Electronic Supplementary Material (ESI) for Soft Matter.

Supporting Information

Rapid, Interface-Driven Domain Orientation in Bottlebrush Diblock Copolymer Films During Thermal Annealing

*Bijal B. Patel,[a] Dylan J. Walsh, [a] Kush Patel [a], Do Hoon Kim [a], Justin J. Kwok[b], Damien Guironnet [a], and Ying Diao *[a]*

[a] Department of Chemical and Biomolecular Engineering, University of Illinois at Urbana-Champaign, 600 South Mathews Avenue, Urbana, IL 61801, United States

[b]Department of Materials Science and Engineering, University of Illinois at Urbana−Champaign, 1304 W. Green Street, Urbana, Illinois 61801, United States

For correspondence: yingdiao@illinois.edu

Table of Contents

1. Materials and characterization methods

All reactions were performed in an argon-filled glovebox (O_2 < 2 ppm, H_2O < 0.5 ppm) at room temperature using oven-dried glassware. THF, toluene, and DCM was dried using a commercial solvent purification system. rac-Lactide {Aldrich}, sec-butyllithium solution {1.3 M in cyclohexane/hexane (92/8), ACROS Organics} and ethylene oxide solution (2.5-3.3 M in THF, Aldrich) was used as received. 1,8-diazabicyclo[5.4.0]undec-7-ene (DBU) {Aldrich} was distilled over CaH_2 and storage under argon at -20 °C. Styrene was pass through neutral alumina plug and stored under argon at -20 °C. [$(H_2IMes)(3\text{-Br-py})_2(Cl)_2Ru$=CHPh], G3 was synthesized according to literature.[1] exo-5-Norbornene-2-carboxylic acid and exo-5-Norbornene-2-methanol was synthesized according to literature.[2] NMR characterization of identical materials, but a different synthesis batch can be found in our prior publication.[3]

Conventional Size Exclusion Chromatography (SEC) was performed using a Tosoh Ecosec HLC-8320GPC at 40 ºC with THF (HPLC grade) as the eluent. This SEC is equipped with both a refractive index and UV detector (detector set to 266 nm). The SEC is fitted with a guard column (6.0 mm ID x 4.0 cm x 5 µm), and two analytical columns (7.8 mm ID x 30 cm x 5 µm; TSKgel GMH_{HR}-H). The reference flow rate is 0.5 mL min^{-1} while the analytical column is at 1.0 mL·min^{-1}. Polystyrene standards (16 points ranging from 200 Mw to 2.1 million Mw) were used as the general calibration. An additional calibration was created for specifically for linear polylactic acid and only used for linear polylactic acid (12 points ranging from 500 Mw to 10,000 Mw).

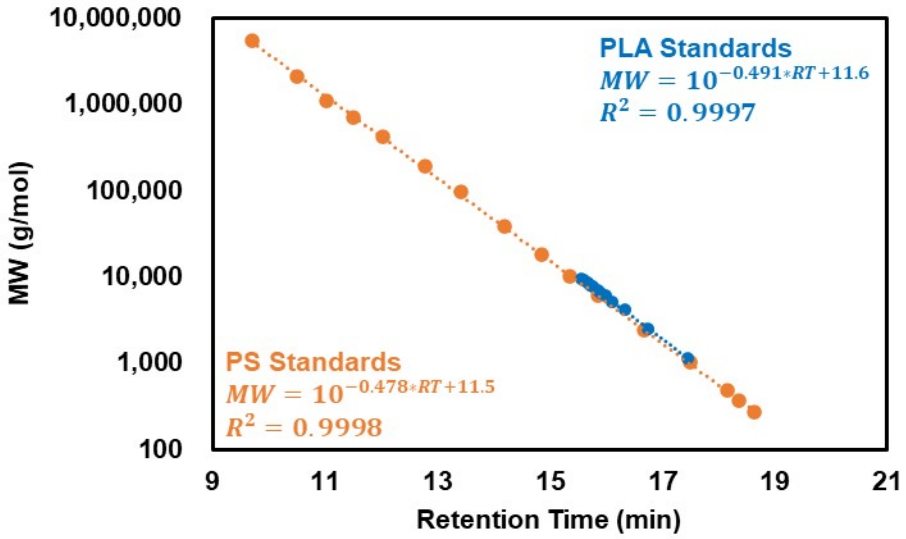

Figure S1. Conventional size exclusion chromatography calibration curves with polystyrene and polylactic acid standards.

Size Exclusion Chromatography (SEC) was performed using a Tosoh Ecosec HLC-8320GPC at 40 ºC with THF (HPLC grade) as the eluent. This SEC is equipped with a refractive index, UV detector, and LenS3 Multi-Angle Light Scattering Detector. The SEC is fitted with a guard

column (6.0 mm ID x 4.0 cm x 5 μm), and two analytical columns (7.8 mm ID x 30 cm x 13 μm; TSKgel Alpha-M). The reference flow rate is 0.3 mL min^{-1} while the analytical column is at 0.6 $mL \cdot min^{-1}$. The detectors were calibrated with a narrow polystyrene standard (Mw= 99,000 Da). Polymer solutions were prepared at a known concentration (ca. 1 mg/mL) and an injection volume of 20 μL was used. dn/dc values for the bottlebrush polymers were obtained for each injection by assuming 100% mass elution of the bottlebrush from the columns (the injection mass was adjusted according to the known wt% from conventional SEC).

Synthesis procedures

The synthesis methodology used in this manuscript to produce the diblock bottlebrush polymer has been reported in several prior publications with extensive characterization and detailed procedures.[3–5] We provide the key characterization here, and direct readers to prior publications for more details and characterization of the methodology.

Characterization data

Table S1. Characterization data for macromonomers.

Chemistry	$M_{n,GPC}$ (g/mol)[a]	M_w/M_n[a]
PS	4,500	1.03
PLA	4,200	1.06
[a]Calculated from conventional GPC calibration respect to PS standards or PLA standards.		

Table S2. Characterization data for PS-PLA diblock bottlebrush (PS: 4.5 kg/mol; PLA: 4.2 kg/mol, 50 wt% PS).

Targeted N_{bb}	M_n[a] (kg/mol)	M_w/M_n[a]	wt %[b]				Block length PS:PLA	$M_{w,LS}$[c] (kg/mol)
			Diblock BB	Block 1	PLA arm	PS arm		
400	602	1.04	93	6	>1	1	194:229	1800
[a]Calculated from conventional GPC calibration respect to PS standards [b]See SI section of reference[5] for calculation details. [c]Determined from triple detect GPC								

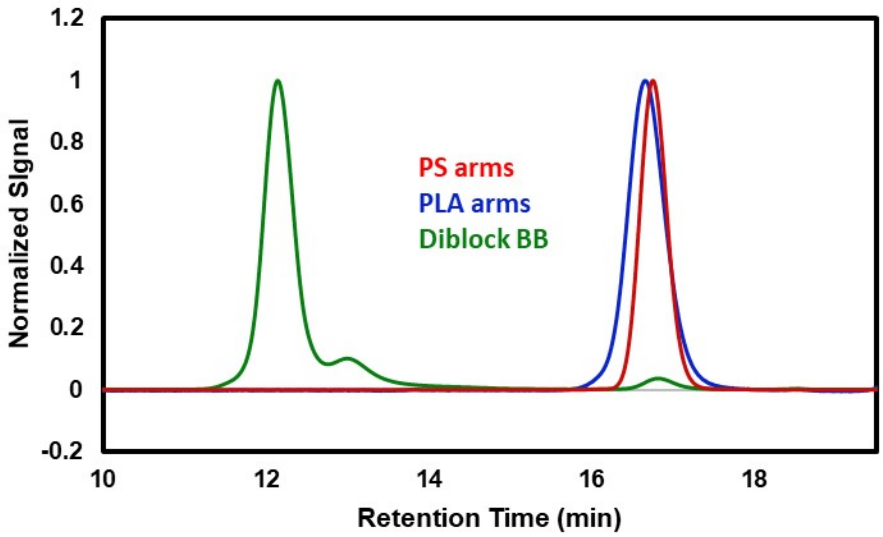

Figure S2. RI-GPC traces for the synthesis of PS-b-PLA diblock bottlebrushes. (The small second hump on the green trace around 13 min is PS-BB cause by catalyst death upon addition of PLA brushes. The small peak at 16.5 min corresponds solely to unfunctionalized PS arms.)

2. Thermal Properties of PS-b-PLA Bottlebrush Block Copolymers (DSC)

As-prepared PS-b-PLA bottlebrush block copolymers exhibited glass transition temperatures (2nd heat) at 95.93 ˚C (PS block) and 49.15 ˚C (PLA block) as shown below.

Measurement Parameters.

A 5.5 mg sample was measured in an Aluminum pan (DSC Consumables Tzero) and scanned on a TA instruments Discovery 2500 Differential Scanning Calorimeter (DSC) at the University of Illinois Materials Research Laboratory. A Heat-Cool-Heat cycle was performed by heating from ambient to 150˚C, cooling to 25˚C, and heating to 150˚C. Heating was performed at 10°C/min with 2-minute isotherms at endpoints.

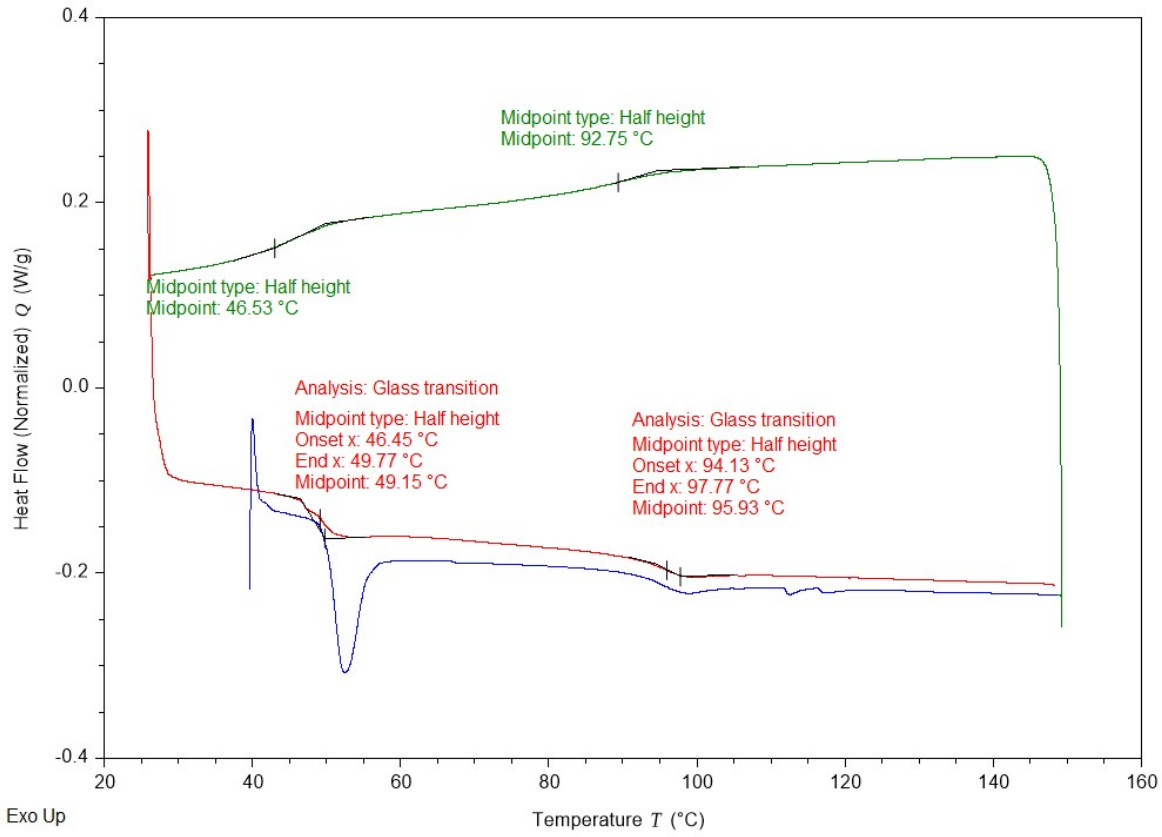

Figure S3. Heat-Cool-Heat data for the synthesized PS-b-PLA Bottlebrush Block copolymer.

3. Details of Thermal Annealing Process

A cartoon of the thermal annealing protocol is provided in **Figure S4** below.

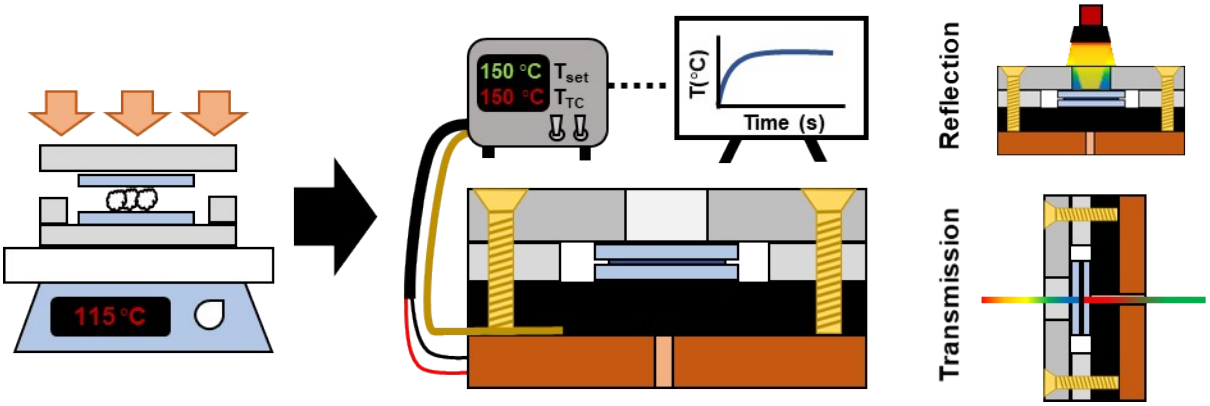

Figure S4. Cartoon of the thermal annealing scheme for two series of films. Films were prepared by first compressing purified BBCP powder between glass slides at 115 $^{\circ}C$ with film thickness defined by either external (metal) or internal (photoresist) spacers. During annealing, samples were installed in a custom heating/compression cell at room temperature and allowed to heat to 150˚C using a closed-loop temperature controller and resistive heater, with a thermocouple embedded below the sample as drawn. In situ monitoring of optical properties was performed in either reflection or transmission mode. Reflection properties were probed using a microscope camera in normal incidence configuration under a ring light. Transmittance was measured by mounting the heating cell within a UV-Vis spectrometer.

The annealing cell was fabricated by the University of Illinois School of Chemical Sciences Machine Shop staff. The cell was designed to accommodate substrates with dimensions of 1” x 1”. Prior to assembly, a T-type thermocouple was sealed into a machined groove in the bottom plate with thermal paste and covered with Kapton tape. A photo of the assembled cell is provided in **Figure S5**, with original mechanical drawings following.

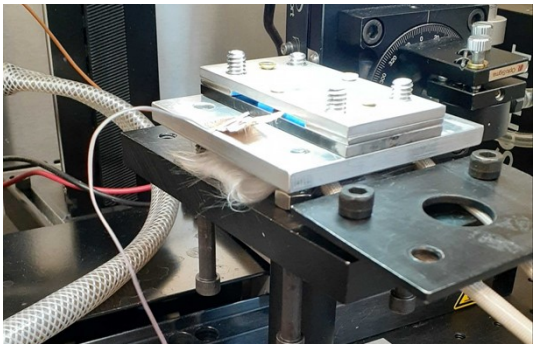

Figure S5. Photograph of the annealing cell in reflection mode.

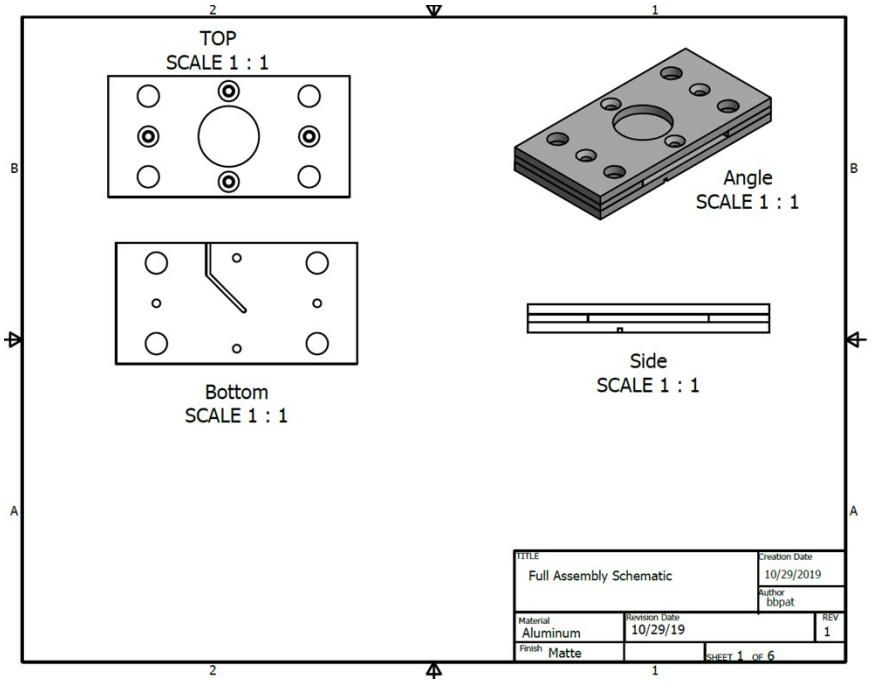

Figure S6. CAD drawing of the cell assembly comprising top and bottom plate and spacers.

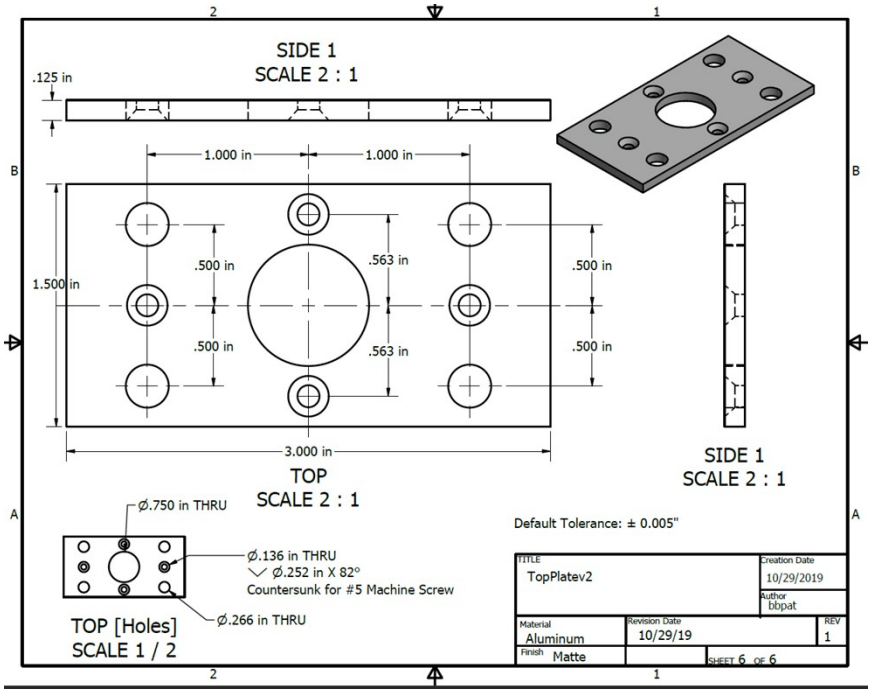

Figure S7. CAD drawing of the top plate of the annealing cell.

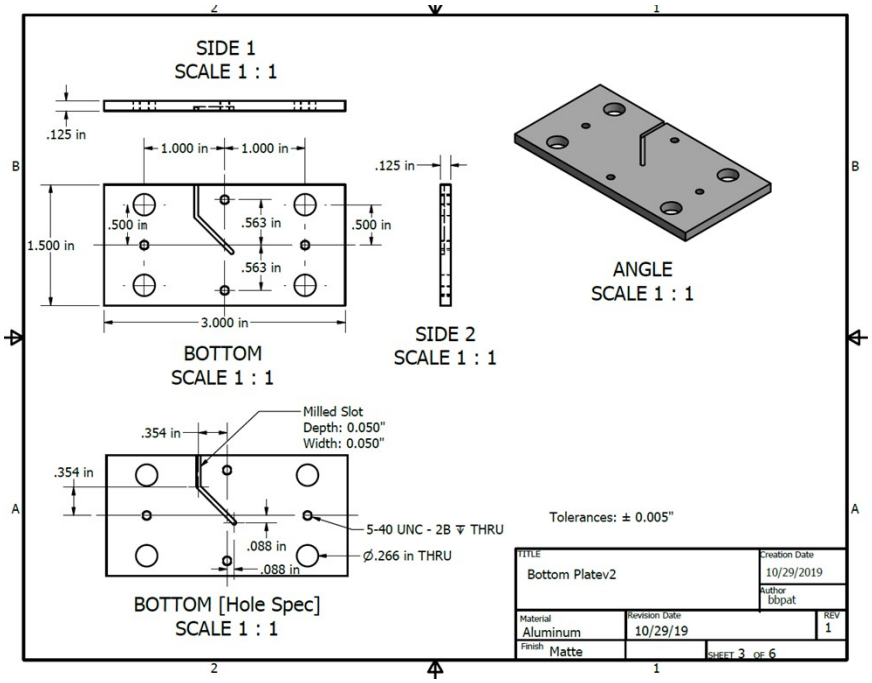

Figure S8. CAD drawing of the bottom plate of the annealing cell (reflection experiments).

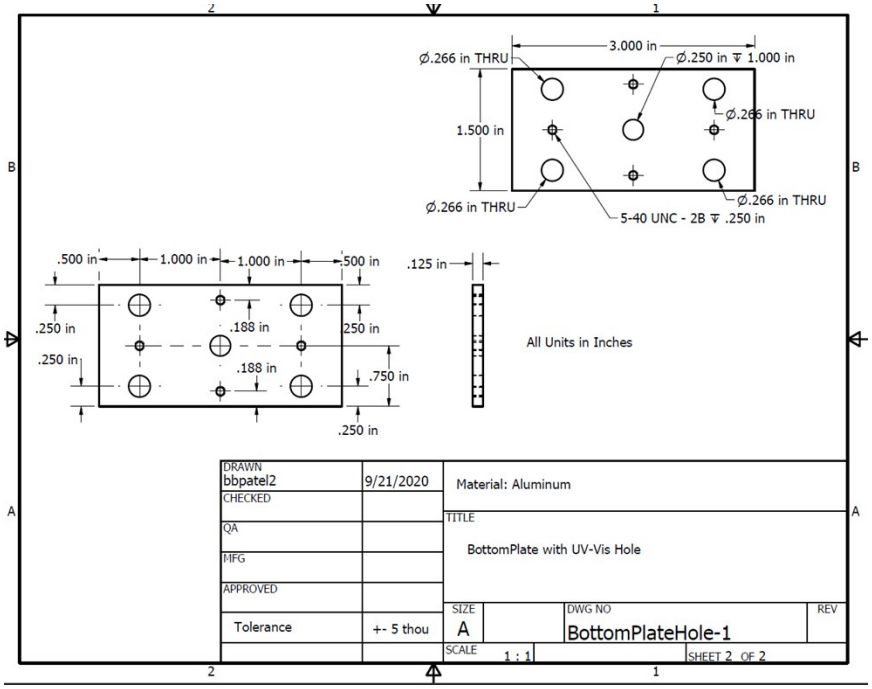

Figure S9. CAD drawing of the bottom plate of the annealing cell (UV-Vis transmission experiments).

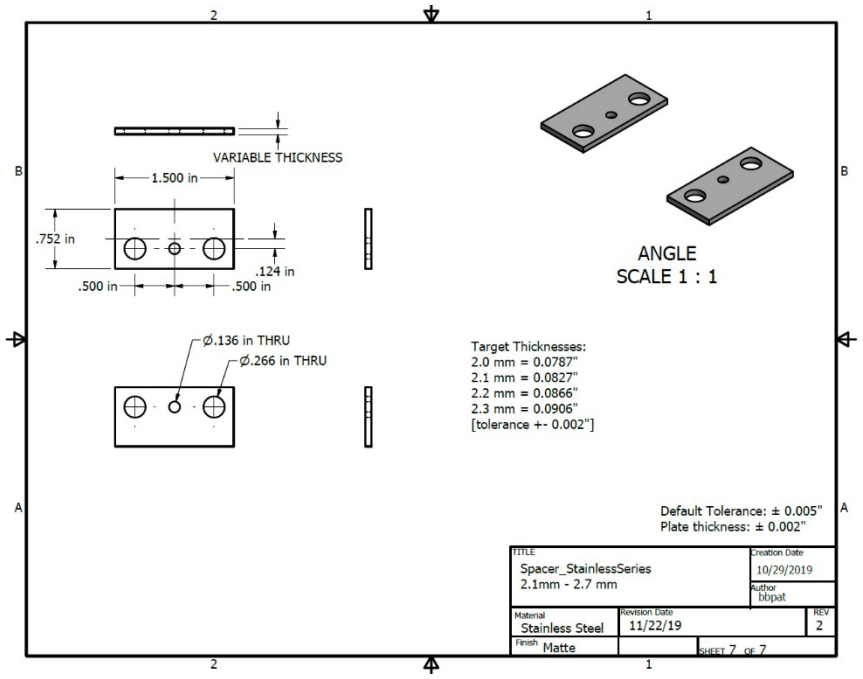

Figure S10. CAD drawing of the series of machined spacers. The 2.0 mm spacer was used for all samples less than 100 μm thick, instead relying on photoresist spacers patterned between the substrates.

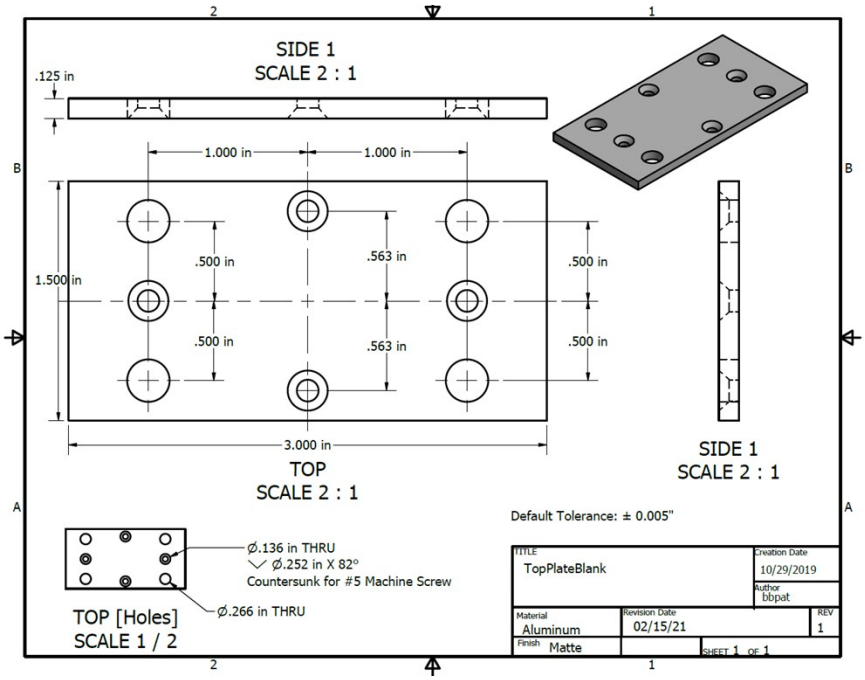

Figure S11. CAD drawing of the 'blank' top plate used for pre-pressing samples before *in situ* annealing.

4. Quantification of *in situ* Reflection Properties via Image Analysis

Matlab code provided below takes as input a folder full of images to be analyzed.

```
%% Software Information
% Name: BCPImageAnalyzer
% Description: Frame-by-Frame image RGB/HSV Image analysis
% Version: 1.2
% Release Date: 200206
% Version Summary: Added direct calculation of HSV Values from image
% Authors: Bijal Patel, Kush Patel, Justin Kwok

% License Terms

% Copyright (c) <2019> <Bijal Patel - University of Illinois>.
% All rights reserved.
%
% Developed by: <Diao Lab>
%               <University of Illinois>
%               <http://diao.scs.illinois.edu/Diao_Lab/Home.html>
%
% Permission is hereby granted, free of charge, to any person obtaining a
% copy of this software and associated documentation files (the
% "Software"), to deal with the Software without restriction, including
% without limitation the rights to use, copy, modify, merge, publish,
% distribute, sublicense, and/or sell copies of the Software, and to permit
% persons to whom the Software is furnished to do so, subject to the
% following conditions:
% * Redistributions of source code must retain the above copyright notice,
% this list of conditions and the following disclaimers.
% * Redistributions in binary form must reproduce the above copyright
% notice, this list of conditions and the following disclaimers in the
% documentation and/or other materials provided with the distribution.
% * Neither the names of <Diao Lab>, <University of Illinois>, nor the
% names of its contributors may be used to endorse or promote products
% derived from this Software without specific prior written permission.
%
% THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS
% OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
% MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN
% NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY
% CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT
```

```
% OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR
% THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

%% Step 0 - Initialize
    disp('Step 0: Initializing...')
    clear all
    close all

    %Get current date/time
    dateTime = datestr(datetime(now,'ConvertFrom','datenum'));
    dateTime = strrep(dateTime,":","_");
    dateTime = strrep(dateTime,"-","_");
    dateTime = strrep(dateTime," ","_");

    BrightnessThreshold = 1.6;

    % Select Folder Path
    % ENTER FOLDER PATH WITH ORIGINAL VIDEOS HERE
    filedir=strcat('[Folder Path Here]','\');
    % or default to userdir
    % filedir=strcat(userpath,'\');

    disp('Choose Folder for analysis...')
    [filedir]=uigetdir(strcat(filedir,'*.*'));
    disp('Folder Chosen!')
    filename = input("Enter Sample Name: ",'s');
    framerate = str2num(input("Enter Frames per second: ",'s'));

    %Make/choose specific folders for data/frame export
    expDir = strcat(filedir,'MATLABexport\',filename,"\");

    disp('Choose Folder with raw frames...')
    rawFramesDir = strcat(uigetdir(strcat(filedir,'*.*')),"\");

    threshFramesDir = strcat(expDir,'ThreshFrames\');
    calcDataDir = strcat(expDir,'calcData\',dateTime,'\');

    mkdir(expDir);
    mkdir(threshFramesDir);
    mkdir(calcDataDir);


    logParams = ["1. Export Directory" "2. RawFramesDirectory" "3. ThresholdFramesDirectory"
"4. CalculatedValuesDirectory"];
    logVals = [expDir rawFramesDir threshFramesDir calcDataDir];
```

```
disp('Step 0: Initializing Complete!')

%% Load Video to RAM
loadToRAM = input("Frame Export Required? (Y/N): ",'s');
if (strcmp(loadToRAM,'Y'))
    disp('Importing Video to RAM...')
    obj = VideoReader(strcat(filedir,filenameext));
    vid = read(obj);
    frames = obj.NumberOfFrames;
    framerate = obj.FrameRate;

    logParams = [logParams "Framerate"];
    logVals = [logVals framerate];
    disp('Import Video Complete!')
else
    disp('Skipping Loading Video to RAM')
    frames = size(dir(fullfile(rawFramesDir, '*.jpg')),1);
end


%% Convert Video file to frames and export
if(strcmp(loadToRAM,'Y'))
    disp('Export Raw Frames to File...')
    %Ask user if frames need to be output
    inp = input("     Output Frames? Y/N: ",'s');

    %if needs output, do so, else skip
    if (strcmp(inp,'Y'))
        for x = 1:1:frames
            imwrite(vid(:,:,:,x),strcat(rawFramesDir,filename,'_raw_',num2str(x),'.jpg'));
            fprintf('\tExporting frame: %d out of %d.\n',x,frames);
        end
        disp('Export Raw Frames Complete!')
        beep
    else
        disp('Skipping Frame Export')
    end
else
    disp('Skipping Frame Export')
end

%% Draw mask
disp('Masking')
inp = input("     Draw New Mask? Y/N: ",'s');

frames = length(dir(fullfile(rawFramesDir, '*.jpg')));
```

```
if(newAnalysis || strcmp(inp,'Y'))
    disp('Choose Frame to draw mask from.')
    %Choose frame to look at for drawing mask

    maskFrame = chooseFrame(1,frames,rawFramesDir,strcat(filename),-1);

        %Export Frame
        expTail = strcat(filename,'_maskFrame.jpg');
        expPath = strcat(calcDataDir,expTail);
        saveas(gcf,expPath);

    disp("     Mask Frame chosen! Now Draw Mask");

    %Size of frame image
    imsize=size(imread(strcat(rawFramesDir,filename,'_1','.jpg')));

    %Draw polygon mask
    %mx and my are the vertex positions of your mask
    %left click for points, right click to close loop
    %right click a vertex to create mask
    [mask, mx, my]=roipoly;

    %name image of frame to mask
    im =imread(strcat(rawFramesDir,filename,"_",num2str(maskFrame),'.jpg'));

    %Display mask overlayed onto image
    imshow(imoverlay(mat2gray(im),mask,[0.1 0.8 0.1]));

   %Save masked image
        %Export Frame
        expTail = strcat(filename,'_maskedFrame.jpg');
        expPath = strcat(calcDataDir,expTail);
        saveas(gcf,expPath);

    %Save mask itself
    %Save mask to file


        maskPath = strcat(expDir,'_',dateTime,'_mask.mat');
        save(strcat(calcDataDir,'mask_',dateTime,'.mat'),'mask');

    %uncomment for no mask
    %mask=logical(ones(imsize(1:2)));
else %Load Mask from File
    disp('Choose Mask');
```

```
        [maskfilenameext, maskfiledir] = uigetfile(strcat(filedir,'*.*'));
        maskPath = strcat(maskfiledir,maskfilenameext);
        load(maskPath);
        save(strcat(calcDataDir,'mask_',dateTime,'.mat'),'mask');
    end

    logParams = [logParams "MaskFrame"];
    logVals = [logVals maskFrame];

    disp('Get Mask Complete!')


%% Identify first frame to calculate
    disp('Choose first frame to calculate values for (NOT tZERO)')

    %Choose frame for t=0
   calcStartFrame = chooseFrame(maskFrame,frames,rawFramesDir,strcat(filename),mask);
        %Export start Frame
        expTail = strcat(filename,'_MaskedCalcStart.jpg');
        expPath = strcat(calcDataDir,expTail);
        saveas(gcf,expPath);

    disp('Frame for calculation start chosen!')

 %%Identify frameTzero time correction
    disp('Choose frame for time zero...')

    %Choose frame for t=0
    frameZero = chooseFrame(maskFrame,frames,rawFramesDir,strcat(filename),mask);
        %Export Masked Frame
        expTail = strcat(filename,'_MaskedTzero.jpg');
        expPath = strcat(calcDataDir,expTail);
        saveas(gcf,expPath);

    disp('tzero Frame Chosen!')

%% Evaluate mean RGB intensity in masked region
    fstart=calcStartFrame;%number of starting frame
    fspace=1;%number of frames to increment by

    numFramestoCalc = frames-fstart; %number of frames to calc
    %Preallocate arrays
    Int_RGB = zeros(1,numFramestoCalc);
    frameAvg_Gint = zeros (1,numFramestoCalc);
    frameAvg_Rint = zeros (1,numFramestoCalc);
    frameAvg_Bint = zeros (1,numFramestoCalc);
```

```
frameAvg_H = zeros (1,numFramestoCalc);
frameAvg_S = zeros (1,numFramestoCalc);
frameAvg_V = zeros (1,numFramestoCalc);

%Folder for thresholded images

for currentFrame=fstart:fspace:frames
   %Read in frame as RGB
   RGB_frame = imread(strcat(rawFramesDir,filename,'_',num2str(currentFrame),'.jpg'));
   %Convert to HSV
   HSV_frame = rgb2hsv(RGB_frame);

   %extract R, G, and B map values separately
   R=double(RGB_frame(:,:,1));
   G=double(RGB_frame(:,:,2));
   B=double(RGB_frame(:,:,3));

   %extract H, S, and V map values separately
   H=double(HSV_frame(:,:,1))*360;
   S=double(HSV_frame(:,:,2));
   V=double(HSV_frame(:,:,3));

   %NaN values outside of region of interest
   R(mask==0)=nan;
   G(mask==0)=nan;
   B(mask==0)=nan;

   %NaN values outside of region of interest
   H(mask==0)=nan;
   S(mask==0)=nan;
   V(mask==0)=nan;

   %'unwrap' H circle so red values dont appear on both top and bottom
   %of axis
   for hValIndex = 1:size(H)
      if (H(hValIndex)>320)
         H(hValIndex) = H(hValIndex) -360;
      end
   end

   %Get avg RGB values for this frame within mask
   frameAvg_Rint(currentFrame-fstart+1)= nanmean(nanmean(R));
   frameAvg_Gint(currentFrame-fstart+1)= nanmean(nanmean(G));
   frameAvg_Bint(currentFrame-fstart+1)= nanmean(nanmean(B));

   %Mean intensity calculated from R,G,B.
```

```
      Int_RGB(currentFrame-fstart+1)=mean([frameAvg_Rint(currentFrame-
fstart+1),frameAvg_Gint(currentFrame-fstart+1),frameAvg_Bint(currentFrame-fstart+1)]);

      %NAN overexposed pixels (white) from light source reflection
      %White pixels are those above threshold value
      [lengthX, lengthY, rgb] = size(RGB_frame);
      for column = 1:lengthY
        for row =1:lengthX
          if (sum([R(row,column) G(row,column) B(row,column)]) > BrightnessThreshold*
3*Int_RGB(currentFrame-fstart+1))
              R(row,column)= nan;
              G(row,column)= nan;
              B(row,column)= nan;
              H(row,column)= nan;
              S(row,column)=nan;
              V(row,column)=nan;
          end
        end
      end

      %Recalculate avg RGB values for this frame within mask
      frameAvg_Rint(currentFrame-fstart+1)= nanmean(nanmean(R));
      frameAvg_Gint(currentFrame-fstart+1)= nanmean(nanmean(G));
      frameAvg_Bint(currentFrame-fstart+1)= nanmean(nanmean(B));

      %Recalculate Mean intensity from R,G,B.
      Int_RGB(currentFrame-fstart+1)=mean([frameAvg_Rint(currentFrame-
fstart+1),frameAvg_Gint(currentFrame-fstart+1),frameAvg_Bint(currentFrame-fstart+1)]);

      %get avg_hsv for this frame within mask
      frameAvg_H(currentFrame-fstart+1) = nanmean(nanmean(H));
      frameAvg_S(currentFrame-fstart+1) = nanmean(nanmean(S));
      frameAvg_V(currentFrame-fstart+1) = nanmean(nanmean(V));

      %Export Thresholded Frame
      imwrite(uint8(cat(3,R,G,B)),fullfile(threshFramesDir,strcat(num2str(currentFrame),".jpg")));
      fprintf('\tOn frame: %d out of %d processed.\n',currentFrame-fstart+1,frames-fstart);
    end

    logParams = [logParams "Pixel Threshold"];
    logVals = [logVals BrightnessThreshold];

    disp('Step 6: Calc Mean RGB/HSV Complete!')
%% Step 7 - Plotting Data
    disp('Step 7: Plotting Results...')
```

```
%Time management
    %frame number array
    xFrame=fstart:fspace:frames;

    %time array
    %calc time with first frame at t = 0s
        xTime=(xFrame-frameZero)./framerate;%seconds
    xvar=xTime;

%Plotting mean RGB vs time
    subplot(2,3,1);
    plot(xvar,Int_RGB,'k')
    title('Mean RGB Intensity')
    xlabel('Time elapsed(s)')
    ylabel('Intensity (Arb)')
    set(gcf,'color','w');
    box on

%Plotting R, G, B, individually and mean vs frame
    subplot(2,3,4);
    hold on
    plot(xFrame,Int_RGB,'k','Linewidth',1.5)
    plot(xFrame,frameAvg_Rint,'r','Linewidth',1.5)
    plot(xFrame,frameAvg_Gint,'g','Linewidth',1.5)
    plot(xFrame,frameAvg_Bint,'b','Linewidth',1.5)

    %add thick vertical lines every 10/100/1000 frames
    if (frames > 1000)
        for i=0:100:frames
            xline(i,'k');
        end

    elseif (frames > 10000)
        for i=0:1000:frames
            xline(i,'k');
        end
    else
        for i=0:10:frames
            xline(i,'k');
        end
    end

    title('RGB Separated Intensity Values')
    xlabel('Frame')
    ylabel('Intensity (Arb)')
    set(gcf,'color','w');
```

```
    hold off
    box on

%Plotting R, G, B, individually and mean vs time
    subplot(2,3,[2 3]);
    hold on
    plot(xvar,Int_RGB,'k','Linewidth',1.5)
    plot(xvar,frameAvg_Rint,'r','Linewidth',1.5)
    plot(xvar,frameAvg_Gint,'g','Linewidth',1.5)
    plot(xvar,frameAvg_Bint,'b','Linewidth',1.5)
    title('RGB Separated Intensity Values')
    xlabel('Time elapsed (s)')
    ylabel('Intensity (Arb)')
    set(gcf,'color','w');
    box on
    hold off
    set(gcf, 'Position',  [100, 100, 1000, 500])

    %Plotting H and V vs time
    subplot(2,3,[5 6]);
    hold on
    yyaxis left
    ylabel('Hue')
    plot(xvar,frameAvg_H,'k','Linewidth',1.5)
    yyaxis right
    plot(xvar,frameAvg_S,'b','Linewidth',1.5)
    plot(xvar,frameAvg_V,'r','Linewidth',1.5)
    title('Hue and Value over time')
    xlabel('Time elapsed (s)')
    ylabel('Value')
    set(gcf,'color','w');
    box on
    hold off
    set(gcf, 'Position',  [100, 100, 1000, 500])
    legend("Hue","Saturation","Value");

    %Save figure
    saveas(gcf,strcat(calcDataDir,'Plot.png'));

%Concatenate results into matrix [frame xtime(s) Mean1(Arb) Mean2(Arb) Rint(Arb)
Gint(Arb) Bint(Arb)]
m = [transpose(xFrame) transpose(xTime) transpose(Int_RGB) ...
    transpose(frameAvg_Rint) transpose(frameAvg_Gint) ...
    transpose(frameAvg_Bint) transpose(frameAvg_H) transpose(frameAvg_S) ...
    transpose(frameAvg_V)];
```

```
    %Export Text
        expTail = strcat(filename,'_Data.txt');
        expPath = strcat(calcDataDir,expTail);

        %Write data file headers
        fid = fopen(expPath, 'w');
        fprintf(fid, "Frame,TimeElapsed(s),Mean Intensity,Red Intensity,Green Intensity,Blue
Intensity, Hue (degrees), Saturation (0-1), Value (0-1)\n");
        fclose(fid);

        %Write data
        dlmwrite(expPath,m,'-append','delimiter',',');

       disp('Step 7 Complete!')


%% Step 8:  Write Log File
   disp('Step 8 Writing Log File...!')

    %Write Log File

    if(writeLog(expDir,logParams, logVals))
        disp('Log File Written');
    else
        disp('Log Failed');
    end

    %% END


disp('Sequence Complete!')


%% Local functions
%Writes parameters to log file
function logged = writeLog(expPath,params, vals)
    %Get current date/time
    dateTime = datestr(datetime(now,'ConvertFrom','datenum'));
    dateTime = strrep(dateTime,":","_");
    dateTime = strrep(dateTime,"-","_");
    dateTime = strrep(dateTime," ","_");

    logName = strcat(expPath,'\AnalysisLog_',dateTime,'_.txt');
    logFileID = fopen(logName,'w');
    fprintf(logFileID, 'Log Entry - Generated \n%s\n',dateTime);
    outputs = [params;vals];
```

```
    fprintf(logFileID, '%s\n',outputs);
    fclose(logFileID);
    logged = 1;
end

%Loads parameters from log file
function data = readLog(logFilePath)
    logFid   = fopen(logFilePath);
    txt = textscan(logFid,'%s','delimiter',',\n');
    fclose(logFid);
    txt = txt{1}; %Pull first element of cell aray
    txt = string(txt); %convert to string array
    %Convert to n x 2 array of param-val pairs
    data = [txt(1:2:end) txt(2:2:end)];
end


%Allows user to choose a frame
function pickedFrame = chooseFrame(frame, maxFrames, framesDir, filename,mask)
    currentFrame = frame;
    pickedFrame = 0;
    done = 0;

    while (done ~= 1)
        %display masked, if mask given

        im =imread(strcat(framesDir,filename,'_',num2str(currentFrame),'.jpg'));

        if (mask~=-1)
            imshow(imoverlay(mat2gray(im),mask,[0.0 0.5 0.0]));
        else
            imshow(im);
        end

        cmd = input('     q for done, asd+fgh for -(10/5/1)+(1/5/10), frames: ','s');
        switch cmd
            case 'q'
                done = 1;
                pickedFrame = currentFrame;
            case 'd'
                if (currentFrame > 1)
                    currentFrame = currentFrame-1;
                end
            case 's'
                if (currentFrame > 11)
                    currentFrame = currentFrame-10;
```

```
            end
        case 'a'
            if (currentFrame > 101)
                currentFrame = currentFrame-100;
            end
        case 'f'
            if (currentFrame < maxFrames)
                currentFrame = currentFrame+1;
            end
        case 'g'
            if (currentFrame < maxFrames-10)
                currentFrame = currentFrame+10;
            end
        case 'h'
            if (currentFrame < maxFrames-100)
                currentFrame = currentFrame+100;
            end
        otherwise
            disp('Unrecognized input\n');
    end

  end
end
```

5. Image Processing and Orientation Mapping of Scanning Electron Micrographs

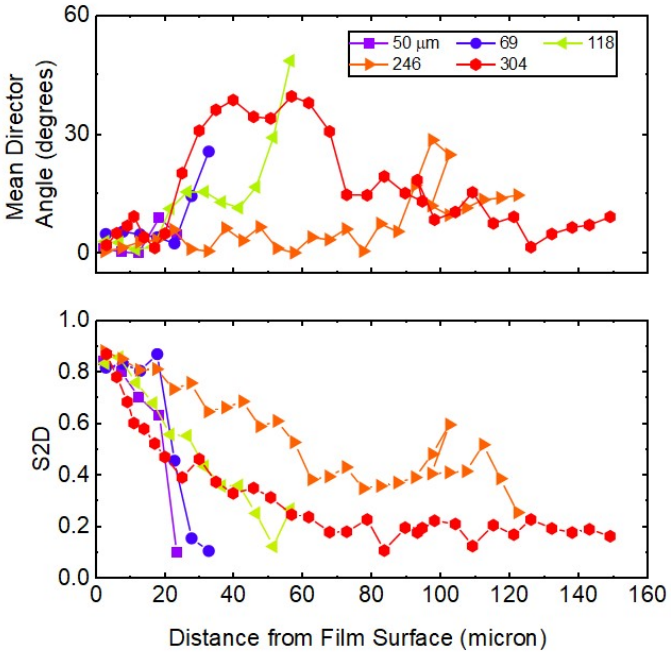

Figure S12. Results of orientation analysis for films characterized by cross-sectional scanning electron microscopy.

Image analysis for cross-sectional scanning electron micrographs was performed in two steps. First, images were pre-processed in FIJI(ImageJ2)[6] to obtain binarized black and white domains. Then, images were processed using GTFiber2.0 by Persson et al.[7] to obtain orientation statistics and colorized fiber maps.

Step 1: Image Preprocessing

In brief, image pre-processing consisted of 4 steps:

1. **Enhancing local contrast** to eliminate edge brightening effects from topographical variation under the electron beam.
2. **Applying a band pass filter** to enhance features with periodicity on the order of the domain size
3. **Binarization to black and white**
4. **Removal of off-film regions** for the top and bottom image of the series (if applicable)

For one series of particularly high-resolution images, it was found that small surface features (likely from Au/Pd deposition) were frequently causing misidentification of fibers. For this set, an optional gaussian blur step was added to the macro. **Figure S12** depicts the intermediate steps of this process.

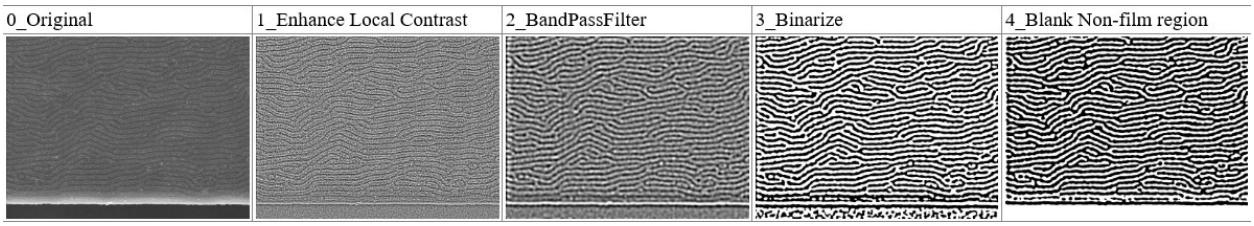

Figure S13. Snapshots of SEM pre-processing steps.

The following imagej2 macro code was written to streamline batch processing of images:

```
// Get Directory of files
// Batch Processing part modified from https://imagej.nih.gov/ij/macros/BatchProcessFolders.txt
sourceDir = getDirectory("Choose a Directory ");
setBatchMode(true);
count = 0;
countFiles(sourceDir);
n = 0;
processFiles(sourceDir);
print(count+" files processed");

function countFiles(sourcedir)
{
    list = getFileList(sourceDir);
    for (i=0; i<list.length; i++)
    {
        if (endsWith(list[i], "/"))
            print("subdirectory skipped");
            //countFiles(""+sourceDir+list[i]);
        else
            count++;
    }
```

```
}

function processFiles(sourceDir)
{
    list = getFileList(sourceDir);
    for (i=0; i<list.length; i++)
    {
        if (endsWith(list[i], "/"))
            print("subdirectory skipped");
            //countFiles(""+sourceDir+list[i]);
        else
        {
            showProgress(n++, count);
            path = sourceDir+list[i];
            processFile(path);
        }
    }
}

function processFile(path)
{
  if (endsWith(path, ".tif"))
  {
        open(path);
        //Get root of filename
        sourceName = File.nameWithoutExtension;
        print("Processing file:" + sourceName + "in folder:" + sourceDir);
```

```
print("Creating Output Folders");
//Create output folders
ClaheDir = sourceDir+"1_ContrastEnhanced"+File.separator;
File.makeDirectory(ClaheDir);
print(ClaheDir);
FFTBandPassDir = sourceDir+"2_BandPassFiltered"+File.separator;
File.makeDirectory(FFTBandPassDir);
BinarizedDir = sourceDir+"3_Binarized"+File.separator;
File.makeDirectory(BinarizedDir);
InvertedDir = sourceDir+"4_Inverted"+File.separator;
File.makeDirectory(InvertedDir);


print("Begin full processing macro");
print("Step 1: Enhancing Local Contrast");
//if data is too high resolution, smaller blocksize accentuates noise: go between 20-50
//run("Enhance Local Contrast (CLAHE)", "blocksize=50 histogram=256 maximum=1000
mask=*None*");
run("Enhance Local Contrast (CLAHE)", "blocksize=20 histogram=256 maximum=1000
mask=*None*");


//Optional Blur for very noisy data
//run("Gaussian Blur...", "sigma=8");


saveAs("jpeg", ClaheDir+sourceName+"_CLAHE");
print("Saved jpeg to " + ClaheDir+sourceName+"_CLAHE");


print("Step 2: Running BandPass Filter");
run("Bandpass Filter...", "filter_large=30 filter_small=15 suppress=Vertical tolerance=5
autoscale saturate");
```

```
    saveAs("jpeg", FFTBandPassDir+sourceName+"_FFTBandPass");
    print("\tSaved jpeg to " + FFTBandPassDir+sourceName+"_FFTBandPass");

    print("Step 3: Running Binarization");
    setOption("BlackBackground", false);

    //if default threshold is 'good enough'
    run("Convert to Mask");

    //if manual threshold needed, use following line
    //setThreshold(0, 112);
    //run("Convert to Mask");

    //remove speckles if needed
    //run("Remove Outliers...", "radius=10 threshold=50 which=Dark");
    //run("Remove Outliers...", "radius=10 threshold=50 which=Bright");

    saveAs("jpeg", BinarizedDir+sourceName+"_FFTBandPass");
    print("\tSaved jpeg to " + BinarizedDir+sourceName+"_FFTBandPass");

    print("Step 4: Inverting B/W");
    setOption("BlackBackground", false);
    run("Invert");
    saveAs("tif", InvertedDir+sourceName+"_FIJIProcessed");
    print("\tSaved tif to " + InvertedDir+sourceName+"_FIJIProcessed");
    print("End full processing macro");
    close();
}}
```

6. UV-Vis-Nir Diffuse+Specular Reflection Spectra for All Samples

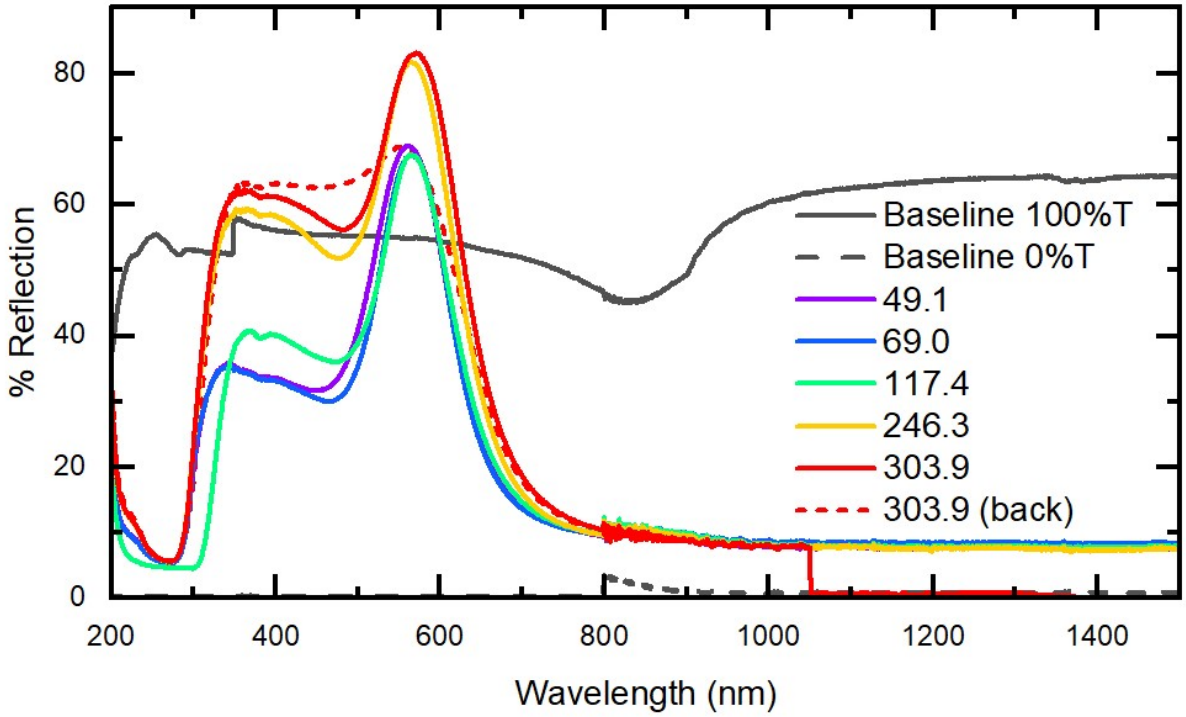

Figure S14. Raw Specular+Diffuse reflection spectra take after samples were annealed for 10 minutes under the microscope camera. Legend contains film thickness in microns.

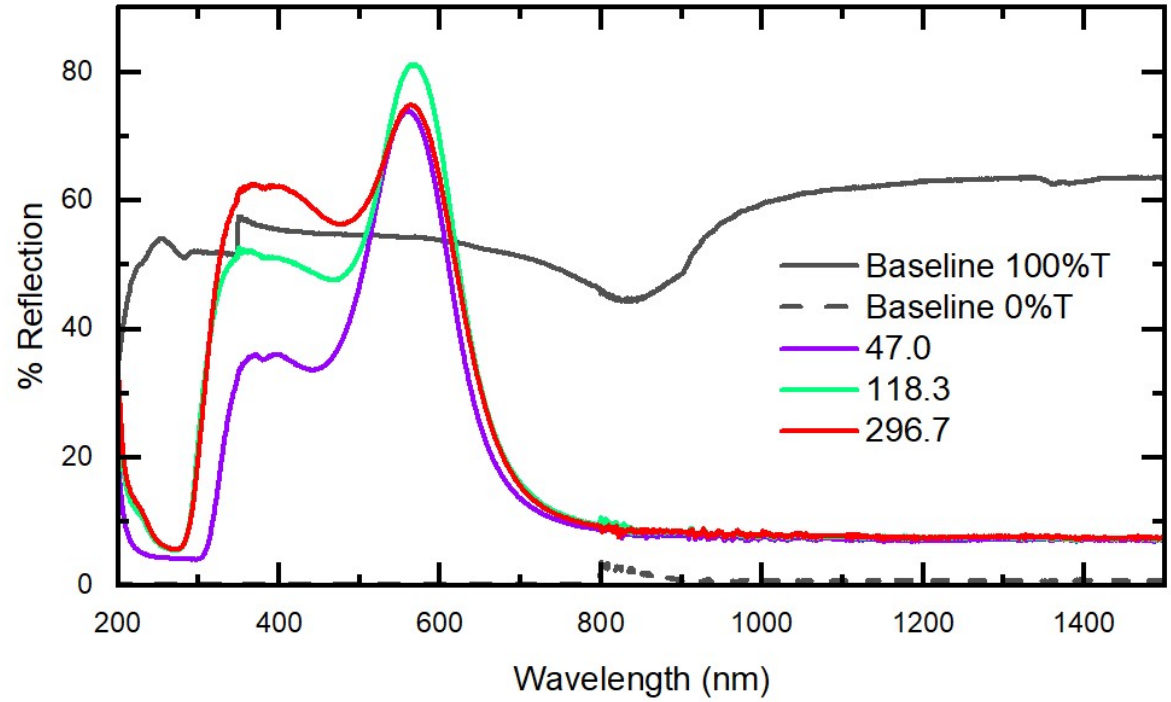

Figure S15. Raw Specular+Diffuse reflection spectra take after samples were annealed for 10 minutes within the UV-Vis transmission instrument. Legend contains film thickness in microns.

7. Volume Fraction Estimation

The volume fraction of PS was estimated by substituting the following data:

Table S3. Synthetic and chemical properties of polymacromonomers

Species	M_n(g/mol)	DP	$\rho\left(\frac{g}{cm^3}\right)$	Density Citation
PS-NB	4500	200	1.04	[8]
PLA-NB	4200	200	1.25	[9]

into the following equation, as per Dalsin et al[10]

$$\Phi_{PS} = \frac{(M_{n,PS-NB} * DP_{PS-NB}/\rho_{PS-NB})}{(M_{n,PS-NB} * DP_{PS-NB}/\rho_{PS-NB})) + (M_{n,PLA-NB} * DP_{PLA-NB}/\rho_{PLA-NB}))} = 0.56$$

Here, we approximate the density of the polymacromonomer to be the same as reported value for the bulk amorphous arm species.

8. SEM cross-sectional images of an initial 'pre-pressed' BBCP film

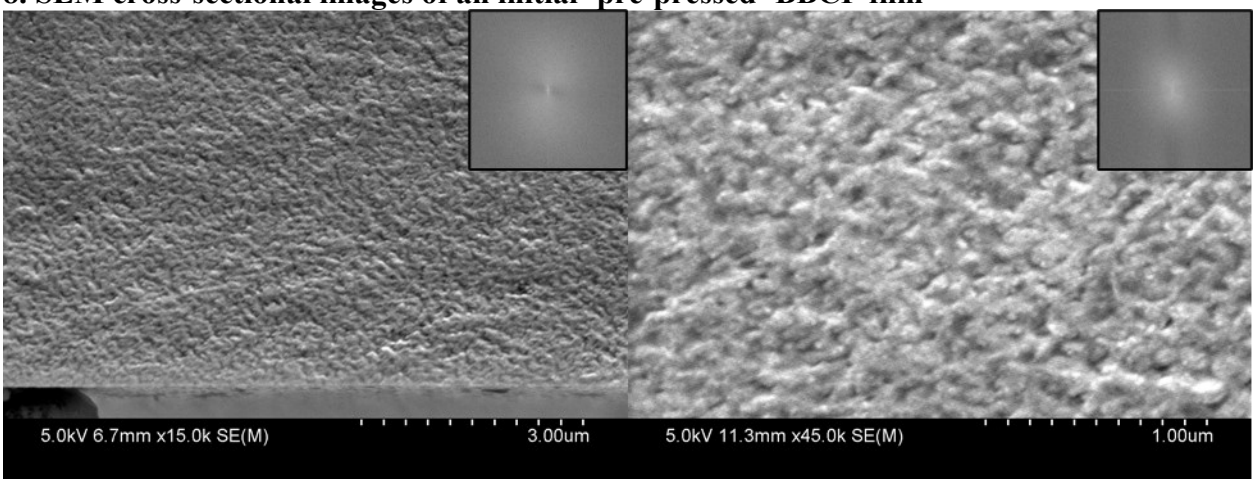

Figure S16. Cross-sectional SEM images near the edge of the pre-pressed BBCP film. Insets contain power spectrum images calculated using ImageJ's FFT plugin.

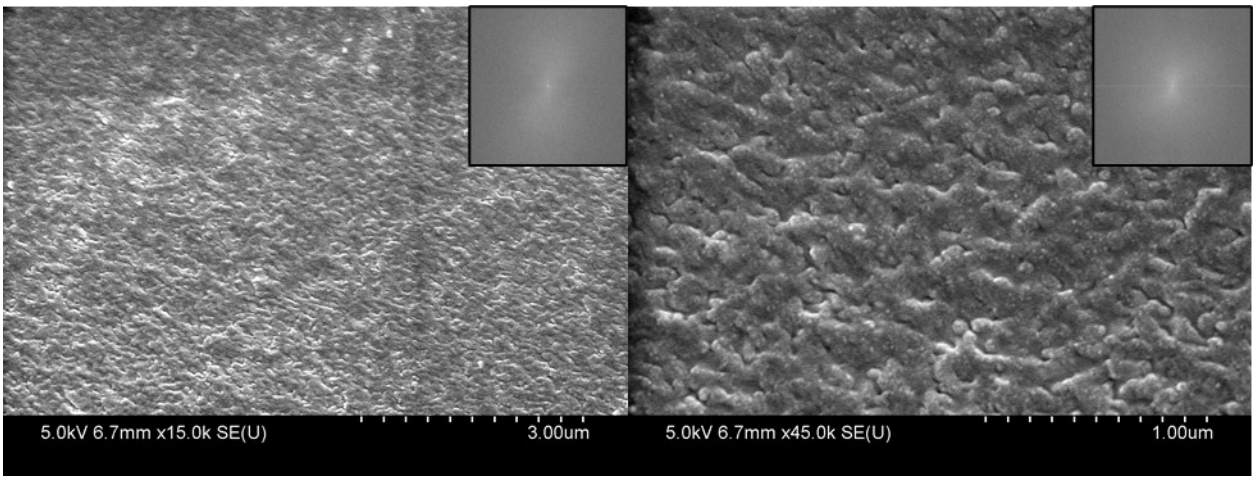

Figure S17. Cross-sectional SEM images near the center of the pre-pressed BBCP film. Insets contain power spectrum images calculated using ImageJ's FFT plugin.

For comparison, **Figure S18** contains cross-sectional SEM images taken from the edge and center of a well-ordered film after annealing, along with their power spectrum images.

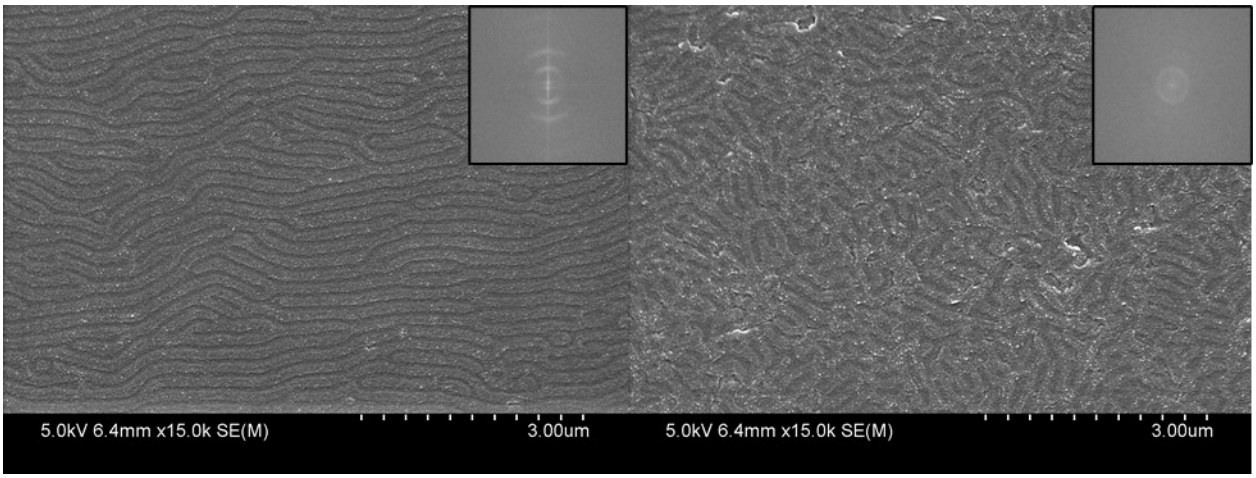

Figure S18. Cross-sectional SEM images taken at the center and edge of an annealed film (69.0 μm thick film). Insets contain power spectrum images calculated using ImageJ's FFT plugin.

9. Supplemental References

1 J. A. Love, J. P. Morgan, T. M. Trnka and R. H. Grubbs, *Angewandte Chemie International Edition*, 2002, **41**, 4035–4037.

2 S. C. Radzinski, J. C. Foster, R. C. Chapleski, D. Troya and J. B. Matson, *Journal of the American Chemical Society*, 2016, **138**, 6998–7004.

3 M. A. Wade, D. Walsh, J. C.-W. Lee, E. Kelley, K. Weigandt, D. Guironnet and S. A. Rogers, *Soft Matter*, 2020, **16**, 4919–4931.

4 B. B. Patel, D. J. Walsh, D. H. Kim, J. Kwok, B. Lee, D. Guironnet and Y. Diao, *Science Advances*, 2020, **6**, eaaz7202.

5 D. J. Walsh, M. A. Wade, S. A. Rogers and D. Guironnet, *Macromolecules*, 2020, **53**, 8610–8620.

6 C. T. Rueden, J. Schindelin, M. C. Hiner, B. E. DeZonia, A. E. Walter, E. T. Arena and K. W. Eliceiri, *BMC Bioinformatics*, 2017, **18**, 529.

7 N. E. Persson, M. A. McBride, M. A. Grover and E. Reichmanis, *Chem. Mater.*, 2017, **29**, 3–14.

8 J. E. Mark, Ed., *Physical properties of polymers handbook*, Springer, New York, 2nd ed., 2006.

9 D. Garlotta, *Journal of Polymers and the Environment*, 2001, **9**, 63–84.

10 S. J. Dalsin, T. G. Rions-Maehren, M. D. Beam, F. S. Bates, M. A. Hillmyer and M. W. Matsen, *ACS Nano*, 2015, **9**, 12233–12245.