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.

# 1) Load video

% vid\_file\_name = 'WAT004\_SO5\_17umL-13Wat\_umL-10kfps x4mag\_sh50\_C001H001S0016.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

# 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);

# 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,'original');

% 'original': for WAT, TRI, SDS, DYE.

% 'modified': for C12Tab.

figure; imshow(bg);

# 3) Load test image

% Load images

I = imread('DYE003\_19.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');

% 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');

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

[bwl n] = bwlabel(cfill1,8); %assign label to each detected ROI

minA = 0.4\*max(rg.Area)

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

% 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)

% 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

% remove any drops outside of main flow channel

ROI(ROI.Centroid(:,1) < leadEdge, :) = []

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 = 1.5\*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