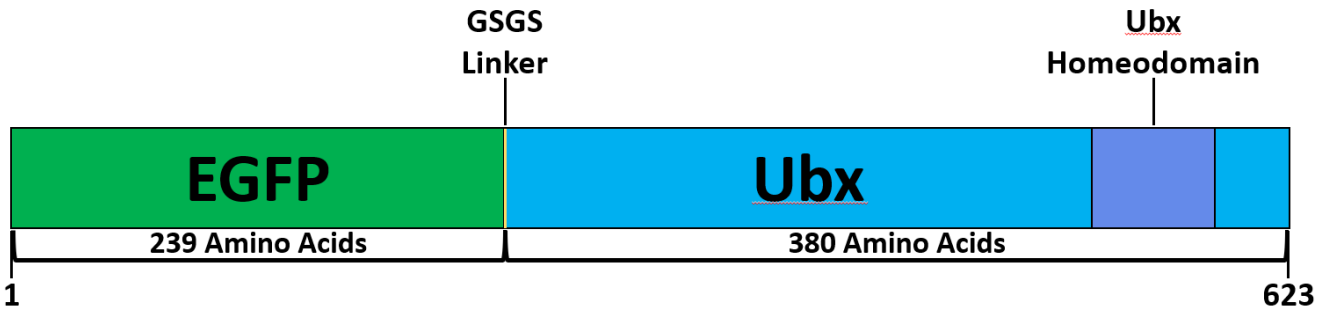
Chemical and temporal manipulation of early steps in protein assembly tune the structure and intermolecular interactions of protein-based materials

Supplemental Materials



Supplemental Figure 1. Diagram of the EGFP-Ubx fusion sequence. The Ubx1a isoform was used to generate the fusion protein. The positively charged Ubx homeodomain is part of the Ubx amino acid sequence. A four amino acid linker (Gly-Ser-Gly-Ser) was included between the two protein sequences.

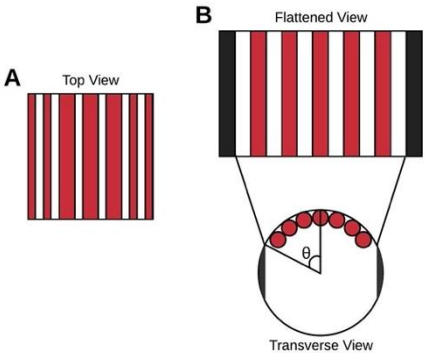
SEM Linearity analysis

To assess the axial alignment and uniformity of the fibril components of the UBX fibers, SEM images of fibers were loaded into MATLAB (2018a) and analysed using the code (see below). The orientation of the fiber in the image frame was user defined and aligned to the y-axis. Dased on the maximum value of the user defined background, a threshold was set and a binary mask of the fiber created. Small areas identified by threshold detection were considered noise and filtered out. Any intensities outside of the mask area on the original image are set to zero to remove the background noise from the analysis. The fiber was then put through an image flattening algorithm to separate and differentiate the fibrils. The

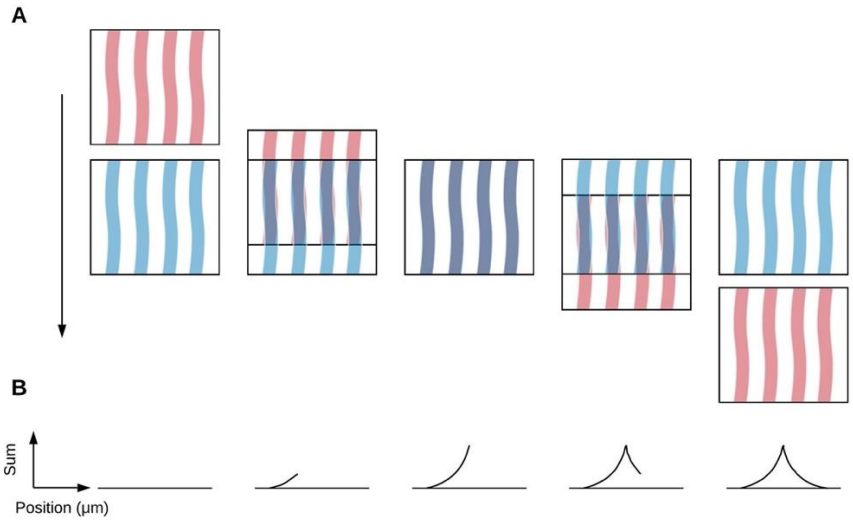
shape of the UBX fiber was assumed to be cylindrical and, thus, the algorithm considered the visible part of the fiber to be a semicircle with the diameter equal to the width of the fiber. The center point of the fiber was defined as the midpoint between the two widest points on the fiber. From that centre point, pixel intensity values are expanded horizontally as if the cylinder was being unrolled into a 2D representation of its surface area (SEM LA Figure 1).

SEM LA Figure 1: Fiber flattening diagram

The view from the SEM (A) can be flattened by assuming a cylindrical shape and calculating horizontal arc lengths along the length of the fiber. The arc length is converted into pixels and rounded to the nearest whole value. When the radius of the fiber is much larger than the radius of the fibril, we can cut off the portion of the image that extends beyond a 60° arc in each direction of the origin (**Error! Reference source not found.**B). At this point, a fibril covers more than half of the following fibril and they become indistinguishable.



A vertical autocorrelation was then employed to assess the long range alignment of the fibrils along the fiber. The flattened image was binarized to differentiate between fibril peaks and the troughs between fibrils. Using the binary image and a copy of the binary image, the autocorrelation vertically translated one image over the other (SEM LA Figure 2A). At each vertical translation, the overlap was plotted against the position of the translating image (SEM LA Figure 2B). The use of binary images enhanced the sensitivity of the autocorrelation because only when fibrils overlap will there be any value added to the sum. The full width at half of the maximum height (FWHM) of the graph was used to quantize the alignment. If the fibrils were well aligned, they would stay on top of themselves for a large portion of the autocorrelation, creating a wide FWHM. If the fibrils were randomly distributed, the autocorrelation would not start to overlap until the images were directly on top of one another, creating a narrow FWHM.



SEM LA Figure 2: Autocorrelation Technique
Diagram of how the vertical autocorrelation is translated (A) along with the graph of the computed sum values (B).

Matlab Main Code

```
clear all; close all;
%% load image
img = imread('file location and name');

umpxl = 20/1000; % um/pxl

%% crop image
figure;
imshow(img);
h_rect = imrect();
% Rectangle position is given as [xmin, ymin, width, height]
pos_rect = h_rect.getPosition();
% Round off so the coordinates can be used as indices
pos_rect = round(pos_rect);
% Select part of the image
img_cropped = img(pos_rect(2) + (0:pos_rect(4)), pos_rect(1) + (0:pos_rect(3)));

%% image mask
figure;
imshow(img_cropped);
h_rect = imrect();
% Rectangle position is given as [xmin, ymin, width, height]
pos_rect = h_rect.getPosition();
% Round off so the coordinates can be used as indices
pos_rect = round(pos_rect);
bkgrnd = max(max(img_cropped(pos_rect(2) + (0:pos_rect(4)), pos_rect(1) + (0:pos_rect(3)))));
binaryImage = img_cropped > bkgrnd;
cleanBinaryImage=bwareafilt(binaryImage, 10);
```

```
start1 = find(cleanBinaryImage(1,:)==1,1,'first');
finish1 = find(cleanBinaryImage(1,:)==1,1,'last');
start2 = find(cleanBinaryImage(end,:)==1,1,'first');
finish2 = find(cleanBinaryImage(end,:)==1,1,'last');
cleanBinaryImage(1,start1:finish1) = 1;
cleanBinaryImage(end,start2:finish2) = 1;
roundmask = imfill(cleanBinaryImage, 'holes');
figure; imshow(roundmask);
img_cropped(~roundmask) = 0;

%% flatten fiber
[flatImg, radius] = flattenFiber_v1(img_cropped,roundmask,umpxl);
% fftflatImg = fft2(flatImg);
% figure;imagesc(abs(fftshift(fftflatImg)));

%% binarize/filter image
contrastImg = localcontrast(flatImg,0.4,0.5);
figure; imshow(contrastImg);
contrastImg(~flatmask) = 0;
binImg = imbinarize(contrastImg,'adaptive');
figure; imshow(binImg);
s1 = strel('disk',1);
binImgDilate1 = imdilate(binImg,s1);
binImgFilt = bwareafilt(binImgDilate1,[50 inf]);
figure; imshow(binImgFilt);
% fftbinImg = fft2(binImg);
% figure;imagesc(abs(fftshift(fftbinImg)));
% fftbinImgFilt = fft2(binImgFilt);
% figure;imagesc(abs(fftshift(fftbinImgFilt)));

%% vertical autocorrelation
vertauto = zeros(length(binImgFilt(:,1))*2-1,1);
temp = [zeros(length(binImgFilt(:,1))-1,length(binImgFilt(1,:))); binImgFilt; zeros(length(binImgFilt(:,1))-
1,length(binImgFilt(1,:)))]; %zero buffer on both sides
for i = 1:(length(vertauto(:,1)))
   if i<=length(vertauto(:,1))/2
      vertauto(i) = sum(sum(binImgFilt.*temp(i:length(binImgFilt(:,1))-1+i,:)))/(length(binImgFilt(1,:))*i);
   else
      vertauto(i) = sum(sum(binImgFilt.*temp(i:length(binImgFilt(:,1))-
1+i,:)))/(length(binImgFilt(1,:))*(length(vertauto(:,1))+1-i));
   end
end

%% subtract background and normalize
vertbaselineauto = vertauto-min(vertauto(50:end-50));
normvertauto = vertbaselineauto/max(vertbaselineauto);
figure; plot(normvertauto);

% baselinevertauto = mean(normvertauto);
```

```
halfnormvertauto = normvertauto(round(length(normvertauto))/2:end);
vertfwhm = 2*find(halfnormvertauto <= 0.5, 1,'first');
% loc1 = find(normvertauto >= baselinevertauto+(1-baselinevertauto)/2, 1,'first');
% loc2 = find(normvertauto >= baselinevertauto+(1-baselinevertauto)/2, 1,'last');
% loc1 = find(normvertauto >= 0.5, 1,'first');
% loc2 = find(normvertauto >= 0.5, 1,'last');
% vertfwhm = loc2-loc1
```

Matlab Function: flattenfiber_v1

```
Function flatImg = flattenFiber_v1(img,mask,umpxl)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
roundcenterpoint = 0;
maxDiameter = 0;
%% find the diameter and centerpoint
for i = 1:length(mask(:,1))
   start = find(mask(i,:) == 1, 1, 'first');
   finish = find(mask(i,:) == 1, 1, 'last');
   diameter = finish-start;
   if diameter > maxDiameter
      maxDiameter = diameter;
      roundcenterpoint = round(start+diameter/2);
   end
end
% rounddistfromcenter = roundcenterpoint - (size(img,1)/2);
flatWidth = ceil(pi * size(img,2)); %2X the number of pixels needed to add resolution to the flattening
flatImg = uint8(zeros(2*size(img,1),flatWidth)); %2X number of rows to keep aspect ratio
flatcenterpoint = round((flatWidth/2));
%% map and flatten the fiber
for j = 1:length(img(:,1))
   start = find(mask(j,:) == 1, 1, 'first');
   finish = find(mask(j,:) == 1, 1, 'last');
   leftradius = roundcenterpoint-start; %just in case the left and right side are different distances from the
centerpoint
   rightradius = finish-roundcenterpoint;
   flatImg((2*j-1):2*j,flatcenterpoint) = img(j,roundcenterpoint); %the centerpoint remains the same, 2 rows
because of aspect ratio
   leftstartpoint = 1;
   %map and flatten the left side of the fiber
   for x = 1:leftradius-1
      y1 = sqrt(leftradius^2-x^2); %pixels
      y2 = sqrt(leftradius^2-(x+1)^2); %pixels
      dist = sqrt(1^2+(y2-y1)^2); %pixels
      theta = 2*asin((dist/2)/leftradius); %radians
      arclength = round(2*theta*leftradius); %pixels, 2X because increased resolution
      flatImg((2*j-1):2*j,flatcenterpoint-leftstartpoint-arclength+1:flatcenterpoint-leftstartpoint) =
img(j,roundcenterpoint-x); % 2 rows because of aspect ratio
      leftstartpoint = leftstartpoint+arclength;
   end
```

```
   rightstartpoint = 1;
   %map and flatten the right side of the fiber
   for x = 1:rightradius-1
      y1 = sqrt(rightradius^2-x^2); %pixels
      y2 = sqrt(rightradius^2-(x+1)^2); %pixels
      dist = sqrt(1^2+(y2-y1)^2); %pixels
      theta = 2*asin((dist/2)/rightradius); %radians
      arclength = round(2*theta*rightradius); %pixels, 2X because increased resolution
      flatImg((2*j-1):2*j,flatcenterpoint+rightstartpoint:flatcenterpoint+rightstartpoint+arclength-1) =
img(j,roundcenterpoint+x); % 2 rows because of aspect ratio
      rightstartpoint = rightstartpoint+arclength;
   end
   if j == 1 || j == round(length(img(:,1))/4) || j == round(length(img(:,1))/2) || j == round(3*length(img(:,1))/4) || j
== length(img(:,1))
      figure;
      cwt(double(flatImg(2*j,flatcenterpoint-leftstartpoint:flatcenterpoint+rightstartpoint-1)),2/umpxl); %figure out
how to use the sampling frequency here
      xlabel('Position (\mum)');
      ylabel('Frequency (cycles/\mum)');
      title(['Magnitude Scalogram:' newline 'Position = ' num2str(j*umpxl) ' \mum']);
   end
end
```