denoising, we may need to remove the angle jump discontinuities within grains [8] and correct the misindexed data. To this effect, we propose preprocessing the orientation map in Appendix A. We then use the vector-valued weighted TV flow to denoise this preprocessed data, which we describe in Sect. 2.1. We note that, on average, the preprocessing step improved the weighted TV denoising by about 10 to 30%. We also observed that the preprocessing step contributed less toward denoising for higher noise levels.

Weighted TV Flow

The orientation data $\boldsymbol{u}_0$ is in the three-dimensional space of Euler angles, i.e. $\boldsymbol{u}_0 : \mathbb{R}^2 \to \mathrm{SO}(3)$. Thus, the orientation data $\boldsymbol{u}_0$ is a vector-valued function on $\Omega \subset \mathbb{R}^2$, with $\boldsymbol{u}_0 = \langle u_0^{(1)}, u_0^{(2)}, u_0^{(3)} \rangle$. In presence of noise $\boldsymbol{\eta}$, the resulting data that we observe is $\boldsymbol{f} = \boldsymbol{u}_0 + \boldsymbol{\eta}$. Variational methods such as total variation (TV) regularization [14] have been used for denoising of image data where one seeks the solution $\hat{\boldsymbol{u}}$ the following minimization problem [16]

$$\hat{\boldsymbol{u}} = \underset{\boldsymbol{u}\in BV(\Omega)}{\arg\min} \left\{ \|\boldsymbol{u}\|_{TV(\Omega)} + \lambda \|\boldsymbol{f} - \boldsymbol{u}\|_{L^2}^2 \right\}, \tag{1}$$

where $\|\boldsymbol{u}\|_{TV(\Omega)} := \left[\sum_{k=1}^{3} \left(\int_\Omega |\nabla u^{(k)}|\right)^2\right]^{\frac{1}{2}}$ is the TV norm, and $BV(\Omega)$ is the space of functions of bounded variation. The Euler–Lagrange differential equation describing the optimal solution $\hat{\boldsymbol{u}}$ for this minimization problem depends on the parameter λ. The parameter λ dictates the level of noise still present in the solution of the minimization problem (1), see [17]. The higher λ is, the closer the solution $\hat{\boldsymbol{u}}$ is the noisy data $\boldsymbol{f}$. Hence, a value of λ needs to be determined that is suitable for the noise level in the data $\boldsymbol{f}$. Chambolle [18] addresses this problem, but their approach requires solving the minimization problem (1) multiple times for different candidate values of λ. Instead, we omit λ in (1) and propose using the vector-valued TV flow corresponding to the minimization problem $\min_{\boldsymbol{u}\in BV(\Omega)} \|\boldsymbol{u}\|_{TV(\Omega)}$. For this, we solve the following partial differential equations

$$\frac{\partial u^{(k)}}{\partial t} = \frac{\|u^{(k)}\|_{TV(\Omega)}}{\|\boldsymbol{u}\|_{TV(\Omega)}} \operatorname{div}\left(\frac{\nabla u^{(k)}}{|\nabla u^{(k)}|}\right), \text{ for } k = 1 \text{ to } 3 \tag{2}$$

starting with the initial conditions $u^{(k)}(\cdot, 0) = f^{(k)}$, and the Neumann boundary conditions $\frac{\partial u^{(k)}}{\partial \hat{\boldsymbol{n}}}\big|_{\partial\Omega} = 0$.

The grain orientation data has well-defined grain boundaries. While the TV flow is effective for denoising the orientation data, it also diffuses the grain boundaries (see Fig. 2 g). To overcome this issue, we use weighted vector-valued TV flow, following [15, 19].

$$\frac{\partial u^{(k)}}{\partial t} = \frac{\|u^{(k)}\|_{TV_\alpha(\Omega)}}{\|\boldsymbol{u}\|_{TV_\alpha(\Omega)}} \operatorname{div}\left(\frac{\alpha \nabla u^{(k)}}{|\nabla u^{(k)}|}\right), \text{ for } k = 1 \text{ to } 3 \tag{3}$$

where $u^{(k)}(\cdot, 0) = f^{(k)}$, $\frac{\partial u^{(k)}}{\partial \hat{\boldsymbol{n}}}\big|_{\partial\Omega} = 0$ and $\|\boldsymbol{u}\|_{TV_\alpha(\Omega)} := \left[\sum_{k=1}^{3} \left(\int_\Omega \alpha |\nabla u^{(k)}|\right)^2\right]^{\frac{1}{2}}$, with $\alpha : \Omega \to \mathbb{R}$ a weight function such that $\alpha(x) \approx 1$ if x is in a homogeneous region, e.g., inside a grain, and $\alpha(x) \approx 0$ if x is on an edge, e.g., the grain orientation boundary. We propose using the time-dependent PDE (3) starting with initial data $\boldsymbol{f}$ until we obtain a denoised solution $\hat{\boldsymbol{u}}$. This approach requires an appropriate stopping criterion.

The weight function $\alpha(x)$ can be defined using edge map $e(x)$, which is 1 if x is an edge location and 0 otherwise. Since an edge map, e obtained from a noisy image, is also noisy, it needs to be cleaned. To obtain a clean edge map where edges coincide with grain boundaries, we first run unweighted ($\alpha = 1$) TV flow and get a smoothed image $\hat{\boldsymbol{f}}$. We then run the Farid edge detector on $\hat{\boldsymbol{f}}$ [20], clean up and enhance the detected edge map with morphological operations of small object removal and skeletonizing the edge map [21]. We define a numerically well-behaved α by smoothing $1 - e$ with a Gaussian kernel.

Data-driven Stopping Criterion

An appropriate stopping criterion is needed to stop the evolution of the time-dependent PDE (3) and to obtain the desired solution $\hat{\boldsymbol{u}}$. We use the approach by [13] to estimate the noise variance $\hat{\sigma}_k^2$ in each of the three channels of $\boldsymbol{f}$. For the TV flow (2) and weighted TV flow (3), we stop the numerical iterations in the kth channel when $\frac{1}{|\Omega|}\|u^{(k)} - f^{(k)}\|_{L^2(\Omega)}^2$ exceeds $\hat{\sigma}_k^2$.

Methods and Experiments

Synthetic Data Acquisition and Performance Comparisons

In order to objectively evaluate a denoising method, we needed the ground truth data. To this effect, we used the software DREAM.3D [22] and generated synthetic EBSD maps for ground truth. The synthetic EBSD maps were drawn from a random orientation distribution function (ODF) with mean = 3 μm and standard deviation = 0.3 μm. We added noise to the ground truth data, then used the denoising method to obtain the denoised data. We could thus compare the denoised data with the ground truth.

Measuring the Denoising Quality with Disorientation Angle δ

In classical image processing algorithms, measuring the quality of denoising usually entails measuring the difference between the denoised data and ground truth, typically using an ℓ^2 difference [10]. However, two numerically different EBSD maps might be crystallographically identical due to crystal symmetry [23]. Since we cannot use a simple numerical difference to measure the denoising quality, we need to devise a new measure suitable for crystallographic data.

Orientation $\boldsymbol{O}$ of a crystal refers to rotations that map coordinates of a crystal reference frame to coordinates of a specimen reference frame [23, 24]. Moreover, $\boldsymbol{OS}$ with $\boldsymbol{S} \in S$ denotes the class of all orientations equivalent to $\boldsymbol{O}$ with respect to the symmetry group S. The difference between two true orientation $\boldsymbol{O_1}$ and the denoised orientation $\boldsymbol{O_2}$ is measured by the misorientation matrix $\boldsymbol{O_1}^{-1}\boldsymbol{O_2}$. The class of symmetrically equivalent misorientations thus are $\boldsymbol{S}_1^{-1}\boldsymbol{O}_1^{-1}\boldsymbol{O}_2\boldsymbol{S}_2$ with $\boldsymbol{S}_1, \boldsymbol{S}_2 \in S$. The smallest rotational angle δ corresponding to the misorientation is called the disorientation angle. We use the mean per-pixel δ as a measure of the difference between the ground truth and the denoised data and refer to this measure as '*the denoising error*'. The smaller the denoising error between the denoised EBSD data $\hat{u}$ and the ground truth f, the better the denoising quality.

Adding Noise to the Synthetic Data

To measure the effectiveness of the denoising algorithms, we needed to add a known amount of noise to the EBSD ground truth data and compare the denoised data with the ground truth. Following [25], we chose to add noise anisotropic Bingham distribution, which is a generalization of the Gaussian distribution suitable for the orientation data. Following [24] we approximated this noise by the de la Vallée Poussin noise [26] with the following distribution function $g(\boldsymbol{q}) = \frac{B(\frac{3}{2},\frac{1}{2})}{B(\frac{3}{2},\kappa+\frac{1}{2})} q_1^{2\kappa}$, where q_1 is the real part of the quaternion representation of the orientation, $\boldsymbol{q}$, B denotes the beta function, and κ is the concentration parameter. The parameter κ is related to the half-width b of the noise by $\kappa = \frac{1}{2}\frac{\ln(\frac{1}{2})}{\ln\cos(\frac{b}{4})}$. We use the half-width parameter b to describe the noise level; a larger half-width b makes the data noisier. In this paper, we added the de la Vallée noise with half-widths of $b = 4, 8, 12$, and 16 to clean synthetic EBSD map and obtained four different levels of noisy EBSD maps. We chose these values to represent a range of misorientation noises. Since it is not obvious how a noise level with a particular value of b presents itself, we show in Fig. 1 examples of EBSD maps with hexagonal symmetry with varying b values. Figures 1 (i)–(iv) depict IPF-z maps corresponding to EBSD data with de la Vallée Poussin noise of half-width $b = 4, 8, 12$, and 16, respectively, added to a EBSD map generated using DREAM.3D. To give an intuitive understanding of the noise levels, we compute the disorientation angle values (δ) corresponding to these half-width values of b and present them in Table 1.

We denoised the noisy data using different denoising algorithms. To obtain the denoising results with mean, median, spline, and Kuwahara filters, we used MTEX, an open-source Matlab® package [27]. See [28] for details of the steps of denoising using MTEX.

Table 1 Half-width and the disorientation angles

Half-width ($\boldsymbol{b}$)	Disorientation angle (δ)
4	0.56°
8	0.67°
12	1.06°
16	1.45°

The disorientation angles δ for the noise corresponding to half-width parameters, b, for the EBSD maps in Fig. 1

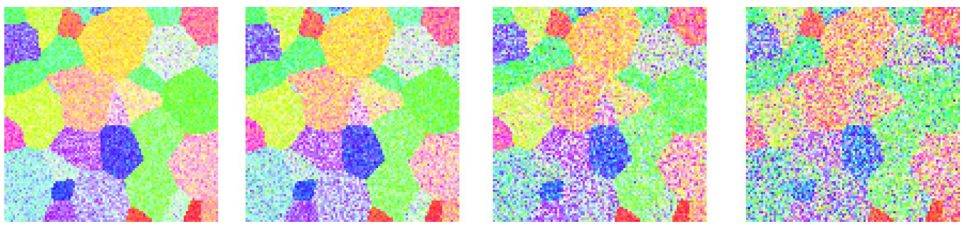
Denoising Experiments

Denoising with the Proposed TV Flow and Weighted TV Flow

We preprocessed the noisy EBSD data using the algorithm described in Appendix A. We estimated the noise variance σ_k^2 in each data channel using the fast noise estimation algorithm in [13]. Using this estimate for the stopping criteria, we applied the TV flow and the weighted TV flow. We provide a detailed semi-implicit numerical scheme for the weighted TV flow in Appendix B. Setting $\alpha = 1$ gives the scheme for the "unweighted" TV flow.

Denoising Experiments with Various Algorithms

After adding a varying level of de la Vallée Poussin noise to the EBSD maps (a dataset of 10 synthetic maps with hexagonal symmetry), we denoised these data using mean, median, spline, Kuwahara, half-quadratic filters, TV flow, and the weighted TV flow. We recorded the denoising error per pixel for each experiment by comparing the denoised image with the clean synthetic image. For the mean, median, spline, Kuwahara, and half-quadratic filters, we used MTEX's [27] built-in algorithms to obtain these results.



(i) $b = 4$ ($\delta = 0.56°$) (ii) $b = 8$ ($\delta = 0.67°$) (iii) $b = 12$ ($\delta = 1.06°$) (iv) $b = 16$ ($\delta = 1.45°$)

Fig. 1 Effect of adding noise to the clean image in Fig. 6 with various levels of de la Vallée Poussin noise. (i) noisy image with de la Vallée Poussin noise with half-width $b = 4$ (the disorientation angle $\delta = 0.56°$), (ii) noisy image with $b = 8$ ($\delta = 0.67°$), (iii) noisy image with $b = 12$ ($\delta = 1.06°$), (iv) noisy image with $b = 16$ ($\delta = 1.45°$)

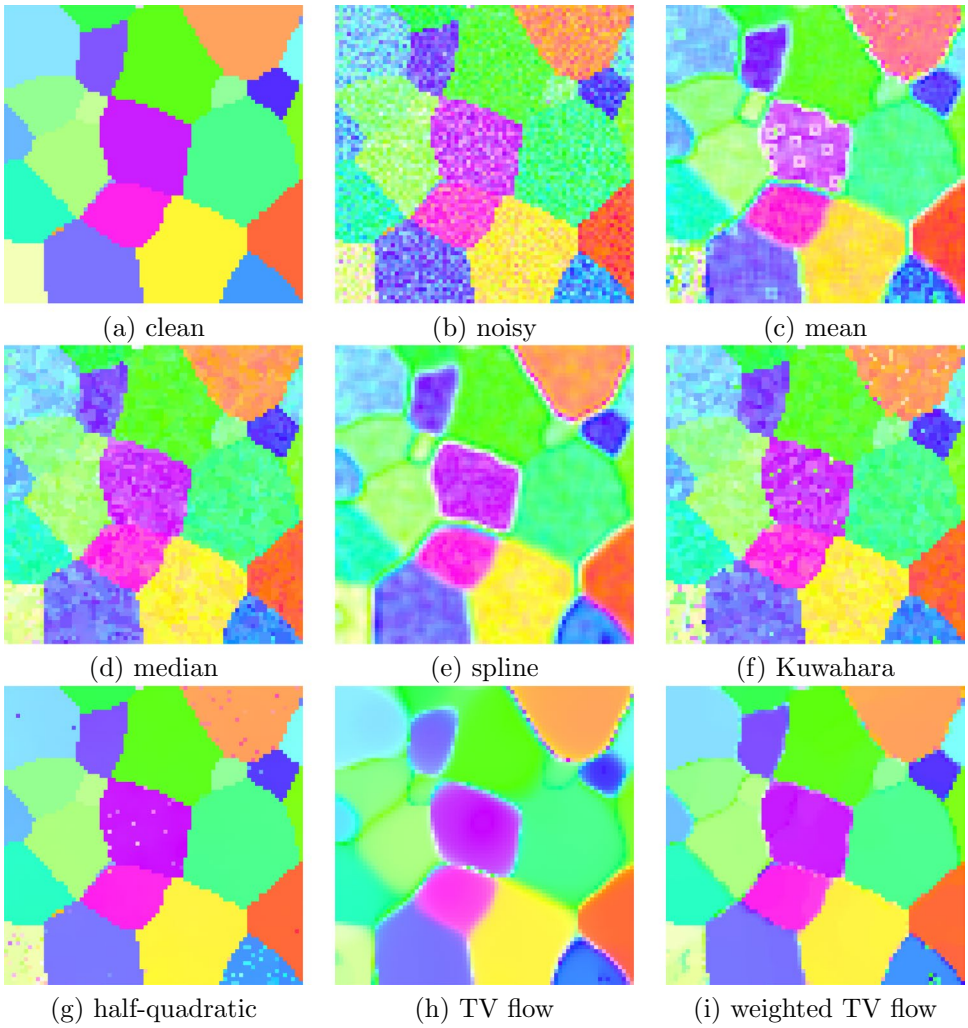


Fig. 2 IPF-z maps of the results of denoising synthetic EBSD map with hexagonal symmetry with de la Vallée Poussin noise with half-width $b = 8$ using various filters

Statistical Comparison of the Weighted TV Flow and the (Unweighted) TV Flow

We first needed to determine whether weighted TV flow produced better denoising results than its unweighted counterpart. The algorithm corresponding to the least denoising error is the best-performing denoising algorithm. However, if two algorithms have similar orientation errors, then the question of whether this difference is statistically significant or not becomes relevant. To find the statistical significance of the results, we employed paired t-tests [29].

To this effect, we measured the orientation errors, X_i for $i = 1, \dots n$, from n experiments from weighted TV flow and orientation errors, Y_i for $i = 1, \dots n$, from the unweighted TV flow. These measurements are paired, i.e., if X_i is the disorientation error of the result from the weighted TV flow for the ith EBSD map, Y_i is the denoising error of the result from the unweighted TV flow for the 'same' ith EBSD map. We define $\overline{d} := \frac{\sum_{i=1}^{n}(X_i - Y_i)}{n}$, and thus the test statistic is $\text{TS} = \frac{\overline{d}}{S/\sqrt{n}}$, where S is the sample standard deviation of the differences $X_i - Y_i$ for $i = 1, \dots n$. The test statistic has a t-distribution with $n - 1$ degrees of freedom. With this setup, the p-value is the probability of obtaining the observed or more extreme value of the test statistic, i.e., the p-value = Prob $(\text{TS} \geq |\,\text{TS}_{\text{observed}}\,|)$. A smaller p-value means the method with less error of the two methods is statistically significantly better than the other. Typically, p-value $< .05$ is considered statistically significant.

Statistical Comparison of the Weighted TV Flow and Other Classical Denoising Algorithms

First, we compared the weighted TV flow with the unweighted TV flow. We found that the weighted TV flow consistently resulted in less denoising error than the unweighted TV flow. Thus, we compared the weighted

TV flow with other classical algorithms: mean, median, spline, Kuwahara, and half-quadratic. For each half-width parameter $b = 4, 8, 12$, and 16, we compared the best algorithm with its nearest competitor amongst mean, median, spline, Kuwahara, and half-quadratic filters. For example, for $b = 12$ the best results are obtained by the weighted TV flow. So, we compared it with the median filter, which is its nearest competitor for this level of noise. Following Sect. 3.5, we used the paired t-test for these comparisons.

Experiments with Real EBSD Maps

We implemented TV flow, weighted TV flow, and algorithms described in Sect. 3.4.2 on real EBSD maps for Titanium with hexagonal symmetry. The real EBSD maps are courtesy of Dr. Adam Creuziger from NIST [30].

Results

Synthetic EBSD Data

In this section, we present the results of our experiments. We compared algorithms such as mean, median, spline, Kuwahara, half-quadratic, TV flow, and weighted TV flow. For each noise level (half-width $b = 4$, 8, 12, and 16), we used ten distinct examples of EBSD maps generated using DREAM.3D software. In Fig. 2 a, we exhibit a synthetic image without any noise, and Fig. 2b depicts the same image with de la Vallée Poussin noise of with half-width, $b = 8$. Images presented in this paper depict the inverse pole figures visualized from the z-direction (IPF-z) unless stated otherwise.

Results of the Denoising Experiments and Average Orientation Errors

For every experiment, we computed the denoising error between the result, $\hat{u}$, and the clean data, f. For each of the noise levels (with added de la Vallée Poussin noise with half-width parameters $b = 4, 8, 12$, and 16), we computed the average denoising error over ten different EBSD maps. Figure 2 depicts the IPF-z maps for a set of denoising results with de la Vallée Poussin noise. Figure 3 depicts the IPF-z maps for a set of denoising results with de la Vallée Poussin noise and 5% impulse noise simulating misindexing.

Results of the Statistical Comparison Between the Weighted TV Flow and the TV Flow

We computed the per-pixel orientation error for each algorithm with respect to the clean image. Thus less error implies that the denoised map is closer to the ground truth. The weighted TV flow produced less denoising error when compared to the unweighted TV flow. We compare the mean denoising errors for the two algorithms in Table 2 for the noise levels $b = 4, 8, 12, 16$, without any impulse noise. The weighted TV flow produced better results even in the presence of impulse noise, as seen in Table 3. Since we used ten noisy datasets from each noise level, we used the test statistic $TS = \frac{\bar{d}}{S/\sqrt{10}}$ which has a t-distribution with $n - 1 = 9$ degrees of freedom. In each case, the difference was statistically significant ($p < .01$).

Results of the Statistical Comparison of the Denoising Methods

For each noise level for half-width $b = 4, 8, 12$, and 16, we used ten different EBSD data to compare various denoising methods. Table 4 shows the average per-pixel denoising errors for all the algorithms considered. The number of stars in each row in Tables 4 and 5 (when present) indicates the significance of the p-value of the t-test comparing the weighted TV flow with the best method from the mean, median, spline, Kuwahara, and half-quadratic filter. The denoising error with the empirically best method is colored in yellow, whereas the error with the second-best method is in blue. For example, as seen in Table 4, for noise with $b = 16$, the per-pixel denoising error for the weighted TV flow is 0.679, which is less than the other methods considered here. Since the spline filter produced the second least per-pixel denoising error of 0.800, it is the nearest competitor of the weighted TV flow for half-width $b = 16$. The paired t-test confirmed that this difference is statistically significant with p-value $< .01$. Whereas, for $b = 8$, even though the half-quadratic produced the less per-pixel error of 0.502 compared to 0.511 with the weighted TV flow, the p- value $\approx .32 > .1$ implies that this difference is 'not' statistically significant.

Denoising Results with the Real EBSD Maps

Finally, we obtained the denoising results by these algorithms on experimentally obtained real EBSD maps of size 400×400. Note that for real EBSD maps, the ground truth is not available for a quantitative comparison of the

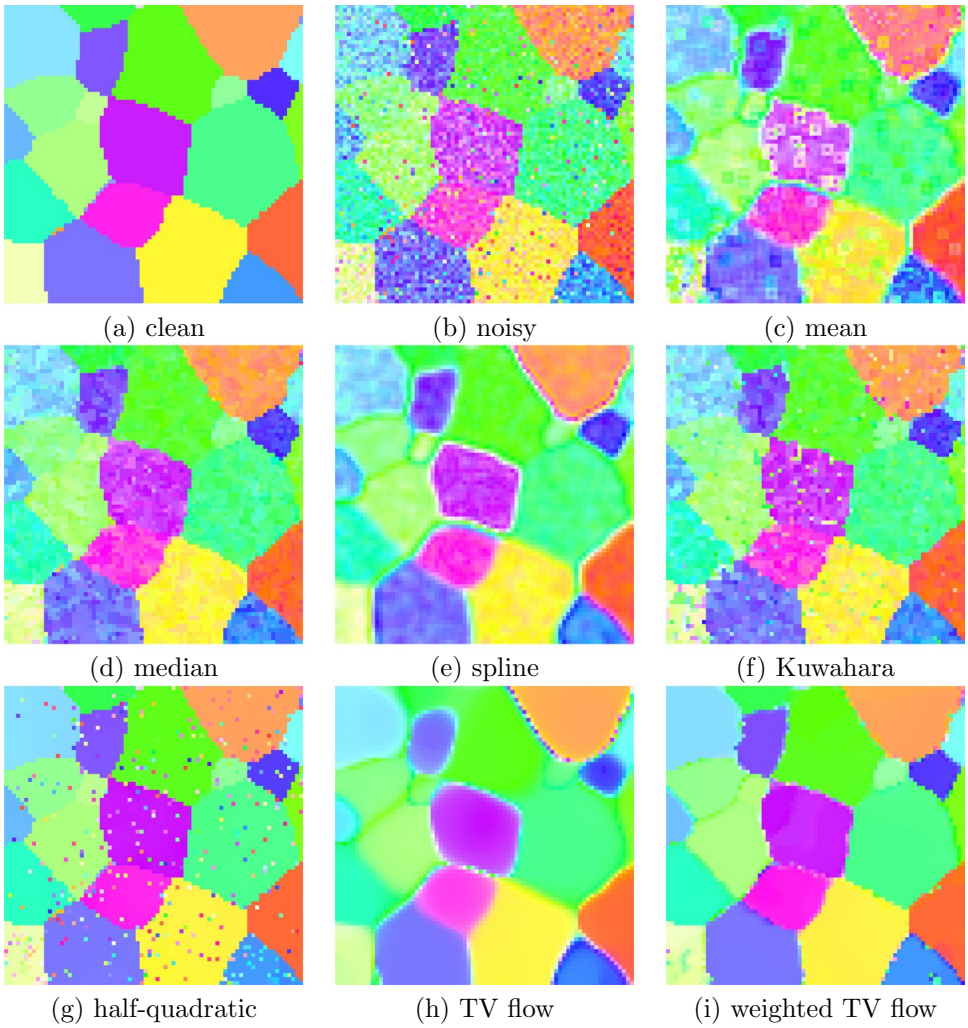


Fig. 3 IPF-z maps of the results of denoising synthetic EBSD map with de la Vallée Poussin noise with half-width $b = 8$ and 5% impulse noise using various filters

Table 2 Performance of TV flow and weighted TV without impulse noise

Half-width (b)	TV flow	Weighted TV flow	p-values
4	*0.387°*	**0.362°*****	.00004
8	*0.531°*	**0.511°*****	.00026
12	*0.605°*	**0.585°****	.00104
16	*0.694°*	**0.679°*****	.00027

Per pixel orientation errors with TV flow and weighted TV flow for input noise with half-width parameters: 4, 8, 12, and 16. The empirically best denoising error is in bold, while the error with the empirically inferior method is in italics. The stars indicate the p-value for the statistically best performance compared with the other method. The p-values reported are for the paired t-test

* $p < 0.1$, ** $p < 0.01$, *** $p < 0.001$

Table 3 Performance of the TV flow and weighted TV flow with 5% impulse noise

Half-width (b)	TV flow	Weighted TV flow	p-values
4	*0.392°*	**0.365°*****	.00003
8	*0.648°*	**0.629°****	.00107
12	*0.615°*	**0.595°****	.00103
16	*0.705°*	**0.690°*****	.00032

Per pixel orientation errors for TV flow, and weighted TV flow for input noise with half-width parameters: 4, 8, 12, and 16 along with 5% impulse noise. The empirically best denoising error is in bold, while the error with the empirically inferior method is in italics. The stars indicate the p-value for the statistically best performance compared with the next best method. The p-values reported are for the paired t-test

* $p < 0.1$, ** $p < 0.01$, *** $p < 0.001$

denoising methods. Also, since differences in the results are not clear from the full-scale images, we selected a small region from these images to show the details of the denoising results in Fig. 4. Unlike synthetic data, the real data contains extra information, such as confidence metric that Dream.3D software can utilize. We used a pipeline of

Table 4 Comparison of various denoising algorithms

Half-width (b)	Mean	Median	Spline	Kuwahara	Half-quadratic	Weighted TV flow	p-values
4	0.566°	*0.358°*	0.480°	0.381°	**0.305°*****	0.362°	.00059
8	0.664°	0.557°	0.586°	0.569°	**0.502°**	*0.511°*	.31659
12	0.775°	*0.649°*	0.687°	0.694°	0.794°	**0.585°***	.05560
16	0.872°	0.805°	*0.800°*	0.813°	1.01°	**0.679°****	.00276

Per pixel denoising errors (in degrees) with various denoising algorithms for input noise with half-width parameters: 4, 8, 12, and 16. The empirically best denoising error is in bold, while the error with the empirically second-best method is in italics. The stars indicate the p-value for the statistically best performance compared with the second-best method. The p-values reported are for the paired t-test between the best methods and its closest competitor. * $p < 0.1$, ** $p < 0.01$, *** $p < 0.001$

Table 5 Comparison of various denoising algorithms in the presence of 5% impulse noise

Half-width (b)	Mean	Median	Spline	Kuwahara	Half-quadratic	Weighted TV flow	p-values
4	.600°	*.370°*	.478°	.414°	.392°	**.365°**	.21007
8	.784°	*.669°*	.702°	.699°	.700°	**.629°***	.01308
12	.812°	*.674°*	.725°	.726°	.868°	**.595°***	.02166
16	.898°	*.821°*	.839°	.836°	1.06°	**.690°*****	.00019

Per pixel denoising errors (in degrees) with various denoising algorithms for input noise with half-width parameters: 4, 8, 12, and 16, as well as 5% impulse noise. The empirically best denoising error is in bold, while the error with the empirically second-best method is in italics. The stars indicate the p-value for the statistically best performance compared with the empirically second-best method. The p-values reported are for the paired t-test between the best methods and its closest competitor. * $p < .1$, ** $p < .01$, *** $p < .001$

three Dream.3D operations, namely, thresholding, neighborhood orientation correlation, and mask dilation. We used the default values for thresholding, neighborhood orientation correlation, and we used five iterations of dilations in Erode/Dilate filter. We include the output of this pipeline in Fig. 4 for comparison. We depict the full-scale images in Fig. 7 in Appendix C.

Discussion

From Tables 2 and 3 we observed that the weighted TV flow consistently results in less denoising error than the TV flow. Moreover, this difference was statistically significant. We thus compared the weighted TV flow with the filters discussed in Sect. 1.1. Table 5 shows that when no impulse noise is involved, the weighted TV flow is significantly better (with p-value $< .01$) than mean, median, spline, Kuwahara, and half-quadratic filters for noise levels $b = 12$ and 16. For medium noise of 8, the weighted TV flow is statistically similar to the other methods. In realistic EBSD data, impulse-like noise is present. In such cases, Table 5 indicates that weighted TV flow produces numerically better results than other methods. Moreover, the results were statistically better for high noise levels. We also note that the grain boundaries after the restoration by the proposed algorithm were qualitatively similar to those found using other algorithms discussed in this paper. We discuss these findings in the Appendix D.

We observe the advantages of using the weighted TV flow on experimentally obtained EBSD maps in Fig. 4. Since the differences between these results were not easily perceived, we chose a smaller region from Fig. 7 from the Appendix C. Since the real EBSD dataset lacked ground truth, we could not objectively measure the accuracy of the denoising. However, from Fig. 4 we observe that the weighted TV flow provides visually better denoising than other algorithms while maintaining the grain boundaries.

One of the limitations of the weighted TV flow is that it promotes piecewise constant solutions. Thus, if the ground truth orientation map is textured, the algorithm will tend to produce a relatively flat orientation within each grain. Higher-order methods may address this limitation, which is beyond the scope of this work.

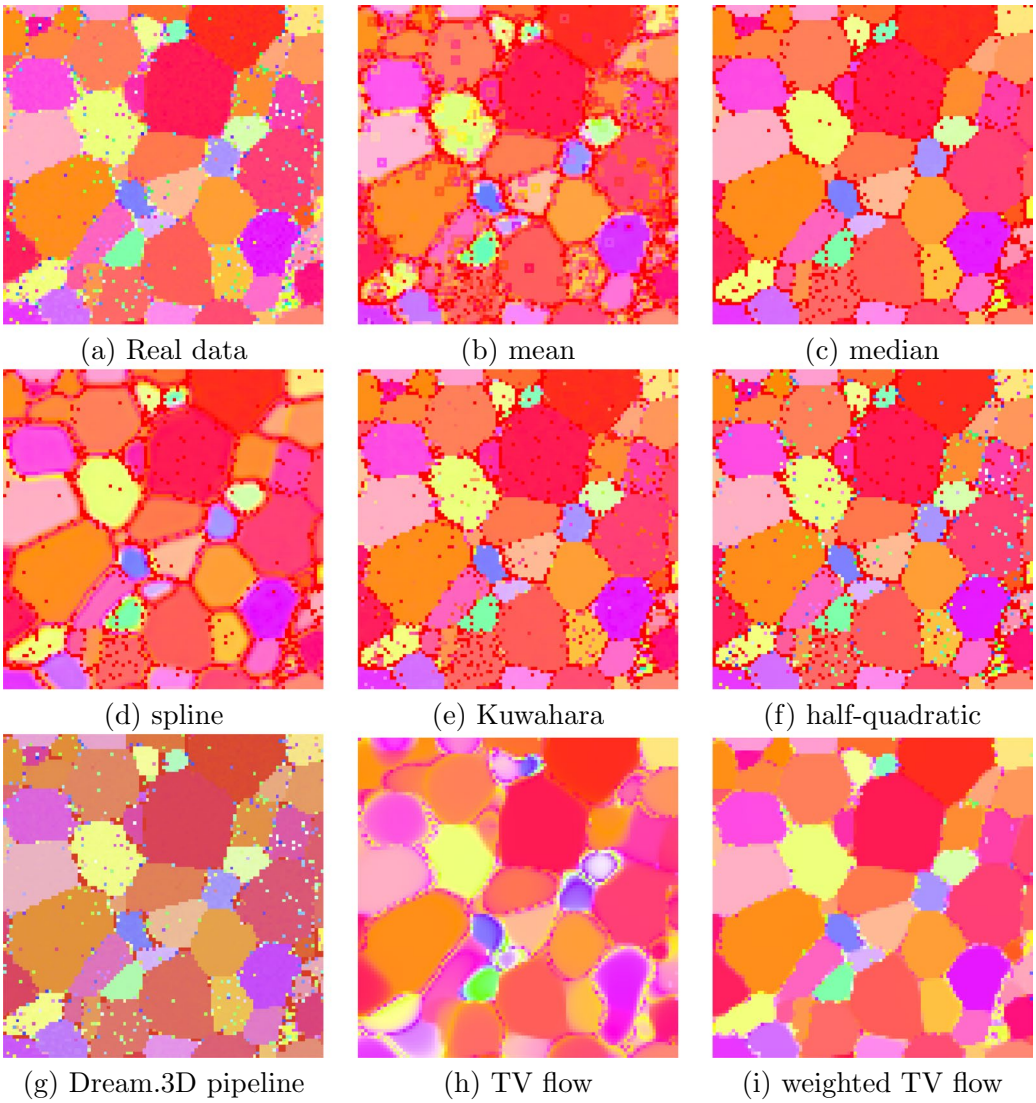
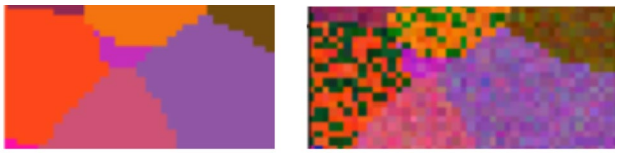


Fig. 4 Details of the IPF-z maps for the results on a selected region of a real EBSD map from Fig. 7 are shown here

Conclusion

Our goal in this paper was to restore EBSD orientation data containing orientation noise and misindexing errors. To this effect, we proposed using weighted vector-valued TV flow with an automatic stopping criterion determined by an estimation of the noise variance. We compared the results obtained by the proposed algorithm with those from EBSD denoising algorithms in the current literature. The results show that the proposed algorithm significantly outperforms the methods compared in the paper for medium to high noise levels. An important advantage of the proposed algorithm is that there are no parameters to be set by the user, and the stopping criterion is determined directly from the noisy data itself. Moreover, the "weight" used in the TV flow retained the grain boundaries effectively, namely, did not blur them like the mean filter. This feature is crucial for EBSD maps, where the grain boundaries carry crucial structural information about the material. Finally, we emphasize that even though we used EBSD maps to demonstrate effectiveness, the proposed algorithm is independent of the modality used. That is, the proposed algorithm applies to various other modalities, such as the automated crystal orientation mapping inside the transmission electron microscope (ACOM-TEM), X-ray diffraction (XRD), and High-Energy X-ray Diffraction Microscopy (HEDM).



(a) (b)

Fig. 5 This figure shows the Euler angles in multiple grains of an example EBSD map as an RGB image. Image **a** shows the ground truth orientations. Image **b** shows the noisy data with up to 15° error in Euler angles. The green pixels show the jumps due to the angle discontinuity around 360° − 0°

Appendix A: The Preprocessing Algorithm

There are two main sources of jump discontinuities in the orientation data: non-unique representations of orientations due to crystal symmetry, and limited range of Euler angle representation. To address the non-unique representation owing to the crystal symmetry, we project the orientations data onto the fundamental region. Note that for each orientation in the data, there is a unique symmetrically equivalent orientation within the fundamental region. In the remainder of the section, we describe the preprocessing algorithm we propose to identify and correct the jump discontinuities due to the range of the Euler angle representation of the data.

The Euler angles range from 0° to 360° (or 180°), depending on the axes representation used. For the range [0, 360), the jump discontinuities around angles 0° and 360° can cause misorientation errors. As example, consider that an Euler angle is 350°, which due to noise of 15° changes to 350° + 15° = 365°. However, since 365° is not in the range [0°, 360°), it is recorded as the equivalent angle of 5° causing a jump discontinuity within a grain, with a large error of (350° − 5°) = 345°, rather than the

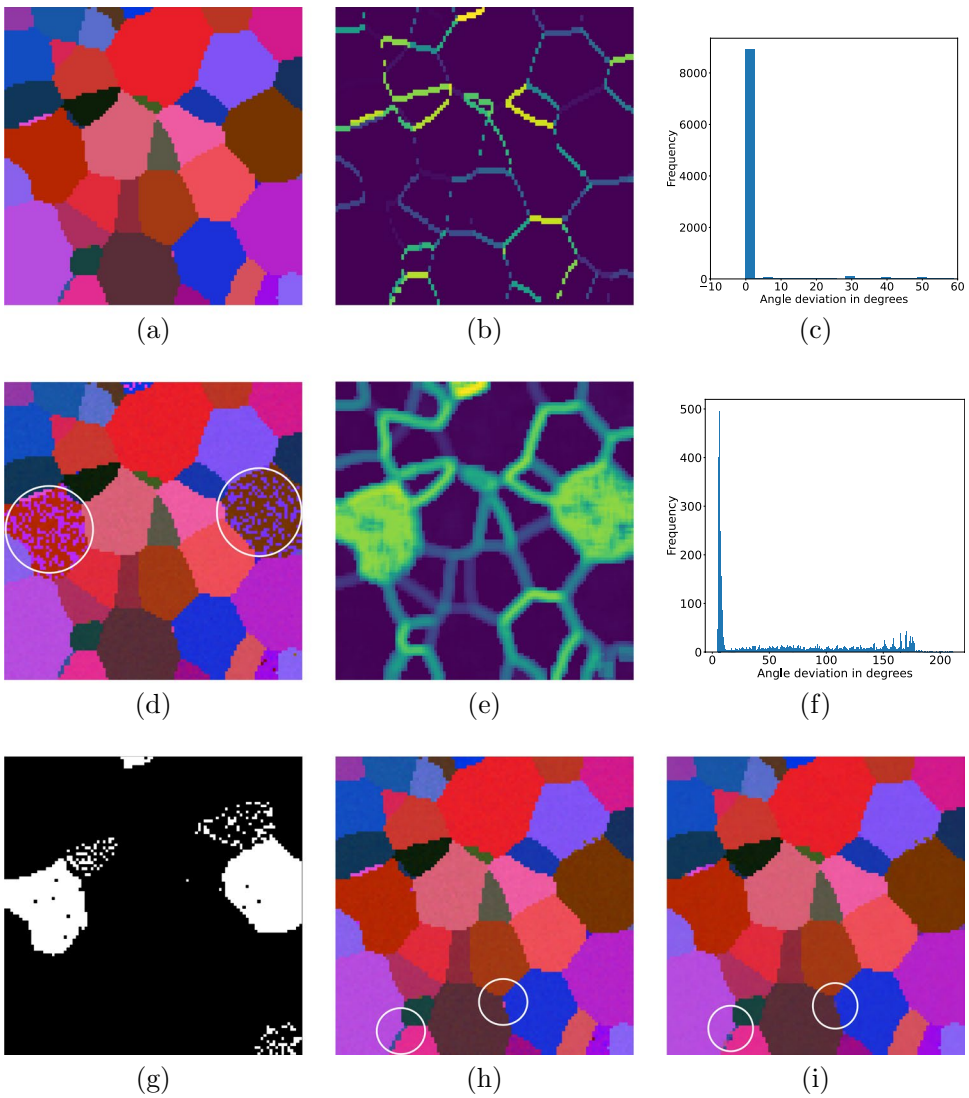


Fig. 6 Illustration of processing of noise due to jump discontinuities in the Euler angles: **a** clean image, **b** image of local standard deviations, **c** standard deviation histogram of clean data, **d** noisy image; the grains with jump discontinuities are highlighted in circles **e** local standard deviations of noisy data, **f** standard deviation histogram of noisy data, **g** region with jump discontinuities, **h** image with corrected jump discontinuities, **i** image with corrected isolated pixels. In some regions in **h**, the isolated pixels were corrected, and the corresponding corrected regions in **i** are shown in circles

original small error of 15°. Fig. 5b shows the jump discontinuities in Euler angles arising from adding 15° disorientation angle error to data in Fig. 5a. Hence, before applying a denoising algorithm, we need to correct the jump discontinuities at 0° and 360° in the data using a preprocessing algorithm. Note that the jump discontinuities are the feature of Euler angles, all images in this Appendix depict normalized Euler angles and are visualized as red, green, and blue (RGB) channels.

We demonstrate the steps of this algorithm in Fig. 6. We depict a clean EBSD map example obtained from DREAM.3D software [22, 31] in Fig. 6a. The EBSD maps contain orientations as Euler angles in Bunge notation i.e. *ZXZ* format [32]. Figure 6b shows the local standard deviations computed in the neighborhood of a 3×3 window centered at each data point. The brighter spots in Fig. 6b correspond to larger values of the local standard deviations. In Fig. 6c, we see the histogram of the local standard deviations. The orientation values within a single grain of clean EBSD data are comparable, as seen in Fig. 6a. Thus, we expect the local standard deviation near each data point within a single grain to be close to 0. Indeed, this phenomenon is apparent in the histogram of the local standard deviations shown in Fig. 6c. Around 90% of the local standard deviations in Fig. 6c are equal to 0, an effect more pronounced due to the map being synthetically generated.

Figure 6d depicts the EBSD data with de la Vallée Poussin noise [26] with a half-width $b = 8$. We compute local standard deviations of the noisy data as shown in Fig. 6 e, with its corresponding histogram in Fig. 6f. Note that Fig. 6e also contains the grain boundaries. The distribution of the local standard deviation is multimodal, with one prominent peak corresponding to the misorientation noise in the EBSD data. The other peaks correspond to the jump discontinuities showing high local standard deviation. To separate the misorientation noise from the jump discontinuities and grain boundaries, we identify the end of the prominent peak. Figure 6g shows the locations with local standard deviations higher than the endpoint of the prominent peak. The Euler angles typically range from 0° to 360° (or 180°). If the range is from 0° to 360°, the two angles are equivalent. In such cases, we only need to identify points corresponding to the jump discontinuities at 0° and 360°. The jump discontinuities identified by our algorithm are depicted in Fig. 6 g. If the angle θ at (x, y), a point of discontinuity, is near 0° (or 360°) while majority of its neighbors are near 360° (or 0°), we correct the angle θ to $360 - \theta$, otherwise θ is unchanged. Fig. 6 h shows the EBSD data with the jump discontinuities corrected. The correction of jump discontinuities does not always capture all the discontinuous data points, and the remaining discontinuities may appear as misindexed points. We apply the algorithm given by Brewer and Michael [9] to identify misindexed points. We label a point misindexed if its angle differs from all its local neighbors by 5° or more. We then replace these misindexed data points with the local median. Figure 6 i depicts the image with the correction of isolated pixels. The circles in Fig. 6 i highlight some regions with these corrections.

Appendix B: Numerical Scheme

For the weighted TV flow,

$$\frac{\partial u^{(k)}}{\partial t} = \frac{\|u^{(k)}\|_{TV_\alpha(\Omega)}}{\|\mathbf{u}\|_{TV_\alpha(\Omega)}} \operatorname{div}\left(\frac{\alpha^{(k)} \nabla u^{(k)}}{|\nabla u^{(k)}|}\right) \quad \text{for } k = 1, 2, 3.$$

For a single channel, we have

$$\begin{aligned}
\operatorname{div}\left(\frac{\alpha \nabla u}{|\nabla u|}\right) &= D_{-x}\left[\frac{\alpha_{i,j} D_{+x} u}{\sqrt{\varepsilon^2 + (D_{+x}u_{i,j})^2 + (D_{0y}u_{i,j})^2}}\right] \\
&\quad + D_{-y}\left[\frac{\alpha_{i,j} D_{+y} u}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j})^2 + (D_{+y}u_{i,j})^2}}\right] \\
&= D_{-x}\left[\frac{\alpha_{i,j}(u_{i+1,j} - u_{i,j})}{\sqrt{\varepsilon^2 + (D_{+x}u_{i,j})^2 + (D_{0y}u_{i,j})^2}}\right] \\
&\quad + D_{-y}\left[\frac{\alpha_{i,j}(u_{i,j+1} - u_{i,j})}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j})^2 + (D_{+y}u_{i,j})^2}}\right].
\end{aligned}$$

Note,

$$
\begin{aligned}
&D_{-x}\left[\frac{\alpha_{i,j}D_{+x}u}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=D_{-x}\left[\frac{\alpha_{i,j}(u_{i+1,j}-u_{i,j})}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=D_{-x}\left[\frac{\alpha_{i,j}u_{i+1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j,k})^2}}\right]\\
&\qquad-D_{-x}\left[\frac{\alpha_{i,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]
\end{aligned}
$$

Apply backward difference on each term separately we obtain

$$
\begin{aligned}
&D_{-x}\left[\frac{\alpha_{i,j}u_{i+1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=\frac{\alpha_{i,j}u_{i+1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\\
&\qquad-\frac{\alpha_{i-1,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i-1,j})^2+(D_{0y}u_{i-1,j})^2}},
\end{aligned}
$$

and

$$
\begin{aligned}
&D_{-x}\left[\frac{\alpha_{i,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=\frac{\alpha_{i,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\\
&\qquad-\frac{\alpha_{i-1,j}u_{i-1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i-1,j})^2+(D_{0y}u_{i-1,j})^2}}.
\end{aligned}
$$

Note that $D_{+x}u_{i-1,j}=D_{-x}u_{i,j}$ and thus,

$$
\begin{aligned}
&D_{-x}\left[\frac{\alpha_{i,j}u_{i+1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=\frac{\alpha_{i,j}u_{i+1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\\
&\qquad-\frac{\alpha_{i-1,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{-x}u_{i,j})^2+(D_{0y}u_{i-1,j})^2}},\\
&D_{-x}\left[\frac{\alpha_{i,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=\frac{\alpha_{i,j}u_{i,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\\
&\qquad-\frac{\alpha_{i-1,j}u_{i-1,j}}{\sqrt{\varepsilon^2+(D_{+x}u_{i-1,j})^2+(D_{0y}u_{i-1,j})^2}},
\end{aligned}
$$

and,

$$
\begin{aligned}
&D_{-x}\left[\frac{\alpha_{i,j}(u_{i+1,j}-u_{i,j})}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\right]\\
&\quad=\frac{\alpha_{i,j}(u_{i+1,j}-u_{i,j})}{\sqrt{\varepsilon^2+(D_{+x}u_{i,j})^2+(D_{0y}u_{i,j})^2}}\\
&\qquad-\frac{\alpha_{i-1,j}(u_{i,j}-u_{i-1,j})}{\sqrt{\varepsilon^2+(D_{-x}u_{i,j})^2+(D_{0y}u_{i-1,j})^2}}.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\operatorname{div}\left(\frac{\alpha_{i,j}\nabla u^k}{|\nabla u|}\right)\\
&\quad= D_{-x}\left[\frac{\alpha_{i,j}(u^k_{i+1,j}-u^k_{i,j})}{\sqrt{\varepsilon^2+(D_{+x}u^k_{i,j})^2+(D_{0y}u^k_{i,j})^2}}\right]\\
&\qquad+D_{-y}\left[\frac{\alpha_{i,j}(u^k_{i,j+1}-u^k_{i,j})}{\sqrt{\varepsilon^2+(D_{0x}u^k_{i,j})^2+(D_{+y}u^k_{i,j})^2}}\right]\\
&\quad=\left[\frac{\alpha_{i,j}(u^k_{i+1,j}-u^k_{i,j})}{\sqrt{\varepsilon^2+(D_{+x}u^k_{i,j})^2+(D_{0y}u^k_{i,j})^2}}\right.\\
&\qquad\left.-\frac{\alpha_{i-1,j}(u^k_{i,j}-u^k_{i-1,j})}{\sqrt{\varepsilon^2+(D_{-x}u^k_{i,j})^2+(D_{0y}u^k_{i-1,j})^2}}\right]\\
&\qquad+\left[\frac{\alpha_{i,j}(u^k_{i,j+1}-u^k_{i,j})}{\sqrt{\varepsilon^2+(D_{0x}u^k_{i,j})^2+(D_{+y}u^k_{i,j})^2}}\right.\\
&\qquad\left.-\frac{\alpha_{i,j-1}(u^k_{i,j}-u^k_{i,j-1})}{\sqrt{\varepsilon^2+(D_{0x}u^k_{i,j-1})^2+(D_{-y}u^k_{i,j})^2}}\right]\\
&\quad=\left[\alpha_{i,j}C_E(u^k_{i+1,j}-u^k_{i,j})-\alpha_{i-1,j}C_W(u^k_{i,j}-u^k_{i-1,j})\right]\\
&\qquad+\left[\alpha_{i,j}C_S(u^k_{i,j+1}-u^k_{i,j})-\alpha_{i,j-1}C_N(u^k_{i,j}-u^k_{i,j-1})\right]\\
&\quad=\alpha_{i-1,j}C_Wu^k_{i-1,j}+\alpha_{i,j}(C_Eu^k_{i+1,j}+C_Su^k_{i,j+1})\\
&\qquad+\alpha_{i,j-1}C_Nu^k_{i,j-1}-(\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S\\
&\qquad+\alpha_{i,j-1}C_N)u^k_{i,j}
\end{aligned}
$$

where

$$
\begin{aligned}
C_E&=\frac{1}{\sqrt{\varepsilon^2+(D_{+x}u^k_{i,j})^2+(D_{0y}u^k_{i,j})^2}},\\
C_W&=\frac{1}{\sqrt{\varepsilon^2+(D_{-x}u^k_{i,j})^2+(D_{0y}u^k_{i-1,j})^2}},\\
C_S&=\frac{1}{\sqrt{\varepsilon^2+(D_{0x}u^k_{i,j})^2+(D_{+y}u^k_{i,j})^2}},\\
C_N&=\frac{1}{\sqrt{\varepsilon^2+(D_{0x}u^k_{i,j-1})^2+(D_{-y}u^k_{i,j})^2}},
\end{aligned}
$$

for $k=1,2,3$. Thus,

$$
\begin{aligned}
\frac{\partial u^{(k)}}{\partial t}&=\frac{\|u^{(k)}\|_{TV_\alpha(\Omega)}}{\|\mathbf{u}\|_{TV_\alpha(\Omega)}}\operatorname{div}\left(\frac{\alpha^{(k)}\nabla u^{(k)}}{|\nabla u^{(k)}|}\right)\\
&=R(u^k)\operatorname{div}\left(\frac{\alpha^{(k)}\nabla u^{(k)}}{|\nabla u^{(k)}|}\right)
\end{aligned}
$$

is discretized as

$$
\begin{aligned}
\frac{u^{k,n+1}_{i,j}-u^{k,n}_{i,j}}{\tau}&=R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}+C_Su^{k,n}_{i,j+1})\\
&\quad+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}-(\alpha_{i,j}C_E+\alpha_{i-1,j}C_W\\
&\quad+\alpha_{i,j}C_S+\alpha_{i,j-1}C_N)u^{k,n+1}_{i,j}\Big]\\
u^{k,n+1}_{i,j}&=u^{k,n}_{i,j}+\tau R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}+C_Su^{k,n}_{i,j+1})\\
&\quad+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}-(\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S\\
&\quad+\alpha_{i,j-1}C_N)u^{k,n+1}_{i,j}\Big]
\end{aligned}
$$

Note, $\|\boldsymbol{u}\|_{TV(\Omega)}:=\left[\sum_{k=1}^3\left(\int_\Omega|\nabla u^{(k)}|\right)^2\right]^{\frac{1}{2}}=\left[\sum_{k=1}^3\left(\sum_{i,j}|\nabla u^k_{i,j}|\right)^2\right]^{\frac{1}{2}}$ and $\|u^k\|_{TV(\Omega)}:=\int_\Omega|\nabla u^k|=\sum_{i,j}|\nabla u^k_{i,j}|$ Thus,

$$
R(u^{k,n})=\frac{\|u^{(k)}\|_{TV_\alpha(\Omega)}}{\|\mathbf{u}\|_{TV_\alpha(\Omega)}}=\frac{\sum_{i,j}|\nabla u^k_{i,j}|}{\left[\sum_{k=1}^3\left(\sum_{i,j}|\nabla u^k_{i,j}|\right)^2\right]^{\frac{1}{2}}}
$$

Hence,

$$
\begin{aligned}
u^{k,n+1}_{i,j}&=u^{k,n}_{i,j}+\tau R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}\\
&\quad+C_Su^{k,n}_{i,j+1})+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}\\
&\quad-(\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S+\alpha_{i,j-1}C_N)u^{k,n+1}_{i,j}\Big]\\
\text{i.e. } u^{k,n+1}_{i,j}&+(\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S+\alpha_{i,j-1}C_N)u^{k,n+1}_{i,j}\\
&=u^{k,n}_{i,j}+\tau R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}+C_Su^{k,n}_{i,j+1})+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}\Big]\\
\text{i.e. } &(1+\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S+\alpha_{i,j-1}C_N)u^{k,n+1}_{i,j}\\
&=u^{k,n}_{i,j}+\tau R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}+C_Su^{k,n}_{i,j+1})+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}\Big]
\end{aligned}
$$

This leads to the final discretization as follows

$$
u^{k,n+1}_{i,j}=\frac{u^{k,n}_{i,j}+\tau R(u^{k,n})\Big[\alpha_{i-1,j}C_Wu^{k,n}_{i-1,j}+\alpha_{i,j}(C_Eu^{k,n}_{i+1,j}+C_Su^{k,n}_{i,j+1})+\alpha_{i,j-1}C_Nu^{k,n}_{i,j-1}\Big]}{(1+\alpha_{i,j}C_E+\alpha_{i-1,j}C_W+\alpha_{i,j}C_S+\alpha_{i,j-1}C_N)}
$$

Appendix C: Experiments with the Real Data

See Figure 7.

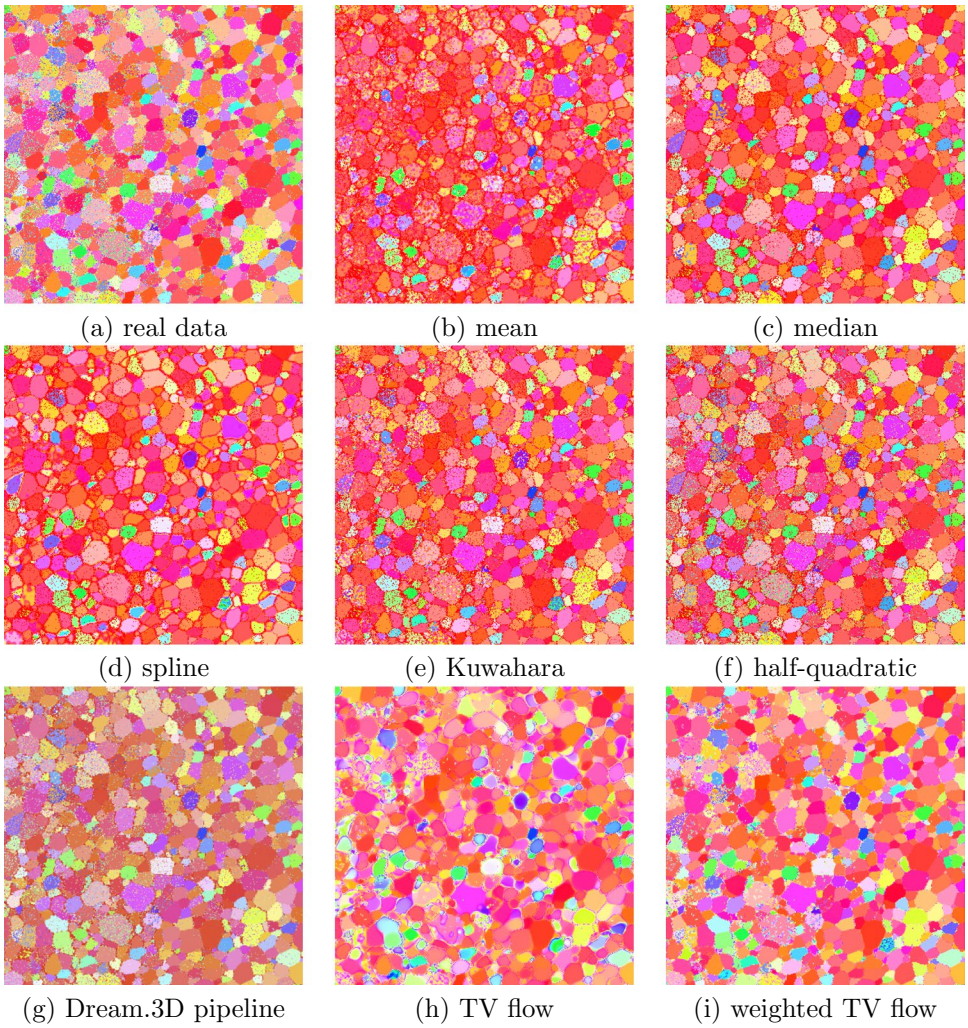


Fig. 7 Results of denoising real EBSD map using various filters

Appendix D: Grain Boundaries After Restoration

In this section, we demonstrate the effect of the algorithms on the boundaries of the restored EBSD maps. We found the location of the boundaries using the Farid edge detector [20]. We see in Fig. 8 the details of the boundaries of ESBD maps shown in Fig. 2. Figure 9 shows the details of the boundaries of maps in Fig. 3. The yellow color in Figs. 8 and 9 indicates the locations where the boundaries from the restored maps match the true boundaries exactly. The red color indicated the true boundaries from the clean EBSD map. Thus, the red-colored pixels in images (b) - (h) in Figs. 8 and 9 show locations of the true boundaries that are missed by the algorithms. The green boundaries are the ones obtained from the restored algorithms. Thus, the green-colored boundaries in images (b) to (h) in Figs. 8 and 9 are the incorrect locations of the boundaries found by the corresponding algorithms. We can infer from these images that the boundaries found by the proposed algorithm are comparable in their accuracy with other algorithms discussed in this paper.

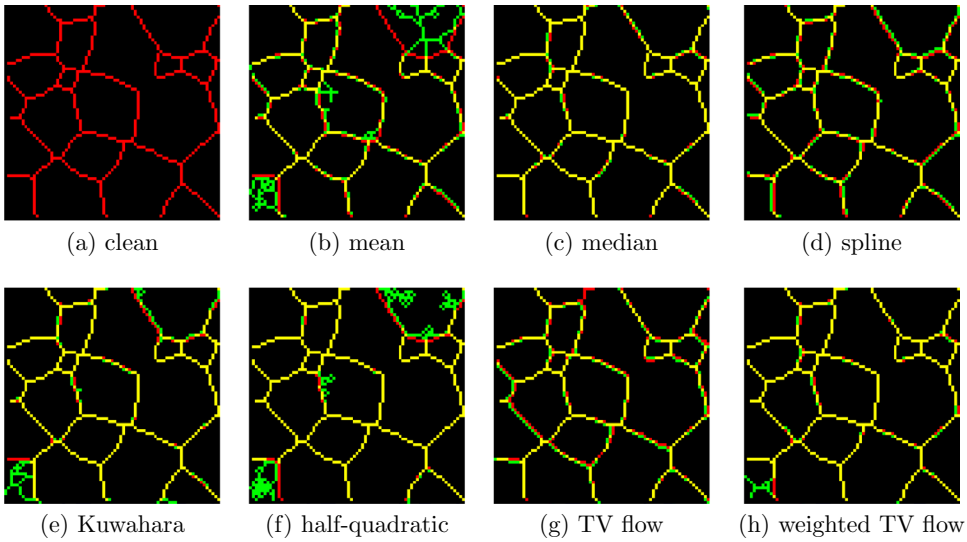


Fig. 8 Details of the grain boundaries of EBSD map from Fig. 2. The yellow color represents accurate boundary detection. Red represents true grain boundaries that are missed by the algorithms. Green represents boundaries obtained by the corresponding algorithms which do not match the ground truth

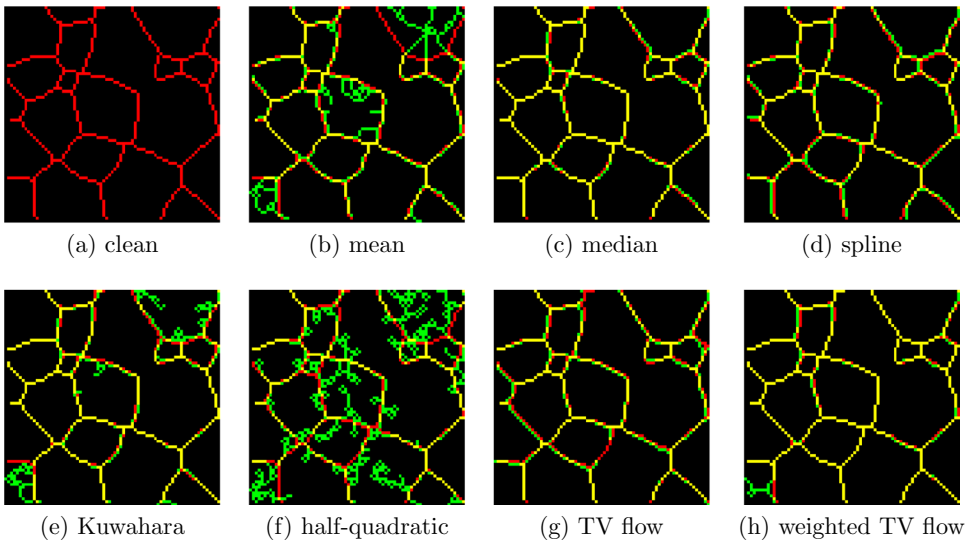


Fig. 9 Details of the grain boundaries of EBSD map from Fig. 3. The yellow color represents accurate boundary detection. Red represents true grain boundaries that are missed by the algorithms. Green represents boundaries obtained by the corresponding algorithms which do not match the ground truth

Acknowledgements The NIST SURF program in the Summer of 2021 supported Peter Lef. Real EBSD maps were courtesy of Dr. Adam Creuziger from NIST, Gaithersburg, MD.

Funding This research was funded by NIST grant # 70NANB21H047.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

NIST Disclaimer Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement of any product or service by NIST, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

1. Zaefferer S (2011) A critical review of orientation microscopy in SEM and TEM. Cryst Res Technol 46(6):607–628. https://doi.org/10.1002/crat.201100125
2. Rauch EF, Portillo J, Nicolopoulos S, Bultreys D, Rouvimov S, Moeck P (2010) Automated nanocrystal orientation and phase mapping in the transmission electron microscope on the basis of precession electron diffraction. Zeitschrift für Kristallographie-Cryst Mater 225(2–3):103–109. https://doi.org/10.1524/zkri.2010.1205
3. Epp J (2016) 4 - X-ray diffraction (XRD) techniques for materials characterization. In: Hübschen G, Altpeter I, Tschuncky R, Herrmann H-G (eds) Materials characterization using nondestructive evaluation (NDE) methods. Woodhead Publishing, Sawston, UK, pp 81–124. https://doi.org/10.1016/B978-0-08-100040-3.00004-3
4. Poulsen HF (2004) Three-dimensional X-ray diffraction microscopy: mapping polycrystals and their dynamics, 1st edn. Springer, Berlin, Heidelberg. https://doi.org/10.1007/b97884

5. Nolze G (2015) Euler angles and crystal symmetry. Cryst Res Technol 50(2):188–201. https://doi.org/10.1002/crat.201400427
6. Creuziger A, Vaudin M (2011) Report on VAMAS round robin of ISO 13067: Microbeam analysis - electron backscatter diffraction - measurement of average grain size. Technical Report NISTIR 7814, National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.7814
7. Langer SA, Fuller ER, Carter WC (2001) OOF: an image-based finite-element analysis of material microstructures. Comput Sci Eng 3(3):15–23. https://doi.org/10.1109/5992.919261
8. Arfken G (1967) Mathematical methods for physicists. Am J Phys 35(11):1097–1098. https://doi.org/10.1119/1.1973757
9. Brewer L, Michael J (2010) Risks of cleaning electron backscatter diffraction data. Microsc Today 18(2):10–15. https://doi.org/10.1017/S1551929510000040
10. Gonzalez RC, Woods RE (2018) Digital image processing, 4th edn. Pearson Publishing, London, UK
11. Kyprianidis JE (2011) Image and video abstraction by multi-scale anisotropic Kuwahara filtering. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering. ACM Press, Vancouver, British Columbia, Canada. https://doi.org/10.1145/2024676.2024686
12. Charbonnier P, Blanc-Feraud L, Aubert G, Barlaud M (1994) Two deterministic half-quadratic regularization algorithms for computed imaging. In: Proceedings of 1st International Conference on Image Processing, vol 2. IEEE, pp 168–172. https://doi.org/10.1109/icip.1994.413553
13. Immerkær J (1996) Fast noise variance estimation. Comput Vis Image Underst 64(2):300–302. https://doi.org/10.1006/cviu.1996.0060
14. Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. Phys D Nonlinear Phenom 60(1–4):259–268. https://doi.org/10.1016/0167-2789(92)90242-f
15. Athavale P, Xu R, Radau P, Nachman A, Wright GA (2015) Multiscale properties of weighted total variation flow with applications to denoising and registration. Med Image Anal 23(1):28–42. https://doi.org/10.1016/j.media.2015.04.013
16. Blomgren P, Chan TF (1998) Color TV: total variation methods for restoration of vector-valued images. IEEE Trans Image Process 7(3):304–309. https://doi.org/10.1109/83.661180
17. Meyer Y (2001) Oscillating Patterns in Image Processing and Nonlinear Evolution Equations: the Fifteenth Dean Jacqueline B. Lewis Memorial Lectures, vol 22. American Mathematical Society, Providence, RI, USA
18. Chambolle A (2004) An algorithm for total variation minimization and applications. J Math Imag Vis 20(1–2):89–97. https://doi.org/10.1023/B:JMIV.0000011325.36760.1e
19. Athavale P, Jerrard RL, Novaga M, Orlandi G (2017) Weighted TV minimization and applications to vortex density models. J Convex Anal 24(4):1051–1084. https://doi.org/10.48550/arXiv.1509.03713
20. Farid H, Simoncelli EP (2004) Differentiation of discrete multi-dimensional signals. IEEE Trans Image Process 13(4):496–508. https://doi.org/10.1109/TIP.2004.823819
21. Lee T-C, Kashyap RL, Chu C-N (1994) Building skeleton models via 3-d medial surface axis thinning algorithms. CVGIP Graph Models Image Process 56(6):462–478. https://doi.org/10.1006/cgip.1994.1042
22. Groeber MA, Jackson MA (2014) DREAM.3D: a digital representation environment for the analysis of microstructure in 3D. Integr Mater Manuf Innov 3:56–72. https://doi.org/10.1186/2193-9772-3-5
23. Morawiec A (2003) Orientations and rotations. Springer, Berlin, Germany
24. Hielscher R, Silbermann CB, Schmidl E, Ihlemann J (2019) Denoising of crystal orientation maps. J Appl Crystallogr 52(5):984–996. https://doi.org/10.1107/S1600576719009075
25. Bachmann F, Hielscher R, Jupp PE, Pantleon W, Schaeben H, Wegert E (2010) Inferential statistics of electron backscatter diffraction data from within individual crystalline grains. J Appl Crystallogr 43(6):1338–1355. https://doi.org/10.1107/S0021889810030276X
26. Schaeben H (1999) The de la vallée poussin standard orientation density function. Textures Microstruct 33(1–4):365–373. https://doi.org/10.1155/TSM.33.365
27. MTEX (2020) MTEX Toolbox. https://mtex-toolbox.github.io/. Online; accessed September 9, 2020
28. MTEX (2021) Denoising Orientation Maps. https://mtex-toolbox.github.io/EBSDDenoising.html. Online; accessed December 14, 2021
29. Neter J, Kutner MH, Nachtsheim CJ, Wasserman W (1996) Applied linear statistical models. McGraw Hill/Irwin Series
30. Creuziger A (2023) Real EBSD map examples. Accessed: 2023-03-15. https://doi.org/10.18434/mds2-2951
31. Dream.3D (2020) Dream.3D. http://dream3d.bluequartz.net/. Online, accessed September 9, 2020
32. Bunge H-J (2013) Texture analysis in materials science: mathematical methods. Elsevier, Amsterdam, Netherlands