File S1. MatLab script for colony image analysis

```
%% System preparation
% In MatLab, install the add-on packages `subplot_tight`, `qr_code` and `QR
Code Generator 1.1`. Also make sure you have the Image processing toolbox.
% Download zxing 'core-3.3.1.jar' and 'javase-3.3.1.jar' and add the
javapath in Matlab and points to the folder storing the java files just
downloaded.

%% Generate QR codes before experiments
% Import sample names from Excel as cell array
d=QRgenerationS2; % d is the cell array imported
map=[0,0,1
    1,1,1]
for i=1:numel(d)
qr = qrcode_gen(d{i},'size',137,'QuietZone',2); %size rull:17+4N
colormap(map);
subplot_tight(16,9,i,.0125);
imagesc(qr);
axis off;
text(2,26,d{i},'FontSize',7,'Color','blue');
end
set(gcf,'Position',[3,0,589,841],'PaperType','a4');
% Print it directly from the figure output window
% Choose RGB color when print: print - print preview - color - RGB
% QR code can be printed on stickers and stick on each petri dish.

%% Make folders to save images and data during image analysis
Folder = '/Users/Documents'; %% Folder with images to analyze
Files = dir('*.JPG');
% Build two new folders
mkdir ConvertedImage;
convertedImageFolder='/Users/Documents/ConvertedImage';
mkdir FinalLabeledImage;
FinalLabeledImageFolder='/Users/Documents/FinalLabeledImage';
mkdir Data;
DataFolder='/Users/Documents/Data';

%% Image analysis __ Image calibration
```

```
% Image calibration need to be done every time when re-set or move the
camera.
im = imread('.JPG'); % choose a sample image
imc=imcrop(im,[4298 1154 4868 1940]); % cut unnecessary blank space in
image
imshow(imc);
hold on
% Click two dots in the image, which are separated with known distance as 1
cm. Use calibration papers when take photos of images.
    ok = 0;
    while ok == 0

        disp('Click corners of large grid square')

        % Input corners of large grid squares
        [xCorner1, yCorner1] = ginput(1);
        plot(xCorner1, yCorner1, 'r.', 'MarkerSize', 20)

        [xCorner2, yCorner2] = ginput(1);
        plot(xCorner2, yCorner2, 'r.', 'MarkerSize', 20)

        % Check the points
        choice = questdlg('Are the corners accurate?',...
            'Verification',...
            'Yes', 'No', 'Yes');
        % Handle response
        switch choice
            case 'Yes'
                scale = sqrt((xCorner1 - xCorner2)^2 + (yCorner1 -
yCorner2)^2); % scale in pixels per cm
                caliPara = 1/scale; % calculate the calibration parameter
                disp(['Pixels per cm = ', num2str(scale)])
                ok = 1;
            case 'No'
                disp('Try again')
                close all
                imshow(frame)
                hold on
                continue
        end
```

```
    end

hold off

%% Image analysis __ image conversion & particle measurements
%% Run the analysis for each image in a loop
for r = 1:numel(Files)
    baseFileName = Files(r).name;
    im=imread(baseFileName);
    a=imcrop(im,[1430 580 2270 2250]);% Crop away the noisy background
    name=regexprep(baseFileName,'.JPG','');
    %Read QR code
    c=imcrop(im,[2147 2726 3215 3452]); % All the QR code need to be on the
top of the plate
    tad=imadjust(rgb2gray(t));
    filename=decode_qr(tad);

    if isnan(filename)==0
        name=filename;
    else
        name=regexprep(baseFileName,'.JPG','');
    end

    % Choose a circular area and mask all the background to black
    ci = [1185, 1105, 1080];
    imageSize = size(a);
    [xx,yy] = ndgrid((1:imageSize(1))-ci(1),(1:imageSize(2))-ci(2));
    mask = uint8((xx.^2 + yy.^2)<ci(3)^2);
    croppedImage = uint8(zeros(size(a)));
    croppedImage(:,:,1) = a(:,:,1).*mask;
    croppedImage(:,:,2) = a(:,:,2).*mask;
    croppedImage(:,:,3) = a(:,:,3).*mask;
    imshow(croppedImage);

% Convert image
    hsvIm = rgb2hsv(croppedImage);
    ycb = rgb2ycbcr(hsvIm);

% Generate mask;
```

```
    bwIm = ycb(:,:,1) < 0.43; % Use the 'Image threshold' app in Matlab to
pre-estimate the threshold by changing Channel 1 or 3.
    bwTmp = imclearborder(bwIm);
    bwIm2=bwareaopen(bwTmp,50);
    bw = imfill(bwIm2,'holes');
    imshowpair(ycb,bw,'montage');
    convertedName=[name 'Converted.tif'];
    saveas(gcf,fullfile(convertedImageFolder,convertedName),'tif');

% Find connected components
    labeledbw = bwlabel(bw,8);
    bwMeasurement=regionprops(labeledbw,'Area','Perimeter');
% Choose only components with certain circularities
    allAreas = [bwMeasurement.Area];
    allPerims = [bwMeasurement.Perimeter];
    circularity = (allPerims .^ 2) ./ (4 * pi * allAreas);
    allowabelCircularityIndexes= circularity<1.10; % Set the circularity
threshold: to exclude connected colonies, the circularity needs to be set
to <3.
    allowabelAreaIndexes= (allAreas>30) & (allAreas<5000); % Set the area
threshold (unit in pixel).
    keeperIndexes=find(allowabelCircularityIndexes & allowabelAreaIndexes);
    keeperComponents=ismember(labeledbw, keeperIndexes);
    labelKeeperComponents=bwlabel(keeperComponents,8); % Label the chosen
components in the newly generated image.

KeeperComponentsMeasurement=regionprops(keeperComponents,'Centroid','PixelI
dxList','Area');

% Draw boundaries for each labeled components
    imshow(a);
    hold on;
    boundary= bwboundaries(labelKeeperComponents);
    numberOfBoundary = size(boundary, 1);
    for aa = 1 : numberOfBoundary
        thisBoundary = boundary{aa};
        plot(thisBoundary(:,2), thisBoundary(:,1), 'g', 'LineWidth', 2);
    end
    hold off
```

```
% Draw label numbers in each component on the RGB image
    numberOfkeeperComponents = size(KeeperComponentsMeasurement, 1);
    KeeperComponentsCentroids = [KeeperComponentsMeasurement.Centroid];
    KeeperComponentsArea = [KeeperComponentsMeasurement.Area];
    centroidsX = KeeperComponentsCentroids(1:2:end-1);
    centroidsY = KeeperComponentsCentroids(2:2:end);
    textFontSize = 14;
    labelShiftX = -13;

% Add index number to each component
    for j = 1 : numberOfkeeperComponents
        text(centroidsX(j), centroidsY(j), num2str(j),'color','y',
'FontSize', textFontSize);
    end

    LabeledName=[name 'Labeled' '.tif'];
    saveas(gcf,fullfile(FinalLabeledImageFolder,LabeledName),'tif');

% Calculate Grey Value
    d = rgb2gray(a); % The nature of the grey value here is illuminance,
which can be affected by the texture, thickness and color of colonies, and
light condition when taking pictures.
    p=[];
    q={};
    for k = 1:numel(KeeperComponentsMeasurement)
        oneIdxlist = KeeperComponentsMeasurement(k).PixelIdxList;
        oneGray = d(oneIdxlist);
        meanGray = uint8(mean(oneGray));
        p=[p meanGray];
        q=[q k];
    end

% Write data out
    pp=num2cell(transpose(p));
    qq=num2cell(transpose(q));
    KeeperComponentsMeasurement2 =
rmfield(KeeperComponentsMeasurement,'PixelIdxList');
    [KeeperComponentsMeasurement2(:).MeanGrey]=pp{:};
    [KeeperComponentsMeasurement2(:).ColonyNum]=qq{:};
```

```
    worldArea=num2cell([KeeperComponentsMeasurement.Area].*caliPara.^2); %
Transform pixel number into cm^2
    [KeeperComponentsMeasurement2(:).worldAreacm]=worldArea{:};
    xlsfilename=[name '-CC' '.xls'];
    if exist(xlsfilename,'file')
       delete(xlsfilename);
    end
    if ~exist(xlsfilename,'file')

writetable(struct2table(KeeperComponentsMeasurement2),fullfile(DataFolder,
xlsfilename));
    end

end

clear all;

%% Integrate data of each image into a file
% Go to DataFolder
DataFolder='/Users/zzhong/Desktop/reanalyze_picture/8dpi/KCl/Data';
DataCC = dir('*-CC.xls');
namelist1={DataCC.name};
PlateNameCC=regexprep(namelist1,'-CC.xls','');
MeanAreaCC=[];
MeanWorldAreaCC=[];
MeanGreyCC=[];
ColonyNumCC=[];
for w = 1: numel(DataCC)
    basefilenamedata=DataCC(w).name;
    T=readtable(basefilenamedata);
    if (isempty(T.ColonyNum))
        MeanAreaCC =[MeanAreaCC NaN];
        MeanWorldAreaCC =[MeanWorldAreaCC NaN];
        MeanGreyCC=[MeanGreyCC NaN];
        ColonyNumCC=[ColonyNumCC 0];
    else
        MeanAreaCC = [MeanAreaCC trimmean(T.Area,10)];
        MeanWorldAreaCC =[MeanWorldAreaCC trimmean(T.worldAreacm,10)];
        MeanGreyCC = [MeanGreyCC trimmean(T.MeanGrey,10)];
        ColonyNumCC = [ColonyNumCC max(T.ColonyNum)];
    end
```

```
end

x=table(transpose(PlateNameCC), transpose(ColonyNumCC),
transpose(MeanAreaCC), transpose(MeanWorldAreaCC),transpose(MeanGreyCC));
x.Properties.VariableNames =
{'IsoCondiRep','ColonyNumCC','MeanAreaCC','MeanWorldAreaCC','MeanGreyCC'};
writetable(x,fullfile(DataFolder,'DataSummary.xls'),'Sheet',1,'Range','A1')
;

clear all;
clc;
```