Supplementary information:

Conformal Chemical Vapor Deposition of B_4C Thin Films onto Carbon Nanotubes

Arun Haridas Choolakkal[§], Ingemar Persson[§], Jarkko Etula[⊥], Emma Salmi[⊥], Taneli Juntunen[⊥], Per. O. Å Persson[§], Jens Birch[§] and Henrik Pedersen[§*]

[§]*Department of Physics, Chemistry and Biology, Linköping University, SE-581 83 Linköping, Sweden.*

[⊥]*Canatu, Tiilenlyöjänkuja 9A, FI-01720 Vantaa, Finland*

**E-mail: henrik.pedersen@liu.se*

MATLAB SCRIPTS

Pore distribution:

```
Img_A = imread(File_Name);
% Convert to grayscale if the image is in RGB format
Img_B = Img_A;
if ndims(Img_A) == 3
   Img_B = rgb2gray(Img_A);
end


% Apply contrast stretching
% Compute the minimum and maximum intensity values
min_intensity = double(min(Img_B(:)));
max_intensity = double(max(Img_B(:)));
% Stretch the contrast to the full 0-255 range
Img_B_stretched = uint8(255 * (double(Img_B) - min_intensity) / (max_intensity -
min_intensity));
% The Img_B_stretched image is created by linearly mapping the original intensity values to
the full 0-255 range
```

```
figure;
imhist(Img_B_stretched);
% Use islocalmax to find peaks
V = imhist(Img_B_stretched);
L = islocalmax(V);
% Set a threshold for peak prominence (adjust as needed)
threshold = 200;
% Find peaks above the threshold
[~, locs] = findpeaks(V, 'MinPeakProminence', threshold);
% Extract the elements corresponding to major peaks
major_peaks = V(locs);
major_peak_centers = linspace(0, 255, numel(V));
major_peak_centers = major_peak_centers(locs);

% Display the number of major peaks
num_major_peaks = numel(major_peaks);
fprintf('Number of peaks in the histogram: %d\n', num_major_peaks);

% Number of intensity levels in the image
prompt = 'Number of intensity levels in the image: ';
userInput = input(prompt);
Int_level = userInput; % Consider number of peaks in the histogram for Intensity level value

% Multilevel thresholding
level = multithresh(Img_B_stretched, Int_level);
B_Quant = imquantize(Img_B_stretched, level);
% Create depth map visualization
RGB1 = label2rgb(Img_B_stretched);
% Save depth map
```

```
imwrite(RGB1, [File_Name(1:end-4) '_Processed.png']);


% Binary segmentation
BS = zeros(size(B_Quant)); % Initialize an all-zero matrix BS with the same size as B_Quant
% Find the indices where B_Quant equals 1 (foreground)
[row_indices, col_indices] = find(B_Quant == 1);
% Set the corresponding pixels in BS to 1
BS(sub2ind(size(BS), row_indices, col_indices)) = 1;
% Inverse binarized
BS = 1 - BS;
BS = bwmorph(BS, 'majority', 1);


Conn = 8; % The  value Conn = 8 represents the connectivity parameter used in the watershed segmentation algorithm.
% Conn = 8 considers all eight neighboring pixels (including diagonals) around a central pixel.
% Compute the gradient magnitude of the binary image BS
gradmag = imgradient(BS); % gradient magnitude is used to identify potential object boundaries.
% Apply median filtering to the gradient magnitude
Img_B = medfilt2(gradmag, [3 3]); % For each output pixel, it computes the median value within the 3-by-3 neighborhood centered around that pixel.
% Perform watershed segmentation using connectivity Conn
Img_B = watershed(Img_B, Conn);


% Pore distribution
PD = zeros(size(BS)); % Initialize PD with the same size as BS
% Find indices where BS is 0 and Img_B is not 0
indices = (BS == 0) & (Img_B ~= 0);
% Set PD to 1 at the identified indices
PD(indices) = 1;
```

```
% Remove small connected components
PD = bwareaopen(PD, 9, Conn);
[PD_l, PD_n] = bwlabel(PD, Conn);
% Colorize pore space segmentation
RGB2 = label2rgb(PD_l, 'jet', 'white', 'shuffle');
% Save pore space segmentation
imwrite(RGB2, [File_Name(1:end-4) '_Pore Distribution.png']);

% Display original SEM image, processed, binarized, and pore distribution
figure;
imshow(Img_A); title('Original');
figure;
imshow(Img_B_stretched);title('Contrast stretched');
figure;
imshow(RGB1); title('Processed');
figure;
imshow(BS); title('Binarized');
figure;
imshow(RGB2); title('Pore distribution');
```

Porosity estimation:

```
% Step 1: Read the image
inputImage = imread('C:\Users\aruch90\Desktop\Img\1.png');

% Convert to grayscale if not already in grayscale
if size(inputImage, 3) > 1
    grayImage = rgb2gray(inputImage);
else
    grayImage = inputImage;
```

```
end

% Step 2: Apply contrast stretching
% Compute the minimum and maximum intensity values
min_intensity = double(min(grayImage(:)));
max_intensity = double(max(grayImage(:)));
% Stretch the contrast to the full 0-255 range
grayImage_stretched = uint8(255 * (double(grayImage) - min_intensity) / (max_intensity -
min_intensity));

% Step 3: Display the grayscale depth map with a colormap
colormap(jet(256)); % Choose a colormap (e.g., 'jet', 'parula', etc.)
imshow(grayImage_stretched, []);
% Optional: Adjust color limits to match the depth range
caxis([min(grayImage_stretched(:)), max(grayImage_stretched(:))]);

Inverted_grayImage_stretched = imcomplement(grayImage_stretched);

% Display the original and inverted images side by side
imshow(J);
title('Grayscale Depth Map');
xlabel('X-axis (pixels)');
ylabel('Y-axis (pixels)');
colorbar; % Add a color scale

% Save the grayscale depth map
imwrite(Inverted_grayImage_stretched,
'C:\Users\aruch90\Desktop\Img\grayscale_depth_map.png');
Porosity = ((mean(Inverted_grayImage_stretched(:)))/255);
fprintf('Porosity: %d\n', Porosity);
```