Electronic Supplementary Information

Fluorescence-Activated Droplet Sorting of Lipolytic Microorganisms Using a Compact Optical System

Yuxin Qiao[a, b], Xiaoyan Zhao[a], Jun Zhu[a], Ran Tu[a], Libing Dong[a], Li Wang[a], Zhiyang Dong[a, b], Qinhong Wang[d], Wenbin Du[a, c, *]

[a] State Key Laboratory of Microbial Resources, Institute of Microbiology, Chinese Academy of Sciences, Beijing 100101, China
[b] College of Life Sciences, University of the Chinese Academy of Sciences, Beijing 100049, China
[c] Savaid Medical School, University of the Chinese Academy of Sciences, Beijing 100049, China
[d] Key Laboratory of Systems Microbial Biotechnology, Tianjin Institute of Industrial Biotechnology, Chinese Academy of Sciences, Tianjin 300308, China

* Corresponding author: wenbin@im.ac.cn (E-mail)

Chemicals and materials

All reagents were purchased from commercial sources and used as received unless otherwise stated. Silicon wafer was purchased from Qimin Silicon Material Co., Ltd. (Shanghai, China). Photomasks were designed in AutoCAD and ordered from Heheng Optoelectronics (Changsha, China). SU-8 photoresist was purchased from MicroChem Corp. (Newton, MA, USA). Poly-dimethylsiloxane (PDMS) was purchased from Momentive Performance Materials Inc. (Waterford, NY, USA). Syringe pumps (Pump 11 PicoPlus Elite, Harvard Apparatus, Holliston, MA, USA) was used for controlling fluid flow in microfluidic devices. Gastight glass syringes (Agilent, Santa Clara, CA, USA) with volume of 250 μL or 500 μL were used to deliver solutions. LB broth medium and Potassium phosphate were purchased from Sangon (Shanghai, China). Marine broth 2216 was purchased from BD (Franklin Lakes, New Jersey, USA). Peptone was purchased from Aoboxing Bio-Tech (Beijing, China). Magnesium sulfate heptahydrate was purchased from Sinopharm (Shanghai, China). Ammonium sulfate was purchased from Guanghua Sci-Tech (Guangzhou, China). *p*-nitrophenyl palmitate (*p*-NPP) was purchased from Aladdin (Shanghai, China). Tributyrin was purchased from Tokyo Chemical Industry (TCI, Tokyo, Japan).

Construction and operation of the compact optical system for FADS.

The compact optical system for FADS was composed of a detection module to measure the fluorescence of picoliter droplets, a high voltage module to deliver high voltage-high frequency electric field to solder-injected electrodes, an Edgertronic™ high speed camera (SC1 Monochrome version, Sanstreak Corp., San Jose, CA, USA) to capture the images and videos, and a power supply to control the microscope and the FADS system. The detection module had all optical components aligned and die bonded in a compact size (160 mm×143 mm×54 mm), and was connected to the side port of an standard inverted microscope (IX81; Olympus, Tokyo, Japan) via its V-mount without further optical alignment. The detection module used a high-power diode laser (200 mW, 470 nm, NDB7675, Nichia, Tokushima, Japan), three photomultipliers (R5600U, Hamamatsu Photonics, Hamamatsu, Japan), and a 2-MHz 24bit analog-to-digital conversion circuit. This compact module measured three different fluorescence signals of various fluorophores simultaneously as listed in Table S1. In this work, only one photomultiplier (PMT1) was used for measurement of FDB hydrolysis by lipase. The FADS process was controlled using a program written in LabVIEW software (National Instruments, USA). Time series recordings were analysed using a program written in Matlab (MathWorks, Natick, MA, USA) to extract the fluorescence intensities of all droplets. The histograms were generated to show the distribution of droplet fluorescence intensity, as well as to determine the threshold of sorting. The sorting threshold was defined based on the histogram we obtained for each experiment, and might vary based on PMT gain (controlled by high voltage applied to the photomultiplier).

The leakage test for the fluorogenic substrate FDB in picoliter droplet system

We designed and made a microfluidic device with two flow-focusing junctions which can produce 1:1 proportion of positive droplets containing 25 μM FDB and 40 U mL^{-1} lipase, and negative droplets containing 25 μM FDB without lipase. The droplets were directly mixed and collected in an Eppendorf tube. 10 μL of mixed droplets were pipetted into a Countess™ cell counting chamber (Cat. No. C10228, ThermoFisher, USA), and sealed with capillary wax to avoid evaporation. To investigate the leakage of fluorescence dye between droplets, the monolayer droplet array in the chamber was imaged by fluorescence microscopy (Eclipse Ti, Nikon, Tokyo, Japan).

Amplification, sequencing and phylogenetic analysis of 16S rRNA genes

The 16S rRNA genes of obtained bacterial strains were amplified by PCR with the forward primer 27F and the reverse primer 1492R as described[1]. The obtained sequences were assembled using SeqMan software (DNASTAR). The sequence similarity was analyzed by comparing 16S rRNA gene sequences of obtained strains with known sequences available from the EzTaxon-e database (http://www.ezbiocloud.net/eztaxon)[2], The phylogenetic analysis based on complete 16S rRNA gene sequences indicates that the strains are closely related to reported strains (98.56% to 100%, as listed in Supplementary Table S2). Sequences were then aligned using CLUSTAL W[3], and the phylogenetic tree was established with MEGA 6 (Molecular Evolutionary Genetics Analysis version 6) program[4], using the neighbor-joining[5] and the maximum-likelihood method, with bootstrap values based on 1000 replications[6].

Lipase activity assay.

Analysis of extracellular lipase activity was carried out following the colorimetric method using *p*-NPP as the substrate. First, we mixed 100 μL *p*-NPP (0.15 to 3 mg/mL in isopropanol) with 1.8 mL Tris-HCl (50 mM, pH7.5), and incubated it at 37 °C for 5 min. A concentrated enzyme solution of 100 μL volume obtained by ultrafiltration was then added, and incubated at 37°C for another 10 min. During this step, *p*-nitrophenol was enzymatically released from the substrate *p*-NPP. Then, we added 500 μL trichloroacetic acid (TCA, 10% w/v in water) into the tube to stop the reaction. We added 500 μL Na_2CO_3 solution (10% w/v in water) as the chromogenic agent, and measured the absorbance of solution at 410 nm.

Quantification of Sorting efficiency.

The efficiency of droplet sorting is defined as enrichment factor as previously described by Jean-Christophe Baret et al.[7] Briefly, the ratio of positive droplets to negative droplets is defined as ε as:

$$\varepsilon = \frac{N_+}{N_-}$$

We obtained ε based on counting of more than 2000 droplets from fluorescence images. The enrichment is defined as the ratio of ε′ after sorting to ε before sorting:

$$\eta = \frac{\varepsilon'}{\varepsilon}$$

In the sorting experiments of a binary mixture of positive and negative droplets made with lipase and FDB, we obtained an 82-fold enrichment of positive droplets after sorting (Fig. S2).

References

1. Lane, DJ; Pace, B; Olsen, GJ; et al. Rapid determination of 16S ribosomal RNA sequences for phylogenetic analyses, *Proc. Natl. Acad. Sci. U. S. A.*, 1985, **82**, 6955-6959.
2. Kim, OS; Cho, YJ; Lee, K; et al. Introducing EzTaxon-e: a prokaryotic 16S rRNA gene sequence database with phylotypes that represent uncultured species, *Int. J. Syst. Evol. Microbiol.*, 2012, **62**, 716-721.
3. Thompson, JD; Higgins, DG; Gibson, TJ. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Res.*, 1994, **22**, 4673-4680.
4. Tamura, K; Peterson, D; Peterson, N; et al. MEGA5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods, *Mol. Biol. Evol.*, 2011, **28**, 2731-2739.
5. Saitou, N; Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, 1987, **4**, 406-425.
6. Felsenstein, J. Confidence Limits on Phylogenies: An Approach Using the Bootstrap, *Evolution*, 1985, **39**, 783-791.
7. Baret, JC; Miller, OJ; Taly, V; et al. Fluorescence-activated droplet sorting (FADS): efficient microfluidic cell sorting based on enzymatic activity, *Lab Chip*, 2009, **9**, 1850-1858.

Table S1. Configuration of fluorescence channels of the compact FADS detection system.

Channel	Laser (nm)	Bandpass Filter (nm)	Dichroic Filter (nm)	Fluorochromes
PMT 1	470	505/40	485	FITC, Alexa Fluor 488,CFSE, Fluo-3
PMT 2	470	600/42	565	PE, PI
PMT 3	470	635/20	610	ECD, PE-Texas Red, PE-CF594, PI, PE-Cy5

Notes: FITC, Fluorescein isothiocyanate; CFSE, Carboxyfluorescein succinimidyl ester; PE, R-phycoerythrin; PI, Propidium iodide; ECD, PE-Texas Red®-x available from Beckman-Coulter; PE-Texas Red, A tandem conjugate of Texas Red with R-phycoerythrin; PE-CF594, a tandem conjugate that combines PE and CF594, available from BD Biosciences; PE-Cy5 a tandem conjugate that combines phycoerythrin and a cyanine dye available from BD Biosciences.

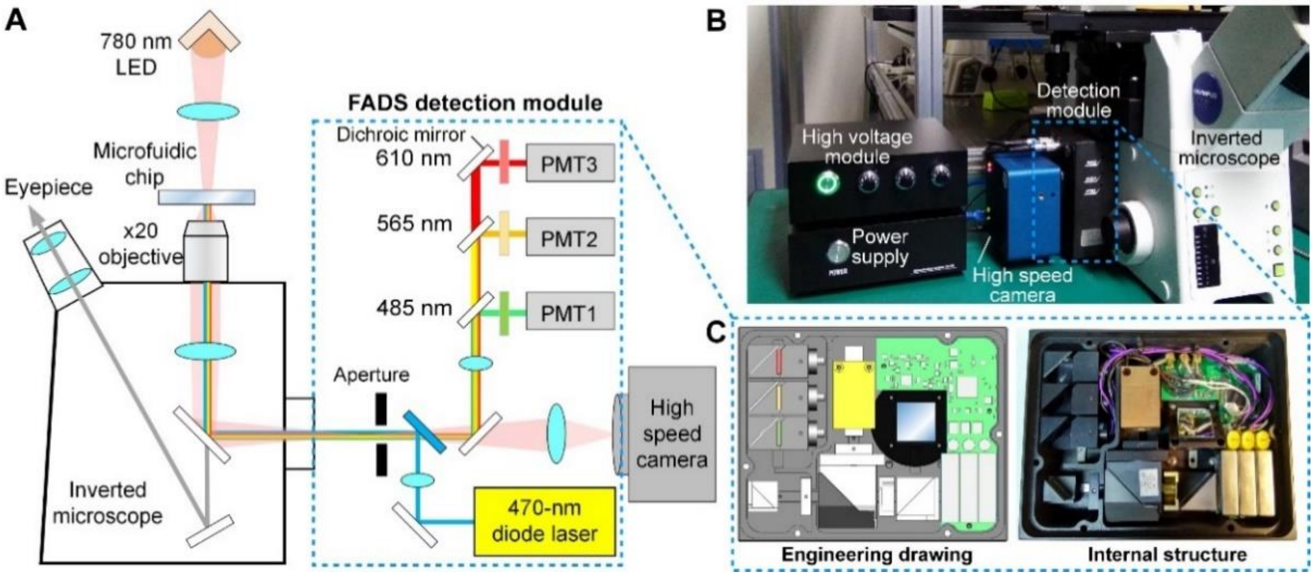


Figure S1. The compact optical system designed for FADS. (A) Diagram of the optical setup. The microscope has a side port, where we place the portable FADS detection module which integrating the laser PMT: photomultiplier; (B) Photograph of the optical experiment setup in panel A; (C) Engineering drawing and photograph of the internal structure of the portable FADS detection module.

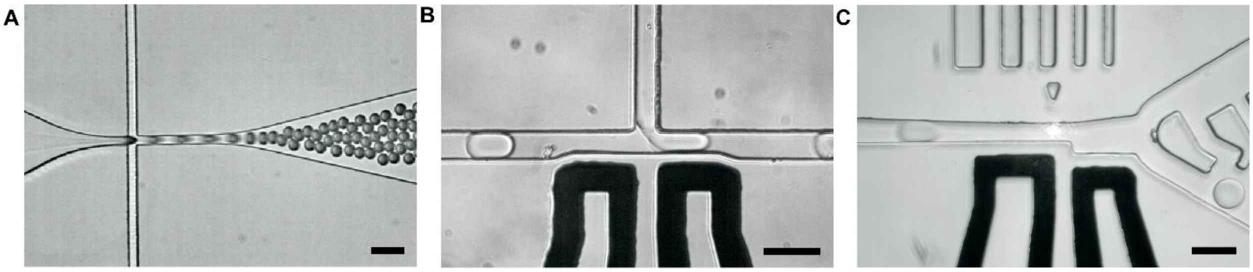


Figure S2. Microscopic images of microfluidic devices for (A) droplet generation, (B) picoinjection of substrate into the droplets, and (C) FADS triggered by dielectrophoresis. Scale bar: 50 µm.

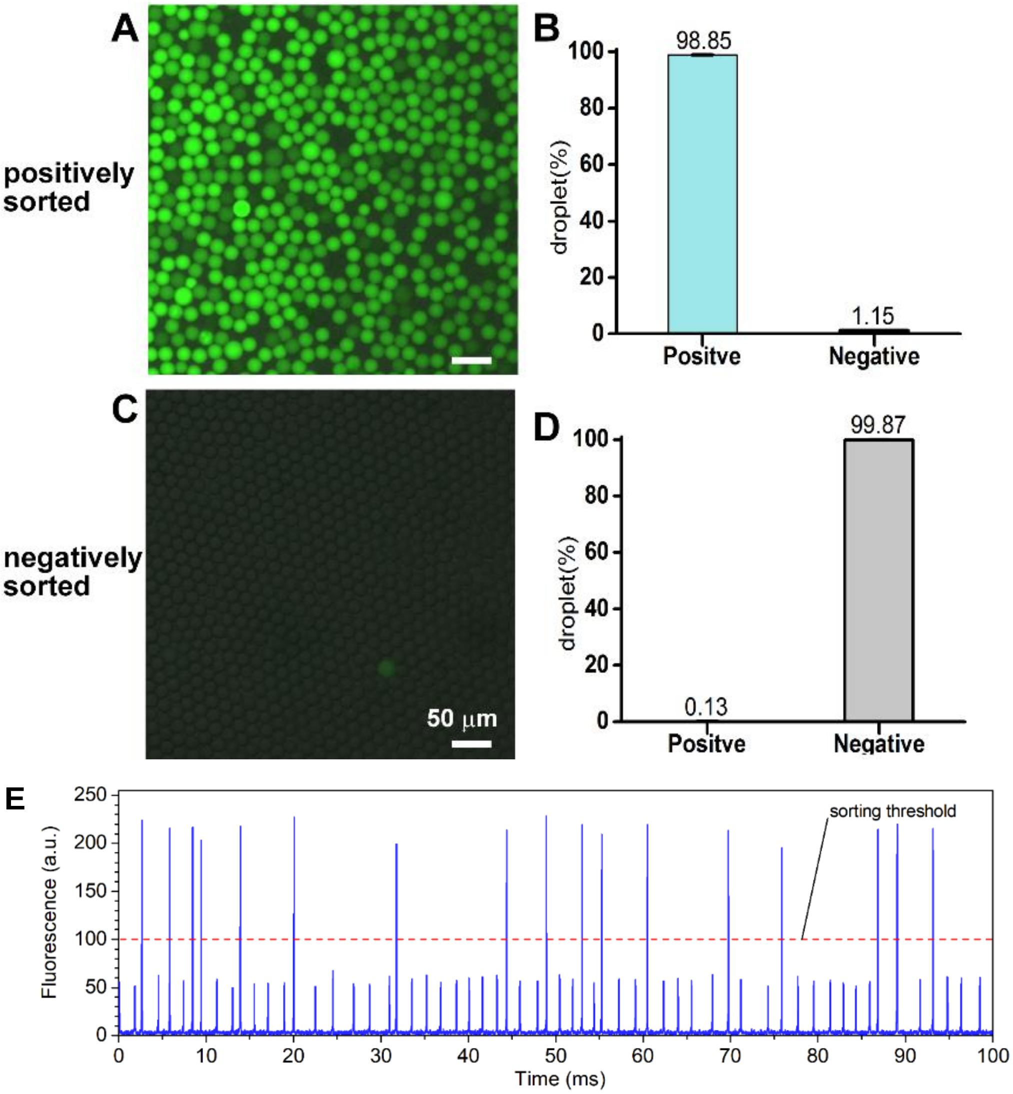


Figure S3. Sorting efficiency of fluorescence-activated droplet sorting (FADS) evaluated by commercial available lipase and its fluorogenic substrate FDB. (A, B) A bright-field and fluorescence merged image of the positive-sorted droplets and the statistical analysis. (C, D) the negative-sorted droplets and the statistical analysis. (E) Time series during a sort showing the detected fluorescence signal (blue). Those peaks about the threshold (red dashed line) was sorted and collected.

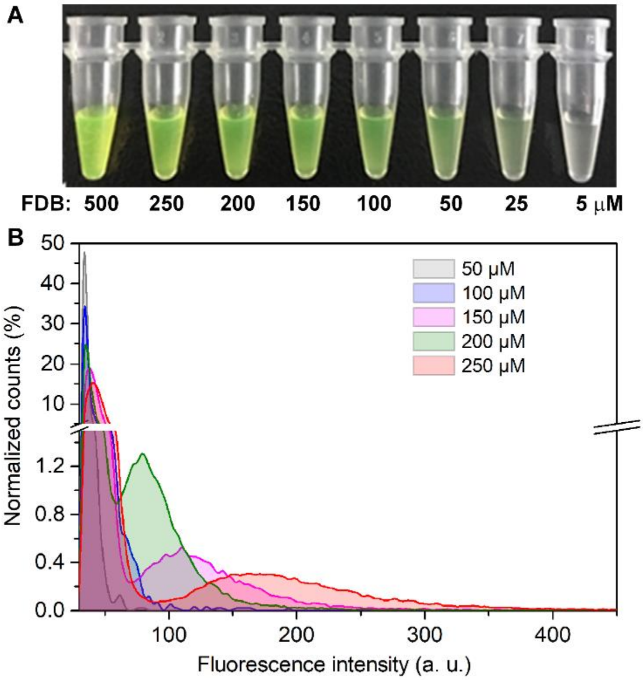


Figure S4. Validation of FDB as the fluorogenic substrate for FADS-based lipase screening. (A) Photograph of the tube assay which mixed *Pseudoalteromonas lipolytica* S29 with FDB of different concentrations. (B) Sorting histogram of droplet fluorescence intensities for S29 cells using FDB of different concentrations. FDB solution was added to droplets by picoinjection. After 1 hour of the picoinjection step, the cells were sorted. The counts were normalized to 100% based on around 200,000 droplets of various FDB concentrations. Intensity threshold of 32 was used to enumerate droplets.

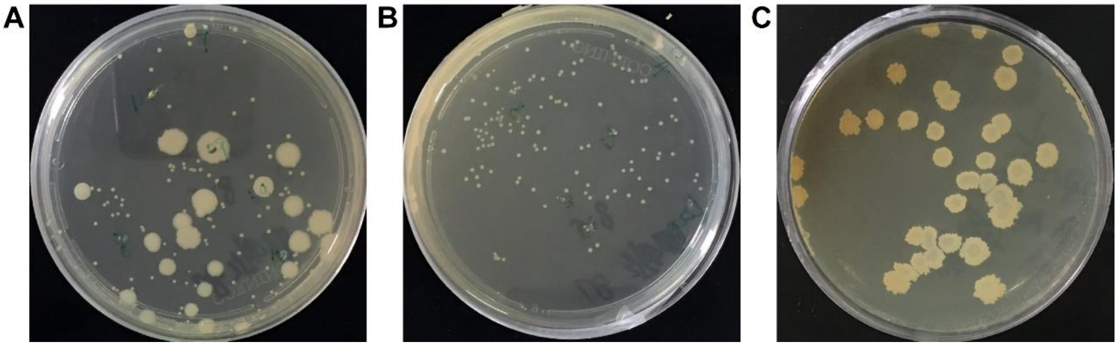


Figure S5. Images of typical agar plates with growth of bacterial cells demulsified from positive-sorted droplets. (A, B) for Daqu starter sample of Gujin liquor from Anhui Province, China, (C) for Lalong hot spring in Tibet, China.

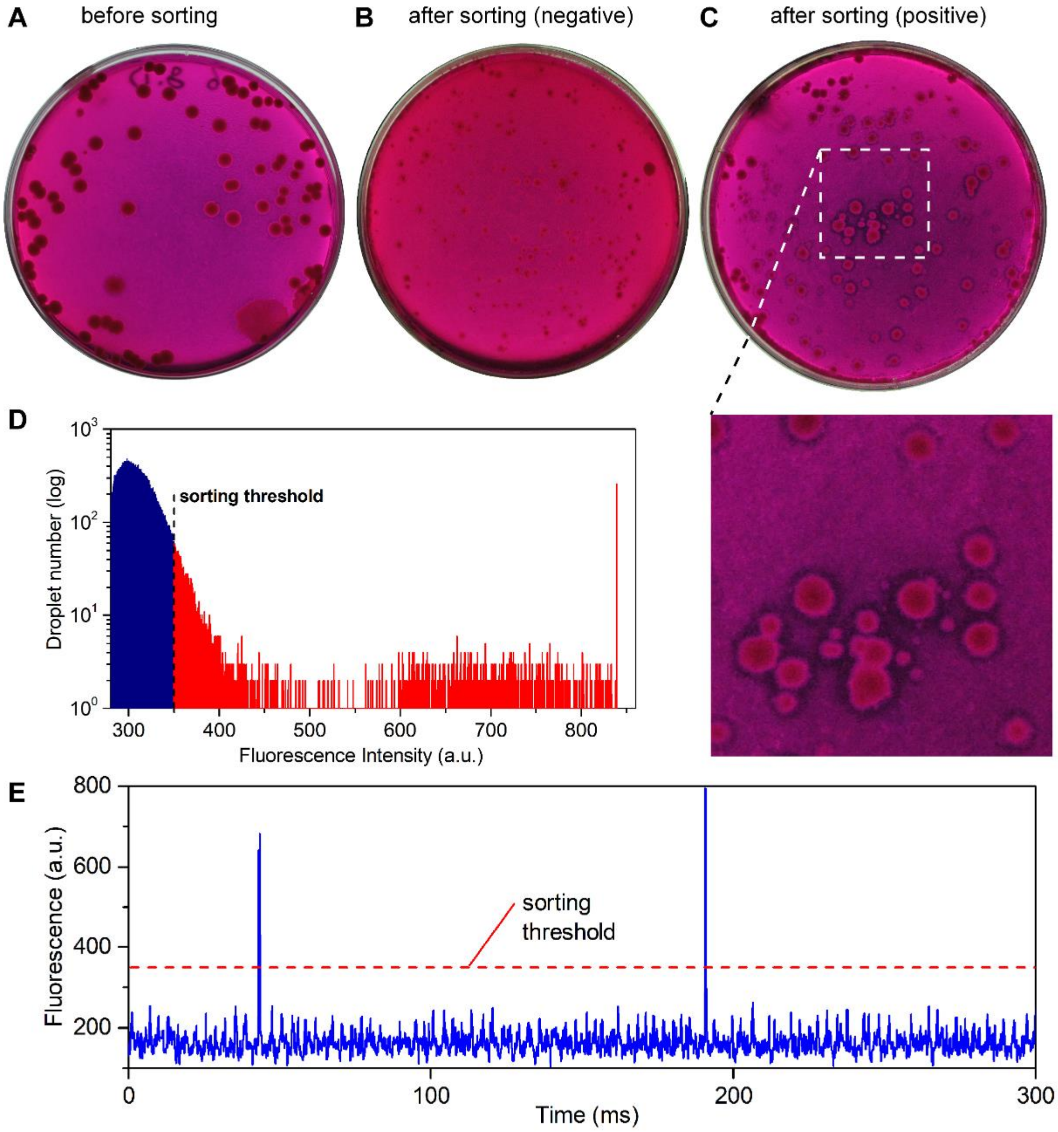


Figure S6. Lipase assays on LB agar plates with 1% (v/v) tributyrin and Rhodamine B (0.5%) for Lalong Hot Spring sample. (A) Before FADS sorting, sample was plated with 10^{-6} X diluted, showing no transparent halos. (B) Negatively sorted colonies showing no transparent halos. (C) Positively sorted colonies showing transparent halos around colonies. The images were taken after 3 d cultivation. Histogram (D) and the time series (E) are showing the distribution of droplet fluorescence intensities for sorting of Lalong Hot Spring sample with 250 μM FDB (final concentration) as the substrate. Droplet detection threshold of 275 and sorting threshold of 350 were used.

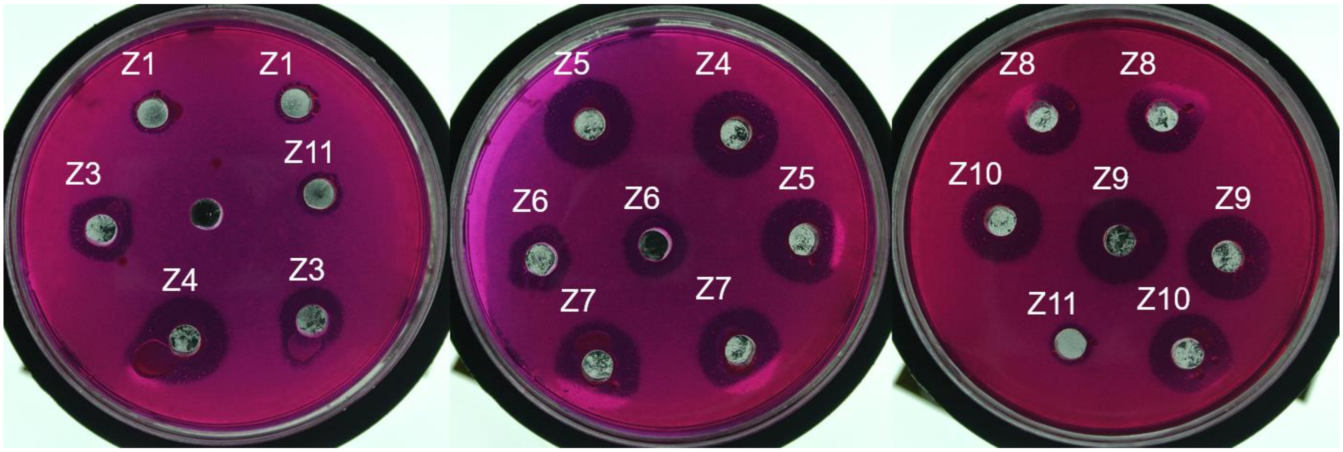
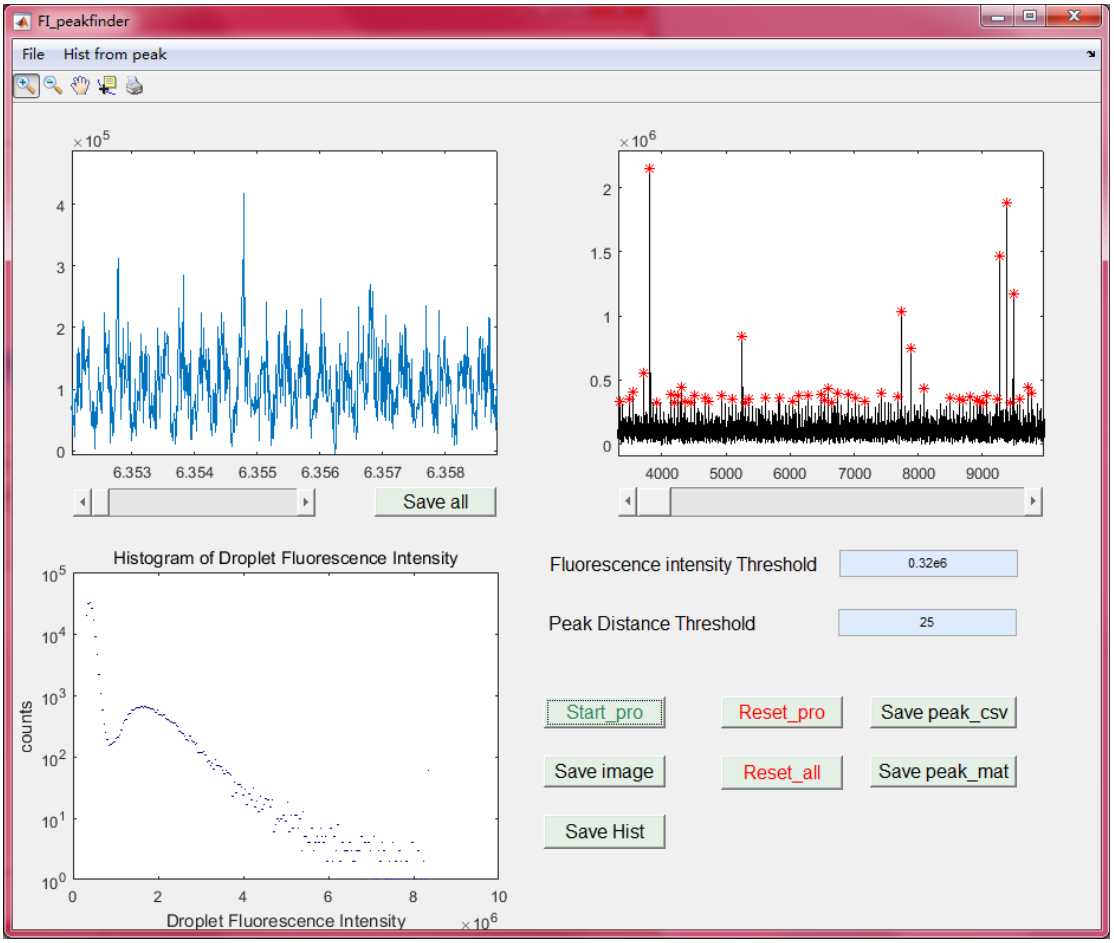


Figure S7. Images show typical results of plate halo assay for strains obtained by FADS (Zoige wetland samples).



Fluorescence intensity Threshold 0.32e6

Peak Distance Threshold 25

Figure S8. The front panel of Matlab program for data processing of FADS fluorescence signal recording.

Peak analysis program written in Matlab.

```
function varargout = FI_peakfinder(varargin)
% FI_PEAKFINDER MATLAB code for FI_peakfinder.fig
%      FI_PEAKFINDER, by itself, creates a new FI_PEAKFINDER or raises the existing
%      singleton*.
%
%      H = FI_PEAKFINDER returns the handle to a new FI_PEAKFINDER or the handle to
%      the existing singleton*.
%
%      FI_PEAKFINDER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FI_PEAKFINDER.M with the given input arguments.
%
%      FI_PEAKFINDER('Property','Value',...) creates a new FI_PEAKFINDER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before FI_peakfinder_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to FI_peakfinder_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help FI_peakfinder

% Last Modified by GUIDE v2.5 15-Nov-2017 17:23:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @FI_peakfinder_OpeningFcn, ...
    'gui_OutputFcn',  @FI_peakfinder_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before FI_peakfinder is made visible.
function FI_peakfinder_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FI_peakfinder (see VARARGIN)

% Choose default command line output for FI_peakfinder
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes FI_peakfinder wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = FI_peakfinder_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
global View_length len_data
val_sli=get(handles.slider1,'Value');
val_sli=val_sli*(len_data-View_length);
set(handles.Ori_sig,'Xlim',[val_sli val_sli+View_length-1]);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end



% --- Executes on button press in Save_all.
function Save_all_Callback(hObject, eventdata, handles)
% hObject    handle to Save_all (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FIDATA View_length len_data
View_length=4000;
[filename,pathname]=uiputfile({'*.jpg';'*.bmp'},'save current image','untitled.jpg');
fig_count=1;
for nn=1:(floor((len_data)/View_length))
   fidstart_d=(nn-1)*(View_length)+1;
   figend_d=nn*(View_length);
   fig_count = fig_count+1;
   figure(fig_count);
   %        set(gcf,'Position',[50,50,900,600]);
   y1_lim=max(FIDATA(fidstart_d:figend_d));

   plot(fidstart_d:figend_d,FIDATA(fidstart_d:figend_d),'k');
   xlim([fidstart_d figend_d]);
   %        saveas(gcf,[pathname,num2str(nn)],'fig');
```

```
    saveas(gcf,[pathname,num2str(nn)],'jpg');
    close gcf
end

for nn=floor((len_data)/(View_length))+1
    if mod(length(FIDATA),4000)==0
        break
    else
        fidstart_d=(floor((len_data-1)/(View_length)))*(View_length)+1;
        figend_d=len_data;
        figure(floor((len_data-1)/(View_length))+1);
        %         set(gcf,'Position',[50,50,900,600]);
        y1_lim=max(FIDATA(fidstart_d:figend_d));

        plot(fidstart_d:figend_d,FIDATA(fidstart_d:figend_d),'k');
        xlim([fidstart_d figend_d]);

        %        saveas(gcf,[pathname,num2str(nn)],'fig');
        saveas(gcf,[pathname,num2str(nn)],'jpg');
        close gcf
    end
end


% --------------------------------------------------------------------
function File_Callback(hObject, eventdata, handles)
% hObject    handle to File (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Load_Callback(hObject, eventdata, handles)
% hObject    handle to Load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
close all;
close(gcf);

% --------------------------------------------------------------------
function load_xls_Callback(hObject, eventdata, handles)
% hObject    handle to load_xls (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[name path]=uigetfile({'*.xls'},'load the fluorescence signal');
if isequal(name,0)||isequal(path,0)
   errordlg('Not an effective path','error');
   return;
else
   global FIDATA View_length len_data
   View_length=4000;
   FIDATA=xlsread([path name]);
   FIDATA(FIDATA==-2^23)=2^23;%% fluorescence intensity saturation control
   len_data=length(FIDATA);
   msgbox('Signal successfully loaded','Message');
   axes(handles.Ori_sig);
   plot(FIDATA);
   guidata(hObject,handles)
end

% --------------------------------------------------------------------
function load_csv_Callback(hObject, eventdata, handles)
% hObject    handle to load_csv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[name path]=uigetfile({'*.csv'},'load the fluorescence signal');
if isequal(name,0)||isequal(path,0)
   errordlg('Not an effective path','error');
   return;
else
   global FIDATA View_length len_data
   View_length=4000;
   FIDATA=csvread([path name]);
    FIDATA(FIDATA==-2^23)=2^23;%% fluorescence intensity saturation control
```

```
    len_data=length(FIDATA);
    msgbox('Signal successfully loaded','Message');
    axes(handles.Ori_sig);
    plot(FIDATA);
    guidata(hObject,handles)
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
global View_length len_data
val_sli=get(handles.slider2,'Value');
val_sli=val_sli*(len_data-View_length);
set(handles.Processing_part,'Xlim',[val_sli val_sli+View_length-1]);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function thre_num_Callback(hObject, eventdata, handles)
% hObject    handle to thre_num (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of thre_num as text
%        str2double(get(hObject,'String')) returns contents of thre_num as a double
```

```
global thre1
thre11=get(hObject,'String');
thre1=str2double((get(hObject,'String')));
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function thre_num_CreateFcn(hObject, eventdata, handles)
% hObject    handle to thre_num (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in Start_pro.
function Start_pro_Callback(hObject, eventdata, handles)
% hObject    handle to Start_pro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FIDATA len_data thre1 peak_loc_all peak_num_all peak_value_all PEAK_VALUE cal_count DistanceThreshold
len_data=length(FIDATA);
if mod(length(FIDATA),1e6)==0%% Process the whole dataset as several sub-datasets
    PEAK_LOC=cell(floor(length(FIDATA)/1e6),1);
    PEAK_VALUE=cell(floor(length(FIDATA)/1e6),1);
else
    PEAK_LOC=cell(floor(length(FIDATA)/1e6)+1,1);
    PEAK_VALUE=cell(floor(length(FIDATA)/1e6)+1,1);
end
for cal_count=1:length(PEAK_VALUE)
    clear FI_NUM point_loc Value_peak FI_DEC diff_pointloc
    if cal_count==length(PEAK_VALUE)
        FI_NUM=FIDATA((cal_count-1)*1e6+1:end);
        FI_NUM(FI_NUM<0)=0;
    else
        FI_NUM=FIDATA((cal_count-1)*1e6+1:cal_count*1e6);
```

```
    FI_NUM(FI_NUM<0)=0;
end

len_segment=length(FI_NUM);
FI_DEC=diff(FI_NUM);%% Get first order difference of intensity

FI_DEC(FI_DEC<0)=0;
FI_DEC(FI_DEC>0)=1;

point_loc=zeros(len_segment-1,1);
for ii=2:len_segment-2
    if (FI_DEC(ii)==1)&&(FI_DEC(ii+1)==0)%%Get peak position^
        point_loc(ii)=ii;
    else
        point_loc(ii)=0;
    end
end
point_loc=point_loc(point_loc~=0);
point_loc=point_loc+1;
Value_peak=FI_NUM(point_loc);

%% peak selection
for pp=1:length(point_loc)
    if FI_NUM(point_loc(pp))<thre1%% threshold screening
        point_loc(pp)=0;
        Value_peak(pp)=0;
    end
end

point_loc=point_loc(point_loc~=0);
Value_peak=Value_peak(Value_peak~=0);
diff_pointloc=diff(point_loc);%% Get first order difference of peak position and peak-to-peak distance
%temp=(mean(diff_pointloc)-2*std(diff_pointloc));
temp = DistanceThreshold;

for MM=1:length(point_loc)-1
    if diff_pointloc(MM)>temp
        continue
    else
        array_temp=[Value_peak(MM),Value_peak(MM+1)]; %If two peaks are too close, chose the larger peak value, and discard the smaller one
```

```
            [num loc]=max(array_temp);
            point_loc(MM+2-loc)=NaN;
            Value_peak(MM+2-loc)=NaN;
        end
    end
    point_loc=point_loc(isnan(point_loc)==0);
    Value_peak=Value_peak(isnan(Value_peak)==0);
    point_loc=point_loc+(cal_count-1)*1e6;
    PEAK_LOC{cal_count,1}=point_loc;
    PEAK_VALUE{cal_count,1}=Value_peak;
end
peak_loc_all=[];
peak_value_all=[];
for MM = 1:length(PEAK_VALUE)
    peak_value_all=[peak_value_all;PEAK_VALUE{MM,1}];
    peak_loc_all=[peak_loc_all;PEAK_LOC{MM,1}];
end
peak_num_all=peak_value_all;
axes(handles.Processing_part)
plot(FIDATA(1:end),'k');
hold on
plot(peak_loc_all(1:end),FIDATA(peak_loc_all(1:end)),'r*');

axes(handles.Hist)
hist(peak_num_all,200)
set(gca,'YScale','log');
title('Histogram of Droplet Fluorescence Intensity');
ylabel('counts');
xlabel('Droplet Fluorescence Intensity');
guidata(hObject,handles)

% --- Executes on button press in Reset_pro.
function Reset_pro_Callback(hObject, eventdata, handles)
% hObject    handle to Reset_pro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    delete(allchild(handles.Processing_part));
    delete(allchild(handles.slider2));
   %% set(handles.thre_num,'string',' ');
    delete(allchild(handles.Start_pro));
```

```
    delete(allchild(handles.Hist));
    delete(allchild(handles.log_scale));
    clear peak_loc_all peak_num_all thre1
end


% --- Executes on button press in Reset_all.
function Reset_all_Callback(hObject, eventdata, handles)
% hObject    handle to Reset_all (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FIDATA View_length len_data thre1 Value_peak point_loc
try
    delete(allchild(handles.Ori_sig));
    delete(allchild(handles.slider1));
    delete(allchild(handles.Save_all));
    delete(allchild(handles.Processing_part));
    delete(allchild(handles.slider2));
    set(handles.thre_num,'string',' ');
    delete(allchild(handles.Start_pro));
    delete(allchild(handles.Hist));
    delete(allchild(handles.log_scale));
    clear FIDATA View_length len_data thre1 peak_loc_all peak_num_all
    clear all
end


% --- Executes on button press in save_hist.
function save_hist_Callback(hObject, eventdata, handles)
% hObject    handle to save_hist (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global peak_num_all
[filename,pathname]=uiputfile({'*.jpg';'*.bmp'},'save current image','untitled.jpg');
if isequal([filename pathname],[0,0])
    errordlg('image not saved','error')
    return
else
    figure
    hist(peak_num_all,200)
    set(gca,'YScale','log');
```

```
    title('Histogram of Droplet Fluorescence Intensity');
    ylabel('counts');
    xlabel('Droplet Fluorescence Intensity');
    saveas(gcf,[pathname,filename],'jpg');
    close gcf
    guidata(hObject,handles)
end


% --- Executes on button press in save_image.
function save_image_Callback(hObject, eventdata, handles)
% hObject    handle to save_image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FIDATA View_length len_data peak_loc_all peak_num_all
[filename,pathname]=uiputfile({'*.jpg';'*.bmp'},'save current image','untitled.jpg');
fig_count=1;
for nn=1:(floor((len_data)/View_length))
    fidstart_d=(nn-1)*(View_length)+1;
    figend_d=nn*(View_length);
    fig_count = fig_count+1;
    figure(fig_count);
    set(gcf,'Position',[50,50,900,600]);
    y1_lim=max(FIDATA(fidstart_d:figend_d));
    y2_lim=max(FIDATA(fidstart_d:figend_d));

    plot(fidstart_d:figend_d,FIDATA(fidstart_d:figend_d),'k');
    xlim([fidstart_d figend_d]);
    hold on
    plot(peak_loc_all(1:end),FIDATA(peak_loc_all(1:end)),'r*');
    text(figend_d-10,y2_lim,'signal');
    saveas(gcf,[pathname,num2str(nn)],'jpg');
    close gcf
end

for nn=floor((length(FIDATA))/(4000))+1
    if mod(length(FIDATA),4000)==0
        break
    else
        fidstart_d=(floor((length(FIDATA)-1)/(4000)))*(4000)+1;
        figend_d=length(FIDATA);
```

```
        figure(floor((len_data-1)/(4000))+1);
        set(gcf,'Position',[50,50,900,600]);
        y1_lim=max(FIDATA(fidstart_d:figend_d));
        y2_lim=max(FIDATA(fidstart_d:figend_d));

        plot(fidstart_d:figend_d,FIDATA(fidstart_d:figend_d),'k');
        xlim([fidstart_d figend_d]);
        hold on
        plot(peak_loc_all(1:end),FIDATA(peak_loc_all(1:end)),'r*');
        text(figend_d-10,y2_lim,'signal');

        saveas(gcf,[pathname,num2str(nn)],'jpg');
        close all
    end
end

% --- Executes on button press in save_peak_csv.
function save_peak_csv_Callback(hObject, eventdata, handles)
% hObject    handle to save_peak_csv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global peak_loc_all peak_value_all
[filename,pathname]=uiputfile('*.csv','save peak information','untitled.csv');
% save([pathname,filename],'peak_value_all','peak_loc_all');
csvwrite([pathname,filename],peak_value_all);
guidata(hObject,handles)


% --------------------------------------------------------------------
function hist_peak_Callback(hObject, eventdata, handles)
% hObject    handle to hist_peak (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[name path]=uigetfile({'*.mat'},'load the peak information');
if isequal(path,0)
    errordlg('Not an effective path','error');
    return;
else
    global peak_find peak_num_all
    peak_find=[];
    MAT_File=dir([path '*.mat']);
```

```
    len_file=length(MAT_File);
    for file_count=1:len_file
        load([path MAT_File(file_count).name]);
        peak_find=[peak_find;peak_value_all];
    end
    peak_num_all=peak_find;
    msgbox('Peak information successfully loaded','Message');
    axes(handles.Hist);
    hist(peak_num_all,200)
    set(gca,'YScale','log');
    title('Histogram of Droplet Fluorescence Intensity');
    ylabel('counts');
    xlabel('Droplet Fluorescence Intensity');
    guidata(hObject,handles)
end


% --- Executes on button press in save_peak_mat.
function save_peak_mat_Callback(hObject, eventdata, handles)
% hObject    handle to save_peak_mat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global peak_loc_all peak_value_all
[filename,pathname]=uiputfile('*.mat','save peak information','untitled.mat');
save([pathname,filename],'peak_value_all','peak_loc_all');
guidata(hObject,handles)



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
global DistanceThreshold
DistanceThreshold=get(hObject,'String');
DistanceThreshold=str2double((get(hObject,'String')));
guidata(hObject,handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```