Data S1: MATLAB Code for CaSCaDe Analysis

```
function res=Cal_anl_main2sa_forreview(im0)
% input :
%     im0 : is n frame video data stored in cell array (n by 1  cell)
% output :
%     res: resutls output
    clc
    im0=im0(:);

%% parameter setting;

    % basic analysis criteria setting
        p.foffset=60; % how many initial frames to exclude in analysis
        p.norm_signal='std'; % ('std','bkg','sub') % different way to
normalize intenisty
        p.spf=1 ; %  frame rate at acquisition

    % event detection
         p.min_int_ed=0.5;     % minimum intenisty value for start-end of a
event;
         p.peak_int_ed=5.0  ;  % minimum peak intesnity value for being
considered as signal
         p.min_peak_dist_ed= 4 ;
         p.min_peak_length=4;
    % background trending correction
         p.int_correct= 0; % if 1, correct bkg, if 0, no correction.


% main code
 %% read information regarding images and conditions.
  % system parameter
   knn=1;
   fii=1;

        [hh,ww]=size(im0{1});
        iminfo.Height=hh;
        iminfo.Width=ww;

 %% analysis individual condition section

   % generate data name based on condition id, image id and drug id.
   %% acquire image data for this condition
     tic
     % set frame of interests for each conditions
        fini=1+p.foffset;
        fend=length(im0);
        frameset0=(fini:fend);

    %% spatial temporal convolution
        % set data output folder
         % processing images.
           clear M
```

```
            im0temp=im0(frameset0);

      %% identify domain candidates

            im3f=bpass3d_v1(im0temp);
            [bff]=sum(im3f,3);
            bff=bff/length(frameset0);
            bw=domain_segment(bff)  ;
            L=bwlabel(bw);

      % obtain domains information
          stats=regionprops(bw,'area'); % get area of each mode
          A=[stats.Area];
          obnum=max(L(:)); % all detected node from spatial descrepency

     %%%%%%%%%
     %% recording the intenisty of each modes and normalization
     % initiate output variables
     if obnum>0
         intout=get_domain_int(im0temp,L);
         % get normalized intensity by doamin size.(inout0)
         intout0=intout./(ones(length(intout(:,1)),1)*A(:)');
          if p.int_correct % correction for intensity shift, possible
photobleaching effect
              [bkg_int]=get_bkg_int(intout0);
              intout0=intout0-bkg_int;
           end

         % need to normalize the intout / normalized the intensity.
         bg00=zeros(size(intout0(1,:)))'; rbg00=bg00;
         parfor k99=1:length(intout0(1,:))
             [bg,rbg]=estibkg(intout0(:,k99),7);    % estimate the background;
             bg00(k99)=bg;
             rbg00(k99)=rbg;
         end
         medmat=ones(size(intout0(:,1)))*bg00';
         stdmat=ones(size(intout0(:,1)))*rbg00';

     % intenity profile with different normalization
         intoutb1=(intout0-medmat)./stdmat; %normalized signal
         intoutb2=(intout0-medmat)./medmat; %normalized signal
         intoutb3=(intout0-medmat); %normalized signal

         switch(p.norm_signal)
             case('std')
                 intoutf=intoutb1;
             case('bkg')
                 intoutf=intoutb2;
             case('sub')
                 intoutf=intoutb3;
         end

       %% peak detecting at individaul domains and processing.

         [pkout0, intoutbw]=peak_detect_v2(intoutf,p);
```

```
        % get intenity value of event peak at differnet normalization method

        pk_int1=get_peak_intprofile(pkout0,intoutb1);
        pk_int2=get_peak_intprofile(pkout0,intoutb2);
        pk_int3=get_peak_intprofile(pkout0,intoutb3);

        pkout=[pkout0,[pk_int1 pk_int2 pk_int3]];
        % pkout format: domain id/ peak location/peak length/peak height/
        % peak initial frame/ end frame

    end


     %% get features of peaks
        pknum=0;
       if ~isempty(pkout)
            pk_DA=A(pkout(:,1)); % corresponding domain size of individual
events
            pk_bg=bg00(pkout(:,1)); % background intensity
            pk_rbg=rbg00(pkout(:,1));  % standard dev.
            pknum=length(pkout(:,1));

            peakfeatures=zeros(pknum,75);

            peak_int_t=cell(pknum,1); % variable to store all detected peak
profiles
            % go through individual events,
            for kpk=1:pknum;
                % get peak features for peaks

                intp= intoutf(:,pkout(kpk,1)) ; % intensity profile of this
event.
                intp2=intoutbw(:,pkout(kpk,1));

                i1b = bpass1d(intp,1,11);
                i1b2= bpass1d(intp,1,21);

                L1=bwlabel(intp2);
                bwint= L1==L1(pkout(kpk,2));

                itest=intp(bwint);
                itestb=i1b(bwint);
                itestb2=i1b2(bwint);

                fs1=get_peak_feature(itest);
                fs2=get_peak_feature(itestb);
                fs3=get_peak_feature(itestb2);

                peakfeatures(kpk,:)=[fs1',fs2',fs3'];
                peak_int_t{kpk}=itest;
            end
                pkinfo=[knn*ones(pknum,1),fii*ones(pknum,1),1*ones(pknum,1),
pkout, pk_DA(:),pk_bg(:), pk_rbg(:)];
        else
```

```
       end

 %% SVM-classification for peaks;
  if ~isempty(pkout)
% % %
        %  load previous established svm model
        % model 1
        load(Bulit_svm_model)

        Lower=modelparm.Lower ;
        Upper=modelparm.Upper ;
        MaxV=modelparm.MaxV ; % defined by training dataset.
        MinV=modelparm.MinV ;

        pkf=peakfeatures;
        maxVm=ones(size(pkf(:,1)))*MaxV(:)';
        minVm=ones(size(pkf(:,1)))*MinV(:)';

        % if there are NaN,inf in pkf, make it to min V or max V
        % maybe this is not the best way to deal with it. % thie part are
        % slightly differnet then previously defined.

        cc=isnan(pkf) ;
        pkf(cc)=minVm(cc);
        cc=isinf(pkf) ;
        pkf(cc)=maxVm(cc) ;
        [pkf] = ScaleW(pkf,Lower,Upper,MaxV,MinV);

        cc=isnan(pkf) ;
        pkf(cc)=0;

        testdata=pkf(:,:);
    SVMmodel=modelparm.SVMmodel;
    [Group,~,~] = svmpredict2(ones(size(testdata(:,1))), testdata, SVMmodel,
'-b 1');

    svm_pk_class=Group(:); % classificaitno by svm , good /no good
  else
      svm_pk_class=[]; % classificaitno by svm , good /no good
  end

% save organized output resutls
p.iminfo=iminfo;
res.param=p; % used parameters setting
res.frameset0=frameset0; % frames that is used in analysis
res.bff=bff; % 2D projection image after 3D convolution.
res.bw=bw; % domain segmetnation image
res.L=L; % labeled domain segmentation image
res.A=A; % Area of each domain from domain segmetnation image
res.obnum=obnum; % number of domain based on domain segmentaion image
res.medmat=medmat; % median intenisty for each domain
res.stdmat=stdmat; % standard deviation in intensity profile for each domain

res.intout=intout; % differnet type of intenisty profiles of domains
```

```
res.intoutf=intoutf;
res.intoutb1=intoutb1;
res.intoutb2=intoutb2;
res.intoutb3=intoutb3;
res.intout0=intout0;
res.intoutbw=intoutbw; % binarized domain intensity profiles : 1 means
detecting as signal. 0 means no signal.

res.pkinfo=pkinfo; % detected peak informaiton
res.peakfeatures=peakfeatures; % features of each detected peaks
res.peak_int_t=peak_int_t;

res.svm1_pk_class=svm_pk_class; % peak goodness based on svm classication

toc
end
function res = bpass1d(arr,lb,hb)
%
  b = double(lb);
  r = round(hb);
  w = 2*r + 1;

  r = ((0:w-1) - r)/(2 * b);
  gx = exp( -r.^2) / (2 * b * sqrt(pi));

  mx = ones(1,w)/w;

  res = arr(:)';
  g = conv(res,gx,'valid');

  tmpres = res;
  res = conv2(tmpres,mx,'valid');

  res0=zeros(size(arr));
  g0 = zeros(size(arr));

  res0((lobject+1):end-lobject) = res;
  g0((lobject+1):end-lobject) = g;

  res=max(g0-res0,0);
end
function [scaled, Lower, Upper, MaxV, MinV ] = ScaleW(Data, Lower, Upper,
MaxV, MinV)

    if (nargin<3)
     Lower = -1;
     Upper = 1;
    elseif (Lower > Upper)
     disp ('Wrong Lower or Upper values!');
    end
    if nargin<=3 % calulate MaxV and MinV
        [MaxV, ~]=max(Data);
        [MinV, ~]=min(Data);
    end
```

```
    [R,C]= size(Data);
        scaled=(Data-ones(R,1)*MinV).*(ones(R,1)*((Upper-
Lower)*ones(1,C)./(MaxV-MinV)))+Lower;
        scaled=min(Upper, scaled);
        scaled=max(Lower, scaled);
end
function pk_int=get_peak_intprofile(pkout0,intoutf)
    % obtain peak and sum_intenisty of a given detected signal
    pk_int=[];
    if ~isempty(pkout0)
        pk_int=zeros(size(pkout0(:,1:2)));
        for kk=1:length(pkout0(:,1))
            xid=pkout0(kk,1);
            yid1=pkout0(kk,5):pkout0(kk,6);
            ytemp=intoutf(yid1,xid);
            sii=sum(ytemp); % sum_intenisty
            pkii=max(ytemp);  % peak_intensity
            pk_int(kk,:)=[pkii,sii];
        end
    end
end
function im3f=bpass3d_v1(im,param)
% 3D band pass convolution
% im : stacked images
%        format1 : N*1 cells and each cell corrposdence to a image with size
%        of h*w
%        format2 : h*w*N matrices
% % processing is done using single precision- for reducing memory loading
% bim : processed image
% developed by : Pei-Hsun Wu, Ph.D
%    10/28/2014 @ Johns Hopkins University

% set bandpass process parameteres
if nargin==1
    lb=1; % low bound size for in-x,y dim
    hb=11; % high bound size for in-x,y dim
    zlb=1;% low bound size for in-z(t) dim
    zhb=21; % high bound size for in-z(t) dim
else
   lb=param.lb;
   hb=param.hb;
   zlb=param.zlb;
   zhb=param.zhb;
end

% obtain the 3D image size
if iscell(im)
    znum=length(im);
    [h,w]=size(im{1});
else
    [h,w,znum]=size(im);
end

im3=single(zeros(h,w,znum));
```

```
% convolution in 2D (x,y direction first through whole stack
if iscell(im)
   parfor kbs=1:znum % time period without drug
        a=single(im{kbs});
        b=bpassW(a,lb,hb);
        im3(:,:,kbs)=single(b);
   end
else
    parfor kbs=1:znum % time period without drug
        a=single(im(:,:,kbs));
        b=bpassW(a,lb,hb);
        im3(:,:,kbs)=single(b);
   end
end

    im3f=im3;
% bandpass filtering over z (t) dim
    parfor kx=1:w
        for ky=1:h
            temp=squeeze(im3(ky,kx,:));
            im3f(ky,kx,:)=single(bpass1d(temp,zlb,zhb));
        end
    end
end
function bw=domain_segment(bff,Areacut)
% segment the calcium domains area
if nargin==1
    Areacut=25;
end
    [hh,ww]=size(bff);

    if hh==ww
        mskc=mskcircle(length(bff));
    else
        rr=5;
        mskc=zeros(size(bff));
        mskc(rr+1:end-rr,rr+1:end-rr)=1;
    end
    test=bff.*mskc;
    [bge,rbge]=estibkgkmean(test(test>0));
    bw=bff>bge+2*rbge;
%     bw=bff>80; %(100) set threshold to binarize the locations demonstrate
spatial temporal distintive

     % xyc is the just to highlighted the center region of images. morelikely
where the cell is located

    b000=bpassW(bff,3,21); % default 3,21;
        nbw=imregionalmax(b000,8);
        nbw=imdilate(nbw,strel('disk',2));
        nbw=nbw & bw ;

        % watershed segmentation
          cbw=bw | nbw;
          D=bwdist(nbw);
          DL=watershed(D);
```

```
          cbw(DL==0)=0;
          cbw00=cbw<0; % preallocateing cbw00

        ln=bwlabel(nbw);
        lc=bwlabel(cbw);

        DLrev=DL < 0 ;
        for k=1:max(lc(:))
            idx=ln(lc==k);
            [uidx]=unique(idx);
            if length(uidx)==1;
                bwtemp=lc==k;
                cbw00=cbw00 | bwtemp; % collect these non-nucleated reguion
                bwtemp=imdilate(bwtemp,strel('disk',2));
                DL00= DL==0 & imdilate(bwtemp,strel('disk',2));
                DLrev=DL00 | DLrev;
            end
        end

        bw=cbw & ~cbw00; % does not take the one without nuclesate
        bw=bwareaopen(bw,Areacut); % get rid of nodes with small area size
        bw=imfill(bw,'hole');
end
function res = bpassW(arr,lnoise,lobject)
%
% bandpass filter.
%
% Written by  Pei-Hsun Wu,
% Post-doc associate, @ JHU, IMBT.
%
  b = double(lnoise);
  w = round(lobject);
N=2*w+1;
hg=fspecial('gaussian',N, b*sqrt(2));
ha = fspecial('average',N);
arra = imfilter(arr,hg-ha,'symmetric','conv');

rest = max(arra,0);

res=rest;
end
function intout=get_domain_int(im0temp,pattern)
    L=pattern;
    obnum=max(L(:));
    intout=zeros(length(im0temp),obnum);  % matrix with size of (frame # *
domain #)
    % normalized the intensity shift for correct the bleaching .
       parfor kbs=1:length(im0temp); % get intenisty of each domains at
different time frames
            a=(im0temp{kbs});
            for kll=1:obnum
                intout(kbs,kll)=sum(a(L==kll));
            end
       end
      intout=double(intout);
end
```

```
function [bkg_int]=get_bkg_int(intout0)
   span=100;
   porder=1;
   bkg_int=zeros(size(intout0));
   parfor kbs=1:length(intout0(1,:)); % get mean intensity of each frame,
            bkg_int(:,kbs)=smooth(intout0(:,kbs),span,'sgolay',porder);
   end
end
function [pkout,intoutbw]=peak_detect_v2(intout0,p)
% pkout : N*4 matrix, (N: number of peaks;
%          4 columns are 1. domain id, 2.time id, event width, event peak
%          int.
    intoutbw=zeros(size(intout0));
    obnum=size(intout0,2);
    pkout=[];
    for kii=1:obnum
        ytemp=intout0(:,kii);
        bw=ytemp> p.min_int_ed;
        L0=bwlabel(bw);
        bwnew=bw<0; %(zero logical matrix)

[~,locs]=findpeaks(ytemp,'minpeakheight',p.peak_int_ed,'MINPEAKDISTANCE',p.mi
n_peak_dist_ed);
        for kll=1:length(locs)
            bwnew= bwnew | L0==L0(locs(kll));
        end
        intoutbw(:,kii)=(bwnew(:)); % eliminate signal with less than 5 span
away;
        bw=bwnew;

        L0=bwlabel(bw);
        % peak segmentation
        %  segment when there are more than one peaks in an detected evnt
(bw).
            test=L0(locs);
            kold=1;
            for ktt=1:length(test)-1
                kcurrent=ktt+1;
                if test(kold)==test(kcurrent); % if repeated. then segmented
                    id=locs(kold):locs(kcurrent);
                    col=find(ytemp(id)==min(ytemp(id)));
                    col=col(1);
                    h1n=ytemp(id(1));
                    h2n=ytemp(id(end));
                    h1=ytemp(id(1))-min(ytemp(id));
                    h2=ytemp(id(end))-min(ytemp(id));
                    hr1=h1/ytemp(id(1));
                    hr2=h2/ytemp(id(end));
                    d1=col-1;
                    d2=length(id)-col;
                    thp=3;
                    tha=3;
                    ccid=col+locs(kold)-1;

                    if h1 > thp && h2 > tha && d1>1 && d2>1  && (hr1 >0.5 &&
hr2 > 0.5); % condiitons for segments.
```

```
                    if h1n>p.peak_int_ed*2 && h2n > p.peak_int_ed*2
                        bw(ccid)=0; % apply segment
                        kold=kcurrent; % update the kcurrent value
                    end
                elseif h2>h1
                    kold=kcurrent;
                end
            else
               kold=kcurrent; % update the kcurrent value
            end
        end

    % reanalyze the peak info based on new segmentation result
      L0 = bwlabel(bw); % new label on updated bw;

    % remove the event start from first frame or last til last
        % frame
        temp2=L0(1);
        if temp2>0
            L0(L0==temp2)=0;
        end
        temp2=L0(end);
        if temp2>0
            L0(L0==temp2)=0;
        end

        bw=L0>0; % update binary activation signal
        L0= bwlabel(bw); % update label;

        bwnew = zeros(size(bw));
        % only collect the ones with signal
        for kll=1:max(L0(:))
            % recalculate peak info.
            bwtemp1=L0==kll;
            loctemp=find(bwtemp1==1); % get peaklocation;
            [pkh,ploc]=max(ytemp.*bwtemp1); % peak height and peak
locations
            ploc=ploc(1); % if there are more than one peaks with same
maximum value, take first one.

            if sum(bwtemp1)> p.min_peak_length
                % only harvest the peak with length more the threshold
                % value.
                pkl=sum(bwtemp1); % peak length
                bwnew= bwnew | bwtemp1; % update binary info
                f_start=loctemp(1);
                f_end=loctemp(end);

                pkres=[kii,ploc,pkl,pkh, f_start f_end];
                pkout=[pkout;pkres];
            end
        end

        intoutbw(:,kii)=(bwnew(:)); % eliminate signal with less than 5
span away;
```

```
    end
end
function [bkg,rb]=estibkg(I,iter,gmode)
    % estimate the image background using a iterative process
    %
    %*******************************************************
    % written by :
    %                Pei-Hsun Wu, PhD
    %                Institue for Nano-bio technology
    %                Johns Hopkins University
    %
    % Last update: 08/19/2013
    %*******************************************************%
    %
    if nargin==2;
        gmode='iterNspace';
    end

    drr=2;
    binnum=40;
    switch(gmode)
        case('iter')

            I=double(I(:));

            itop= mean(I)+ drr* std(I) ;  % 16 bit
            for mm=1:iter
                    [c1,c2]=hist(I(I < itop),40);
                    mimg= c2(c1 == max(c1)) ;     mimg=mimg(1);
                    ira=std(I(I < itop));
                    itop= mimg + drr*ira ;
            end
            bkg=mimg  ;
            rb= ira ;

        case('gaufit')

            I=double(I(:));
            mimj = mode(I);     % initial guess
            rb=std(I);
            [c1,c2]=hist((I((I < mimj +3*rb))),30);
            [fr]=fit(c2(:),c1(:),'gauss1');
            bkg=fr.b1   ;
            rb=fr.c1/sqrt(2) ;

        case('dir')

            I=double(I(:));
            bkg= mean(I);
            rb=std(I);
        case('iterNspace')
            I1=I;

            I=double(I(:));
```

```
                itop= mean(I)+ drr* std(I) ;  % 16 bit
            for mm=1:iter
                [c1,c2]=hist(I(I < itop),binnum);
                mimg= c2(c1 == max(c1)) ;     mimg=mimg(1);
                ira=std(I(I < itop)) ;
                itop= mimg + drr*ira ;
            end
            bkg=mimg  ;
            cc=I1 < bkg;
            cc=imclose(cc,strel('disk',3));
            bkg= mean(I1(cc));
            rb=std(I1(cc));
    end
end
function [bkg,rb,rg]=estibkgkmean(I,Nmode)
% using kmeans classifiction to identify the backgorund
%
    if nargin==1
        Nmode=2;
    end

    [idx,C]=kmeans(I,Nmode);
    [~,sid]=sort(C,'ascend');
    idx0=zeros(size(idx));
    for k=1:Nmode
        idx0(idx==sid(k))=k;
    end

    Ibg=I(idx0==1);
    rg=[min(I(idx0==1)),max(I(idx0==1))];
    bkg=mean(Ibg);
    rb=std(Ibg);
end
```