SUPPORTING INFORMATION

A Three-Channel Spectrometer for Wide-Field Imaging of Anisotropic Plasmonic Nanoparticles

Christina M. Sweeney,[†,§] *Colleen L. Nehl,*[†,§] *Warefta Hasan,*[†] *Taiyang Liang,*[†] *Amanda Eckermann,*[†] *Thomas J. Meade,*[†] *and Teri W. Odom*[†,‡,*]

§ - Co-first Authors

†Department of Chemistry, ‡Department of Materials Science and Engineering, Northwestern University, 2145 Sheridan Road, Evanston, Illinois 60208-3113

* To whom correspondence should be addressed. E-mail: todom@northwestern.edu.

Telephone number: 847-491-7674

Fax number: 847-497-7713

Email address: todom@northwestern.edu

Data Analysis of DF Scattering Spots

Tristimulus theory, which uses the summation of three colors with variable intensities to replicate most visible colors, is the basis by which digital cameras interpret color using the Bayer filter. The Bayer filter is composed of regions that are sensitive to specific wavelengths of light that roughly correspond to R, G, and B regions of the visible spectrum. By placing this filter over the CCD of a digital camera, the camera can interpret color because it only permits colors of certain wavelengths to impinge on the CCD in specific areas. The camera is programmed to demosaic, or average, neighboring RGB intensity values to simulate the color of the object. Using these ideas for 3CS, a MATLAB algorithm was composed to extract the average RGB values from each scattering spot in a dark-field optical image of metal NPs. As shown in the screen captures of the annotated code (Fig. S1), the RGB values for each particle can be determined, and then the algorithm can be used to determine particle characteristics based a predetermined metric value. The code in Fig. S1 was designed to distinguish between tip-up (U) from tip-down (D) nanopyramids.

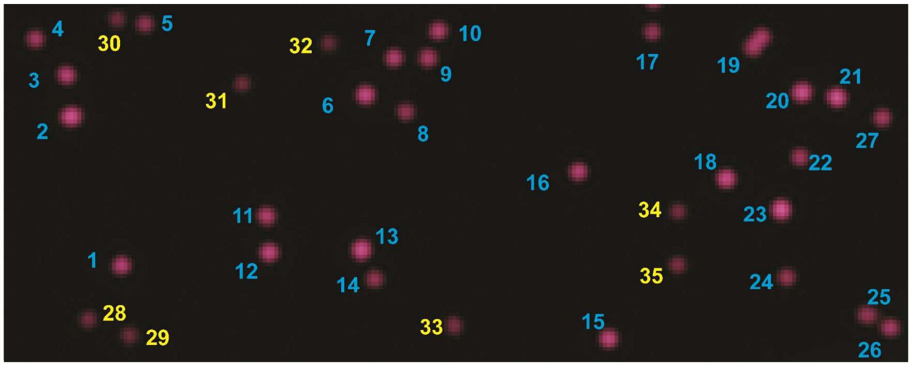
Part 1 describes how the image (tiff format) was accessed in its file directory and how a region of interest in the image was selected for analysis. Included in this part of the code was a separation of the RGB layers of the tiff image. Part 2 demonstrates how the program distinguished bright spots from the background. Here, a gray scale version of the image was used to select spots brighter than a variable threshold value. Part 3 describes how bright areas above the threshold were identified as particles based on the symmetry of the bright spot. Once the bright spot was established as a particle, the RGB values were extracted and summed. Part 4 finds the average RGB value for each spot and then compared the values to the metric value determined from the calibration. Here, R/G values above 3.2 were assigned as U and below as D. Part 5 displays the data as histograms and on the image itself. The histograms were particularly useful for determining the differentiation scheme during calibration.

```
%%Code outline:
% 1. Opens file, names experiment and defines region of interest (ROI)
% 2. Locates particles
% 3. Excludes non-circular particles and sums RGB values for remaining
% particles.
% 4. Computes averaged values of R,G,B, and inputs metric valuation for
% differentiation.
% 5. Histograms of R,G,B, and the metric.

clear
all=[0 0 0];            %Clears stored values, defines initial values for variables
area=0;
P1=[0 0 0];
z=[0];
numU=0;
numD=0;

%Part 1
name = input ('Enter name of file to analyze:    ','s');  %Prints to command line prompt for file name
expname = input ('Enter name of experiment:    ','s');     %Prints to command line prompt for experiment name
Pic1New = imread (strcat('C:\Users\Odom Group\Desktop\matlabsimulateddata\', name), 'tiff'); %Opens file/stores
ynnum=str2num(expname); %Changes yn from a string to a number.
figure, imshow (Pic1New)  %Shows the figure in a window
scrsz = get(0,'ScreenSize');  %Determines size of your monitor screen (useful for displaying images later).
figure, imshow (Pic1New)  %Shows picture
masklog=roipoly;  %Opens the ROIpoly function
mask = uint8 (masklog); %Converts mask data to unsigned 8 bit integers.
maskRGB = cat(3, mask, mask, mask );  %Creates a RGB version of the BW mask by stacking three levels.
masksavename= strcat (name, '_', 'mask', '_', expname);  %mask save name is name_mask_expname.
imwrite (maskRGB, masksavename, 'tiff');    %Saves maskRGB as masksavename.
Pic1= (maskRGB.*Pic1New);  %Applies the new mask to the picture.
imshow (Pic1);  %Shows the picture with the new mask.

%Part 2
grayPic = rgb2gray(Pic1);  %Creates grayscale version of picture.
figure, imshow (grayPic)   %Displays grayscale version of the picture.
hold on
bw=imextendedmax(grayPic,8);                 %Select brightness here, larger values exlude dim particles
figure, imshow(bw);

[B,L] = bwboundaries(bw,'noholes');   %Traces the exterior boundaries of objects, converts to black and white.
imshow(label2rgb(L, @jet, [.5 .5 .5]))  %Shows the boundary-traced object with the jet color scale.
imshow (Pic1)  %Shows the original picture on top of the boundary-traced object.

hold on

%Code below plots boundary around pyramids
for k = 1:length(B)  %for the first boundary (object) to the last object
  boundary = B{k};  %boundary
  plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 0.1) %plots a boundary around the object (traces border)
end
stats = regionprops(L,'Area','Centroid');  %Finds the area and centroid for every object
area;  %dummy
threshold = 0.6;  %sets threshold for how circular particles should be
% loop over the boundaries
figure ('name', 'With labels')  % displays the figure with labels
imshow (Pic1New) %shows the picture
hold on

%Part 3. Part 4 is a loop inside part 3.
for k = 1:length(B)  %for every object from 1 to the last one
  boundary = B{k};  %boundary
  delta_sq = diff(boundary).^2;      %Finds difference between adajacent elements of boundary
  perimeter = sum(sqrt(sum(delta_sq,2)));   %finds perimeter
  area = stats(k).Area;                        %Determines area
  metric = 4*pi*area/perimeter^2;   %NOT the ID metric for up vs down, for determination of circularness
```

```
        metric = 4*pi*area/perimeter^2;   %Not the ID metric for up vs down, for determination of circularness.
        % display the results
        metric_string = sprintf('%2.2f', metric);  %Shows the circularness metric.
                k;  %k
            centroidk = stats(k).Centroid;  %finds the centroid of k
            k_string = sprintf('%2.0f',k);   %prints the value of k

         if metric > threshold    %If metric > threshold (particles are circular enough), find RGB values
            z=z+1;
            P = impixel(Pic1, centroidk(1), centroidk(2));     %Finds the RGB values of each particle spot.
            j=8;   %j sets how large an area will be summed in the next for loop.
            z_string = sprintf('%2.0f',z);  %Prints z
          for i=1:j  %This is the next for loop, where we use j. i runs from 1 to j
              k;  %we use k too. dummy for now.
              i;  %this is what we're stepping. Here for trouble shoooting purposes.
              numU;  %number of tip-up
              numD;  %number of tip-down
            Pa = impixel(Pic1, (centroidk(1)+i), (centroidk(2)+i));   %We sum RGB values in 4 quadrants, a,b,c,d.
            Pb = impixel(Pic1, (centroidk(1)+i), (centroidk(2)-i));   %For each quadrant we're summing RGB
            Pc = impixel(Pic1, (centroidk(1)-i), (centroidk(2)+i));   %We sum RGB from one outside the centroid to i
            Pd = impixel(Pic1, (centroidk(1)-i), (centroidk(2)-i));
            P=((Pa+Pb+Pc+Pd)/4);  %Averages RGB for 4 quadrants
            Pavg=P1+P;  %running average of RGB. P1 defined at beginning.
            P1=Pavg;  %Sets P1 to the running average. For the first execution P1=P. For second, P1=P1+P.
          end
             centroid = stats(k).Centroid;            %Finds the centroid.
             P=Pavg;  %Sets P to Pavg.
             P=P/(j); %Sets P to P divided by j, the number of pixels summed outside the centroid.

         %Part 4. Part 4 code begins within the loops for part 3, after part 4 terminates the loops for part 3 end.
             UDMetric=(P(1)/P(2));   %This is metric designed to automate particle identification (R/G)

             red_string = sprintf('%2.2f',(P(1)));  %This line pulls out the red value from the P string.
             green_string = sprintf('%2.2f',(P(2))); %This line pulls out the green value from the P string.
             blue_string = sprintf('%2.2f',(P(3)));  %This line pulls out the blue value from the P string.
             METRIC_string = sprintf('%2.2f',(P(1)/P(2)));  %This line pulls the values needed to compute the metric.
              text(boundary(1,2)+5,boundary(1,1)+10,red_string,'Color','r',...
              'FontSize',8,'FontWeight','bold');   %Prints the value of the red.
              text(boundary(1,2)+5,boundary(1,1)-0,green_string,'Color','g',...
              'FontSize',8,'FontWeight','bold');    %Prints the value of the green.
              text(boundary(1,2)+5,boundary(1,1)-10,blue_string,'Color','b',...
              'FontSize',8,'FontWeight','bold');    %Prints the value of the blue.
              text(boundary(1,2)+15,boundary(1,1)+10,METRIC_string,'Color','y',...
              'FontSize',8,'FontWeight','bold');    %Prints the value of the metric.
         if UDMetric > 3.2  %This line sets the value that determines U from D.
                  text(boundary(1,2)+15,boundary(1,1)-20,'U','Color','c',...  %This line prints U for tip-up.
                  'FontSize',6,'FontWeight','bold');
              numU=numU+1;   %This line keeps track of the number of tip-up pyramids
         end

         if UDMetric <= 3.2
                  text(boundary(1,2)+15,boundary(1,1)-20,'D','Color','w',...      %This line prints D for tip-down.
                  'FontSize',6,'FontWeight','bold');
              numD=numD+1;    %This line keeps track of the number of tip-down pyramids
         end

              centroidk(1);
        all = cat(3 ,all, [P]);
         P1=[0 0 0];
      Pavg=[0 0 0];

         end

      end

```

```
%Part 5 - the code makes histograms
        blue= all(1,3,(2:z+1));      %pulls out the blue
        green= all(1,2,(2:z+1));     %pulls out the green
        red= all(1,1,(2:z+1));       %pulls out the red
        B=squeeze(blue);        %squeezes the blue into one number (not an array anymore)
        R=squeeze(red);         %squeezes the red into one number (not an array anymore)
        G=squeeze(green);       %squeezes the green into one number (not an array anymore)


GR=R+G+B;   %sum of red, green, blue
RdGinf=R./(G);  %R/G

RdGfin= isfinite (RdGinf) ;
RdG= RdGinf.*RdGfin;
%above 2 correct for infinite values (G=0)
Gnorminf=G./(R+G+B);

Gnormfin= isfinite (Gnorminf) ;
Gnorm= Gnorminf.*Gnormfin;
%above 2 correct for inf
Rnorminf=(R./G);

histsize=50;   %sets bin size in histogram.

Rfin= isfinite (Rnorminf) ;
UDMETRIC= Rnorminf.*Rfin;  %computes the metric.



figure          %Makes the histogram of blue values.
    drawnow
    shading interp
    hold on
hist(B,histsize);
h = findobj(gca,'Type','patch');
set(h,'FaceColor','b','EdgeColor','w')
ylabel('Number of Pyramids','fontsize',12,'fontweight','b')
xlabel('Blue Intensity from 0 to 255','fontsize',12,'fontweight','b')

figure            %Makes the histogram of red values.
    drawnow
    shading interp
    hold on
hist(R,histsize);
h = findobj(gca,'Type','patch');
set(h,'FaceColor','r','EdgeColor','w')
ylabel('Number of Pyramids','fontsize',12,'fontweight','b')
xlabel('Red Intensity from 0 to 255','fontsize',12,'fontweight','b')

figure           %Makes the histogram of green values.
    drawnow
    shading interp
    hold on
hist(G,histsize);
h = findobj(gca,'Type','patch');
set(h,'FaceColor','g','EdgeColor','w')
ylabel('Number of Pyramids','fontsize',12,'fontweight','b')
xlabel('Green Intensity from 0 to 255','fontsize',12,'fontweight','b')

figure     %Makes the histogram the metric values.
    drawnow
    shading interp
    hold on
hist(UDMETRIC,histsize);
h = findobj(gca,'Type','patch');
set(h,'FaceColor','k','EdgeColor','w')
ylabel('Number of Pyramids','fontsize',12,'fontweight','b')
xlabel('Metric','fontsize',12,'fontweight','b')
```

Figure S1. MATLAB Code for Extracting RGB data.



	VIS				NIR				NIR + BP (850 nm)			
Pyramid*	Blue	Green	Red	R/G	Blue	Green	Red	R/G	Blue	Green	Red	R/G
1	16.66	95.28	164.13	1.722	10.41	42.31	168.53	3.983	110.13	64.16	159.97	2.493
2	23.44	114.81	185.59	1.6165	16.88	53.44	199.09	3.725	138.72	84.91	189.38	2.230
3	27.28	121.56	192.50	1.5836	12.00	44.00	172.44	3.919	112.91	65.56	162.38	2.477
4	24.13	119.91	183.84	1.5331	8.47	31.81	147.50	4.637	90.75	51.25	135.50	2.644
5	15.50	80.22	162.25	2.023	7.31	29.63	134.84	4.551	86.19	46.34	137.31	2.963
6	60.56	147.44	201.03	1.3635	10.94	41.34	200.00	4.838	125.94	75.16	177.19	2.358
7	4.31	76.06	153.78	2.022	8.03	33.75	154.78	4.586	106.84	60.38	157.69	2.612
8	15.66	88.81	162.75	1.833	7.91	33.91	148.09	4.367	87.94	47.25	138.34	2.928
9	40.59	127.63	196.28	1.5379	9.47	41.78	195.72	4.685	103.91	56.41	155.44	2.756
10	8.34	76.47	155.56	2.034	9.63	35.44	161.75	4.564	107.78	61.97	158.69	2.561
11	19.16	108.16	203.25	1.8792	14.53	52.72	187.81	3.562	111.00	63.63	161.69	2.541
12	23.50	115.13	187.16	1.6256	13.38	44.41	170.06	3.829	116.44	68.34	166.78	2.440
13	36.22	135.56	200.94	1.4823	14.88	55.72	205.38	3.686	131.75	79.81	183.06	2.294
14	24.16	73.16	172.66	2.360	8.19	28.38	123.81	4.363	90.28	48.78	140.81	2.887
15	37.41	98.38	184.09	1.871	10.44	40.56	167.34	4.126	118.94	70.91	168.88	2.382
16	34.56	86.63	170.19	1.965	10.28	37.81	153.69	4.065	106.53	60.78	154.66	2.545
17	23.72	81.56	156.16	1.915	6.78	28.66	134.34	4.687	85.88	44.72	133.91	2.994
18	34.09	116.16	184.50	1.5883	12.16	44.44	171.75	3.865	122.88	73.59	173.63	2.359
19	59.91	146.31	205.84	1.4069	15.91	55.63	195.19	3.509	134.00	77.84	184.38	2.369
20	25.5	100.25	182.34	1.8189	13.81	52.63	185.16	3.518	131.47	78.78	184.25	2.339
21	41.97	116.41	186.72	1.6040	8.47	32.13	144.84	4.509	132.22	80.91	183.56	2.269
22	28.56	105.66	173.66	1.6436	8.22	40.09	179.88	4.487	94.28	51.16	144.91	2.832
23	39.53	128.81	202.41	1.5714	13.69	49.91	200.31	4.013	140.59	88.72	192.44	2.169
24	37.47	75.97	158.66	2.089	7.81	31.41	133.63	4.254	89.84	48.38	138.84	2.870
25	38.94	105.06	173.97	1.6559	5.34	25.75	150.31	5.837	101.13	53.38	154.38	2.892
26	16.34	74.56	149.06	1.999	8.25	35.34	163.13	4.616	108.03	59.63	160.53	2.692
27	18.84	84.63	162.50	1.920	7.09	40.53	177.72	4.385	96.66	52.97	146.66	2.769
28	***15.00***	***99.19***	***181.28***	***1.828***	***7.38***	***39.47***	***169.63***	***4.289***	***68.34***	***34.72***	***119.33***	***3.437***
29	***34.41***	***118.31***	***183.50***	***1.5510***	***7.00***	***35.28***	***163.09***	***4.623***	***66.28***	***33.84***	***116.81***	***3.452***
30	***25.51***	***119.59***	***183.47***	***1.5342***	***6.38***	***32.09***	***143.03***	***4.457***	***68.44***	***35.25***	***119.81***	***3.399***
31	***13.22***	***96.13***	***174.97***	***1.820***	***5.56***	***31.44***	***146.03***	***4.645***	***65.25***	***33.56***	***112.59***	***3.355***
32	***33.19***	***116.59***	***189.97***	***1.629***	***6.44***	***34.16***	***156.72***	***4.588***	***61.28***	***31.31***	***110.06***	***3.515***
33	***44.34***	***130.59***	***186.81***	***1.4305***	***7.00***	***36.00***	***153.31***	***4.259***	***72.25***	***37.53***	***121.09***	***3.226***
34	***49.00***	***129.06***	***193.44***	***1.4988***	***6.34***	***37.84***	***173.16***	***4.576***	***69.94***	***35.47***	***118.38***	***3.337***
35	***43.91***	***121.63***	***190.88***	***1.5693***	***7.47***	***41.53***	***181.97***	***4.382***	***73.22***	***36.91***	***124.44***	***3.371***

** Tip-up (U) particles indicated in bolded italics.*

Figure S2. RGB and R/G ratio values used to determine the orientation for Au nanopyramids. RGB intensity values were extracted from NIRcam (BP = 850 nm). Particles are arranged in the table by orientation. The first twenty-seven particles were identified as D and the remaining eight as U (indicated in bolded italics). Particle 1, 11, 12, 28, and 29 were used to calibrate the program to identify orientation and were examined in detail in Figure 3.

Pyramid*	Blue	Green	Red
1	***5.38***	***52.50***	***84.19***
2	***4.97***	***48.13***	***75.34***
3	***4.38***	***41.19***	***66.53***
4	***6.34***	***62.63***	***96.34***
5	***4.78***	***48.41***	***76.03***
6	3.28	20.72	36.25
7	2.91	23.91	41.94
8	5.13	48.72	77.22
9	3.16	26.91	47.06
10	3.63	28.00	45.25
11	3.88	34.22	54.72
12	2.94	32.06	54.06
13	3.25	26.88	44.97

VIScam + BP (580 nm)

* Au particles indicated in bolded italics.

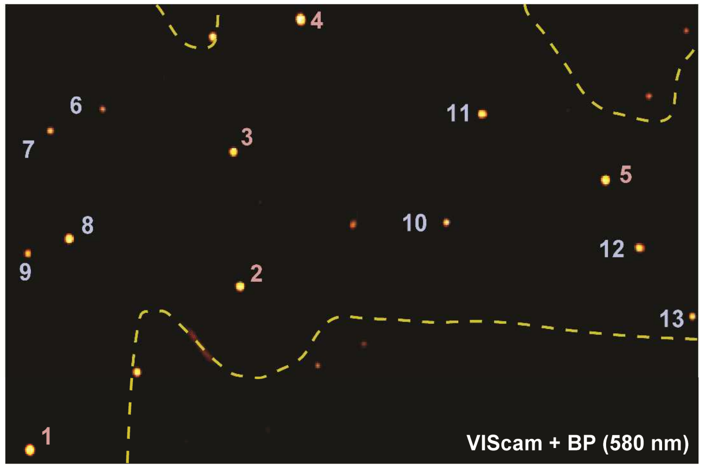


Figure S3. RGB channel intensities used to determine composition for Au and Au/Ag/Au nanopyramids. Intensity for the RGB values were extracted from VIScam (BP = 580 nm). Particles are arranged in the table by orientation. The first five particles were identified in SEM as Au (indicated in bolded italics) and the remaining particles were identified as Au/Ag/Au.

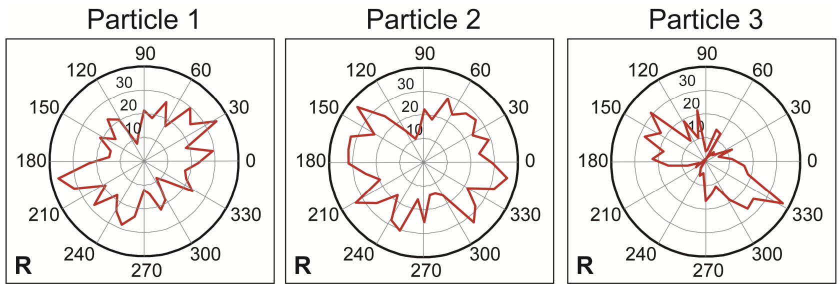


Figure S4. Polar plots of R-channel with NIRcam (no BP) for the three AuNSs. All images for polar plots were measured every 10° from 0° to 360° with an integration time of 30 s using the NIRcam and a polarizer. The orientations of the lobes are similar to those in Figure 5 but are less distinct because of contributions from other resonances.

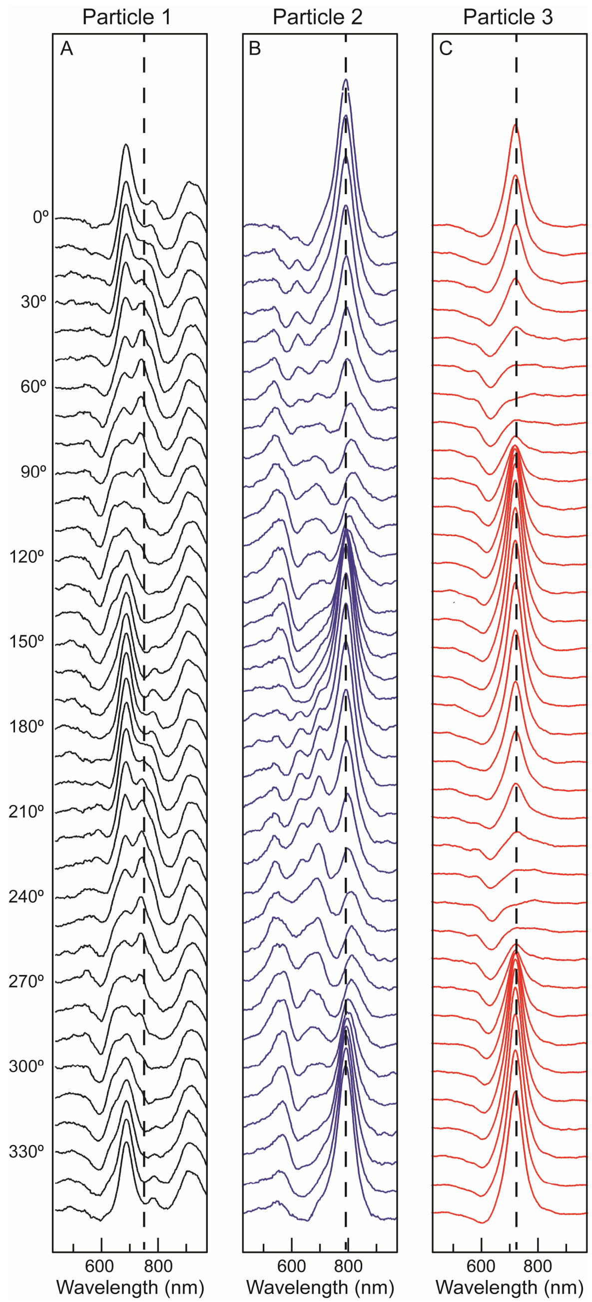


Figure S5. Polarization-dependent single-particle spectroscopy for Au NSs in Figure 5. Spectra were taken every 10° from 0° to 360°. Resonances corresponding to the longest arms are highlighted with a dashed line.

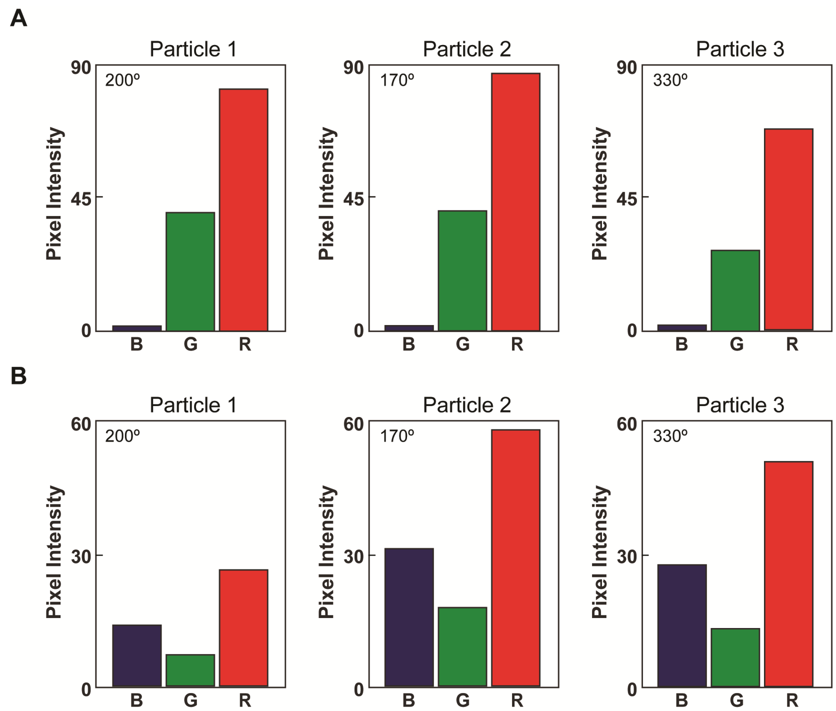


Figure S6. RGB histograms for AuNSs at maximum LSP intensity. Histograms of the RGB values for: **(A)** VIScam and **(B)** NIRcam (BP = 850 nm) of the three AuNSs. The intensity values were extracted from images at the polarization angle indicated and which corresponded to the orientation of the longest arms of each AuNS. The histograms demonstrate that contributions from other resonances, which can be attributed to different asperities, are present in the shorter wavelength regimes (G channels) in the VIScam data but are reduced by the addition of the BP filter.