JOURNAL OF APPLIED CRYSTALLOGRAPHY

Volume 48 (2015)

Supporting information for article:

Removing multiple outliers and single-crystal artefacts from X-ray diffraction computed tomography data

Antonios Vamvakeros, Simon D. M. Jacques, Marco Di Michiel, Vesna Middelkoop, Christopher K. Egan, Robert J. Cernik and Andrew M. Beale

Removing multiple outliers and single crystal artefacts from X-ray diffraction computed tomography data

Authors

Antonios Vamvakeros[ab]*, Simon D. M. Jacques[abc]*, Marco Di Michiel[d], Vesna Middelkoop[e], Christopher K. Egan[c], Robert J. Cernik[c] and Andrew M. Beale[ab]*

[a]Department of Chemistry, University College London, 20 Gordon Street, London, WC1H 0AJ, UK

[b]UK Catalysis Hub, Research Complex at Harwell, Didcot, Oxfordshire

[c]School of Materials, University of Manchester, Manchester, Lancashire, M13 9PL, UK

[d]ESRF - The European Synchrotron, Grenoble, F-38000, France

[e]Flemish Institute for Tecnological Research, VITO NV, Boeretang 200, Mol, Belgium

Correspondence email: antonyvam@gmail.com;simon.jacques@gmail.com; andrew.beale@ucl.ac.uk

XRD-CT method

A schematic representation of an XRD-CT experiment is provided in Figure S1. The method relies on a pencil beam scanning approach using a highly collimated or focussed monochromatic beam with, for best counting statistics/speed, scattered X-rays recorded on an area detector. This is typically normal to and centred with respect to the beam. Typically, the object is translated across the X-ray beam (i.e. perpendicular to the beam axis) with a step size close/same as the horizontal size of the incoming X-ray beam and for every translational step *t*, diffraction patterns are collected using an area detector. The length of the translational scan can be the same as the width of the sample but in practise the value for the length *s* used is higher (i.e. number of translational steps $n = s / t$). This is because one should take into account the imperfection of the sample alignment and also the fact that the sample may move during successive tomographic collections. After a translational scan is completed, the sample is rotated (angular step) and the translational scan is repeated. The angular range usually covered is from 0 to π and the number of angles scanned *m* should be, as in the case of the first generation X-ray computed tomography, equal to the number of translational measurements times $\pi/2$ (i.e. $m = n \times \pi/2$). However, in practice, the number of angular steps can be decreased without significant changes in the quality of the collected data (Álvarez-Murga *et al.*, 2012).

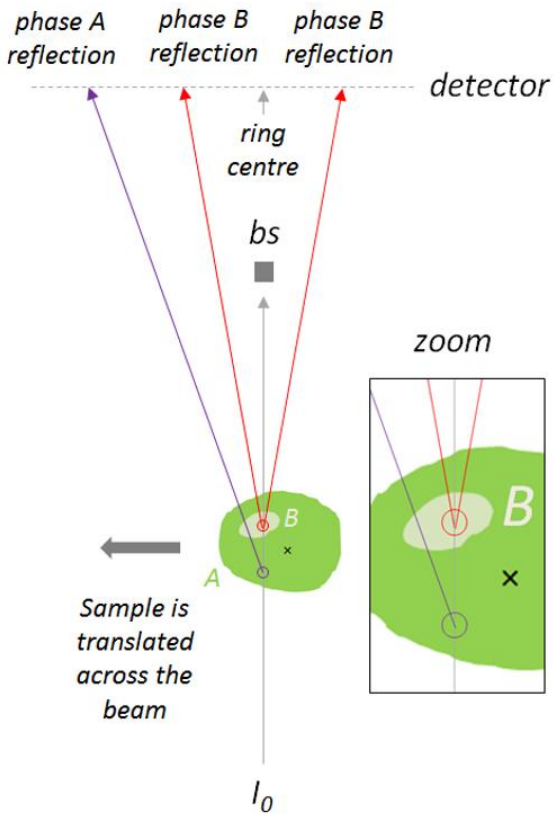

Figure S1: Tomographic plane schematics of XRD-CT experiment. In the schematic the green object consists primarily of phases A (coloured green) with inclusion of a small amount of phase B (coloured light grey/green). The incident beam is denoted I_0, the detector is protected from the incident beam by use of a backstop (denoted bs in the schematics). As the object is translated across the beam, diffraction is recorded from the entire path of the incident beam. If A and B are powders, characteristic Bragg reflections from both phases will be recorded on the detector manifesting as diffraction rings on the detector. This is illustrated in the figure where purple ray path indicates a reflection at high 2θ angles from phase A, and the red paths indicate a single reflection from phase B recorded at low 2θ angles on different sides of the detector ring centre.

Preparation of catalyst and membranes

2%Mn-1.6%Na-3.1%W/SiO_2 was prepared by a sequential incipient wetness impregnation method. Firstly, the SiO_2 support (Silica gel Davisil 646, ~ 250-500 μm) was impregnated by an aqueous solution of sodium tungstate dihydrate $Na_2WO_4 \cdot 2H_2O$ and sodium oxalate $Na_2C_2O_4$ salts taken in appropriate concentrations at a temperature of 80ºC. The Na-W/SiO_2 was dried at 120 ºC for 6 h and was then impregnated by an aqueous solution of manganese (II) acetate tetrahydrate $Mn(CH3COO)_2 \cdot 4H_2O$ salt. For the 2%La-2%Mn-1.6%Na-3.1%W/SiO_2 catalyst, the Na-W/SiO_2 was dried at 120 ºC for 6 h and was then impregnated by an aqueous solution of manganese (II) acetate tetrahydrate $Mn(CH3COO)_2 \cdot 4H_2O$ and lanthanum nitrate $La(NO_3)_3 \cdot 6H_2O$ salts taken in appropriate concentrations. The catalysts were then dried at 120°C for 6 h and calcined in air at 850 for 6 h with a heating rate of 2 ºC/min. The BCFZ membranes reported here have been manufactured using the spinning and phase inversion methods previously described.(Van Noyen *et al.*, 2012, Middelkoop *et al.*, 2014) The starting polymer suspension was prepared from cellulose acetate (CA, Mr ~52000, Fluka), dimethylsulphoxide (DMSO, Synthesis grade, Merck) and de-ionised water that were used as a phase-inversion polymer, solvent and non-solvent additive to the polymer solution, respectively.

Summed diffraction pattern

The summed diffraction pattern of the XRD-CT dataset is shown at the left side of Figure S2. The cristobalite (green), $BaWO_4$ (red), Mn_2O_3 (cyan) and BCFZ (magenta) peaks presented at the right side of Figure S2 correspond to the sinograms used in this work.

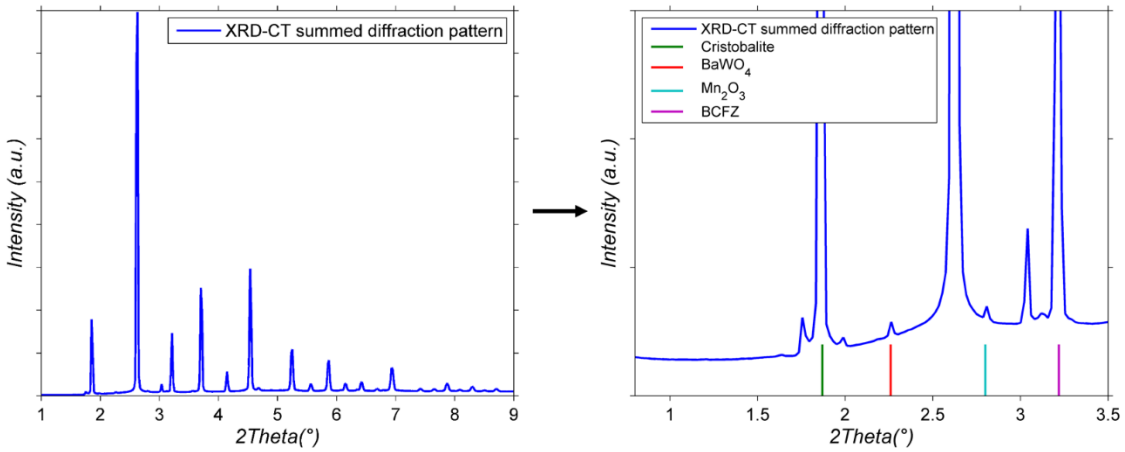

Figure S2: Left: The summed diffraction pattern from the XRD-CT dataset presented in the main paper. Right: A region of interest (ROI) of that diffraction pattern, showing four peaks corresponding to cristobalite (green), $BaWO_4$ (red), Mn_2O_3 (cyan) and BCFZ (magenta) phases.

Tomographic reconstruction algorithms

The effect of different reconstruction algorithms to deal with spotty sinograms is presented in Figure S3. The sinogram used corresponds to $BaWO_4$ also shown in Figure 3 in the main paper. The XRDUA software program was used to implement the various tomographic algorithms. Initially, the CeO_2 (NIST) diffraction pattern was used for calibration and then the XRD-CT data were processed (i.e. azimuthal integration of the raw 2D diffraction images and creation of the sinograms). As expected (see first row in Figure S8), the $BaWO_4$ sinogram and reconstructed XRD-CT image (i.e. using the filtered back projection algorithm) are identical with the ones generated using the MATLAB script used in this work. Different tomographic algorithms were implemented to reconstruct the spotty BaWO4 sinogram using the XRDUA software. More specifically, the algebraic reconstruction technique (ART), the simultaneous algebraic reconstruction technique (SART), the simultaneous iterative reconstruction technique (SIRT), the ordered subset expectation minimization (OSEM) and the maximum likelihood expectation maximization (MLEM) algorithms were tested. As it is shown in Figure S8, the reconstructed images are full of line artefacts regardless of the algorithm used (or how many iterations are used).

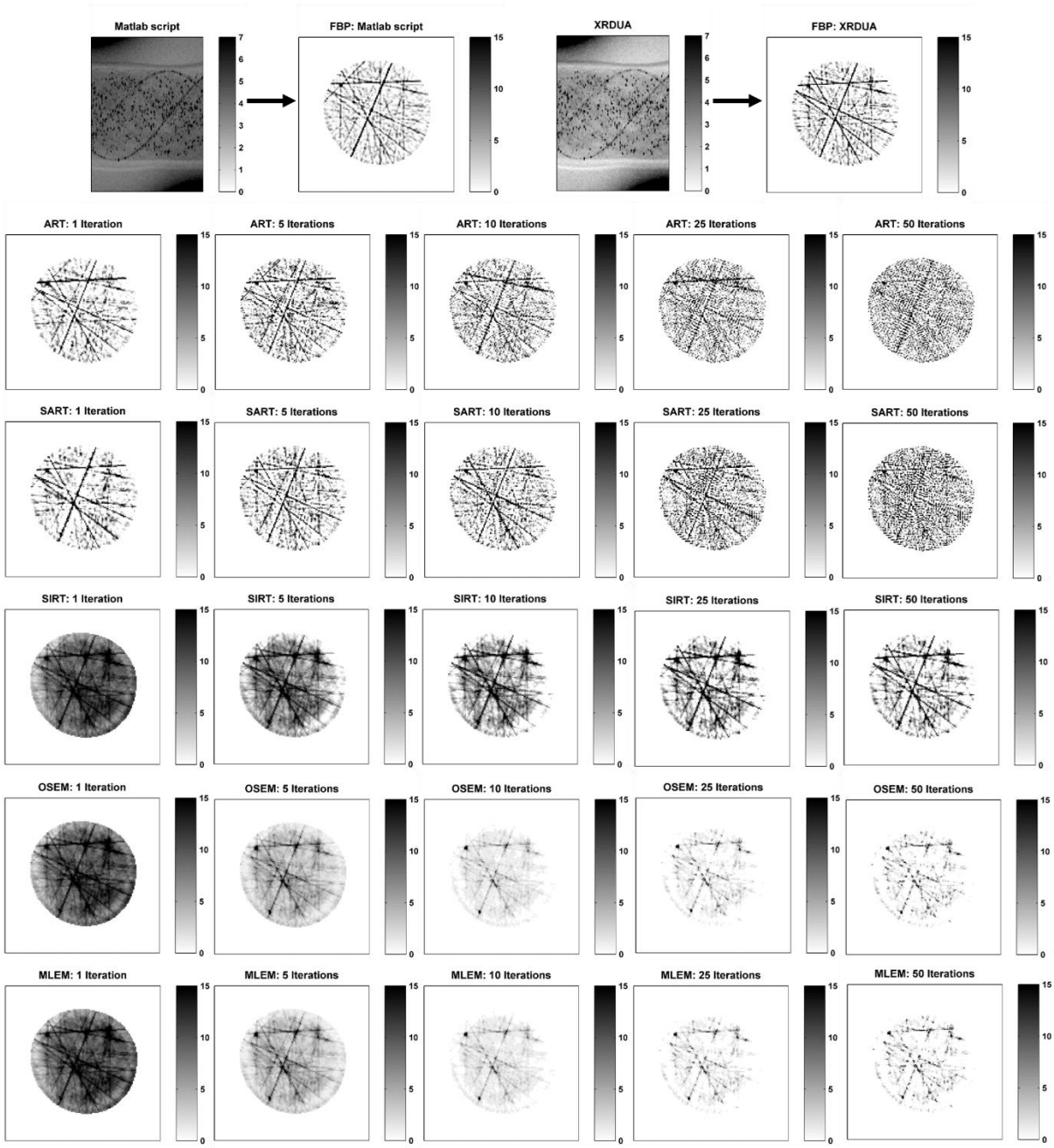

Figure S3: Top row: $BaWO_4$ sinograms and reconstructed images (FBP) using the MATLAB script provided in the ESI and the XRDUA software program. The reconstructed images when ART, SART, SIRT, OSEM, MLEM are used is also shown.

Alpha-trimmed mean filter

In Figure S4, the derived 1D diffraction patterns for the CeO_2 standard using different values for the alpha-trimmed mean are shown (i.e. 0, 1, 2, 3, 5, 10, 25, 50 and 75% respectively). The 2D diffraction image shown in Panel A of Figure 2 in the main text is used as the benchmark tool. As it is shown in Figure S4, there are no obvious changes in the 1D diffraction patterns using the different filters.

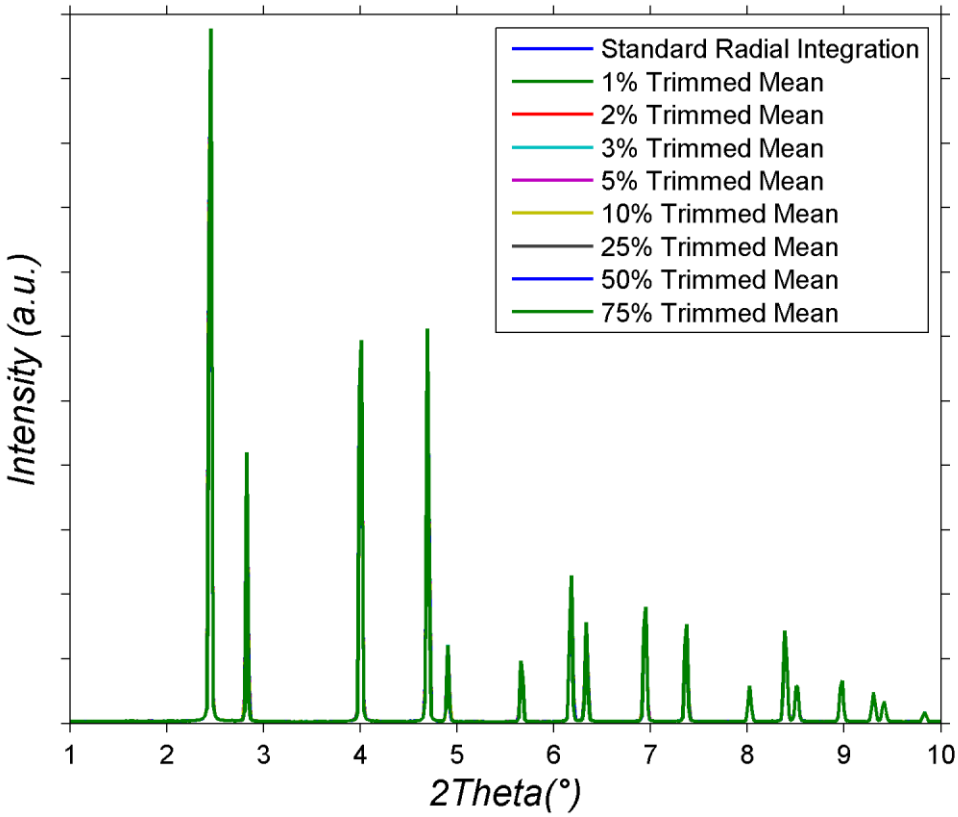

Figure S4: The 1D diffraction patterns for the CeO_2 standard are shown using different values for the alpha-trimmed mean filter.

In Figure S5, the relative difference between the 1D diffraction pattern calculated with standard radial integration and 1, 2 and 3% alpha-trimmed mean filter is presented. It can be seen that the values remain less than 10% even for the 3% alpha-trimmed mean filter.

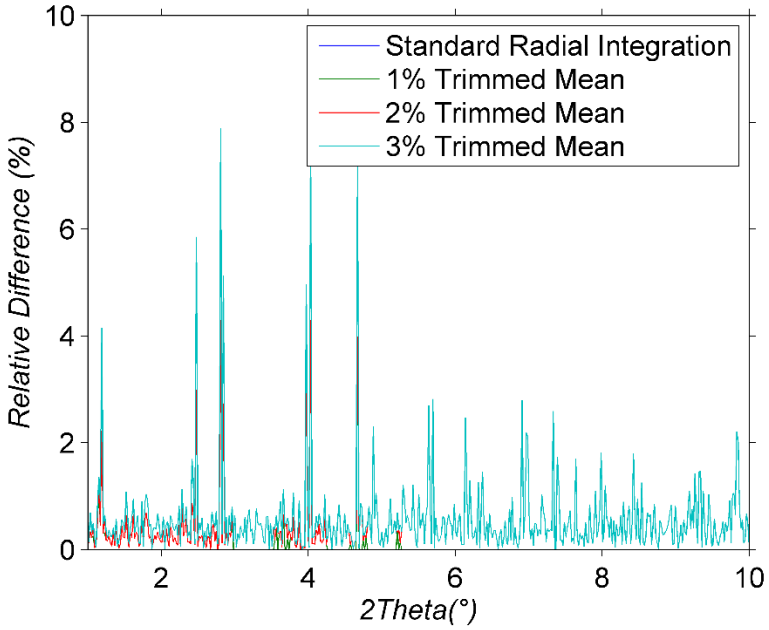

Figure S5: The relative difference between the 1D diffraction pattern calculated with standard radial integration and 1, 2 and 3% alpha-trimmed mean filter for the CeO_2 is shown.

In Figure S6, the relative difference between the 1D diffraction pattern calculated with standard radial integration and 10, 25, 50 and 75% alpha-trimmed mean filter is presented. As expected, the values increase and there are differences more than 50% for the 75% alpha-trimmed mean filter at certain scattering angles 2θ.

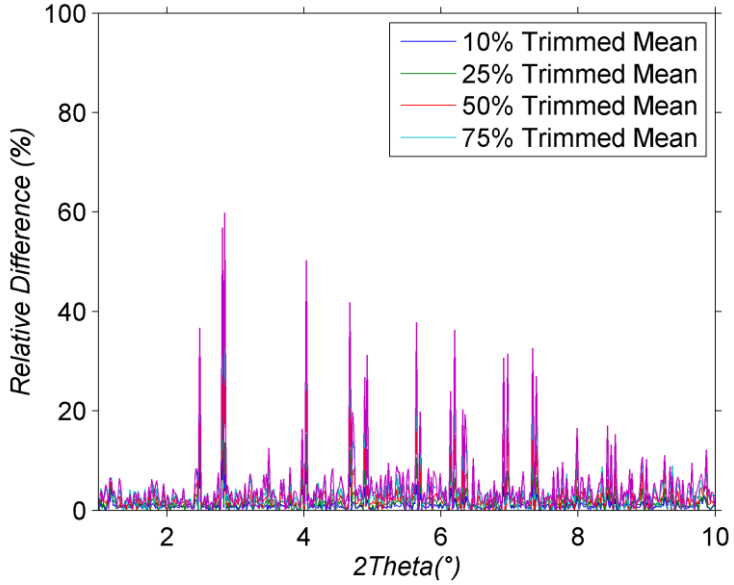

Figure S6: The relative difference between the 1D diffraction pattern calculated with standard radial integration and 1, 2 and 3% alpha-trimmed mean filter for the CeO_2 is shown

The maximum values of the relative difference between the 1D diffraction pattern calculated with standard radial integration and 1, 2, 3, 5, 10, 25, 50 and 75% alpha-trimmed mean filter for the CeO2 are presented in Figure S7.

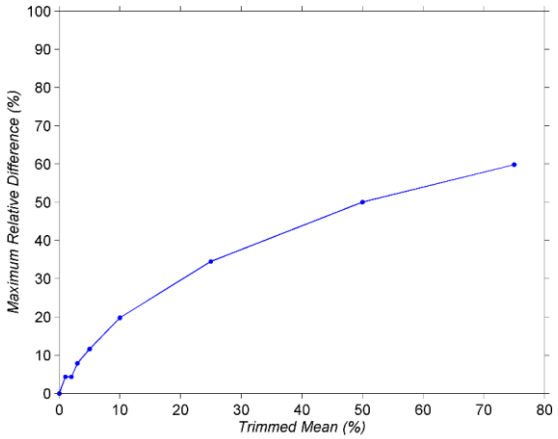

Figure S7: The maximum values of the relative difference between the 1D diffraction pattern calculated with standard radial integration and 1, 2, 3, 5, 10, 25, 50 and 75% alpha-trimmed mean filter for the CeO_2 is shown.

However, as it was shown also in Figure S4, even for the extreme case of the 75% trimmed mean filter, the changes are not radical. The maximum intensities of the diffraction peaks are maintained and the main changes (e.g. the 60% reported in Figure S5) are observed at the tails of the bragg peaks. This is clearly shown in Figure S8 where the 1D diffraction pattern calculated with standard radial integration and the one calculated with 75% trimmed mean filter are presented.

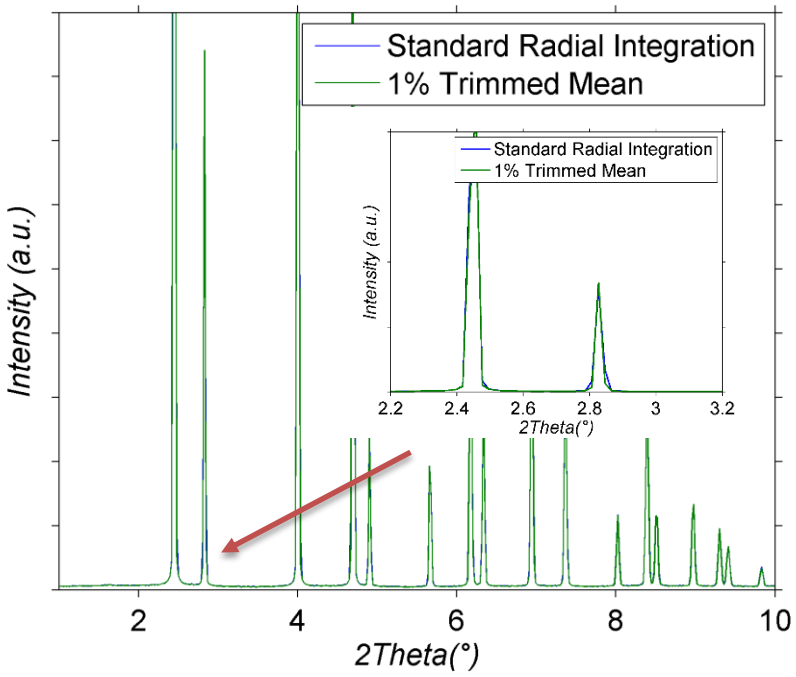

Figure S8: The 1D diffraction patterns for the CeO_2 standard is shown using standard radial integration and 75% alpha-trimmed mean filter. Also shown is a 2θ region of interest where the maximum relative difference between the two diffraction patterns is observed (i.e. scattering angle 2.85° 2θ.

MATLAB Scripts

The main MATLAB in-house function used in this work are provided below.

```
function [rBin,phiBin,ibad] = binning(image,cnt_x,cnt_y)
% The function binning is used to set up the polar transformation. It
% requires a raw 2D diffraction image and the coordinates of the beam
% centre after the calibration i.e. (cnt_x,cnt_y)

nox = size(image,1);
noy = size(image,2);
X = cumsum(ones(nox,noy),2) - cnt_x  ;
Y = cumsum(ones(nox,noy),1) - cnt_y  ;
rh = sqrt(X.^2 + Y.^2);
phi = atan2d(Y,X)+180;
phiBinSize = 1;
phiBin = round(phi/phiBinSize);
rBinSize = 1;
rBin = round(rh/rBinSize);
ibad = sub2ind(size(phiBin),cnt_y,cnt_x);


function T = image2pol(a,rBin,phiBin,ibad)
% The image2pol function transforms an image from Cartesian to polar
% coordinates

T = zeros(max(rBin(:)),max(phiBin(:)));
for i = 1: length(rBin(:))
    if i == ibad;
        continue
    end
    T(rBin(i),phiBin(i)+1) = a(i);
end


function M = masknan(a)
% The masknan function creates a binary mask (i.e. NaN and 1 values only)
% and requires as an input the original 2D diffraction image.

a(find(a == 0)) = 0.01;
a(find(a < 0)) = 0;
M = image2pol(a,rBin,phiBin,ibad);
M(find(M==0)) = NaN;
M(find(M>0)) = 1;


function F = tnans(T,z)
% The tnans function is used to transfer the NaN values in each row of a 2
% dimensional matrix (T) to the end of each row. The user can choose if the
% NaN values are converted to zeros at the end of the operation.
% T is the 2D matrix (e.g. grayscale image)
% if z == 1 then the nan values are going to be zeros

F = zeros(size(T));
for i = 1:size(T,1)
    F(i,1:length(find(~isnan(T(i,:))))) = T(i,~isnan(T(i,:)));
    if z == 1
        F(i,length(find(~isnan(T(i,:))))+1:end) = 0;
    else
        F(i,length(find(~isnan(T(i,:))))+1:end) = NaN;
    end
```

```
end

function MN = masknan2(M)
% The masknan2 function creates a binary mask (i.e. 0 and 1 values only)
% and requires as an input the binary mask created with the masknan
% function. This process is necessary if the user wants to apply the
% ordfilt2 function.
MN = tnans(M,0);

function A = filterextrv(B,sd)
% The filterextrv function applies the Standard Deviation based Trimmed
% Mean filter to the polar transformed images.
% B is the polar transformed image
% sd: how many times the standard deviation should be applied as the
% threshold value. The user should input 10 times the desired value.

for i = 1: size(B,1)
    B(i,find(abs(B(i,:)-nanmean(B(i,:)))>(sd/10)*nanstd(B(i,:)))) = nan;
end
A(1:size(B,1)) = nanmean(B,2);
```

Below is provided, for the interested reader, a simplified example of the MATLAB code used to apply different filter in the XRD-CT data presented in this work.

```
%% XRD-CT reconstruction example
% Parameters from the tomographic scan
nFastAxisSteps = 140;
slowAxisStart = -94.5;
slowAxisEnd = 94.5;
nSlowAxisSteps = 105;
angleStep = abs(slowAxisEnd-slowAxisStart)/nSlowAxisSteps;
theta = angleStep:angleStep:abs(slowAxisEnd-slowAxisStart);
theta = theta(find(theta<=180));

% Directory name
xrdct_datasets = {'test'};
% Domains for ordfilt2
fm1 = [1 1 1 1 1]; % Row Median filter
fm2 = ones(3,3); % Median filter
fm3 = [0 1 0 ; 1 1 1;0 1 0]; % Cross Median filter
sdtm = 10*[1,1.5,2,3]; % Standard Deviatioan based Trimmed Mean Filter

mm = 1;
XRDCT_filters(mm).name = char(xrdct_datasets(mm));
% specify the directory where the images have been stored
mypath = sprintf('/data/%s/',char(xrdct_datasets(mm)));
mystem = sprintf('%s_',char(xrdct_datasets(mm)));
% n is the number of images in the xrd-ct scan
n = 20000;

% Generate the names and allocate zeros
clear RAW
vnames = {'r','fm', 'f','atm'};
tth = size(T,1);
for ii = 1:length(sdtm)
    RAW.(sprintf('%s%s',char(vnames(4)),num2str(sdtm(ii)))) =
zeros(n,length(tth));
end
```

```
for ii = [0,1,2,3,5,10,25,50]
    RAW.(sprintf('%s%s',char(vnames(1)),num2str(ii))) =
zeros(n,length(tth));
    if (ii < 10)

RAW.(sprintf('%s%s%s1',char(vnames(1)),num2str(ii),char(vnames(2)))) =
zeros(n,length(tth));

RAW.(sprintf('%s%s%s2',char(vnames(1)),num2str(ii),char(vnames(2)))) =
zeros(n,length(tth));

RAW.(sprintf('%s%s%s3',char(vnames(1)),num2str(ii),char(vnames(2)))) =
zeros(n,length(tth));
    end
end

for jj = 0 : n
    % Directory for the raw images
    fn = sprintf('%s%s%.jpg',mypath,mystem,jj);
    % Read the raw images
    I = imread(fn);
    % Transform the raw I image from Cartesian to polar coordinates
    T = image2pol(I,rBin,phiBin).*masknan;
    % The different nth rank-order filters
    F1 = ordfilt2(tnans(T,1),3,fm1).*maskn;
    F2 = ordfilt2(tnans(T,1),5,fm2).*maskn;
    F3 = ordfilt2(tnans(T,1),3,fm3).*maskn;
    % The SDTM filter
    for ii = 1:1:length(sdtm)
        RAW.(sprintf('%s%s',char(vnames(4)),num2str(sdtm(ii))))(jj+1,:) =
filterextrv(T,length(tth),sdtm(ii));
    end
    % The alpha-trimmed mean filters
    for ii = [0,1,2,3,5,10,25,50]
        RAW.(sprintf('%s%s',char(vnames(1)),num2str(ii)))(jj+1,:) =
trimmean(T(1:length(tth),:),ii,2)';
        if (ii < 10)

RAW.(sprintf('%s%s%s1',char(vnames(1)),num2str(ii),char(vnames(2))))(jj+1,:
) = trimmean(F1(1:length(tth),:),ii,2)';

RAW.(sprintf('%s%s%s2',char(vnames(1)),num2str(ii),char(vnames(2))))(jj+1,:
) = trimmean(F2(1:length(tth),:),ii,2)';

RAW.(sprintf('%s%s%s3',char(vnames(1)),num2str(ii),char(vnames(2))))(jj+1,:
) = trimmean(F3(1:length(tth),:),ii,2)';
        end
    end

    fprintf('\r---> %.4d',jj);
end

% Store the projection data
for ii = 1:1:length(sdtm)

XRDCT_filters(mm).(sprintf('%s%s_projectiondata',char(vnames(4)),num2str(sd
tm(ii)))) =
reshape(RAW.(sprintf('%s%s',char(vnames(4)),num2str(sdtm(ii)))),nFastAxisSt
eps,nSlowAxisSteps+1,size(T,1));
end
for ii = [0,1,2,3,5,10,25,50]
```

```
XRDCT_filters(mm).(sprintf('%s%s_projectiondata',char(vnames(1)),num2str(ii
))) =
reshape(RAW.(sprintf('%s%s',char(vnames(1)),num2str(ii))),nFastAxisSteps,nS
lowAxisSteps+1, size(T,1));
    if (ii < 10)

XRDCT_filters(mm).(sprintf('%s%s%s1_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2)))) =
reshape(RAW.(sprintf('%s%s%s1',char(vnames(1)),num2str(ii),char(vnames(2)))
),nFastAxisSteps,nSlowAxisSteps+1, size(T,1));

XRDCT_filters(mm).(sprintf('%s%s%s2_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2)))) =
reshape(RAW.(sprintf('%s%s%s2',char(vnames(1)),num2str(ii),char(vnames(2)))
),nFastAxisSteps,nSlowAxisSteps+1, size(T,1));

XRDCT_filters(mm).(sprintf('%s%s%s3_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2)))) =
reshape(RAW.(sprintf('%s%s%s3',char(vnames(1)),num2str(ii),char(vnames(2)))
),nFastAxisSteps,nSlowAxisSteps+1, size(T,1));
    end
end

% Centre the sinograms
for ii = 1:1:length(sdtm)

XRDCT_filters(mm).(sprintf('%s%s_centredsino',char(vnames(4)),num2str(sdtm(
ii)))) =
XRDCT_filters(mm).(sprintf('%s%s_projectiondata',char(vnames(4)),num2str(sd
tm(ii))))(:,1:length(theta),:);
end
for ii = [0,1,2,3,5,10,25,50]

XRDCT_filters(mm).(sprintf('%s%s_centredsino',char(vnames(1)),num2str(ii)))
=
XRDCT_filters(mm).(sprintf('%s%s_projectiondata',char(vnames(1)),num2str(ii
)))(:,1:length(theta),:);
    if (ii < 10)

XRDCT_filters(mm).(sprintf('%s%s%s1_centredsino',char(vnames(1)),num2str(ii
),char(vnames(2)))) =
XRDCT_filters(mm).(sprintf('%s%s%s1_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2))))(:,1:length(theta),:);

XRDCT_filters(mm).(sprintf('%s%s%s2_centredsino',char(vnames(1)),num2str(ii
),char(vnames(2)))) =
XRDCT_filters(mm).(sprintf('%s%s%s2_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2))))(:,1:length(theta),:);

XRDCT_filters(mm).(sprintf('%s%s%s3_centredsino',char(vnames(1)),num2str(ii
),char(vnames(2)))) =
XRDCT_filters(mm).(sprintf('%s%s%s3_projectiondata',char(vnames(1)),num2str
(ii),char(vnames(2))))(:,1:length(theta),:);
    end
end

% Filtered back projection
ii = 0;
```

```
[finalSize,NAng,NCh] =
size(XRDCT_filters(mm).(sprintf('%s%s_centredsino',char(vnames(1)),num2str(
ii))));
clear s0
for kk = 1 : NCh
    for ii = 1:1:length(sdtm)

XRDCT_filters(mm).(sprintf('%s%s_fbp',char(vnames(4)),num2str(sdtm(ii))))(:
,:,kk) =
iradon(XRDCT_filters(mm).(sprintf('%s%s_centredsino',char(vnames(4)),num2st
r(sdtm(ii))))(:,:,kk),theta,'linear','Shepp-Logan',0.9,finalSize);
    end

    for ii = [0,1,2,3,5,10,25,50]

XRDCT_filters(mm).(sprintf('%s%s_fbp',char(vnames(1)),num2str(ii)))(:,:,kk)
=
iradon(XRDCT_filters(mm).(sprintf('%s%s_centredsino',char(vnames(1)),num2st
r(ii)))(:,:,kk),theta,'linear','Shepp-Logan',0.9,finalSize);
        if (ii < 10)

XRDCT_filters(mm).(sprintf('%s%s%s1_fbp',char(vnames(1)),num2str(ii),char(v
names(2))))(:,:,kk) =
iradon(XRDCT_filters(mm).(sprintf('%s%s%s1_centredsino',char(vnames(1)),num
2str(ii),char(vnames(2))))(:,:,kk),theta,'linear','Shepp-
Logan',0.9,finalSize);

XRDCT_filters(mm).(sprintf('%s%s%s2_fbp',char(vnames(1)),num2str(ii),char(v
names(2))))(:,:,kk) =
iradon(XRDCT_filters(mm).(sprintf('%s%s%s2_centredsino',char(vnames(1)),num
2str(ii),char(vnames(2))))(:,:,kk),theta,'linear','Shepp-
Logan',0.9,finalSize);

XRDCT_filters(mm).(sprintf('%s%s%s3_fbp',char(vnames(1)),num2str(ii),char(v
names(2))))(:,:,kk) =
iradon(XRDCT_filters(mm).(sprintf('%s%s%s3_centredsino',char(vnames(1)),num
2str(ii),char(vnames(2))))(:,:,kk),theta,'linear','Shepp-
Logan',0.9,finalSize);
        end
    end
end


save XRDCT_effectfilters.mat -v7.3 XRDCT_filters
```

References

Middelkoop, V., Chen, H., Michielsen, B., Jacobs, M., Syvertsen-Wiig, G., Mertens, M., Buekenhoudt, A. & Snijkers, F. (2014). *J. Membr. Sci.* **468**, 250-258.

Van Noyen, J., Middelkoop, V., Buysse, C., Kovalevsky, A., Snijkers, F., Buekenhoudt, A., Mullens, S., Luyten, J., Kretzschmar, J. & Lenaerts, S. (2012). *Catal. Today* **193**, 172-178.