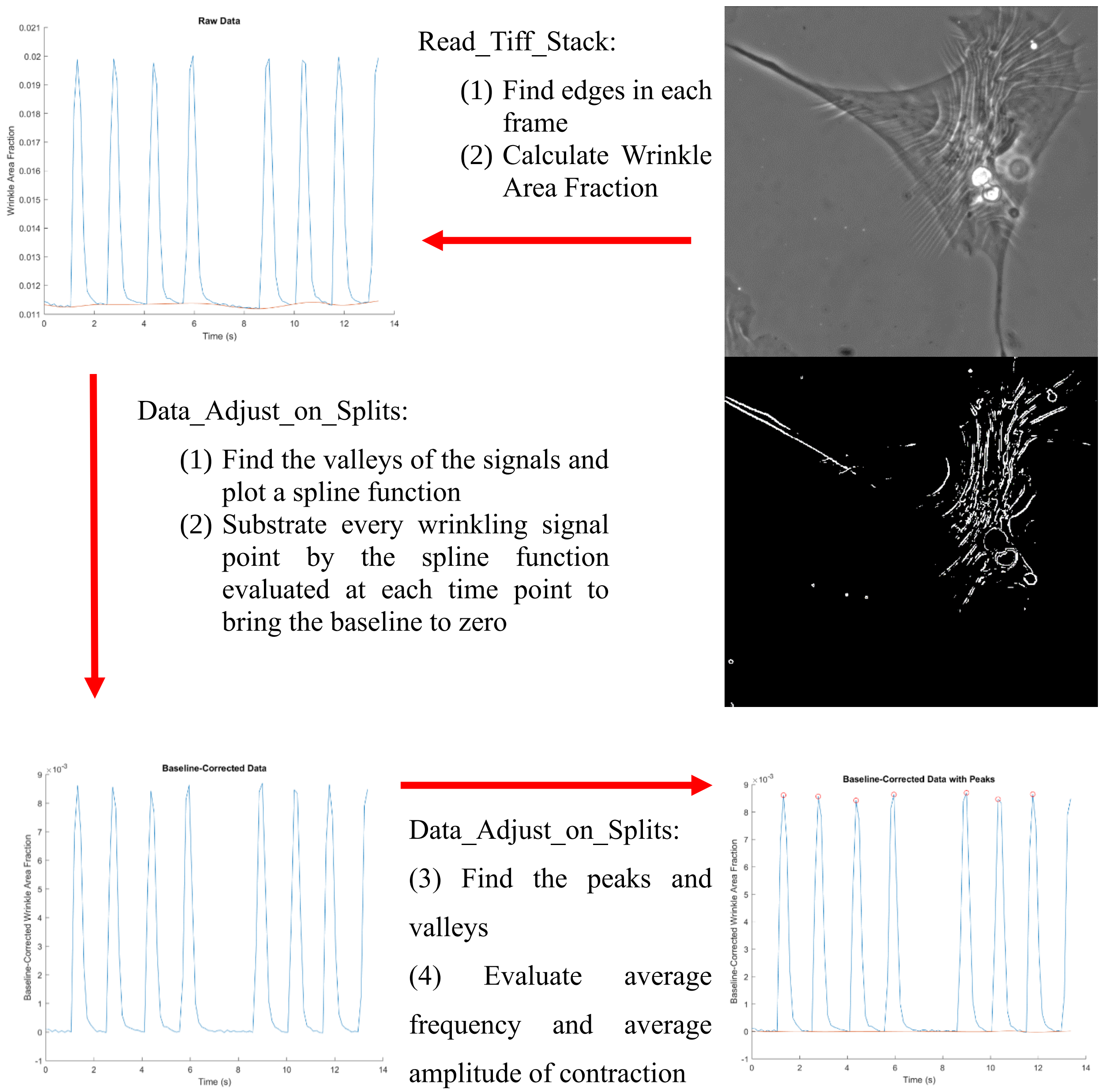

Figure 1: Process Flow of the Automated Image Analysis
This figure details the procedure in a step-by-step manner explaining what built-in function was used and what data fitting or calculation was done.

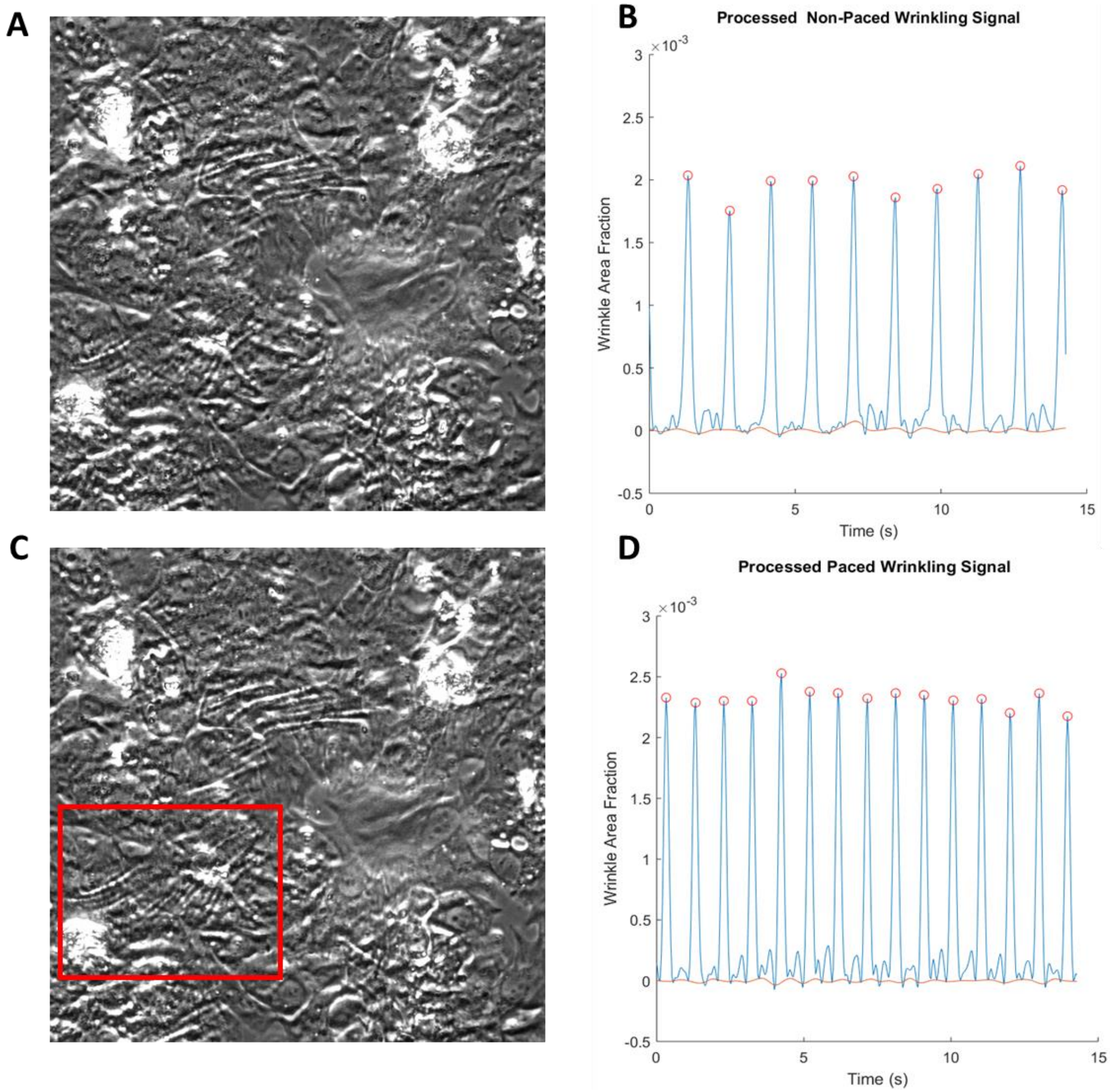

Figure 2: Validation of the Automated Image Analysis
(A-B) show the phase contrast image of fully contracted CM colony without electrical pacing wrinkling on soft silicone substrate and its corresponding wrinkling signal obtained with the automated image analysis. (C-D) show the same CM colony with electrical pacing at 1 Hz and 6 V and its corresponding wrinkling signal. Red square highlights the region with increase in wrinkles compared to the control. Extracting beating signal shows a frequency of 1.05 Hz in accordance to the pacing frequency of 1Hz. In addition, there is an apparent increase of 0.04 % in wrinkle area fraction or amplitude of contraction compared to the control.

Appendix 1: Automated Image Analysis

Automated_Master_Script_For_Patterning_Cms_ROI.m

```
close all
clear all
clc
z=input('What is the prominence criteria for peaks and valleys selection? Enter "1" for 30% max
pks/vs, "2" for 50% max pks/vs"3" for avg_pks/avg_vs');

D=dir;
skip=[];
original_dir=pwd;
fps=input('frame per second');%frames per second
tpf=1/fps;%time per frame

for k=3:length(D)
    if ~(D(k).isdir)
        continue
    end
        cd(strcat(original_dir,'\',D(k).name));
    A=dir;
    well_folder=pwd;
for i=3:length(A)

if A(i).isdir
cd(strcat(well_folder,'\',A(i).name));
B_pwd=pwd;
B=dir;
else
    continue
end
for b=3:length(B)
    if ~strcmp(B(b).name,'ROI')
        continue
    end
    cd(strcat(B_pwd,'\',B(b).name));
    J=dir;
    J_pwd=pwd;
for j=3:length(J)
    cd(J_pwd);
if ~isempty(strfind(J(j).name,'4X'))||isempty(strfind(J(j).name,'.tif'))
  continue
end
fname=J(j).name;
info=imfinfo(fname);

[I_g_find_edge,Percent_wrinkles,I_collect]=Read_Tiff_Stack(info,fname,tpf);
if ~isempty(strfind(fname(1:end-4),'.'))
    i=strfind(fname(1:end-4),'.');
    fname(i)='_';

end
[I_g_find_edge,Percent_wrinkles,I_collect]=Read_Tiff_Stack(info,fname,tpf);
pos2=strfind(fname,'.');
analysis_data_name=fname(1:(pos2(1)-1));


mkdir(strcat(analysis_data_name,'_','MATLAB'))
cd(strcat(analysis_data_name,'_','MATLAB'))

data_length=length(Percent_wrinkles(1,:));
xlswrite(('MATLAB.xlsx'),Percent_wrinkles(1,:)','Original
Data',strcat('A2:A',num2str(data_length+2-1)));
xlswrite(('MATLAB.xlsx'),Percent_wrinkles(2,:)','Original
Data',strcat('B2:B',num2str(data_length+2-1)));
time=Percent_wrinkles(1,1:end); %To cut off end irregularirties
data=Percent_wrinkles(2,1:end);
Data_Adjust_on_Splits
```

```
figurename=strcat(analysis_data_name,'_processed');
saveas(gcf,figurename);
saveas(gcf,strcat(figurename,'.tif'));

close all
plot(Percent_wrinkles(1,:),Percent_wrinkles(2,:)*100);
saveas(gcf,strcat(analysis_data_name,'_Original_Data','.tiff'));
saveas(gcf,strcat(analysis_data_name,'_Original_Data','.fig'));

xlswrite(('MATLAB.xlsx'),cellstr('Average Frequency'),'Collection','B1');
xlswrite(('MATLAB.xlsx'),cellstr('Average Amplitude'),'Collection','C1');
xlswrite(('MATLAB.xlsx'),cellstr('Average Period'),'Collection','D1');
xlswrite(('MATLAB.xlsx'),cellstr('Peak Diff'),'Collection','E1');

if isempty (pks)
xlswrite(('MATLAB.xlsx'),0,'Collection','B2');
xlswrite(('MATLAB.xlsx'),0,'Collection','C2');
xlswrite(('MATLAB.xlsx'),0,'Collection','D2');
xlswrite(('MATLAB.xlsx'),0,'Collection','E2');

else
xlswrite(('MATLAB.xlsx'),frequency','Collection','B2');
xlswrite(('MATLAB.xlsx'),avgamp','Collection','C2');
xlswrite(('MATLAB.xlsx'),avg_period','Collection','D2');
xlswrite(('MATLAB.xlsx'),avg_peak_diff','Collection','E2');
end
%Outputing Peaks

if isempty(pks)
xlswrite(('MATLAB.xlsx'),0,'Collection','B6');
xlswrite(('MATLAB.xlsx'),0,'Collection','C6');

else
len_pks=length(pks);
cell_range_B=strcat('B6:','B',num2str(2+len_pks));
cell_range_C=strcat('C6:','C',num2str(2+len_pks));
xlswrite(('MATLAB.xlsx'),pks','Collection',cell_range_B);
xlswrite(('MATLAB.xlsx'),pks_time','Collection',cell_range_C);
end
clear time data Percent_wrinkles
end
end
end
end
```

Read_Tiff_Stack.m

```
function [I_g_find_edge,Percent_wrinkles,I_collect]=Read_Tiff_Stack(info,fname,tpf)

    for j=1:numel(info)
        I_indiv=imread(fname ,j,'info',info);

        if info(j).BitDepth==24
        I_g=rgb2gray(I_indiv);
        else
           I_g=I_indiv;
        end
        I_g=imadjust(I_g);
        I_g_find_edge=edge(I_g,'Sobel','nothinning');
        Percent_wrinkles(1,j)=(j-1)*tpf;
        Percent_wrinkles(2,j)=sum(sum(I_g_find_edge(:,:)))/numel(I_g_find_edge(:,:));
        I_collect(:,:,j)=I_g_find_edge;
    end

end
```

Data_Adjust_on_Splits.m

```
%Enter the name of the file in the first column, sheet name in the second
%column, and the location and range of the data in the third column. If the
%range is undetermined, for example, the B column, simply put 'B:B'

figure('units','normalized','outerposition',[0 0 1 1])
subplot(3,1,1)
title('Original Data');
hold on
plot(time, data)

[pks , pks_time,w_peaks]=findpeaks(data,time);
[vs, vs_time,w_vs]=findpeaks(-data,time);
vs=-vs;

if z==1 %turn on 30% of max peak prominence for peak selection
pks_criteria=0.3*max(pks);
vs_criteria=0.3*max(vs);
elseif z==2
    pks_criteria=0.50*max(pks);
vs_criteria=0.50*max(vs);
elseif z==3
    pks_criteria=avg_peaks;
    vs_criteria=avg_vs;
elseif z==4
    pks_criteria=0.70*max(pks);
vs_criteria=0.70*max(vs);
elseif z==5
    pks_criteria=0.95*max(pks);
vs_criteria=0.95*max(vs);
elseif z==6
    pks_criteria=0.97*max(pks);
vs_criteria=0.97*max(vs);
end


% Level Baseline to zero with spline function
%Ensure no high valley points that skew the base line
vs_new=[];
vs_time_new=[];
pk_diff_max=max(pks)-min(pks);
vs_diff_max=max(vs)-min(vs);

for x=1:length(vs)
```

```
if vs(x)<=(mean(data))
%Minus 10% from the mean of the data as the threshold
vs_new=[vs_new vs(x)];
vs_time_new=[vs_time_new vs_time(x)];
end
end

if length(vs_new)<=3
vs=vs;
vs_time=vs_time;
elseif pk_diff_max>=0.8
    if 0.7<=pk_diff_max/vs_diff_max<=1.5
    vs=vs;
    vs_time=vs_time;
    end
else
vs=vs_new;
vs_time=vs_time_new;
end

smoothing_spline=fit(vs_time',vs','smoothingspline');
plot(time,smoothing_spline(time))
detrenddata=data-smoothing_spline(time)';

[pks , pks_time,w_peaks]=findpeaks(detrenddata,time);
[vs, vs_time,w_vs]=findpeaks(-detrenddata,time);
vs=-vs;

if z==1 %turn on 30% of max peak prominence for peak selection
pks_criteria=0.3*max(pks);
vs_criteria=0.3*max(vs);
elseif z==2
    pks_criteria=0.50*max(pks);
vs_criteria=0.50*max(vs);
elseif z==3
    pks_criteria=avg_peaks;
    vs_criteria=avg_vs;
elseif z==4
    pks_criteria=0.70*max(pks);
vs_criteria=0.70*max(vs);
elseif z==5
    pks_criteria=0.95*max(pks);
vs_criteria=0.95*max(vs);
elseif z==6
    pks_criteria=0.97*max(pks);
vs_criteria=0.97*max(vs);
end


subplot(3,1,2)
title('Detrend Data')
plot(time, detrenddata); grid on;

% For calculating frequency. Use data without detrending to get more
% accurate peak counts
[pks_2 , pks_time_2,w_peaks_2]=findpeaks(data,time,'MinPeakProminence',pks_criteria);



[pks , pks_time,w_peaks]=findpeaks(detrenddata,time,'MinPeakProminence',pks_criteria);
%Need to have a peak prominence of at least 50% of the highest peak
%0.5*max(pks)-->Temporarilry substituted as avg_pks for now.
peaks=[pks_time;pks]; % Store indices of the peaks,time @ the indices, and the peak value
%from first row to the third row
avg_w_peaks=mean(w_peaks);

%deleted avg_w_peaks and threshold
[vs, vs_time,w_vs]=findpeaks(-detrenddata,time);
%Need to have a peak prominence of at least 50% of the highest peak-->Temporarilry substituted as
avg_vs for now
vs=-vs;
```

```
valleys=[vs_time;vs];
 vs_new=[];
vs_time_new=[];
data_gap=max(detrenddata)-min(detrenddata);

for x=1:length(vs)
if vs(x)<=(mean(detrenddata))
%Minus 10% from the mean of the data as the threshold
vs_new=[vs_new vs(x)];
vs_time_new=[vs_time_new vs_time(x)];
end
end

if length(vs_new)<=3
vs=vs;
vs_time=vs_time;
else
vs=vs_new;
vs_time=vs_time_new;
end

smoothing_spline=fit(vs_time',vs','smoothingspline');

%Calculate avg_ampltidue
min_baseline=(smoothing_spline(pks_time))';
avgamp=mean(pks-min_baseline);
% %Find the theoretical 3 max values
[sort_amp,sort_loc]=sort(pks-min_baseline,'descend');

if length(sort_loc)>=3
max_loc=sort_loc(1:3);
max_amp=sort_amp(1:3);
else
max_loc=sort_loc(1:length(sort_loc));
max_amp=sort_amp(1:length(sort_loc));
end
avgamp=mean(max_amp);

%Find the amplitutde from the old peak detection
index_pk=[];
for i_3=1:length(pks_time_2)
index_pk(i_3)=find(time==pks_time_2(i_3));
end

%Plot local max,mins, and a pair of max and min that gives largest ampltiude
subplot(3,1,3)
hold on
title('Original Data with Minima and Maxima')
plot(time,detrenddata)
plot(pks_time,pks,'go',pks_time_2,detrenddata(index_pk),'ro');
plot(time,smoothing_spline(time));

if length(pks_time)<3
    pks_time=pks_time_2;
    pks=detrenddata(index_pk);
end
%Finding Frequency
%Using average time span from one peak to ther other

avg_frequency = length(pks_2)/(time(end)-time(1)); %workfromhere
avg_period=1/avg_frequency;
period=[diff(pks_time)];
avg_period=mean(period);
peak_diff=[diff(pks)];
avg_peak_diff=mean(peak_diff);

frequency=avg_frequency;

str=strcat('  Avg Amp of 3 Highest Peaks (in green circles)= ', num2str(avgamp),' ','  Highest 
Ampltiude =', num2str(max_amp), '  Avg Freqnuecy = ',num2str(avg_frequency));
suptitle(str)
```