Test/ Debugger: imgCropMULTI

Summary: Works through step by step the preprocessing and cropping stages. Useful for code debugging and investigating preprocessing quality of a particular frame.

User notes:

1) Make sure relevant function files, videos and individual video frames are located in the same folder as this live script. So far, only works on horizontal flow videos.

2) There are some user-defined inputs, particularly video/ image file name. Other processing parameters can also be adjusted as see fit.

V2.0. SWC, 19 Feb 2021.

```
close all
clear
```

1) Load video

```
vid_file_name = 'TRI001_SO5cSt 12.5uL_21.5 uL TRITON_ 5 mM 032.avi'; %<--------
user-defined input!!
% vid_file_name = 'DYE003_52%Gl_W_0.003_Ink_SO_SPAN80_0.003_2kfps_.avi';
vid = VideoReader(vid_file_name); %read video
totframes = vid.NumFrames %check total num of frames in video
```

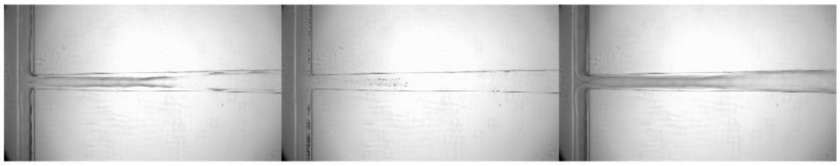
```
totframes = 748
```

2a) Background generation - 'Basic' Statistical Approach

Shows 3 methods: i) Median, ii) Mode and iii) Mean

```
n = 200;    % number of frames to use for background generation  <--------
user-defined input!!

[med_bg mod_bg avg_bg] = bgGenBasic(vid_file_name,n); %custom function
```

```
Processed video: TRI001_SO5cSt 12.5uL_21.5 uL TRITON_ 5 mM 032.avi
Median; Elapsed time = 25.23
Mode; Elapsed time = 65.57
Mean; Elapsed time = 43.41
```



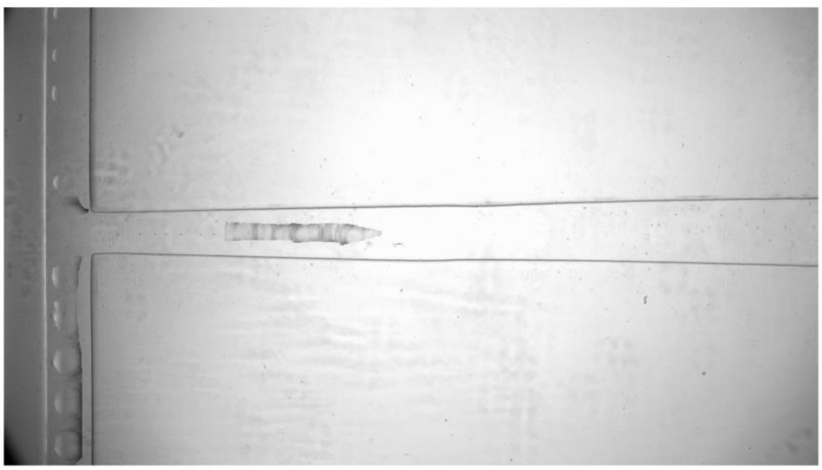
2b) Background generation - 'Complex' Statistical Approach

(Adapted from ADM method)

```
n = 40;    % number of frames to use for background generation <-------- user-defined input!!

bg = bgGenCmplx(vid_file_name,n,'modified'); %custom function
```

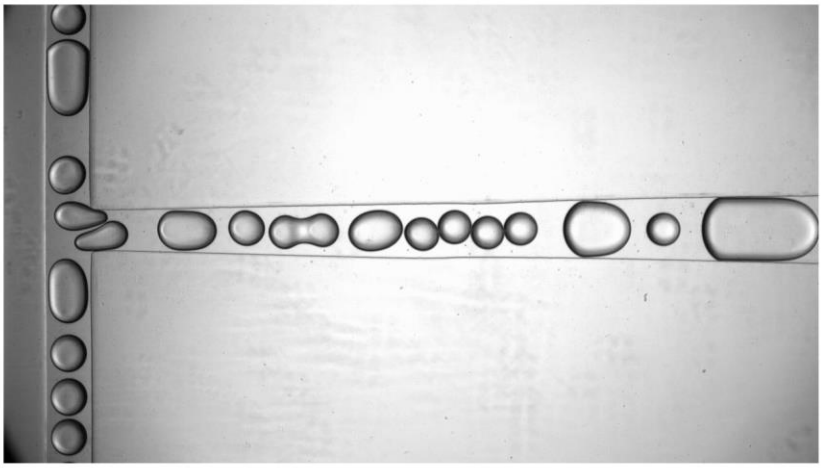
```
Processed video: TRI001_SO5cSt 12.5uL_21.5 uL TRITON_ 5 mM 032.avi
Using standard algorithm...
Elapsed time is 49.500167 seconds.
```

```
% 'original': for WAT, TRI, SDS, DYE.
% 'modified': for C12Tab.

figure; imshow(bg);
```



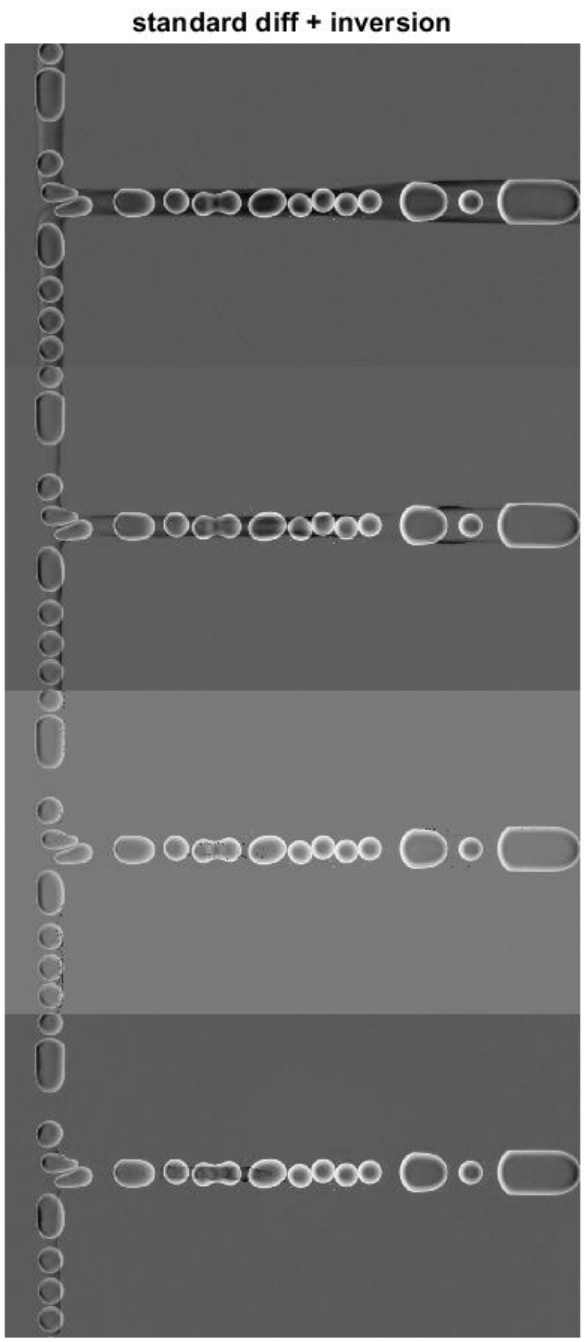
3) Load test image

```
% Load images
I = imread('TRI001_582.jpg');   %<-------- user-defined input!!

% check images loaded correctly
figure; imshow(I);
```



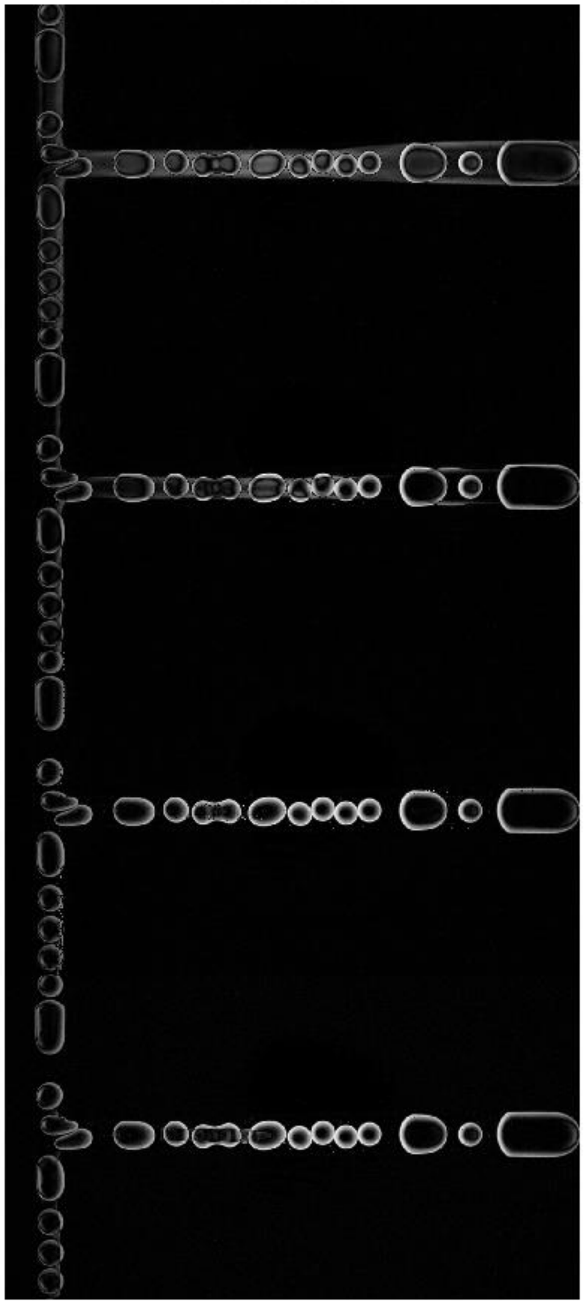
4) Background Subtraction

```
%% From Mean background generation
subMean = rescale(1-(double(I) - double(avg_bg))); % std diff with inversion
subMean2 = rescale(abs(double(I) - double(avg_bg))); % abs subtraction

%% From Median background generation
subMed = rescale(1-(double(I) - double(med_bg))); % std diff with inversion
subMed2 = rescale(abs(double(I) - double(med_bg))); % abs diff

%% From Mode background generation
subMax = rescale(1-(double(I) - double(mod_bg))); % std diff with inversion
subMax2 = rescale(abs(double(I) - double(mod_bg))); % abs diff

%% From complex background generation method
subCom = rescale(1-(double(I) - double(bg))); % std diff with inversion
subCom2 = rescale(abs(double(I) - double(bg))); % absdiff

figure;
montage({subMean subMed subMax subCom}, 'Size', [4 1]); title('standard diff +
inversion');
```




```
 figure; montage({subMean2 subMed2 subMax2 subCom2}, 'Size', [4 1]);
title('absolute diff');
```

absolute diff

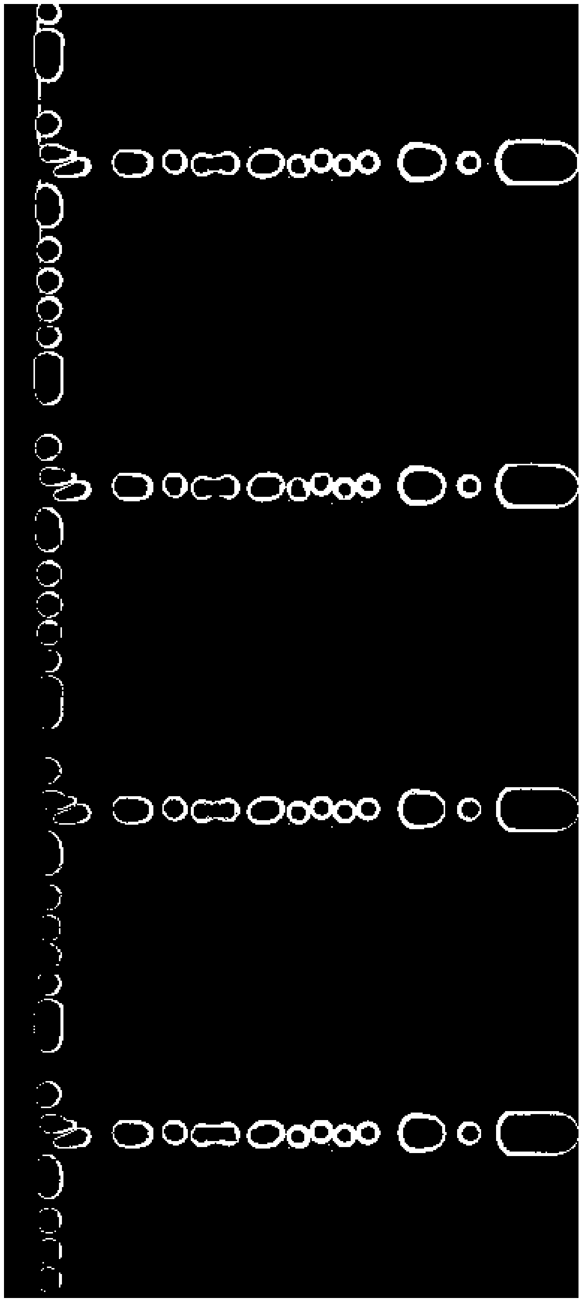


```
% figure; imshow(subMean); title('mean')
% figure; imshow(subMed); title('median')
% figure; imshow(subMax); title('mode')
% figure; imshow(subCom); title('complex')
```

5) Convert to binary image

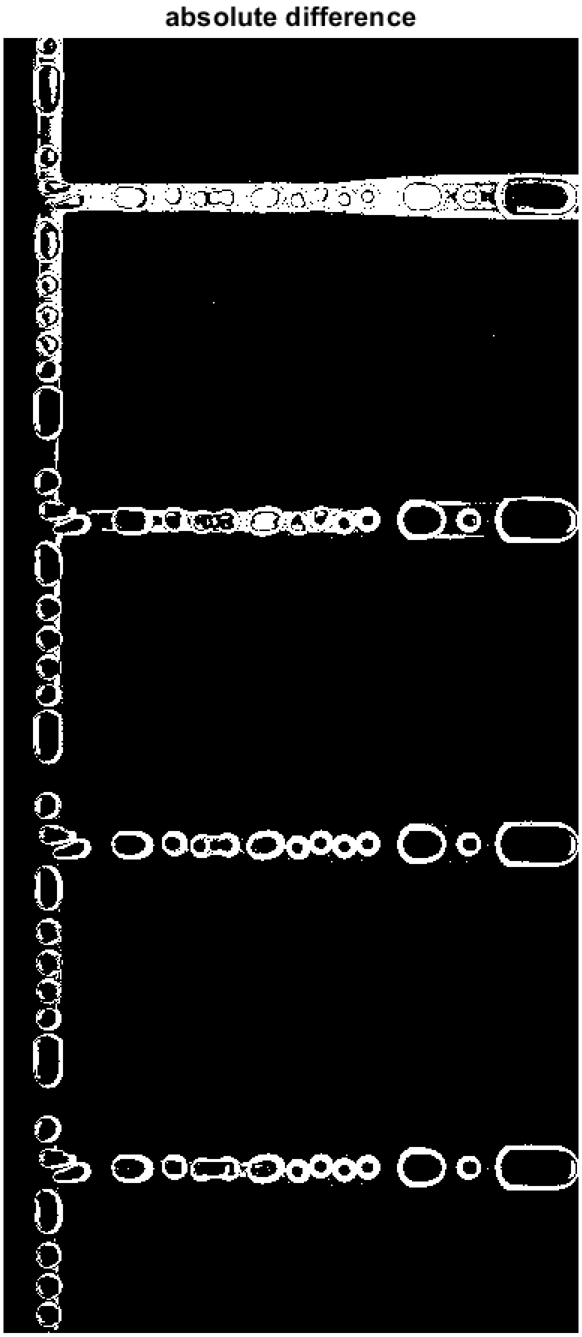
```
bin1 = imbinarize(imadjust((subMean)), graythresh(subMean));  %graythresh uses Otsu's Method
bin2 = imbinarize(imadjust((subMed)), graythresh(subMed));
bin3 = imbinarize(imadjust((subMax)), graythresh(subMax));
bin4 = imbinarize(imadjust((subCom)), graythresh(subCom));

figure; montage({bin1 bin2 bin3 bin4}, 'Size', [4 1]); title('standard difference + inversion');
```

standard difference + inversion



```
bin21 = imbinarize(imadjust((subMean2)), graythresh(subMean2));
bin22 = imbinarize(imadjust((subMed2)), graythresh(subMed2));
bin23 = imbinarize(imadjust((subMax2)), graythresh(subMax2));
bin24 = imbinarize(imadjust((subCom2)), graythresh(subCom2));
```

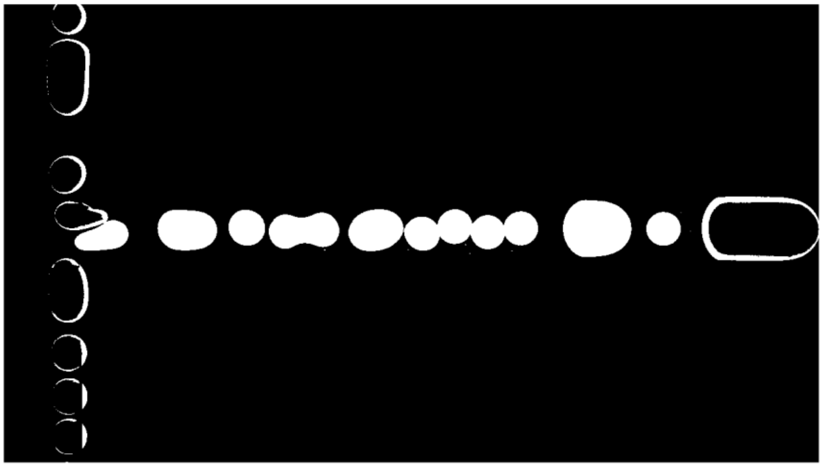
```
figure; montage({bin21 bin22 bin23 bin24}, 'Size', [4 1]); title('absolute difference');
```



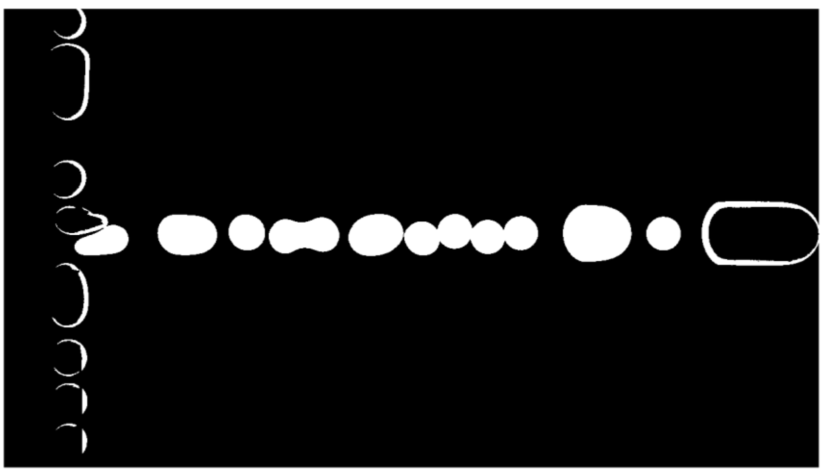
6) Morphological fill

- only use best result from previous parts

```
Img = bin4; % choose a  binary image to process

fill1 = imfill(Img, 'holes'); %fill fully closed drops
figure; imshow(fill1)
```



```
minA = 50; % anything with area less than this will be treated as noise object
cfill1 = bwareaopen(fill1,minA); %noise object removal
figure; imshow(cfill1);
```



```
rg = regionprops('table',cfill1,'Area') %detect ROI
```

rg = 20×1 table

	Area
1	1077
2	768

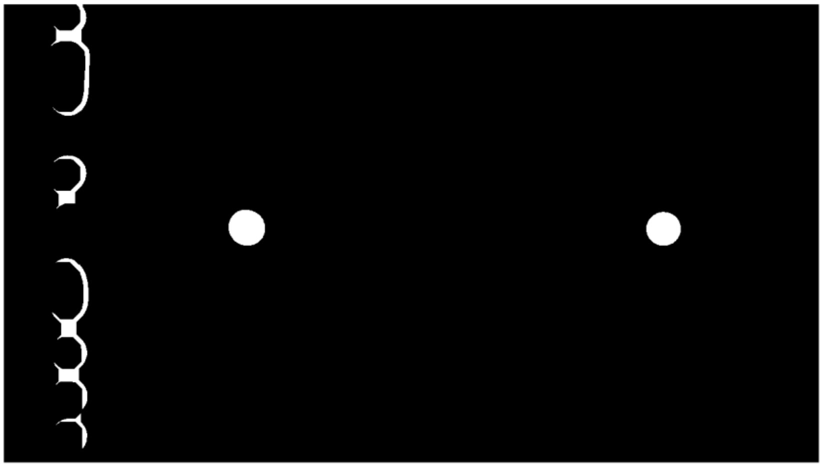
	Area
3	476
4	598
5	95
6	3474
7	131
8	106
9	80
10	78
11	380
12	239
13	216
14	4948
15	2556
16	4962
17	13720
18	7837
19	2258
20	3575

```
[bwl n] = bwlabel(cfill1,8); %assign label to each detected ROI
minA = 0.2*max(rg.Area)
```

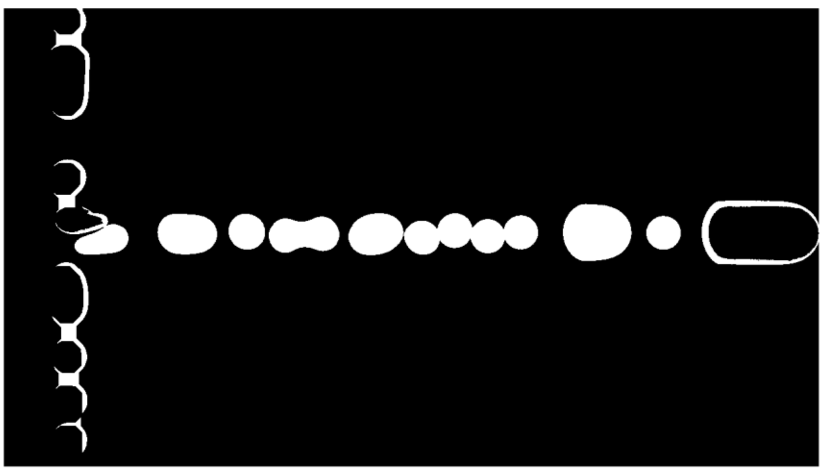
```
minA = 2744
```

```
pos = find(rg.Area < minA); %find ROIs that have holes so have not been filled;

bwl2 = ismember(bwl,pos); %segment out ROIs that need further manipulation
(morphological close)
% figure; imshow(bwl2);

bwl2 = imclose(bwl2,strel('disk',20)); %morphological close
fill2 = imfill(bwl2,'holes'); %fill fully closed drops (drops at border not
filled!)
% figure; imshow(fill2)

fill2_bord = fill_border_drops(bwl2); % fill objects at border (custom
function)
% figure; imshow(fill2_bord)

fill2 = fill2 | fill2_bord; %resulting filled image
```

```
figure; imshow(fill2);
```



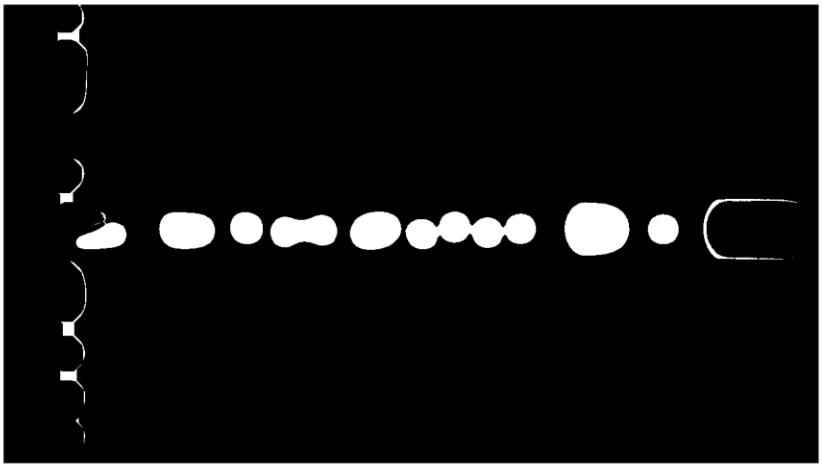
```
L = cfill1 | fill2; %combine to form a final mask
figure; imshow(L);
```



7) Drop Segmentation

- attempt to separate drops that were wrongly fused together from previous steps

```
L2 = imerode(L, strel('diamond',3));
figure; imshow(L2)
```



8) Identify Bounding Box of ROI

```
ROI = regionprops('table', L2)
```

ROI = 24×3 table

	Area	Centroid		...
1	651	110.3226	48.4209	
2	470	106.6660	290.9553	
3	422	104.4645	584.1730	
4	736	111.2636	494.9293	
5	142	124.7817	146.6549	
6	34	117.9118	655.7647	
7	2356	157.4550	365.6651	
8	10	120	2.0000	
9	37	127.4324	616.7838	
10	25	127.2800	679.0800	
11	1	140	347.0000	
12	4	145.5000	345.2500	
13	2	149.5000	344.0000	
14	1	152	343.0000	
15	4255	288.0045	355.0381	
16	2089	382.0469	351.5280	
17	4181	472.0445	356.1837	
18	3855	584.5274	355.4724	
19	7716	734.0257	355.4930	
20	6985	931.0017	352.8597	
21	1822	1.0366e+03	353.3452	

	Area	Centroid	
22	1097	1.1365e+03	345.9836
23	56	1.2030e+03	399.6786
24	1	1240	398.0000

```
% first entry in ROI table is the drop nearest to left frame border
leadEdge = ROI.Centroid(1,1) + 1.4*(ROI.BoundingBox(1,3) /2)  % estimate point
of entrance to main channel
```

```
leadEdge = 143.2226
```

```
% remove any drops outside of main flow channel
ROI(ROI.Centroid(:,1) < leadEdge, :) = []
```

```
ROI = 14×3 table
```

	Area	Centroid	
1	2356	157.4550	365.6651
2	4	145.5000	345.2500
3	2	149.5000	344.0000
4	1	152	343.0000
5	4255	288.0045	355.0381
6	2089	382.0469	351.5280
7	4181	472.0445	356.1837
8	3855	584.5274	355.4724
9	7716	734.0257	355.4930
10	6985	931.0017	352.8597
11	1822	1.0366e+03	353.3452
12	1097	1.1365e+03	345.9836
13	56	1.2030e+03	399.6786
14	1	1240	398.0000

```
% remove small objects
ROI(ROI.Area < 200, :) = []
```

```
ROI = 9×3 table
```

	Area	Centroid	
1	2356	157.4550	365.6651
2	4255	288.0045	355.0381
3	2089	382.0469	351.5280
4	4181	472.0445	356.1837

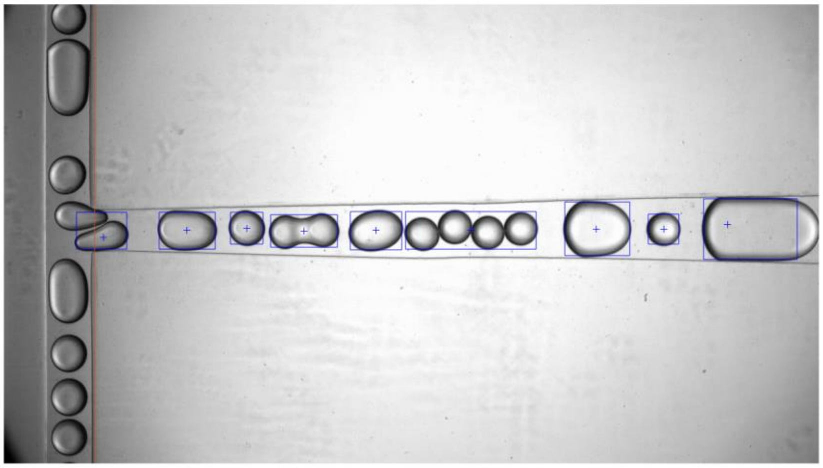
	Area	Centroid		...
5	3855	584.5274	355.4724	
6	7716	734.0257	355.4930	
7	6985	931.0017	352.8597	
8	1822	1.0366e+03	353.3452	
9	1097	1.1365e+03	345.9836	

Assumptions:

- horizontal flow in main channel
- entrance region is captured in video (i.e., video does not just purely show a segment of main flow)
- drop identified as that nearest to left frame border is not a random noise object (and thus the width is not representative of a regular drop)

9) Plot results

```
figure; imshow(I);
hold on

% visualise centroid
plot(ROI.Centroid(:,1), ROI.Centroid(:,2), 'b+')

% visualise bounding box
for i=1:height(ROI)
    rectangle('Position',ROI.BoundingBox(i,:),'EdgeColor','b')
end

% visualise estimated leading edge for analysed region
plot([leadEdge leadEdge], [0 vid.Height])
```



6) Crop and show individual drops

```
roiTable = ROI.BoundingBox;
cropSize = [128 128];

for i = 1: height(roiTable)
    dim = 1.4*max([roiTable(i,3:4)]); % this will be the width/ height of
square cropped area

    % x-coord of centroid of cropping region
    cx = roiTable(i,1) + roiTable(i,3)/2;
    % y-coord of centroid of cropping region
    cy = roiTable(i,2) + roiTable(i,4)/2;

    % coordinates of bottom-left vertex of intended crop region
    xmin = cx - dim/2 ;
    ymin = cy - dim/2 ;

    % crop region defined as [x-coord of bottom left point, y-coord of
    % bottom left point, width, height]
    Img = imcrop(I, [xmin, ymin, dim, dim]);

    % resize cropped image
    Img = imresize(Img, cropSize);

    %show image
    figure; imshow(Img)
end
```