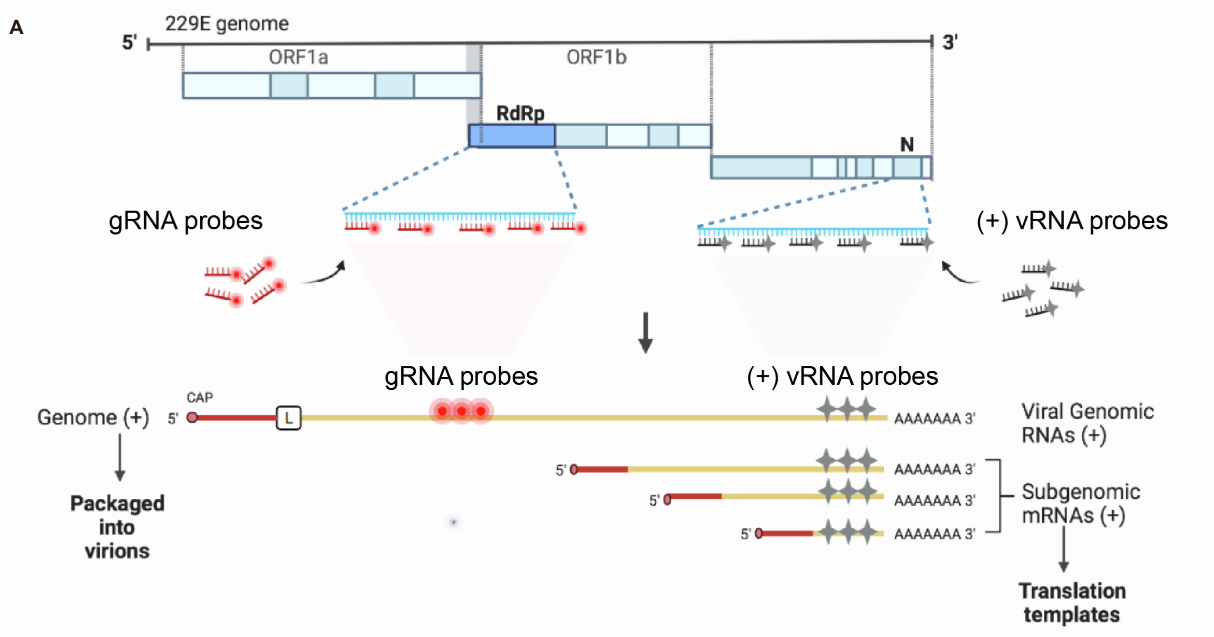
Cell Reports Methods, Volume *2*

Supplemental information

Multi-color super-resolution imaging to study human coronavirus RNA during cellular infection

Jiarui Wang, Mengting Han, Anish R. Roy, Haifeng Wang, Leonhard Möckl, Leiping Zeng, W.E. Moerner, and Lei S. Qi

Supplementary Figures



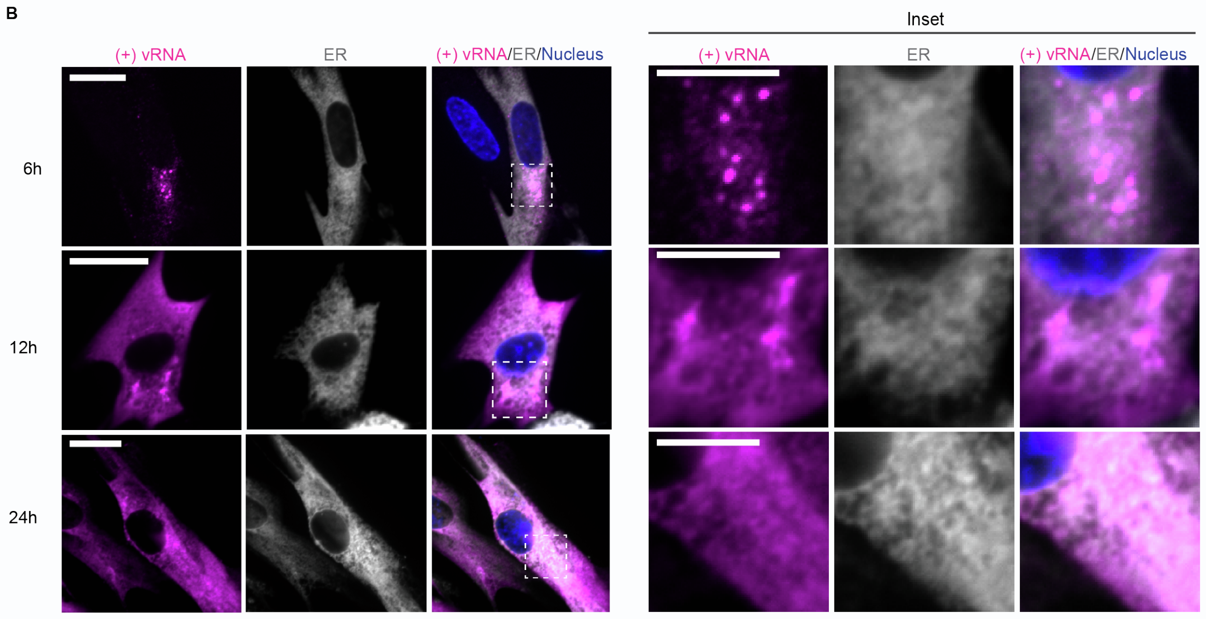
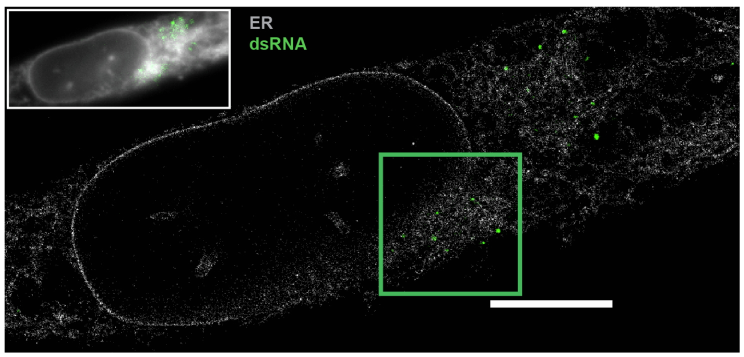
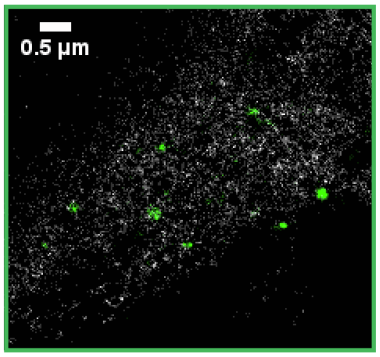
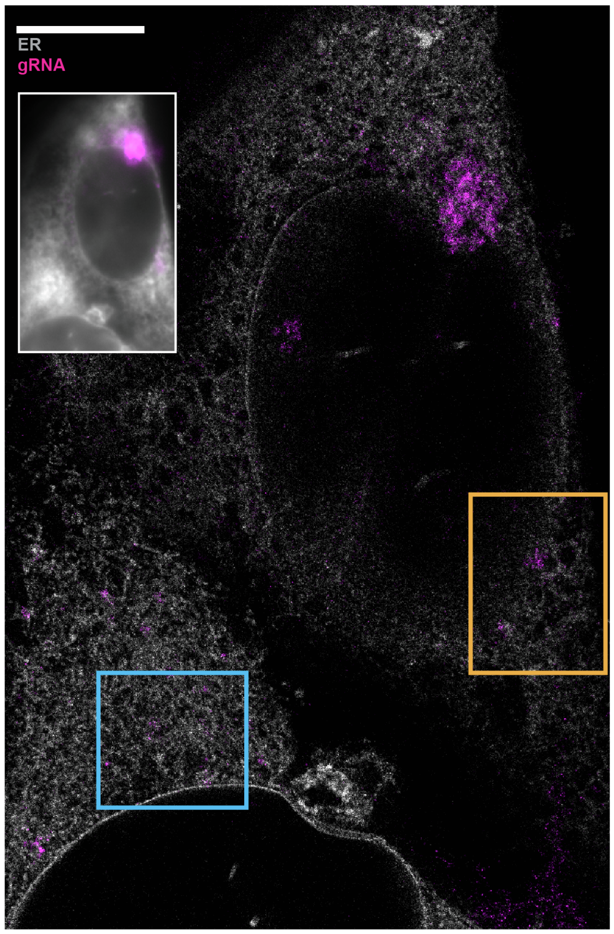


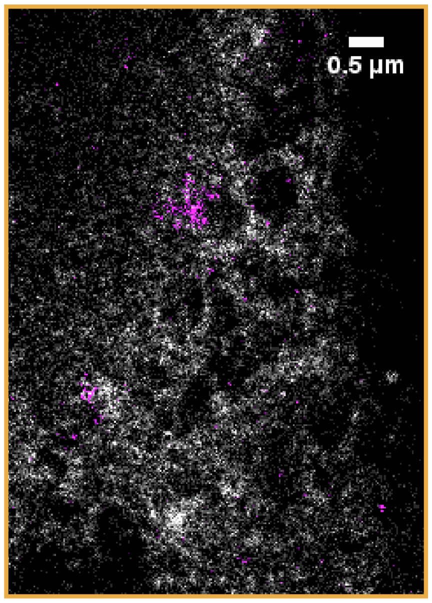
Figure S1. Visualization and labeling scheme of viral gRNA and sgRNA using FISH probes, Related to Figures 1-6.

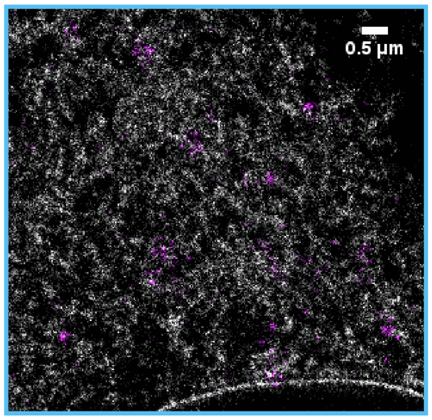
(A) Design of two sets of FISH probes targeting hCoV-229E viral gRNA and sgRNA. Top: 229E genomic construct map used for the detection of viral gRNA and sgRNA in fixed cells. The first FISH probe set (48 FISH probes, red) was designed to target genomic RNA (gRNA) by hybridizing with the RNA-dependent RNA polymerase (RdRp)-coding region, which is only present in the positive-sense gRNA and not in the subgenomic RNAs (sgRNAs). The second set (44 FISH probes, grey) was designed to target all positive sense viral RNA ((+) vRNA), including both gRNA and sgRNA, by hybridizing with the N protein-coding regions and 3'-UTRs, which is present in both genomic RNA and subgenomic RNAs. Bottom: Targeting regions of the two sets of FISH probes in the viral gRNA and sgRNA. The RdRp FISH probe only binds to the viral gRNA, and the N FISH probe binds to both viral gRNAs and sgRNAs. The FISH probes are labeled by CF568 or AF647. Adapted from templates "Discontinuous Transcription" and "Remdesivir Active Molecule Interaction with SARS-CoV-2 RdRp", by BioRender.com (2021). Retrieved from https://app.biorender.com/biorender-templates. (Hartenian et al., 2020) **(B)** Representative confocal images of (+) vRNA (magenta) in GFP-Sec61B-expressing fixed MRC5 cells (ER, gray). Cells were infected with 0.2 MOI 229E and fixed at 6-, 12-, and 24-hour post infection (h p.i.) Nucleus is labeled with DAPI (blue). Scale bar: 10 μm. Right: insets of dotted boxes. Scale bar: 5 μm.

A

ER
dsRNA

B

0.5 µm

C

ER
gRNA

D

0.5 µm

E

0.5 µm

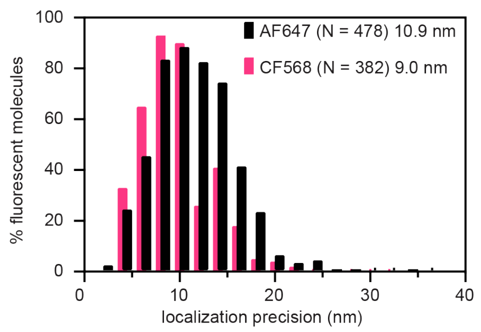
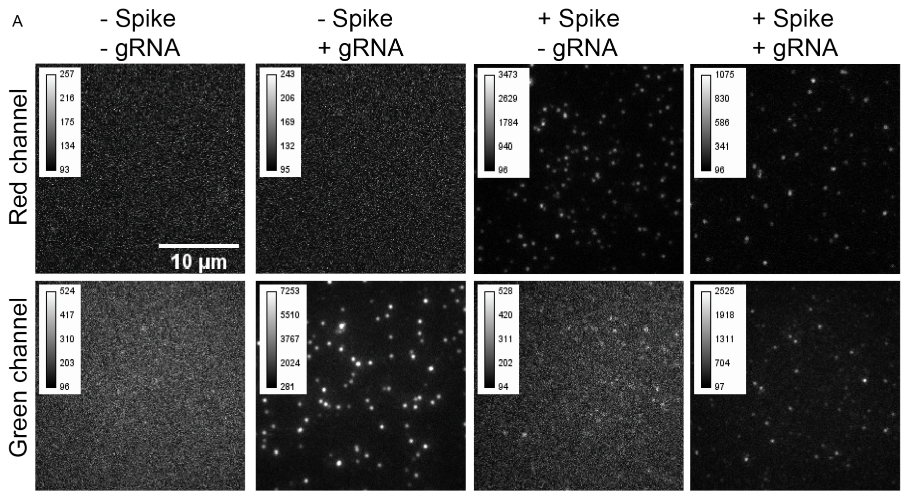
F

AF647 (N = 478) 10.9 nm
CF568 (N = 382) 9.0 nm
% fluorescent molecules
localization precision (nm)
0
20
40
60
80
100
0
10
20
30
40

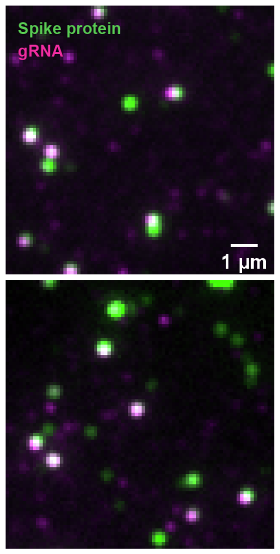
Figure S2. Additional examples of two-color SR reconstructions of both dsRNA and gRNA with the ER. Related to Figure 2.

(A) Two-color SR reconstruction of a cell where the ER (gray) and dsRNAs (green) are labeled. Scale bar: 5 µm. **(B)** Zoom-in of the boxed region. dsRNA forms compact puncta that appear in regions outside of the ER network. **(C)** Two-color SR reconstruction of a cell where the ER (gray) and gRNA (magenta) are labeled. Scale bar: 5 µm. **(D-E)** Zoom-ins of the green boxed region. (**F**) Localization precision of AF647 and CF568 determined experimentally. Dye conjugated FISH probes were immobilized to the cell chamber and imaged under the same conditions as cellular imaging. The localization precision is estimated by taking the standard deviation of the multiple measured positions of the same molecules.

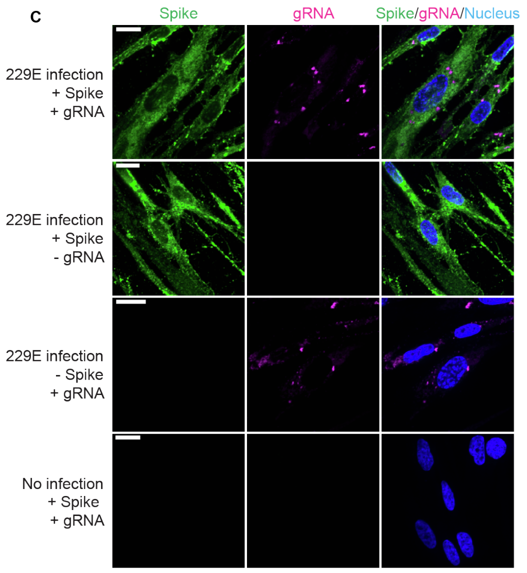
A



B



C



D

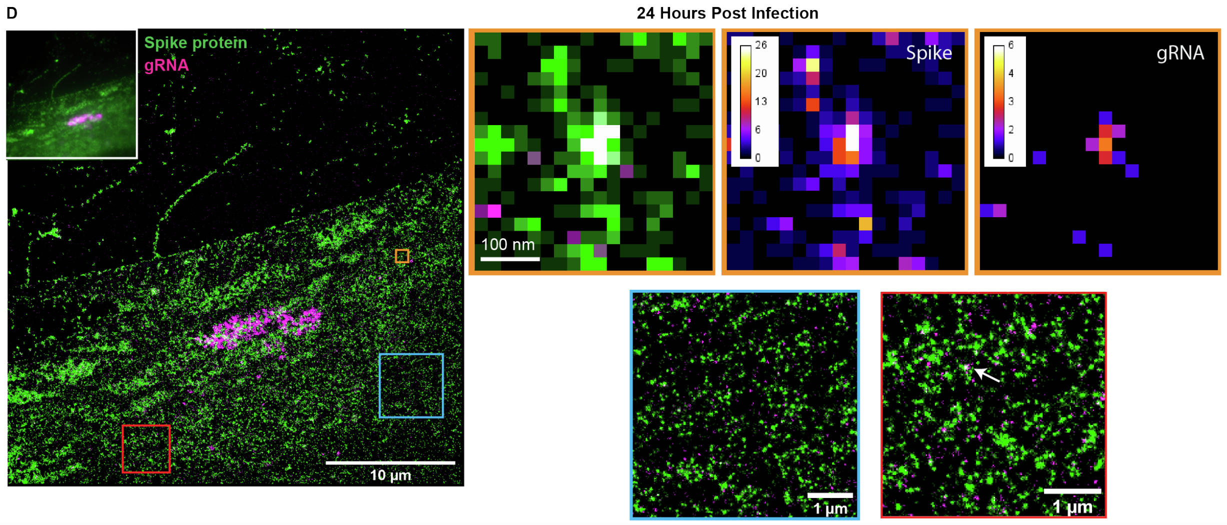


Figure S3. Two-color controls of gRNA and spike protein labeling and examples of two-color SR reconstructions of Spike protein and gRNA in cells. Related to Figure 3.

(A) Low nonspecific binding and cross talk of spike and gRNA labeled virions. For the -spike cases, samples were incubated with only secondary antibody. Virions were purified and immobilized on the coverslip in all conditions shown. Spike was labeled with AF647 antibody (red channel) and gRNA was labeled with FISH probes containing CF568 (green channel). Images show clear low nonspecific binding and low cross talk between the two-color channels. Calibration bar depicts camera counts. Images taken with a widefield fluorescence microscope. **(B)** DL images of labeled spike and gRNA virions. For both images, clear puncta with spike (green) and gRNA (magenta) as well as both are present. Overlap between the labels as shown by the white color is clear with some virions. **(C)** Representative confocal images showing labeled Spike proteins (green) and gRNA (magenta) in MRC5 cells at 24 h p.i and with no infection. Spike proteins were stained by anti-Spike antibody and AF647 labeled secondary antibody. gRNA was stained by CF568 labeled FISH probes. Nuclei are also stained using DAPI (Blue). Scale bar: 20 µm. The same brightness and contrast thresholds were applied for each group. **(D)** Two-color SR reconstruction of labeled spike and gRNA in a hCoV-229E infected cell at 24 h p.i. Left image shows the entire field of view of the cell with a DL image in the white inset. Orange box is an inset showing an example of a possible virion labeled with spike concentrically encapsulating a gRNA punctus. The concentric ring with white at the center is clearly visible and is similar to the purified virion SR images shown in **Figure 3I**. The color map in the single-color reconstructions indicates the number of detection events per pixel for both the spike and gRNA signal. The teal and red insets below show additional larger regions of spike and gRNA. Sometimes, the two targets colocalize as shown by the white arrow.

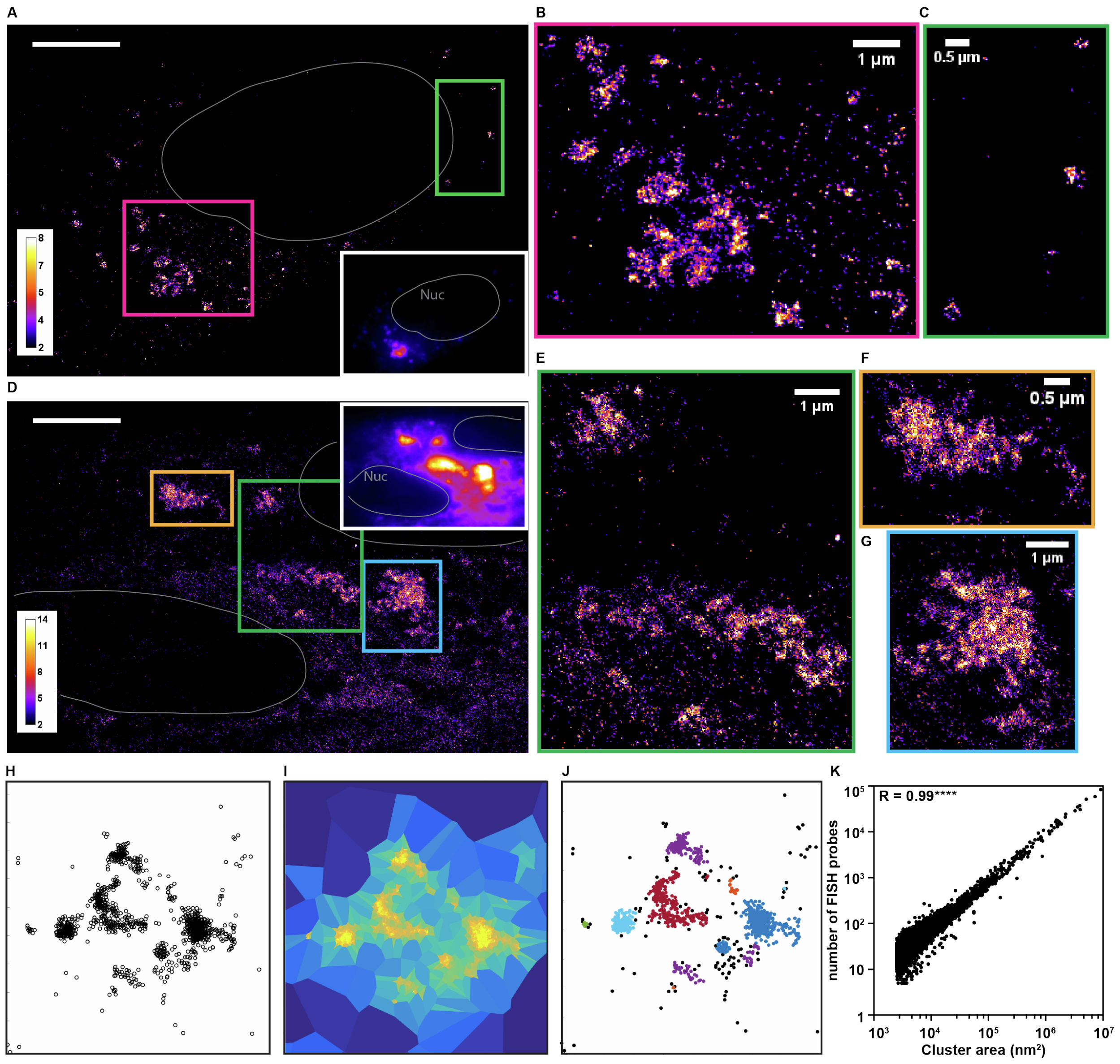


Figure S4. One-color SR reconstructions of gRNA at 6 and 24 h p.i and clustering analysis workflow. Related to Figures 2-3.

(A) SR reconstruction of gRNA in cell at 6 h p.i. Inset: diffraction-limited image of the same field of view. Scale bar: 5 µm. Color bar indicates the number of single-molecule detections within each pixel. **(B-C)** Zoom-ins of the colored boxed regions. **(D)** SR reconstruction of gRNA in cell at 24 h p.i. Inset: diffraction-limited image of the same field of view. Scale bar: 5 µm. Color bar indicates the number of single-molecule detections within each pixel. **(E-G)** Zoom-ins of the colored boxed regions. **(H)** Single molecules used as seeds for Voronoi tessellation. **(I)** Voronoi diagram generated for the single molecules shown in H. The coloring scheme indicates the area of each cell. **(J)** Cells above an area threshold and which share borders with each other were merged to generate the final clusters that are shown in different colors. **(K)** Plot of number of FISH probes versus cluster area. The two parameters are highly correlated, suggesting that the density of these gRNA clusters is relatively constant.

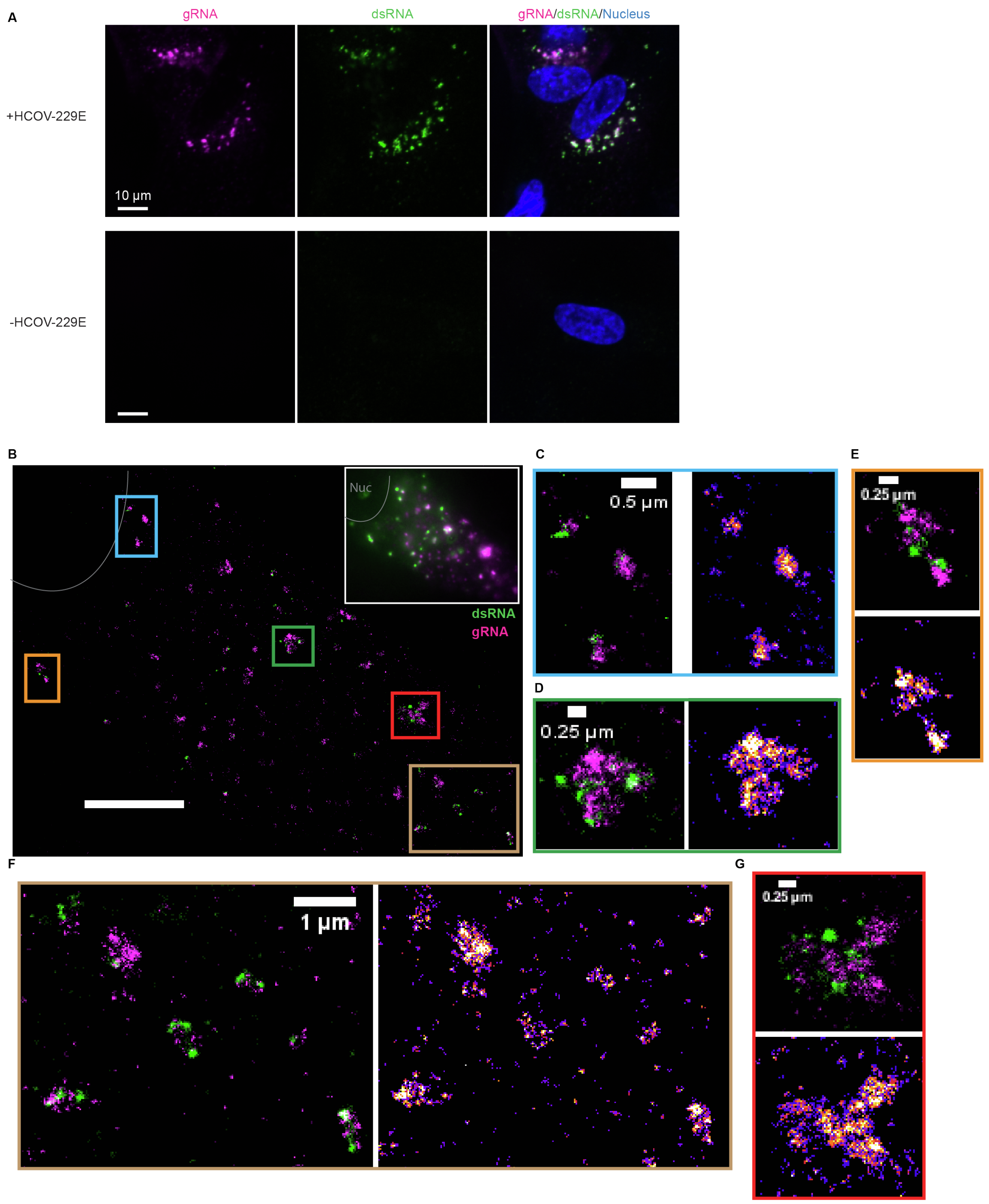


Figure S5. Additional examples of two-color SR reconstructions of dsRNA and gRNA at 6 h p.i. Related to Figures 4-5.

(A) Two-color confocal imaging of MRC5 cells infected with or without hCoV-229E virus for 12 h and stained with RdRP FISH probes (AF647) highlighting gRNA and anti-dsRNA antibody (CF568). The same brightness and contrast threshold were applied in all panels.

Scale bar: 10 µm. **(B)** Two-color SR reconstruction of a cell 6 h p.i. gRNA (magenta) and dsRNA (green) are labeled. dsRNA signals decorate the periphery of extended gRNA clusters. **(C-G)** Zoom-ins of the boxed regions. The color map in the single-color reconstructions indicates the number of detection events per pixel for the gRNA signal. dsRNA formed compact puncta that are in the vicinity of the gRNA clusters, but spatially separated (anticorrelated).

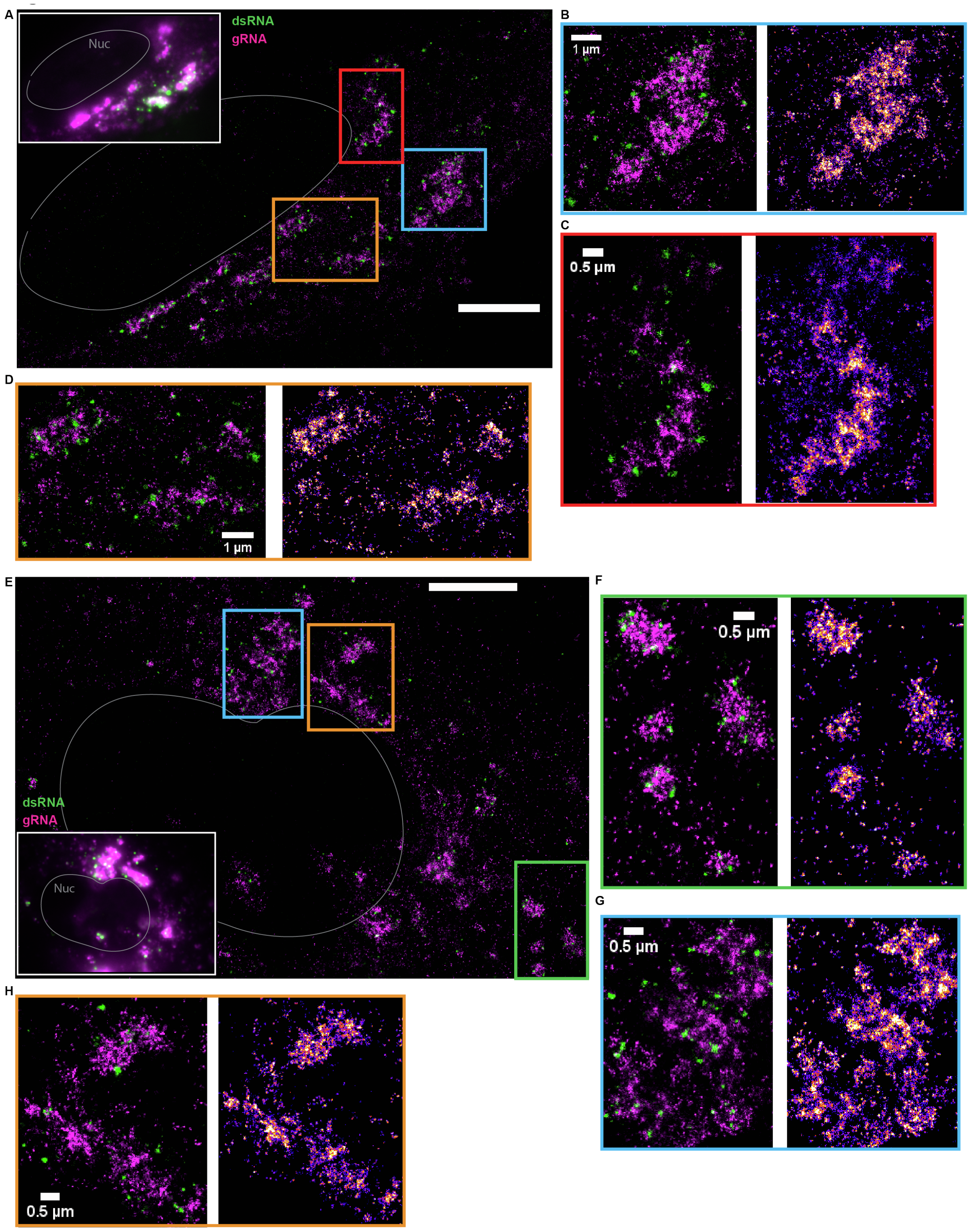
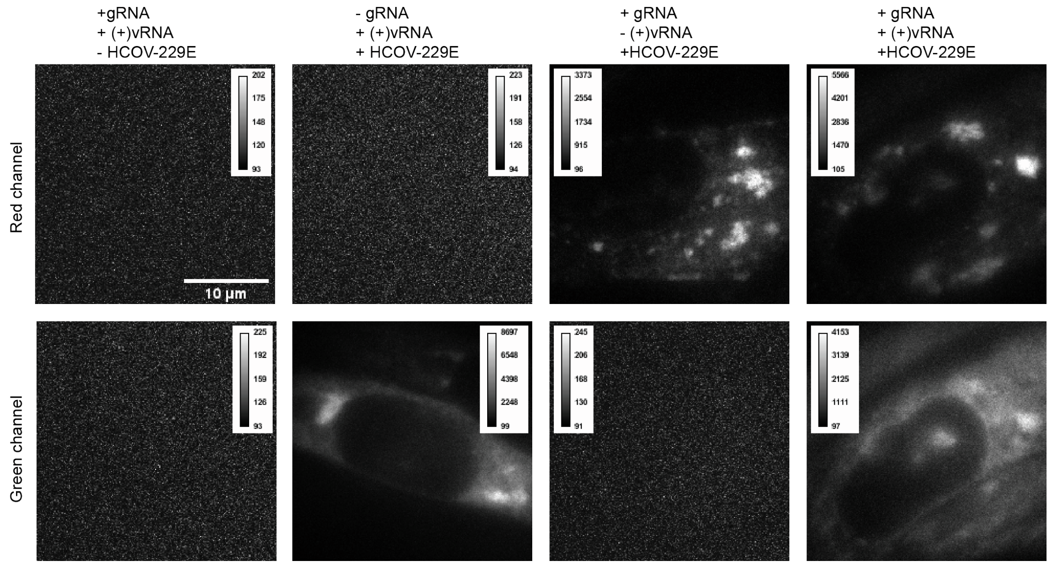


Figure S6. Additional examples of two-color SR reconstructions of dsRNA and gRNA at 12 h p.i. and 24 h p.i. Related to Figure 5.

(A) Two-color SR reconstruction of a cell 12 h p.i. gRNA (magenta) and dsRNA (green) are labeled. dsRNA signals decorate the periphery of extended gRNA clusters**. (B-D)** Zoom-ins of the boxed regions. The color map of single-color reconstruction indicates the number of detection events per pixel for the gRNA signal. dsRNA formed compact puncta that are in the vicinity of the gRNA clusters, but spatially separated. **(E)** Two-color SR reconstruction of a cell 24 h p.i. gRNA (magenta) and dsRNA (green) are labeled. dsRNA signals decorate the periphery of extended gRNA clusters. **(F-H)** Zoom-ins of the boxed regions. The color map in the single-color reconstructions indicates the number of detection events per pixel for the gRNA signal. dsRNA formed compact puncta that are in the vicinity of the gRNA clusters, but spatially separated.

A



B

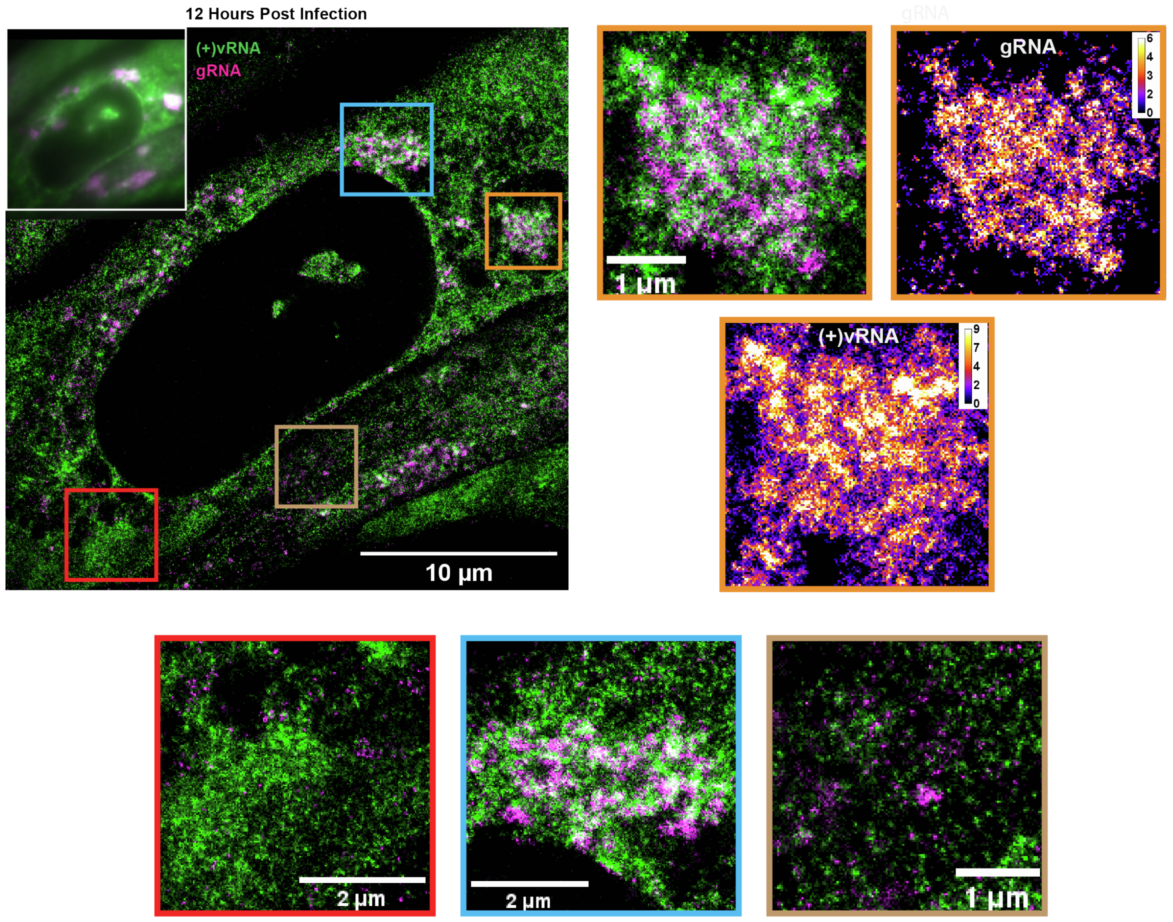


Figure S7. Examples of two-color SR reconstructions of gRNA and (+)vRNA. Related to Figures 2-3 and 5-6.

(A) Low nonspecific binding and cross talk of labeled gRNA and (+) vRNA in cells. gRNA was labeled with FISH probes with AF647 (red channel) and (+) vRNA was labeled with

FISH probes with CF568 (green channel). -hCoV-229E indicates that the cell was not infected. Images show clear low nonspecific binding of the FISH probes and low cross talk between the two-color channels. Calibration bar depicts camera counts. Images taken with a widefield DL microscope**. (B)** Two-color SR reconstruction of labeled gRNA (magenta) and (+) vRNA (green) in a hCoV-229E infected cell at 12 h p.i. Left image shows the entire field of view of the cell. DL image shown in the top left. Orange box is an inset showing an example of a large gRNA cluster. As expected, the white color indicates that the gRNA signal colocalizes with (+) vRNA signal. The color map for the single-color reconstructions indicates the number of detection events per pixel for both the gRNA and (+) vRNA signal. Cyan inset is another example of frequent colocalization in a large RNA cluster. Red inset shows a region where (+) vRNA is predominant. Brown inset shows a region where isolated gRNA is predominant.

Data S1. MATLAB code for clustering molecules using Voronoi diagram and calculating spatial point statistics. Related to Figures 3,5,6 and STAR Methods.

(A) MATLAB code for clustering molecules using Voronoi diagram

This following code clusters molecules based on Voronoi diagram and a user-defined density threshold. The code was adapted from code written Dr. Camille Bayas' and Dr. Maurice Youzong Lee's dissertations (Bayas, 2019; Lee, 2019).

```
%% find 2D voronoi vertices and cells
clear all
savename = 'date_sample';
[thunderFileName_R,thunderPathName_R] = uigetfile({'*.csv';'*.*'},'Open red file');
PSFFits_R_temp =csvread([thunderPathName_R thunderFileName_R],1,1);
xy=[PSFFits_R_temp(:,2) PSFFits_R_temp(:,3)];
[V,C] = voronoin(xy);
Vol_threshold = 500; %threshold for cell volume in nm2
mindist_threshold = 25; %threshold for closest seed distance in nm
% V is a numv-by-n array of the numv Voronoi vertices in n-dimensional space,
% each row corresponds to a Voronoi vertex. C is a vector cell array where
% each element contains the indices into V of the vertices of the corresponding Voronoi cell.
%% Figure
% % % scatter(xy(:,1),xy(:,2),5,'filled')
% % % axis image xy
% % % xlim([min(xy(:,1)) max(xy(:,1))])
% % % ylim([min(xy(:,2)) max(xy(:,2))])
% % % xlabel('X (nm)')
% % % ylabel('Y (nm)')
% % % caxis([-12 -4])

max_vertex=0;
density = []; % density of each cell
meanDistance = []; % mean distance to other seeds
nnDistance = []; % distance to closest seed
volume = zeros(length(xy),1);
% loop through each point, find vertices of each cell
for jj=1:length(C)
   if all(C{jj}~=1) %1 is [inf inf] represent unbounded (A row of Inf values represents an
unbounded cell.)
       VertCell=V(C{jj},:);
   else
       VertCell=V(C{jj}(2:end),:);
   end

   % calc volume of each cell
   try
       [~,volume(jj,1)]=convhulln(VertCell,{'Qt','Pp'});
   catch
       volume(jj,1)=nan;
```

```
    end
    if length(VertCell)>max_vertex
        max_vertex=length(VertCell);
    end
end % end loop through all points in this one color
%% calculate volume, density, nn distances...
vol_plot_lim = 5000; % this is just an upper limit for the histogram later
numbins = 500;
figure(1)
subplot(1,2,1)
hist(volume(volume<vol_plot_lim),numbins)
xlim([0 vol_plot_lim])
xlabel('Area of each voronoi cell (nm^2)')
subplot(1,2,2)
numbins = 500;
hist(-log(volume(volume<vol_plot_lim)),numbins)
xlabel('-log(Area) of each voronoi cell (nm^2)')


% calculate density from volume
%density(1:length(volume),1)= ones(length(volume),1)./volume;

% % calculate all distances from pt i to pt j
% allDists=nan(length(xy),length(xy));
% for i=1:length(xy)
%     for ii=1:length(xy)
%         if ii~=i
%             allDists(i,ii)=sqrt((xy(i,1)-xy(ii,1)).^2 + ...
%                 (xy(i,2)-xy(ii,2)).^2);
%         end
%     end
% end
% % calculate mean distance of seed i to all other seeds
% meanDistance=nanmean(allDists,2);

% calculate distance from seed i to closest seed (nn dist)
% do this by finding all nn distances and picking shortest one

% for i=1:length(xy)
%     XYcurr=xy(i,:);
%     otherXYcurr=xy;
%     otherXYcurr(i,:)=[];
%     [~,nnDistance(i)]=knnsearch(otherXYcurr,XYcurr);
%     %[~,allDists(i,1)]=knnsearch(otherXYcurr,XYcurr);
% end
% nnDistance=nnDistance';
% clear allDists
%
% figure(100)
%     subplot(1,3,1);histogram(density);
%     set(gca, 'XScale', 'log')
%     xlabel('density (locs/nm^2)');ylabel('counts'); title('Density')
%     subplot(1,3,2);histogram(meanDistance);
%     xlabel('mean distance to other seeds (nm)');ylabel('counts'); title('Mean dist')
%     set(gca, 'XScale', 'log')
%     subplot(1,3,3);histogram(nnDistance(nnDistance<500));
%     xlabel('distance to closest seed (nm)');ylabel('counts'); title('Nn dist');
%
%% VISUALIZE the voronoi plot with color as density but using PATCHES instead
figure(4)
for i = 1:length(C)
   if all(C{i}~=1)   % If at least one of the indices is 1,
      % then it is an open region and we can't patch that.
%        patch( Vertex_allcells(Vert_eachcell{i},1) , Vertex_allcells(Vert_eachcell{i} , 2), ...
%            -log(Vol_expt(i)),'EdgeColor','none'); % use color i.
      patch( V(C{i},1) , V(C{i} , 2), ...
         -log(volume(i)),'EdgeColor','none'); % use color i.
   end
end
colormap(parula)
% colormap(dusk)
colorbar
axis image xy
xlim([min(xy(:,1)) max(xy(:,1))])
ylim([min(xy(:,2)) max(xy(:,2))])
xlabel('X (nm)')
ylabel('Y (nm)')
caxis([-12 -4])
%% Set threshold of for density of Voronoi cells that are considered clustered

% any voronoi patch smaller than this threshold is considered clustered
% choose low area cells, check which have vertices touching

vertexforeachcell = zeros(numel(volume),max_vertex,'single');
for i = 1:numel(volume)
   thisvertices = C{(i)};
   vertexforeachcell(i,1:numel(thisvertices)) = thisvertices;
end
vertexforeachcell(vertexforeachcell==0) = NaN;

%%Assign cluster numbers to the Voronoi cells that are considered small

% initialize a log to contain the ID of each cluster
ClusterNumberLog = zeros(numel(volume),1);
ClusterNumberLog_big = zeros(numel(volume),1);
% give a cluster number to all small cells
small_molecule_ID = find(volume<Vol_threshold);
```

```
big_molecule_ID = find(volume>Vol_threshold); %note some contain NaN as volume, will not be
counter here
ClusterNumberLog_big_IND=[];
ClusterNumberLog(small_molecule_ID) = small_molecule_ID; % row numbers of patches
smaller than threshold
ClusterNumberLog_big(big_molecule_ID,1) = big_molecule_ID;
ClusterNumberLog_big_IND = find(ClusterNumberLog_big>0); % find small patches

%%Prepare for clustering small voronoi patches that are touching
ClusterNumberLog_small_IND = find(ClusterNumberLog>0); % find small patches
ClusterNumberLog_small = ClusterNumberLog(ClusterNumberLog_small_IND); % use only the
small patches
ClusterNumberLog_small = repmat(ClusterNumberLog_small,[1 size(vertexforeachcell,2)]); %
convert into same size as vertexforeachcell_small

vertexforeachcell_small = vertexforeachcell(ClusterNumberLog_small_IND,:); % only vertices
from small patches

% reshape for parfor loop
vertexforeachcell_small_reshape =
reshape(vertexforeachcell_small,[numel(vertexforeachcell_small),1]);
ClusterNumberLog_small_reshape =
reshape(ClusterNumberLog_small,[numel(vertexforeachcell_small),1]);
ClusterNumberLog_small_reshape_new = ClusterNumberLog_small_reshape;

disp(['Start with ',num2str(numel(unique(ClusterNumberLog_small_reshape))),' clusters.'])
changeiszero = 1;
tic
while changeiszero > 0
  %% Change cluster numbers to be the smallest number for touching patches
    %%
    %1 - list vert from smallest to largest and list their corresponding
    %cluster number log ===[B,index] = sortrows(___); such that B = A(index,:)
    [verticesANDclusters, sortrowsindex] = sortrows([vertexforeachcell_small_reshape
ClusterNumberLog_small_reshape]);
    %2 - ismember on list itself returns the ind of the first smallest!!!
    %vertices the current ind is with. index = 0 when vertex = NaN
    [~,index] = ismember(verticesANDclusters(:,1), verticesANDclusters(:,1));
    %get vertice cluster log where index is not 0
    newclusters = verticesANDclusters(index(index>0),2);
    %switch the vertice log to the new log that
    verticesANDclusters(1:length(newclusters),2) = newclusters;

    %put back clusterNumLog given the order now
    ClusterNumberLog_small_reshape_new(sortrowsindex) = verticesANDclusters(:,2);

    disp(['Now with ',num2str(numel(unique(ClusterNumberLog_small_reshape_new))),'
clusters.'])
    %put back into orignal format where each row is a cell
    ClusterNumberLog_small_reshape_new = reshape(ClusterNumberLog_small_reshape_new ,
size(vertexforeachcell_small));
    %find minn in each cell
    ClusterNumberLog_small_reshape_new = min(ClusterNumberLog_small_reshape_new,[],2);

    ClusterNumberLog_small_reshape_new =
repmat(ClusterNumberLog_small_reshape_new,[size(vertexforeachcell,2) 1]);

    changeiszero = max((ClusterNumberLog_small_reshape_new -
ClusterNumberLog_small_reshape).^2);

    ClusterNumberLog_small_reshape = ClusterNumberLog_small_reshape_new;
end

%%Calculate core cluster area and then periphery molecules
% expand ClusterNumberLog_small_reshape to include patches that were not small enough
% 0s are the large ones
ClusterNumberLog(small_molecule_ID) =
ClusterNumberLog_small_reshape_new(1:numel(ClusterNumberLog_small_IND));
cluster_vol2 = [];
cluster_CoM=[];
% Change the cluster numbers to be the unique values without space between
% them
uniqueclusternumbers = unique(ClusterNumberLog);

for ii = 2:numel(uniqueclusternumbers)
    ClusterNumberLog(ClusterNumberLog==uniqueclusternumbers(ii)) = ii-1;
end


%%Calculate area for clusters (not including periphery)
for ii = 1:max(ClusterNumberLog)
    cluster_vol2 = [cluster_vol2;sum(volume(ClusterNumberLog==ii))];
    cluster_CoM = [cluster_CoM; mean(xy(cell_ind,:),1)];
end
disp(['Clustering Finished!'])
save(savename);

%%
tic
changeiszero = 1;
while changeiszero > 0
    oldClusterNumberLog = ClusterNumberLog;
    for ii = 1:max(ClusterNumberLog) % for each cluster
        % for each cluster, find the max nearest neighbor within
        thisclusterX = xy(ClusterNumberLog==ii,1);
        thisclusterY = xy(ClusterNumberLog==ii,2);
        maxNNdist = 0;
        for jj = 1:numel(thisclusterX)
```

```
            diffX_1 = thisclusterX - thisclusterX(jj);
            diffY_1 = thisclusterY - thisclusterY(jj);
            diffXY_1 = sqrt(diffX_1.^2+diffY_1.^2);
            diffXY_NN = min(diffXY_1(diffXY_1>0));
            maxNNdist = max([maxNNdist diffXY_NN]);
        end
        % check X and Y (that were in big cells) and change their cluster number if need be
        for jj = 1:numel(thisclusterX)
            diffX_2 = xy(ClusterNumberLog_big_IND,1) - thisclusterX(jj);
            diffY_2 = xy(ClusterNumberLog_big_IND,2) - thisclusterY(jj);
            diffXY_2 = sqrt(diffX_2.^2+diffY_2.^2);
            id_temp = find(diffXY_2<maxNNdist);
            ClusterNumberLog(ClusterNumberLog_big_IND(id_temp)) = ii; %changing from 0 to
small cluster id
            ClusterNumberLog_big_IND(id_temp)=[]; %don't need to check them again
        end
    end
    newClusterNumberLog = ClusterNumberLog;
    changeiszero = sum((newClusterNumberLog-oldClusterNumberLog).^2);
    disp(['Current change = ',num2str(changeiszero)])

end
save(savename);

%% Scatter compare before and after adding periphery molecules
% % % figure(4)
% % % subplot(1,2,1)
% % % hold on
% % % scatter(xy(ClusterNumberLog==0,1),xy(ClusterNumberLog==0,2),50,'k','filled')
% % %
scatter(xy(ClusterNumberLog>0,1),xy(ClusterNumberLog>0,2),50,ClusterNumberLog(ClusterNu
mberLog>0),'filled')
% % % hold off
% % % axis image xy
% % % colormap(lines)
% % % subplot(1,2,2)
% % % hold on
% % %
scatter(xy(ClusterNumberLog_big_IND,1),xy(ClusterNumberLog_big_IND,2),50,'k','filled')
% % % scatter(xy(ClusterNumberLog_small_IND,1),xy(ClusterNumberLog_small_IND,2), ...
% % %
50,ClusterNumberLog_small_reshape_new(1:numel(ClusterNumberLog_small_IND)),'filled')
% % % hold off
% % % axis image xy
% % % colormap(lines)
%% Find area of clusters

uniqueclusternumbers=unique(ClusterNumberLog);
cluster_xy={};
```

```
cluster_mol_num=[]; %num mol in the cluster
cluster_CoM = []; %center of mass
cluster_vol2 = [];

% scatter(xy(ClusterNumberLog==0,1),xy(ClusterNumberLog==0,2),5,'k','filled')
% hold on
for i=2:size(uniqueclusternumbers)
    cell_ind = find(ClusterNumberLog==uniqueclusternumbers(i));
    pts_in_cell = xy(cell_ind,:);
    cluster_xy{i-1}= xy(cell_ind,:);
    cluster_mol_num=[cluster_mol_num;(size(pts_in_cell,1))];
%     k = boundary(pts_in_cell(:,1),pts_in_cell(:,2));
%     cluster_vol = [cluster_vol;polyarea(pts_in_cell(k,1),pts_in_cell(k,2))];
    cluster_vol2 = [cluster_vol2;sum(volume(cell_ind))];
    cluster_CoM = [cluster_CoM; mean(xy(cell_ind,:),1)];
%     try
%         [k,cluster_vol(i-1,1)]=convhulln(pts_in_cell);
%     catch
%         cluster_vol(i-1,1)=nan;
%     end
end
save(savename);

%% plot cluster
% %
figure(1)
hold on

plot_cluster_num = 60;
k = boundary(cluster_xy{plot_cluster_num}(:,1),cluster_xy{plot_cluster_num}(:,2));
%k = convhull(cluster_xy{plot_cluster_num}(:,1),cluster_xy{plot_cluster_num}(:,2));

hold on
scatter(xy(ClusterNumberLog_big_IND,1),xy(ClusterNumberLog_big_IND,2),50,'k','filled')
scatter(xy(ClusterNumberLog_small_IND,1),xy(ClusterNumberLog_small_IND,2), ...
    50,ClusterNumberLog_small_reshape_new(1:numel(ClusterNumberLog_small_IND)),'filled')
scatter(cluster_xy{plot_cluster_num}(:,1),cluster_xy{plot_cluster_num}(:,2),20,'m')
plot(cluster_xy{plot_cluster_num}(k,1),cluster_xy{plot_cluster_num}(k,2));
hold off
axis image xy
colormap(lines)
% % %
% % %%
%
% histogram(cluster_vol(cluster_vol>0));
% set(gca, 'Yscale', 'Log');
```

(B) MATLAB code for calculating spatial point statistics

The following code quantifies spatial point statistics between two-color super-resolution reconstruction data and compares with complete spatial randomness (CSR) that is generated by function csr_bw_sim.m.

```
%% Selecting files and set parameters
clear all
%load('20210119_6hr_r1_SPS.mat')
[fish_FileName,fish_PathName] = uigetfile({'*.tif';'*.*'},'Open red/FISH file');

[ab_FileName,ab_PathName] = uigetfile({'*.tif';'*.*'},'Open green/Ab file');

m_thresh=3;
g_thresh=3;
max_rad = 70; %maximum radius to calculate
fileinfo = imfinfo(fish_FileName);
img_magenta = double(imread([fish_PathName fish_FileName],1, 'Info', fileinfo));
img_magenta = img_magenta>m_thresh;
fileinfo = imfinfo(ab_FileName);
img_green = double(imread([ab_PathName ab_FileName],1, 'Info', fileinfo));
img_green = img_green>g_thresh;


savename='date_sample';

 %% User choose areas to calculate
% imagesc(img_green,[min(img_green(:)),max(img_green(:))./5]);axis image;colormap hot;
twoc = [];
twoc(:,:,1)=img_green;
twoc(:,:,2)=img_magenta;
twoc(:,:,3)=img_magenta;
imagesc(twoc);axis image;colormap jet;
title('finished? To continue - Left, to terminate - Right')
num_ROIs = 0;
but = 1;
while but == 1
    hold on;
    % increment number of ROIs chosen so far
    num_ROIs=num_ROIs+1;
    [ROI_collect(:,:,num_ROIs), xi, yi] = roipoly;
    ROIs{num_ROIs}=[[floor(min(xi)) ceil(max(xi))];[floor(min(yi)) ceil(max(yi))]];
    plot(xi, yi, 'Color','m', 'LineWidth',2)
    [gx,gy,but] = ginput(1);
end

%% Test ROI
% i =1;
% img_magenta_curr = ROI_collect(:,:,i).*img_magenta;
% img_green_curr = ROI_collect(:,:,i).*img_green;
% figure(2);
% subplot(1,3,1)
% imagesc(img_green_curr,[min(img_green_curr(:)),max(img_green_curr(:))]);axis
image;colormap gray;
% subplot(1,3,2)
% imagesc(img_magenta_curr,[min(img_magenta_curr(:)),max(img_magenta_curr(:))]);axis
image;colormap gray;
% subplot(1,3,3)
% imagesc(csr_img_magenta,[min(csr_img_magenta(:)),max(csr_img_magenta(:))]);axis
image;colormap gray;

%% calculate SPS for each ROI
% For circle coordinates
[CoordsX CoordsY] = meshgrid(1:size(img_magenta,1), 1:size(img_magenta,1));
res_cumulative_all=cell(num_ROIs,1);
res_cumulative_all_1derivative=cell(num_ROIs,1);
res_cumulative_all_CSR=cell(num_ROIs,1);
res_cumulative_all_1derivative_CSR=cell(num_ROIs,1);
tic
for roi_curr = 1:num_ROIs
%get ROI
storearea = [];
%the rectangle shape that harbors the irregular polygon inside
x_start = ROIs{roi_curr}(1,1);
y_start = ROIs{roi_curr}(2,1);
x_range = ROIs{roi_curr}(1,2)-x_start;
y_range = ROIs{roi_curr}(2,2)-y_start;

% 2 color data within polygon roi of choice
img_magenta_curr=ROI_collect(:,:,roi_curr).*img_magenta;
img_green_curr=ROI_collect(:,:,roi_curr).*img_green;
%% simulate csr image for both green and magenta use ROI signal density
roi_w = x_range+1;
roi_h = y_range+1;

csr_m = csr_bw_polygon_sim(ROI_collect(:,:,roi_curr),img_magenta, roi_w,
roi_h,x_start,y_start);
csr_g = csr_bw_polygon_sim(ROI_collect(:,:,roi_curr),img_green, roi_w, roi_h,x_start,y_start);

%% Calculate cross-correlation from green data to magenta data
%%Initialize the results.
res_raw = cell(x_range+1,y_range+1);

 for dx = 0:x_range
   for dy = 0:y_range  % going through the pixels
      if img_green_curr(y_start+dy, x_start+dx)==1 % checking if the green pixel is 1
         res_raw{dx+1, dy+1} = zeros(max_rad-1,1); % initializing the results matrix
```

```
            for rad = 1:max_rad
                % Drawing the rings and summing up all pixels in the other
                % channel.
                circle1 = (CoordsX - (x_start+dx)).^2 + (CoordsY - (y_start+dy)).^2 <= rad.^2;
                circle2 = (CoordsX - (x_start+dx)).^2 + (CoordsY - (y_start+dy)).^2 > (rad-1).^2;
                ring = circle1.*circle2;
                num_pix_ring = sum(circle1(:)==1)-sum(circle2(:)==0); %number of pixel in this ring
(area)
                storearea =[storearea;num_pix_ring];
                ring_in_roi = ROI_collect(:,:,roi_curr).*ring;
                num_pix_ring_inroi = sum(ring_in_roi(:)==1);
%                   figure(2)
%                   subplot(1,2,1)
%                   imagesc(ring);
%                   hold on
%                   plot(a,b, 'LineWidth',2,'MarkerSize',10);
%                   axis image
%                   subplot(1,2,2)
%                   imagesc(ring_in_roi);
%
%                   hold on
%                   plot(a,b, 'LineWidth',2,'MarkerSize',10);
%                   axis image
                image_magenta_circle = img_magenta_curr.*ring;
                area_adjust=num_pix_ring/num_pix_ring_inroi;
                res_raw{dx+1, dy+1}(rad) = sum(image_magenta_circle(:))*area_adjust;

            end
        end
    end
end

%% calculate cross-correlation from green csr to magenta csr
res_raw_CSR = cell(x_range+1,y_range+1);
for dx = 0:x_range
    for dy = 0:y_range  % going through the pixels
        if csr_g(y_start+dy, x_start+dx)==1 % checking if the green pixel is 1
             % initializing the results matrix
            res_raw_CSR{dx+1, dy+1} = zeros(max_rad-1,1);
            for rad = 1:max_rad
                % Drawing the rings and summing up all pixels in the other
                % channel.
                circle1 = (CoordsX - (x_start+dx)).^2 + (CoordsY - (y_start+dy)).^2 <= rad.^2;
                circle2 = (CoordsX - (x_start+dx)).^2 + (CoordsY - (y_start+dy)).^2 > (rad-1).^2;
                ring = circle1.*circle2;
                num_pix_ring = sum(circle1(:)==1)-sum(circle2(:)==0); %number of pixel in this ring
(area)
                storearea =[storearea;num_pix_ring];
                ring_in_roi = ROI_collect(:,:,roi_curr).*ring;
                num_pix_ring_inroi = sum(ring_in_roi(:)==1);
                area_adjust=num_pix_ring/num_pix_ring_inroi;
                image_magenta_csr_circle = csr_m.*ring;
                res_raw_CSR{dx+1, dy+1}(rad) = sum(image_magenta_csr_circle(:))*area_adjust;
            end
        end
    end
end

%% Postprocessing the raw results to extract a single trace for the ROI.
res_cumulative = zeros (max_rad,1);
res_cumulative_CSR = zeros (max_rad,1);

for dx = 0:x_range
    for dy = 0:y_range  % going through the pixels
        if ~isempty(res_raw{dx+1,dy+1})
            res_cumulative = res_cumulative + res_raw{dx+1,dy+1};
        end
    end
end
res_cumulative(isnan(res_cumulative))=0;

for dx = 0:x_range
    for dy = 0:y_range  % going through the pixels
        if ~isempty(res_raw_CSR{dx+1,dy+1})
            res_cumulative_CSR = res_cumulative_CSR + res_raw_CSR{dx+1,dy+1};
        end
    end
end
res_cumulative_CSR(isnan(res_cumulative_CSR))=0;

res_cumulative_all{roi_curr}=res_cumulative;
%res_cumulative_all_1derivative{roi_curr}=res_cumulative(2:end)-res_cumulative(1:end-1);

res_cumulative_all_CSR{roi_curr}=res_cumulative_CSR;
%res_cumulative_all_1derivative_CSR{roi_curr}=res_cumulative_CSR(2:end)-
res_cumulative_CSR(1:end-1);

toc

end
disp(['Finished!'])
%save(savename);
%% calculate signal density in ring area
% rad  = 1:max_rad;
% area = pi*(rad.^2-(rad-1).^2);
% area = area';
nmPerPixel = 122/5;
```

```
x_rad_plot = 1:max_rad;
x_rad_plot =x_rad_plot*nmPerPixel;

% imagesc(ROI_collect(:,:,roi).*twoc);
% axis image

%%
roi =1;
data = res_cumulative_all{roi}./storearea(1:max_rad);
csr = res_cumulative_all_CSR{roi}./storearea(1:max_rad);
% data = data./(sum(data));
% csr = csr./(sum(csr));

end_of_plot = 30;
figure(10)
plot(x_rad_plot(1:end_of_plot), data(1:end_of_plot),'r','LineWidth',2)
hold on
plot(x_rad_plot(1:end_of_plot), csr(1:end_of_plot),'k','LineWidth',2)
legend({'dsRNA to 229E','Complete Spatial Randomness'},'Location','southeast','FontSize',16)
xlabel('R (nm)')
ylabel('signal density')

function [csr_img] = csr_bw_sim(num_ones, roi_w, roi_h)
initial_csr = zeros((roi_w)*(roi_h),1);
initial_csr(1:round(num_ones)) = ones(round(num_ones),1);
%then scamble
initial_csr_idx = randperm((roi_w)*(roi_h))';
initial_csr = initial_csr(initial_csr_idx);
%then reshape into ROI size
csr_img = reshape(initial_csr,[roi_w roi_h]);
end
```