STIC(C)S MATLAB Analysis Manual

Elvis Pandzic
e.pandzic@unsw.edu.au

ARC Centre for Advanced Molecular Imaging and
Australian Centre for NanoMedicine
University of New South Wales
Australia, Sydney, NSW, Australia

March 29, 2015

Contents

1 Introduction

This software implements spatiotemporal image (cross-)correlation spectroscopy (STICS), described in detail in Ref. (1) and later extended to two color analysis (4). The STICS technique is able to resolve the directed flow of fluorescently labelled macromolecules in living cells. Sometimes, it is desirable to filter the immobile features in an image series. This is achieved by the choice of several different immobile filter implementations. Its cross-correlation variant STICCS is able to measure the co-transport of two different fluorescently labelled macromolecules by cross-correlating their image series. Example applications of these techniques can be found in Refs. (2, 3).

1.1 Copyright

1.2 Disclaimer

1.3 Contact Info

Please send bug reports and suggestions to e.pandzic@unsw.edu.au

1.4 System Requirements

- Any platform supporting MATLAB (preferably latest version).
- There are no hardware requirements, per se. Processor speed will play a large role in the analysis time, and the amount of system RAM will dictate the maximum size of the image series which can be analyzed.

2 Two channels analysis

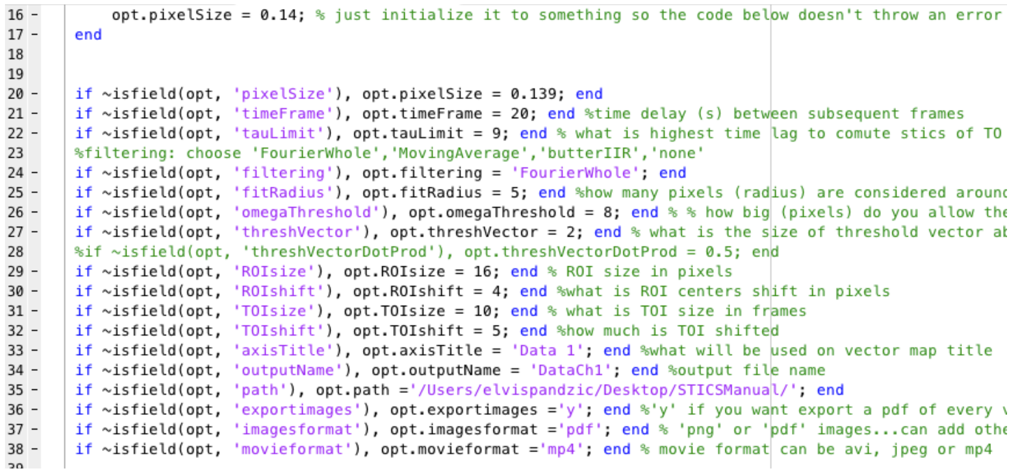
For the two channels analysis we use the *run2ChannelsSTICCS* script which calls the *sticcs-vectormapping* code to perform STICCS analysis. A STIC(C)S analysis is comprised of the following steps:

1. Load image series
2. Select region of interest within a cell
3. Immobile filter (if necessary)
4. STICS or STICCS analysis
5. Displaying and saving the flow vector maps

Each step is discussed in greater detail below through example using the test data files provided (`ch1data.tif` and `ch2data.tif`)

2.1 User defined inputs

The analysis settings must be defined inside the *stics-vectormapping* or *sticcs-vectormapping* (2 color) files, prior to running the scripts. Alternatively, any of the arguments can be defined outside these files, in the `opt` structure array which can be practical when running batch processing. The figure 1 shows the screenshot of the input setting for *stics-vectormapping* code.



```
    opt.pixelSize = 0.14; % just initialize it to something so the code below doesn't throw an error
end


if ~isfield(opt, 'pixelSize'), opt.pixelSize = 0.139; end
if ~isfield(opt, 'timeFrame'), opt.timeFrame = 20; end %time delay (s) between subsequent frames
if ~isfield(opt, 'tauLimit'), opt.tauLimit = 9; end % what is highest time lag to comute stics of TO
%filtering: choose 'FourierWhole','MovingAverage','butterIIR','none'
if ~isfield(opt, 'filtering'), opt.filtering = 'FourierWhole'; end
if ~isfield(opt, 'fitRadius'), opt.fitRadius = 5; end %how many pixels (radius) are considered aroun
if ~isfield(opt, 'omegaThreshold'), opt.omegaThreshold = 8; end % % how big (pixels) do you allow the
if ~isfield(opt, 'threshVector'), opt.threshVector = 2; end % what is the size of threshold vector a
%if ~isfield(opt, 'threshVectorDotProd'), opt.threshVectorDotProd = 0.5; end
if ~isfield(opt, 'ROIsize'), opt.ROIsize = 16; end % ROI size in pixels
if ~isfield(opt, 'ROIshift'), opt.ROIshift = 4; end %what is ROI centers shift in pixels
if ~isfield(opt, 'TOIsize'), opt.TOIsize = 10; end % what is TOI size in frames
if ~isfield(opt, 'TOIshift'), opt.TOIshift = 5; end %how much is TOI shifted
if ~isfield(opt, 'axisTitle'), opt.axisTitle = 'Data 1'; end %what will be used on vector map title
if ~isfield(opt, 'outputName'), opt.outputName = 'DataCh1'; end %output file name
if ~isfield(opt, 'path'), opt.path ='/Users/elvispandzic/Desktop/STICSManual/'; end
if ~isfield(opt, 'exportimages'), opt.exportimages ='y'; end %'y' if you want export a pdf of every
if ~isfield(opt, 'imagesformat'), opt.imagesformat ='pdf'; end % 'png' or 'pdf' images...can add othe
if ~isfield(opt, 'movieformat'), opt.movieformat ='mp4'; end % movie format can be avi, jpeg or mp4
```

Figure 1: Input settings for stics-vectormapping code.

pixelSize The size of a pixel in μm.

timeFrame The time delay between frames in s.

tauLimit The time correlation analysis will calculate up to at most this number of time lags, but may end earlier if either the `omegaThreshold` or `correlation local maximum threshold` are exceeded. Typically for N frames, the value for tauLimit can be set to ∼N/5.

Immobile Removal Settings Large static structures can mask the flowing component of the correlation function. Immobile filtering permits the measurement of the flowing population by removing slowly moving or immobile structures from the image series prior to STICS analysis.

FourierWhole The DC component of the temporal Fourier transform of the image series is set to zero. This is equivalent to subtracting the average image of the series from each image in the series. It will effectively remove all fluorescent structures that are static for the duration of the image series (1).

MovingAverage The average of $(n-1)/2$ frames before and $(n-1)/2$ frames after a given image are subtracted from it. The parameter n is implemented with the input `MoveAverage`. This method of immobile removal works better for large, slowly moving fluorescent entities, such as large, sliding adhesions (5).

butterIIR The Fourier immobile filter is non-ideal static component filter as it removes the slowest component of a given time series. As such it is not a true immobile component filter, as it is depending on the length of the image time series. Butterworth IIR filter is more appropriate for immobile population removal, independent of the length of the image time series (6).

Outlier Criteria Spurious vectors are sometimes calculated, which do not reflect true fluorophore fluxes, but are instead artifacts of different elements of the analysis. To minimize the number of spurious vectors included in the results, three different thresholds are applied at during the analysis:

omegaThreshold Large structures can perturb the correlation function and cause it to become much larger than a diffraction-limited point spread function. If the correlation function exceeds this radius for a given analysis subregion, the analysis will terminate for that subregion. The importance of an appropriate beam waist in the fit of a correlation function is described in Ref. (7). This parameter, in pixels, defines the largest beam waist acceptable from the correlation function fit.

fitRadius For a noisy correlation functions, it is preferable to fit the neighbourhood of the local maxima. This parameter, in pixels, defines the radius of a circular mask centred at the local maximum within ROI, which is considered in the fitting of 2D Gaussian.

threshVector With the default settings of `ROIshift`=4 and `ROIsize`=16, there is significant overlap of the data analyzed in adjacent regions. Consequently, adjacent vectors should be oriented in similar directions. This threshold marks some vectors as outliers if they deviate by a certain amount from their neighbours. The details of this threshold can be found in Ref. (8). Briefly, it is a factor multiplying the standard deviation of velocities of 24 next nearest neighbors of a given ROI. Typically, it is set to 2 but can be higher, to increase the tolerance to vector rejection criteria. It is used to assign a 'good' vector according to its neigborhood.

Correlation local maximum With noisy image series, the correlation function due to a flowing population can be small compared to the noise correlation peaks. When the global maximum of a correlation function is "close" to the height of a neighbouring local maximum, the analysis for that particular region is aborted. This threshold is based on that described in Ref. (9) and is not user defined.

ROIsize The size of the analysis region of interest (ROI), in pixels, used for velocity mapping analysis. Each arrow on the final velocity map is calculated from a region of pixels (ROIsize)$\times$(ROIsize).

ROIshift The space between adjacent analysis regions, in pixels. Note that if `ROIshift` is set to a value lower than `ROIsize`, adjacent arrows in the resulting analysis will not contain all unique information.

TOIsize The size of the temporal window of interest (TOI), in frames, used to define the overall size of the block of data used to calculate the local correlation function. Time equivalent of ROIsize.

TOIshift The number of frames by which the the TOI is shifted in time. Time equivalent of ROIshift.

axisTitle A string defining the title displayed in vector maps images and movies.

outputName A string used in naming the STIC(C)S output files.

path A string defining the output directory.

exportimages If this argument is set at 'y' then the images of vector maps will be saved in output directory.

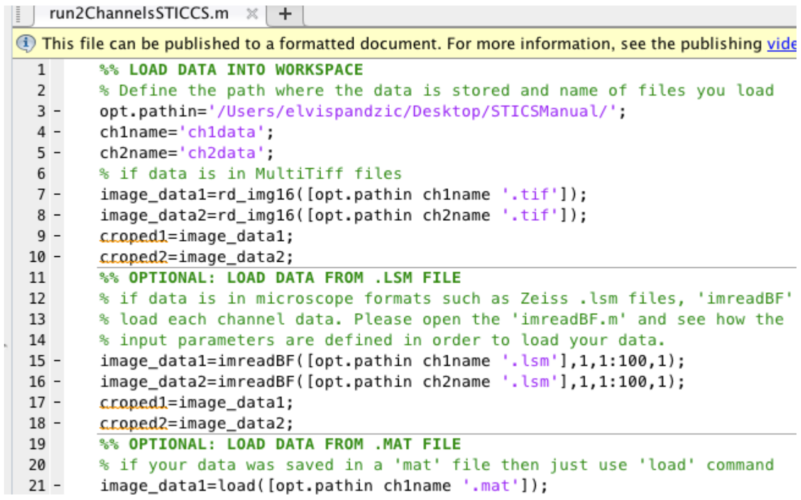
imagesformat This input can be set to 'pdf' or 'png' to output files of vector maps in those formats.

movieformat This input can be set to either 'avi', 'mp4' or 'jpeg' to specify the format of the vector map movie.

timeDelay This input argument is only used when two color STICS analysis is applied to the data. In case there is a time delay between images of channel 1 and channel 2, then a value of the time delay should be indicated in s.

2.2 Loading image series

After the input parameters are set, the image series can be loaded using either *run1ChannelSTICS* or *run2ChannelsSTICCS* lines as shown in figure 2. Depending of the input files format (.tif, .lsm, .mat or other) different lines of the code are ran as shown in figure 2. This step creates two variables `image-data1` and `image-data2`, which are arrays containing the image series for two channels.


run2ChannelsSTICCS.m

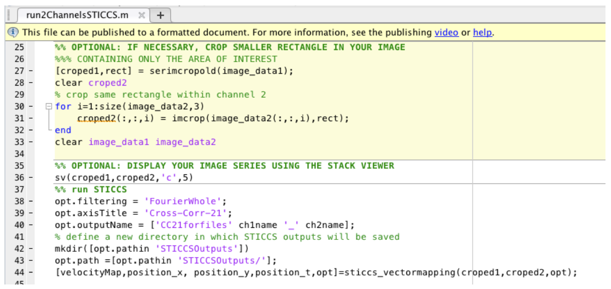
This file can be published to a formatted document. For more information, see the publishing vide

```
%% LOAD DATA INTO WORKSPACE
% Define the path where the data is stored and name of files you load
opt.pathin='/Users/elvispandzic/Desktop/STICSManual/';
ch1name='ch1data';
ch2name='ch2data';
% if data is in MultiTiff files
image_data1=rd_img16([opt.pathin ch1name '.tif']);
image_data2=rd_img16([opt.pathin ch2name '.tif']);
croped1=image_data1;
croped2=image_data2;
%% OPTIONAL: LOAD DATA FROM .LSM FILE
% if data is in microscope formats such as Zeiss .lsm files, 'imreadBF'
% load each channel data. Please open the 'imreadBF.m' and see how the
% input parameters are defined in order to load your data.
image_data1=imreadBF([opt.pathin ch1name '.lsm'],1,1:100,1);
image_data2=imreadBF([opt.pathin ch2name '.lsm'],1,1:100,1);
croped1=image_data1;
croped2=image_data2;
%% OPTIONAL: LOAD DATA FROM .MAT FILE
% if your data was saved in a 'mat' file then just use 'load' command
image_data1=load([opt.pathin ch1name '.mat']);
```

Figure 2: Loading data in *run2ChannelsSTICCS*.

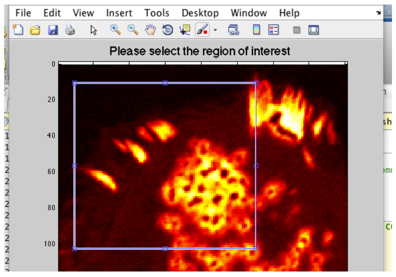
2.3 Selecting ROI to analyze

After loading the image series, the user can select a area of interest (AOI) for the further analysis. This step is preferable in case user wants to restrict the AOI to smaller window within the image time series. When running the lines 25-34 of *run2ChannelsSTICCS*, the average image of the time series is displayed (Figure 3 b) and user is prompted to select a rectangular window. Same rectangle AOI will be cropped in both channels, used later in STICCS analysis. This step will create two variables `croped1` and `croped2`, which are cropped versions of `image-data1` and `image-data2`. These will be the inputs for *sticcs-vectormapping* code.



```
run2ChannelsSTICCS.m
This file can be published to a formatted document. For more information, see the publishing video or help.
25      %% OPTIONAL: IF NECESSARY, CROP SMALLER RECTANGLE IN YOUR IMAGE
26      %%% CONTAINING ONLY THE AREA OF INTEREST
27 -    [croped1,rect] = serimcropold(image_data1);
28 -    clear croped2
29      % crop same rectangle within channel 2
30 -    for i=1:size(image_data2,3)
31 -        croped2(:,:,i) = imcrop(image_data2(:,:,i),rect);
32 -    end
33 -    clear image_data1 image_data2
34
35      %% OPTIONAL: DISPLAY YOUR IMAGE SERIES USING THE STACK VIEWER
36 -    sv(croped1,croped2,'c',5)
37      %% run STICCS
38 -    opt.filtering = 'FourierWhole';
39 -    opt.axisTitle = 'Cross-Corr-21';
40 -    opt.outputName = ['CC21forfiles' ch1name '_' ch2name];
41      % define a new directory in which STICCS outputs will be saved
42 -    mkdir([opt.pathin 'STICCSOutputs'])
43 -    opt.path =[opt.pathin 'STICCSOutputs/'];
44 -    [velocityMap,position_x, position_y,position_t,opt]=sticcs_vectormapping(croped1,croped2,opt);
```

(a) Croping AOI commands in *run2ChannelsSTICCS*

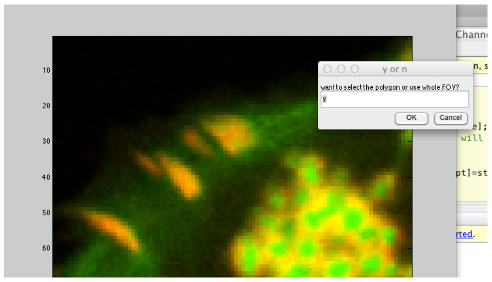

File Edit View Insert Tools Desktop Window Help
Please select the region of interest
0
20
40
60
80
100

(b) Crop AOI prompt

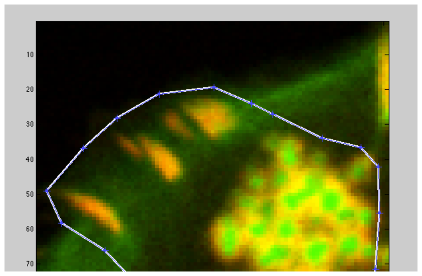
Figure 3: Selecting the AOI within the image time series.

2.4 STICCS Analysis

Lines 37 and on in file *run2ChannelsSTICCS* are used to run 2 color STICS. The user is first prompted to select a polygon inside the cropped region, which encloses the area of the cell which contain the meaningful information (Figure 4 a). This step will ensure that only ROIs inside this polygon will be analyzed, which results in an increase in the computation speed and processing of data sets. Typically areas outside of cells are not of interest, therefore there is no need to apply STIC(C)S to those ROIs. Following the selection of



(a) Select a polygon prompt in *run2ChannelsSTICCS*

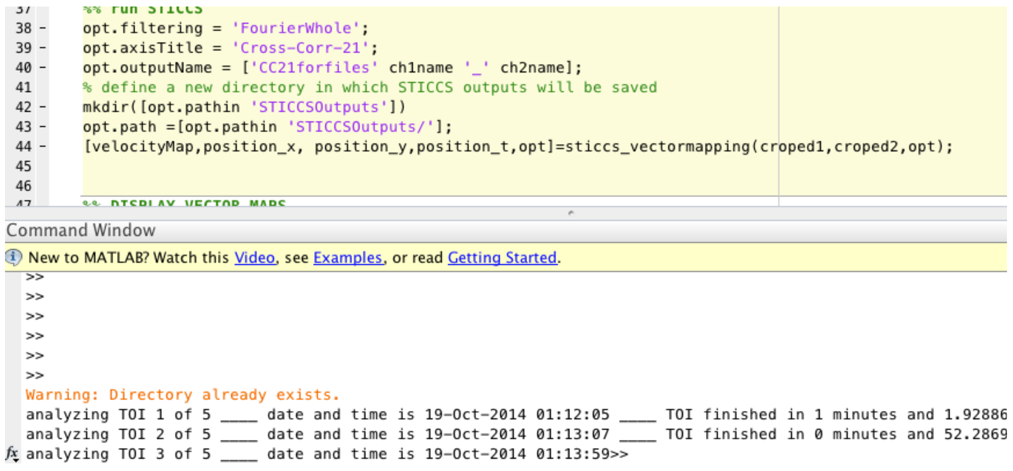


(b) Selecting a polygon

Figure 4: Selecting a polygon surrounding the are of interest inside of the cell. This procedure increases the effective speed of the STICCS processing on a data set.

a polygon containing the AOI, the STICCS proceeds with the analysis. If the code is running normally you will see a display of processing inside the Command Window which will indicate which TOI is being processed (Figure 5).

Note that beside `croped1` and `croped2`, *sticcs-vectormapping* has another input variable called `opt` (Figure 5 lines 38-43). This variable defines file specific parameters that user can set outside of the *sticcs-vectormapping* as shown in figure 1. Whatever is not defined in the opt variable outside of the *sticcs-vectormapping*, will take on the default value that is set inside the file as shown in figure 1. This feature of the code is used well when batch processing many files, as it will be shown in later section. Moreover, `opt.outputName` defines the string used for naming the output file. It can be very useful to be able to adjust the output file name according to the condition used. For instances, for the case shown in figure 5 we could have added the string "Fourierfiltered" insde the file name, to reflect this particular condition. Also, note that an output folder "STICCSOutputs" was created (Figure 5 lines 42 and 43) which will be used to store the output data. Once the STICCS has completed the analysis of data sets, the new



```
37      %% run STICCS
38 -    opt.filtering = 'FourierWhole';
39 -    opt.axisTitle = 'Cross-Corr-21';
40 -    opt.outputName = ['CC21forfiles' ch1name '_' ch2name];
41      % define a new directory in which STICCS outputs will be saved
42 -    mkdir([opt.pathin 'STICCSOutputs'])
43 -    opt.path =[opt.pathin 'STICCSOutputs/'];
44 -    [velocityMap,position_x, position_y,position_t,opt]=sticcs_vectormapping(croped1,croped2,opt);
45
46
47      %% DISPLAY VECTOR MAPS
```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>>
>>
>>
>>
>>
>>
Warning: Directory already exists.
analyzing TOI 1 of 5 ____ date and time is 19-Oct-2014 01:12:05 ____ TOI finished in 1 minutes and 1.92886
analyzing TOI 2 of 5 ____ date and time is 19-Oct-2014 01:13:07 ____ TOI finished in 0 minutes and 52.2869
analyzing TOI 3 of 5 ____ date and time is 19-Oct-2014 01:13:59>>
```

Figure 5: Running STICCS lines 37-45 in *run2ChannelsSTICCS*

variables will appear in the Matlab Workspace (Figure 6). The following output variables are saved in the "STICCSOutputs" folder under variable named "VelocityMap+YourOriginalFileName":

opt This structure variable contains all the input parameters that were used in the STICCS analysis. It is useful to have it later when accessing the results of a particular file in order to know the exact input values that were used for this STICCS analysis.

position-x This vector variable denotes the values, in pixels, of the ROIs centres along the x-axis.

position-y This vector variable denotes the values, in pixels, of the ROIs centres along the y-axis.

position-t This vector variable denotes the values, in frames, of the TOIs centres along the time axis.

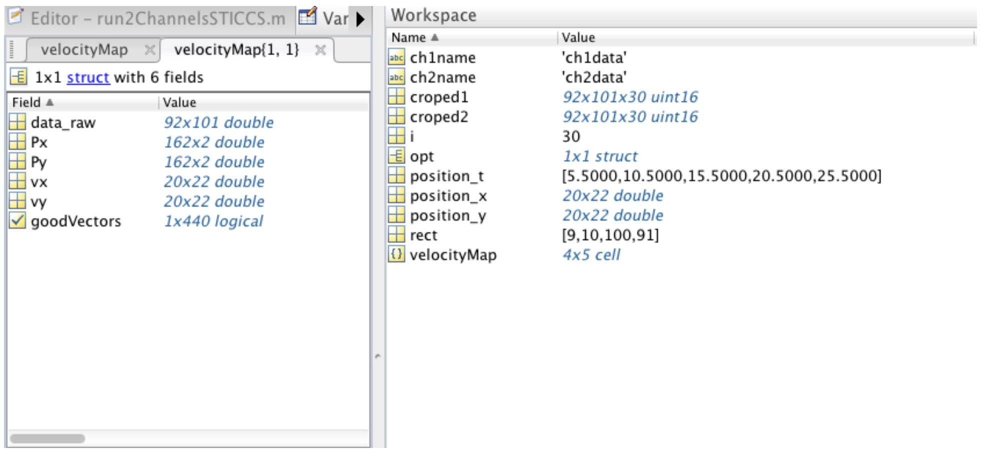
VelocityMap This cell array contains most of the STICCS output variables need for further processing (vector mapping, velocity histograms). Figure 6 shows its dimensions are in 4 by T, where the indexes 1-4 are for auto-correlation channel 1, auto-correlation channel 2, cross-correlation channel 12 and cross-correlation channel 21, respectively. The index T relates to number of TOIs that were outputted. For the settings shown in figure 1 and the test data files used, 5 TOIs were analyzed. Therefore, `velocityMap` is 4x5 cell array. Figure 6 shows the contents of `velocityMap` for auto-correlation channel 1 and for TOI 1. The sub-variables contained in each case are:

data-raw This array contains the average image of a given TOI. It is used as a background image in the vector mapping process.

Px and Py These are results of the linear fits of $v_{x,y}\tau$ vs τ to extract the flow velocities (v_x,v_y). In the example data sets there were 162 'good' ROIs (these are ROIs which were within the selected polygon and survived the vector filtering steps) as many as Px and Py entries. The 2 columns of each represent the slopes of the linear fit (which are velocities v_x,v_y) and offsets in the fit.

$\mathbf{v}_x$,$\mathbf{v}_y$ These are flow velocities in x and y directions extracted for each ROI. They have dimensions M by N, which is same as number of ROIs within the field of view.

goodVectors It is a logical array which indicates 'good' ROIs with 1 and 'bad' with 0. It is useful for further processing such as vector mapping.



1x1 struct with 6 fields

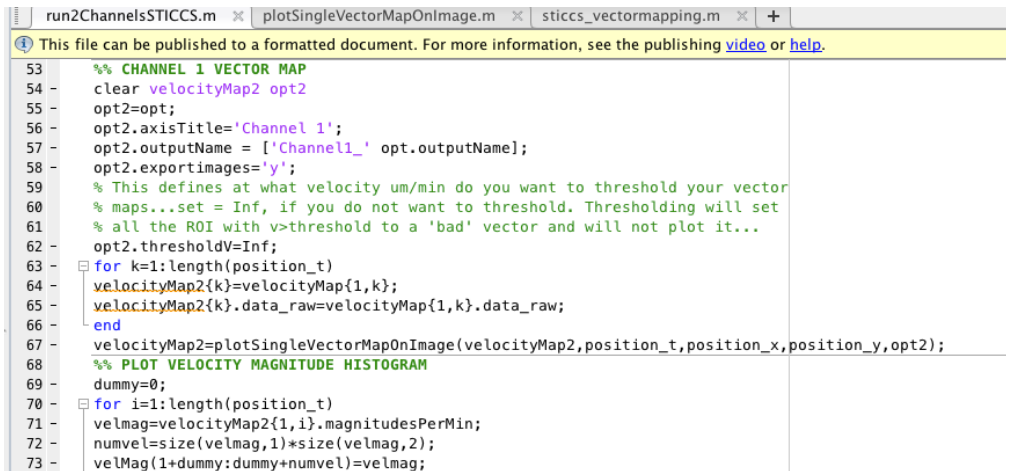
Field	Value
data_raw	92x101 double
Px	162x2 double
Py	162x2 double
vx	20x22 double
vy	20x22 double
goodVectors	1x440 logical

Name	Value
ch1name	'ch1data'
ch2name	'ch2data'
croped1	92x101x30 uint16
croped2	92x101x30 uint16
i	30
opt	1x1 struct
position_t	[5.5000,10.5000,15.5000,20.5000,25.5000]
position_x	20x22 double
position_y	20x22 double
rect	[9,10,100,91]
velocityMap	4x5 cell

Figure 6: STICCS output variables

2.5 Displaying results

Now we can display the results in the form of vector maps or plot a histogram of the magnitude of flow speeds. Figure 7 shows the command lines used to plot the vector map for the auto-correlation channel 1 and its speed histogram. Similar lines are used for other auto-correlation channel and cross-correlation channels. Note that we define temporary variables `opt2` and `velocityMap2` where we access the specific parts of the overall `velocityMap` array to plot a desired channel vector map. Also, we introduce the variable `opt2.thresholdV` (Figure 7 line 62) which can be varied from 0 to Inf. When set to Inf, all the 'good' vectors are displayed in the vector map, and the colormap is adjusted according to the value of the maximum speed found. Otherwise, lower value of this parameter, will result in considering only the vectors below this value (in μm/min) in the plotting of the vector map. Running the lines 53-57 inside *run2ChannelsSTICCS* will generate the vector map images and save the .pdf or .png format as well as the movies of the chosen format (see the details about input variables). These are stored in the output folder previously created ("STICCSOutputs"). Figure 8 shows the results of STICCS analysis on the test data sets provided with the code. The vector maps of TOIs and the speed histograms are displayed for each auto- and cross- correlation channel. Note that cross-correlation histograms might indicate larger maximum vectors detected than any of the auto-correlation channels. This is not unusual, as the STICS pick the flow vector of the most


run2ChannelsSTICCS.m × plotSingleVectorMapOnImage.m × sticcs_vectormapping.m × +

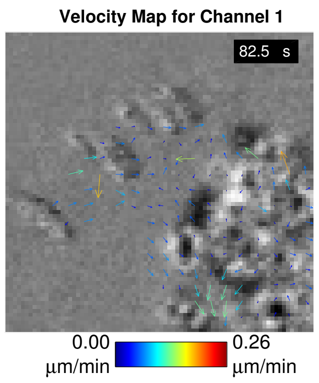
This file can be published to a formatted document. For more information, see the publishing video or help.

```
%% CHANNEL 1 VECTOR MAP
clear velocityMap2 opt2
opt2=opt;
opt2.axisTitle='Channel 1';
opt2.outputName = ['Channel1_' opt.outputName];
opt2.exportimages='y';
% This defines at what velocity um/min do you want to threshold your vector
% maps...set = Inf, if you do not want to threshold. Thresholding will set
% all the ROI with v>threshold to a 'bad' vector and will not plot it...
opt2.thresholdV=Inf;
for k=1:length(position_t)
velocityMap2{k}=velocityMap{1,k};
velocityMap2{k}.data_raw=velocityMap{1,k}.data_raw;
end
velocityMap2=plotSingleVectorMapOnImage(velocityMap2,position_t,position_x,position_y,opt2);
%% PLOT VELOCITY MAGNITUDE HISTOGRAM
dummy=0;
for i=1:length(position_t)
velmag=velocityMap2{1,i}.magnitudesPerMin;
numvel=size(velmag,1)*size(velmag,2);
velMag(1+dummy:dummy+numvel)=velmag;
```

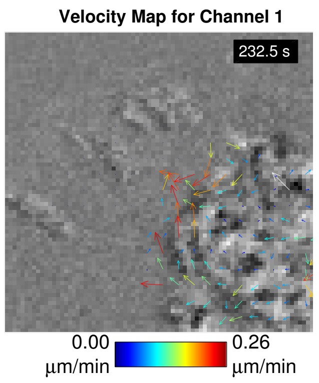
Figure 7: Commands in *run2ChannelsSTICCS* for vector mapping and generation of a speed histogram

dominant population in any given ROI-TOI. If for instance, there is a second particle population, giving rise to a less dominant peak in auto-correlation channel, but happens to co-localize with particles in channel 2, then it will contribute to the dominant peak in the cross-correlation channels, for this ROI-TOI. Therefore it is possible to pick a faster and co-localized species of particles in a given ROI-TOI, that are not 'visible' by STICS in either of auto-correlation channels. Nevertheless, the histograms of the speeds for all ROI-TOIs found show similar trends in auto- and cross- correlation channels.

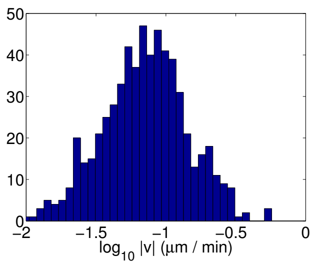
If the variable `opt2.thresholdV` was set to a smaller value than 'Inf' then the vector maps will be plotted considering it as the maximum speed. Consequently, the vectors' color and lengths will be adjusted with respect to this maximum value. For comparison of results obtained with the values Inf and 0.05 μm/min please refer to the vector maps in the figure 9. Note that some vector appearing in the figure 9 a) were removed in figure 9 b), as a result of this velocity thresholding.



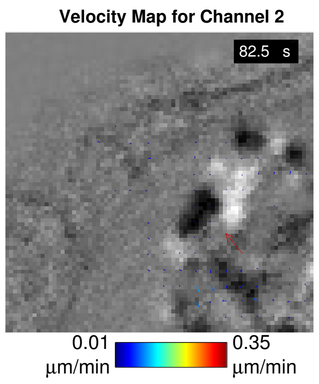
(a) Auto-corr 1 at TOI 1



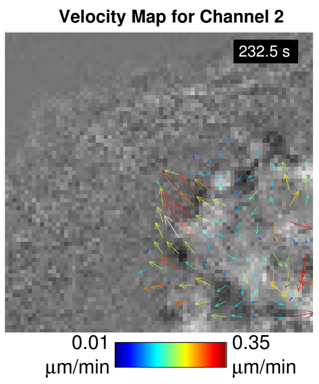
(b) Auto-corr 1 at TOI 3



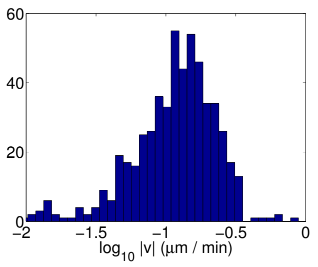
(c) Speed histogram auto-corr 1



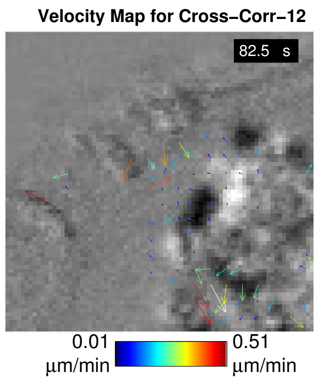
(d) Auto-corr 2 at TOI 1



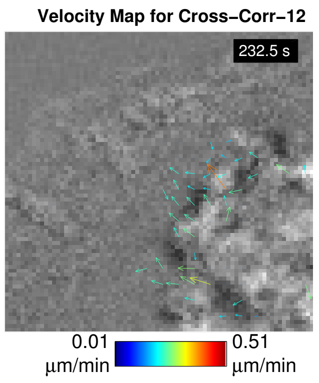
(e) Auto-corr 2 at TOI 3



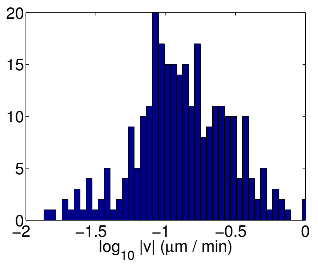
(f) Speed histogram auto-corr 2



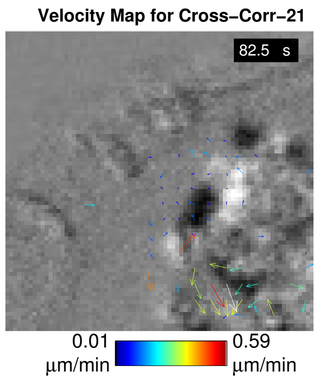
(g) Cross-corr 12 at TOI 1



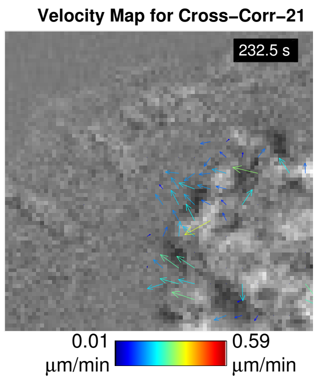
(h) Cross-corr 12 at TOI 3



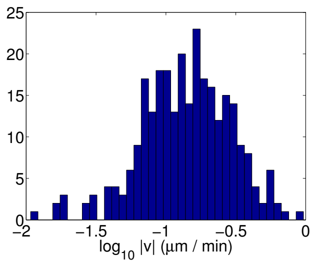
(i) Speed histogram cross-corr 12



(j) Cross-corr 21 at TOI 1

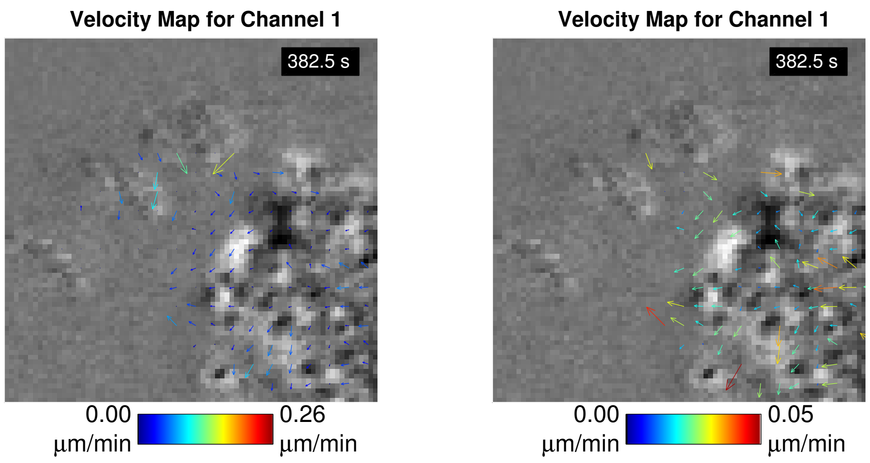


(k) Cross-corr 21 at TOI 3



(l) Speed histogram cross-corr 21

Figure 8: Velocity maps and speed histograms of the trial data set.

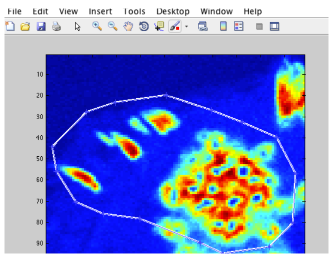


(a) opt2.thresholdV=Inf (b) opt2.thresholdV=0.05 μm/min

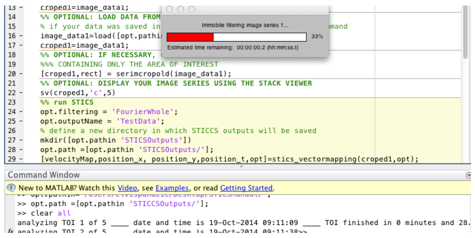
Figure 9: Effect of `opt2.thresholdV` value on the final vector map

3 Single channel analysis

For the single channel analysis we use the *run1ChannelSTICS* script which calls the *stics-vectormapping* code to perform STICS. The steps are very similar to the ones outlined above for two channels analysis, but this time with only one channel. The first 20 lines of *run1ChannelSTICS* are similar to ones in *run2ChannelsSTICCS* as shown in figure 2. As for the selection of a polygon step and running the STICS code please refer to the example in figure 10.



(a) Polygon select prompt in *stics-vectormapping*


Immobile filtering image series 1...
33%
Estimated time remaining: 00:00:00.2 (hh:mm:ss.t)
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
croped1=image_data1;
%% OPTIONAL: LOAD DATA FROM
% if your data was saved in                                   mand
image_data1=load([opt.pathi
croped1=image_data1;
%% OPTIONAL: IF NECESSARY,
%%% CONTAINING ONLY THE AREA OF INTEREST
[croped1,rect] = serimcropold(image_data1);
%% OPTIONAL: DISPLAY YOUR IMAGE SERIES USING THE STACK VIEWER
sv(croped1,'c',5)
%% run STICS
opt.filtering = 'FourierWhole';
opt.outputName = 'TestData';
% define a new directory in which STICCS outputs will be saved
mkdir([opt.pathin 'STICSOutputs'])
opt.path =[opt.pathin 'STICSOutputs/'];
[velocityMap,position_x, position_y,position_t,opt]=stics_vectormapping(croped1,opt);
```

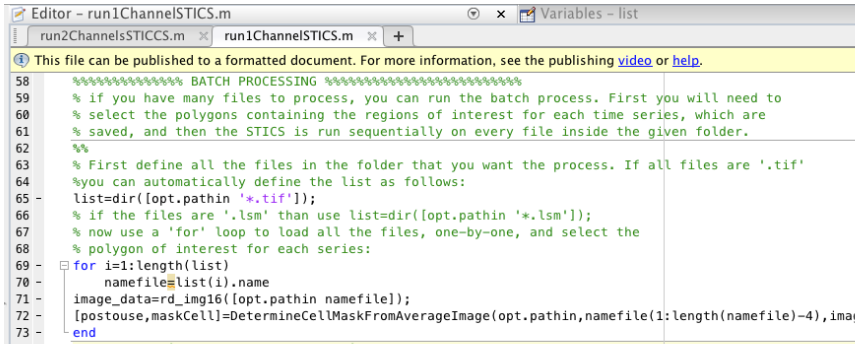
```
>> opt.path =[opt.pathin 'STICCSOutputs/'];
>> clear all
analyzing TOI 1 of 5 ____ date and time is 19-Oct-2014 09:11:09 ____ TOI finished in 0 minutes and 28.
analyzing TOI 2 of 5 ____ date and time is 19-Oct-2014 09:11:38>>
```

(b) Run commands in *run1ChannelSTICS*

Figure 10: Selecting the polygon of interest within cell and running STICS

4 STICS Batch processing

Often there is more than one file to be processed by STICS. Therefore, it is not practical to wait until one file is finished to select the polygon of interest for the second file and then wait again for the prompt of the third file and so on. It is more efficient to pre-select the polygon containing the area of interest within each file and save the information, followed by serial STICS processing of each file. In order to do this, user can refer to lines 58 and on in the file *run1ChannelSTICS* (Figure 11). Following the above operation, there will



This file can be published to a formatted document. For more information, see the publishing video or help.

```
58      %%%%%%%%%%%%%%% BATCH PROCESSING %%%%%%%%%%%%%%%%%%%%%%%%%%%
59      % if you have many files to process, you can run the batch process. First you will need to
60      % select the polygons containing the regions of interest for each time series, which are
61      % saved, and then the STICS is run sequentially on every file inside the given folder.
62      %%
63      % First define all the files in the folder that you want the process. If all files are '.tif'
64      %you can automatically define the list as follows:
65 -    list=dir([opt.pathin '*.tif']);
66      % if the files are '.lsm' than use list=dir([opt.pathin '*.lsm']);
67      % now use a 'for' loop to load all the files, one-by-one, and select the
68      % polygon of interest for each series:
69 -    for i=1:length(list)
70 -        namefile=list(i).name
71 -    image_data=rd_img16([opt.pathin namefile]);
72 -    [postouse,maskCell]=DetermineCellMaskFromAverageImage(opt.pathin,namefile(1:length(namefile)-4),ima
73 -    end
```

Figure 11: Serially loading data files, selecting the polygon of interest' and saving to file.

be new files stored in the input folder which are named as 'MaskWithinCell-Serie+NameOfYourDataFile.mat'. These files contain the info about which pixels are within the polygon of interest. Subsequently, lines 74 and on are run to perform STICS serially on all files in the input folder. Note that code *stics-batch* is being used here rather than *stics-vectormapping*, so make sure you adjust the input parameters within this file to match the desired analysis. When it comes to setting of input parameters for each particular file, one can define a vector of numbers or strings such as example of 'FilterString' variable figure 12. In this case, the first entry defines that the file 1 in the list of files will be 'FourierWhole' filtered while the second file will not be filtered. If for some reason the frame time was different for the two files being analyzed, it is possible to define a vector, say 'FrameTimes'=[frame time 1, frame time 2], and later in the 'for' loop define 'opt.Timeframe=Frametimes(i)'. Therefore, the input parameters can be specifically defined for each file being processed afterwards.

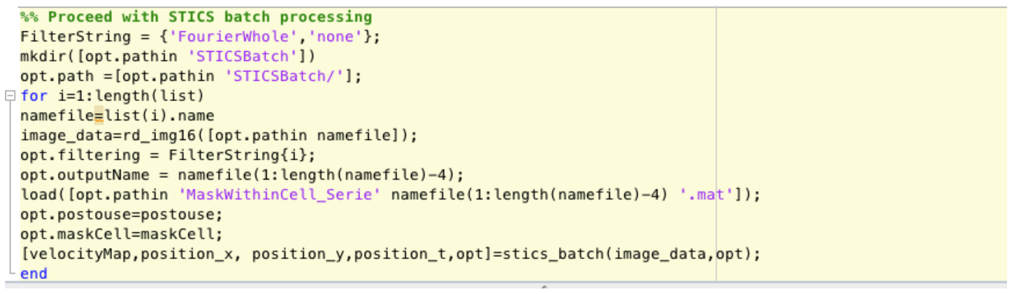

```
%% Proceed with STICS batch processing
FilterString = {'FourierWhole','none'};
mkdir([opt.pathin 'STICSBatch'])
opt.path =[opt.pathin 'STICSBatch/'];
for i=1:length(list)
namefile=list(i).name
image_data=rd_img16([opt.pathin namefile]);
opt.filtering = FilterString{i};
opt.outputName = namefile(1:length(namefile)-4);
load([opt.pathin 'MaskWithinCell_Serie' namefile(1:length(namefile)-4) '.mat']);
opt.postouse=postouse;
opt.maskCell=maskCell;
[velocityMap,position_x, position_y,position_t,opt]=stics_batch(image_data,opt);
end
```

Figure 12: STICS batch processing lines in *run1ChannelSTICS*

5 Acknowledgements

The core of this software and all related files were developed and implemented in GUI version by David L. Kolin. The following authors have made their m-files publicly available and are included in this distribution: John D'Errico (pleas.m), Clay M. Thompson (quiverc.m), Jan Glascher (NaN related m files), Roland Bunschoten (distance.m), M. Bach (circle.m), Chad English (timebar.m), Laurent Potvin-Trottier (butterIIR.m), Oliver Woodford (export-fig.m toolbox), Christoph Moehl (imreadBF.m), Manuel Guizar (dftregistration), Carlos Adrin Vargas Aguilera (moving average).

A Adding *export-fig* subfolder to the Matlab path

In the shared folder STICCSOutsideGUI you will notice a folder *export-fig*. It contains files used to save vector maps in a clean format, removing the borders in Matlab figures. This package of codes is very handy and I strongly suggest you use it for ouputting your images in your favorit format. When you run the *run1ChannelSTICS* code from the main folder (STICCSOutsideGUI), it is trying to run *export-fig* file and since it is in the subfolder which is not defined in the Matlab path, the code crashes.

To remedy this situation, you need to add the subfolder 'export-fig' into your *pathdef* file. There are few ways you can do this. One is by clicking on the 'Set Path' icon in the matlab's 'Home' tab, and once the pop-up window appears, click on 'Add Folder'. You will prompted to select the folder you want to add to the path. Select your folder which could be something like:

```
C:/YOUR HOME FOLDER/STICCSOutsideGUI/export_fig/
```

then click on the 'Save' button and then 'Close'. Now this folder was added to the path and Matlab will recognize it. Alternative approach is to change your Current folder in the Matlab to the one where you saved the STICCS codes (see above). If you made it there, you will see all the files of *export-fig* displayed in the 'Current folder' window on the left of Matlab window. Now type in the Command Window 'savepath' and press enter. This will add the current folder to the *pathdef* file that contains all the folder which Matlab searches through when trying to run the codes.

Subsequently, the code might display an error message about missing ghostscript and will suggest you to go onto ghostcript.com to download ghostscript and install it. I would suggest that you do it, as it is necessary to have the ghostscript to convert ps files to pdf. This is essential if you want to output your files (vector maps) into pdf format, which I find are very good at preserving the resolution of images, as there is no compression involved. If you do not want to save vector maps into pdfs but would prefer only png, then you do not need to install ghostscript, but you do need to make sure that you define this before line 39 of *run1ChannelSTICS*:

```
opt.imagesformat ='png';
```

Alternatively, change lines 114-124 of *plotSingleVectorMapOnImage* to output other formats, such as jpg or tif.

B Data pre-processing

B.1 Alignement of 2 color data

With almost any 2 color data sets one runs into a problem of mis-alignement. Ideally, one would use a two color fiducials, like Au particles or Teraspeck particles, and image them in the same field of view as the cell of interest, but sometimes we are in presence of the data sets that do not have reference fiducials. In this case, it is still possible to align the images from channel 2 to the images in the channel 1 by using the presence of common cell edges or large clusters (organelles) present in both channels. In this case, one can use the cross-correlation of the whole images to correct for this mis-alignement. In this package, you will find the function *Align2ColorDataSets* which can be used as is or modifed to take account for transformations other than translation and rotation between two channel image.

B.2 Registration of image series to the first frame

Another pre-processing of your data could consist of images registration to correct for the drift. In this case you would use functions *registerStack-ToFirstImage* and *dftregistration* in order to register all the images in a series to the first one.

References

[1] Benedict Hebert, Santiago Costantino, and Paul W. Wiseman. Spatiotemporal image correlation spectroscopy (STICS) theory, verification, and application to protein velocity mapping in living CHO cells. *Biophys. J.*, 88:3601–3614, May 2005.

[2] P.W. Wiseman, C.M. Brown, D.J. Webb, B. Hebert, N.L. Johnson, J.A. Squier, M.H. Ellisman, and A.F. Horwitz. Spatial mapping of integrin interactions and dynamics during cell migration by image correlation microscopy. *J. Cell Sci.*, 117(23):5521–5534, 11 2004.

[3] Claire M. Brown, Benedict Hebert, David L. Kolin, Jessica Zareno, Leanna Whitmore, Alan F. Horwitz, and Paul W. Wiseman. Probing the Integrin-Actin Linkage using High Resolution Protein Velocity Mapping. *J. Cell Sci.*, 119:5204–5214, 2006.

[4] T. Toplak, E. Pandzic, L. Chen, M. Vicente-Manzanares, A.R. Horwitz, and P.W. Wiseman. Sticcs reveals matrix-dependent adhesion slipping and gripping in migrating cells. *Biophys. J.*, 103(8):1672–82, 2012.

[5] M.A. Digman, C.M. Brown, P. Sengupta, P.W. Wiseman, A.R. Horwitz, and E. Gratton. Measuring fast dynamics in solutions and cells with a laser scanning microscope. *Biophys. J.*, 89:1317–1327, 2005.

[6] Laurent Potvin-Trottier, Lingfeng Chen, Alan Rick Horwitz, and Paul W. Wiseman. A nu-space for ics: characterization and application to measure protein transport in live cells. *New J Phys*, 15(10), 2013.

[7] N.O. Petersen, P.L. Hoddelius, P.W. Wiseman, O. Seger, and K.E. Magnusson. Quantitation of membrane receptor distributions by image correlation spectroscopy: concept and application. *Biophys. J.*, 65(3):1135–46, Sep 1993.

[8] A.-M. Shinneeb, J.D. Bugg, and R. Balachandar. Variable threshold outlier identification in PIV data. *Meas. Sci. Technol.*, 15:1722–1732, 2004.

[9] L. Ji and G. Danuser. Tracking quasi-stationary flow of weak fluorescent signals by adaptive multi-frame correlation. *J. Microsc.*, 220(3):150–167, 2005.