Appendix

Appendix 1- Script for Analyzing LabChart Data with MATLAB

```
function [deltaforce,mforce] =
FindChange(name,data,com,comt,comtext,samplerate,scaleoffse
t,scaleunits,offbefore,lengthbefore,offafter,lengthafter,Se
cIntervals)

colorplot =
['r','g','b','m','c','k','r','g','b','m','c','k','r','g','b
','m','c','k','r','g','b','m','c','k'];

dataG =
(data+(ones(1,length(data))*scaleoffset))*scaleunits;
figure('Position',[10,250,1900,500]);
plot(dataG,'color',[224 144 22]./255)
title(name);
hold on;

ticks = length(com);

for n=1:(ticks)

plot([com(n),com(n)],[min(dataG),max(dataG)],'color',colorp
lot(n));
end

msOffBefore = samplerate*offbefore;
msLengthBefore = samplerate*lengthbefore;

msOffAfter = samplerate*offafter;
msLengthAfter = samplerate*lengthafter;

ignorecount=0;

for n=1:ticks

    q=char(comtext(comt(n)));

    if q(1)=='$'
        ignorecount = ignorecount+1;
```

```
    end

    MStick = com(n);
    stB = MStick - msOffBefore - msLengthBefore;
    finB = MStick - msOffBefore;
    stA = MStick + msOffAfter;
    finA = MStick + msOffAfter + msLengthAfter;

    rectangle('Position',[stB,min(dataG),finB-
stB,range(dataG)],'edgecolor',colorplot(n))
    rectangle('Position',[stA,min(dataG),finA-
stA,range(dataG)],'edgecolor',colorplot(n))

    fontsize=12;

    if(rem(n,2)==0)
        ofsttext = 0.2;
    else
        ofsttext = 0;
    end

    textpos = 0.1;


text(com(n),textpos+ofsttext,comtext(comt(n)),'color',color
plot(n),'fontsize',fontsize)

    if q(1) ~= '$'

        mforce.titles(n-ignorecount)=comtext(comt(n));
        deltaforce.titles(n-ignorecount)=comtext(comt(n));

        %Analyze Change Rate of Force

        ppB = interp1([0:length(stB:finB)-
1],dataG(stB:finB),0:SecIntervals*samplerate:length(stB:fin
B)-1,'linear');
        ppA = interp1([0:length(stA:finA)-
1],dataG(stA:finA),0:SecIntervals*samplerate:length(stA:fin
A)-1,'linear');

        dppB = diff(ppB);
        dppA = diff(ppA);
```

```
        mforce.BeforeMax(n-ignorecount) = max(dppB);
        mforce.AfterMax(n-ignorecount) = max(dppA);
        mforce.BeforeMin(n-ignorecount) = min(dppB);
        mforce.AfterMin(n-ignorecount) = min(dppA);

        if(abs(mforce.BeforeMax(n-ignorecount)) >
abs(mforce.BeforeMin(n-ignorecount)))
            mforce.BeforeMagnitude(n-ignorecount) =
mforce.BeforeMax(n-ignorecount);
        elseif(abs(mforce.BeforeMax(n-ignorecount)) <
abs(mforce.BeforeMin(n-ignorecount)))
            mforce.BeforeMagnitude(n-ignorecount) =
mforce.BeforeMin(n-ignorecount);
        else
            mforce.BeforeMagnitude(n-ignorecount) = 0;
        end

        if(abs(mforce.AfterMax(n-ignorecount)) >
abs(mforce.AfterMin(n-ignorecount)))
            mforce.AfterMagnitude(n-ignorecount) =
mforce.AfterMax(n-ignorecount);
        elseif(abs(mforce.AfterMax(n-ignorecount)) <
abs(mforce.AfterMin(n-ignorecount)))
            mforce.AfterMagnitude(n-ignorecount) =
mforce.AfterMin(n-ignorecount);
        else
            mforce.AfterMagnitude(n-ignorecount) = 0;
        end

        mforce.BeforeMean(n-ignorecount) = mean(dppB);
        mforce.AfterMean(n-ignorecount) = mean(dppA);

        mforce.ChangeMean(n-ignorecount) =
mforce.AfterMean(n-ignorecount) - mforce.BeforeMean(n-
ignorecount);
        mforce.ChangeMag(n-ignorecount) =
mforce.AfterMagnitude(n-ignorecount) -
mforce.BeforeMagnitude(n-ignorecount);

        interceptB=mean(dataG((stB-500):(stB+500)));

plot([stB,finB],[interceptB,interceptB+(length(stB:finB)*(m
force.BeforeMean(n-
```

```
ignorecount)/(SecIntervals*samplerate)))],'color',colorplot
(n),'linewidth',3,'linestyle',':')

        interceptA=mean(dataG((stA-500):(stA+500)));

plot([stA,finA],[interceptA,interceptA+(length(stA:finA)*(m
force.AfterMean(n-
ignorecount)/(SecIntervals*samplerate)))],'color',colorplot
(n),'linewidth',3,'linestyle',':')

        %Max and Min Force over Area

        FIntervals =
length(MStick:finA)/(SecIntervals*samplerate);

        baselineForce = mean(dataG(stB:finB));

        for m=1:FIntervals
            ChForce(m) = mean(dataG((MStick+((m-
1)*(SecIntervals*samplerate))):(MStick+(m*(SecIntervals*sam
plerate)))));
            if ChForce(m)==max(ChForce)
                MaxMark = MStick+((m-
1)*(SecIntervals*samplerate));
            end
            if ChForce(m)==min(ChForce)
                MinMark = MStick+((m-
1)*(SecIntervals*samplerate));
            end
        end

        deltaforce.RangeMin(n-ignorecount) = min(ChForce) -
baselineForce;
        deltaforce.RangeMax(n-ignorecount) = max(ChForce) -
baselineForce;
        deltaforce.MinMar(n-ignorecount) = MinMark;
        deltaforce.MaxMar(n-ignorecount) = MaxMark;


plot([MaxMark,MaxMark],[min(dataG),max(dataG)],'color',colo
rplot(n),'linewidth',1,'linestyle','-.')

plot([MinMark,MinMark],[min(dataG),max(dataG)],'color',colo
rplot(n),'linewidth',1,'linestyle',':')
```

```
        clear ChForce;
        clear MaxMark;
        clear MinMark;
        deltaforce.SetRange(n-ignorecount) =
mean(dataG(stA:finA)) - baselineForce;

        mforce.StickMar(n-ignorecount) = MStick;
        mforce.MinMax(n-ignorecount) =
(dataG(deltaforce.MinMar(n-ignorecount))-
dataG(deltaforce.MaxMar(n-
ignorecount)))/((deltaforce.MinMar(n-ignorecount))-
(deltaforce.MaxMar(n-ignorecount)));
        mforce.StickMax(n-ignorecount) =
(dataG(deltaforce.MaxMar(n-ignorecount))-
dataG(MStick))/(deltaforce.MaxMar(n-ignorecount)-(MStick));
        mforce.StickMin(n-ignorecount) =
(dataG(deltaforce.MinMar(n-ignorecount))-
dataG(MStick))/(deltaforce.MinMar(n-ignorecount)-(MStick));


    end
end

xticks([0:600000:length(data)])
xticklabels({[0:600000:length(data)]/1000})
ylabel(['Force (g)']);
xlabel(['Seconds']);

limits = axis;
if limits(3) > 0
    axis([limits(1) limits(2) 0 limits(4)]);
end

for n=1:length(deltaforce.SetRange)
    if abs(deltaforce.RangeMin(n)) >
abs(deltaforce.RangeMax(n))
        deltaforce.RangeMag(n) = deltaforce.RangeMin(n);
    else
        deltaforce.RangeMag(n) = deltaforce.RangeMax(n);
    end
end

field1=fieldnames(deltaforce);
```

```
for(i=1:numel(field1))

deltaforce.(field1{i})=transpose(deltaforce.(field1{i}));
end

field2=fieldnames(mforce);
for(i=1:numel(field2))
    mforce.(field2{i})=transpose(mforce.(field2{i}));
end

Tdf = struct2table(deltaforce);
Tmf = struct2table(mforce);

writetable(Tdf,['df ' name '.csv'],'Delimiter',',');
writetable(Tmf,['mf ' name '.csv'],'Delimiter',',');
end
```

Appendix 2 – OpenSCAD Code for Organ Bath for Whole Colon

```
//UNITS - MICRONS
//1 cm = 10000 microns
//1 mm = 1000 microns

include <SPipe.scad>
include <ZPipe.scad>
include <JPipe.scad>

BATH_WIDTH = 38000;
BATH_LENGTH = 130000;
LUMEN_DEPTH = 38000;
BATH_FLOOR = 2000;
BATH_WALL = 2000;

BATH_FIXTURES = 20000;

TUBE_LUMEN = 4000;
TUBE_WALL = 1500;

LUMEN_RADIUS = (BATH_WIDTH/2)-BATH_WALL;
LUMEN_CENTER = BATH_LENGTH-2*BATH_WALL-2*LUMEN_RADIUS-
BATH_FIXTURES;

difference()
{
translate([-LUMEN_RADIUS-BATH_WALL,-LUMEN_RADIUS-
BATH_WALL,-LUMEN_DEPTH-BATH_FLOOR])
{
    color(alpha=0.1)
    cube([BATH_WIDTH,BATH_LENGTH,LUMEN_DEPTH+BATH_FLOOR]);
}

/* //INTERNAL CAVITY*/
//BATH INNER CHAMBER//
//ORIGIN SIDE
translate([BATH_WIDTH/4,-BATH_FIXTURES/2,-LUMEN_DEPTH])
{
    #cylinder(r=BATH_WIDTH/5.75, h=LUMEN_DEPTH/1.5);
    #cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*1.5);
}

translate([-BATH_WIDTH/4,-BATH_FIXTURES/2,-LUMEN_DEPTH])
{
```

```
    #cylinder(r=BATH_WIDTH/5.75, h=LUMEN_DEPTH/1.8);
    #cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*1.5);
}

translate([0,-BATH_FIXTURES/2,-LUMEN_DEPTH/1.1])
rotate([0,90,0])
cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH/1.8,center=true);

translate([-BATH_FIXTURES/1,-BATH_FIXTURES/2,-
LUMEN_DEPTH/2])
    rotate([0,90,0])
cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH/2,center=true);

//TRANSLATE SIDE
translate([0,BATH_LENGTH-BATH_FIXTURES,0]){
translate([BATH_WIDTH/4,-BATH_FIXTURES/2,-LUMEN_DEPTH])
{
    #cylinder(r=BATH_WIDTH/5.75, h=LUMEN_DEPTH/1.5);
    #cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*1.5);
}

translate([-BATH_WIDTH/4,-BATH_FIXTURES/2,-LUMEN_DEPTH])
{
    #cylinder(r=BATH_WIDTH/5.75, h=LUMEN_DEPTH/2);
    #cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*1.5);
}
    translate([0,-BATH_FIXTURES/2,-LUMEN_DEPTH/1.1])
rotate([0,90,0])
cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH/1.8,center=true);

    translate([BATH_FIXTURES/1,-BATH_FIXTURES/2,-
LUMEN_DEPTH/2])
    rotate([0,90,0])
cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH/1.8,center=true);
}

//CROSS BATH CONNECTOR
translate([-BATH_WIDTH/4-2*BATH_WALL,BATH_LENGTH-
BATH_FIXTURES*1.5,-LUMEN_DEPTH/1.1])
rotate([90,0,0])
cylinder(r=TUBE_LUMEN/2, h=BATH_LENGTH-BATH_FIXTURES);

translate([BATH_WIDTH/4+2*BATH_WALL,BATH_LENGTH-
BATH_FIXTURES*1.5,-LUMEN_DEPTH/1.1])
```

```
rotate([90,0,0])
cylinder(r=TUBE_LUMEN/2, h=BATH_LENGTH-BATH_FIXTURES);
//BATH INNER CHAMBER//


//TUB LUMEN
hull()
{
RES_LUMEN = 3;
translate([0,BATH_FIXTURES,0])
    {
    sphere(r=LUMEN_RADIUS,$fa=RES_LUMEN);
    }
translate([0,LUMEN_CENTER,0])
{
    sphere(r=LUMEN_RADIUS,$fa=RES_LUMEN);
}

translate([0,0,-LUMEN_DEPTH/2])
{
translate([0,BATH_FIXTURES,0])
    {
    sphere(r=LUMEN_RADIUS,$fa=RES_LUMEN);
    }
translate([0,LUMEN_CENTER,0])
{
    sphere(r=LUMEN_RADIUS,$fa=RES_LUMEN);
}
}
}

//

//PIPES

rotate([180,90,0]){
translate([LUMEN_DEPTH/2,BATH_FIXTURES/2-BATH_WALL,0])
{
ZPipe(RADIUS=2000,HEIGHT=20000,OUT=20000,ANGLE=45);

//Lower Angle Entries
//translate([5000-sin(45)*2000/2,-2000-sin(45)*2000,0])
//{ZPipe(RADIUS=2000,HEIGHT=10000,OUT=20000,ANGLE=45);}
//
```

```
//translate([10000-sin(45)*2000/2,-2000,0])
//{ZPipe(RADIUS=2000,HEIGHT=0,OUT=20000,ANGLE=45);}
}
}

//rotate([0,90,180]){
//translate([LUMEN_DEPTH/2,BATH_FIXTURES/2,0])
//{

//#SPipe(RADIUS=TUBE_LUMEN/2,HEIGHT=LUMEN_DEPTH/2,OUT=BATH_
FIXTURES);
//}
//}

mirror([0,1,0])
{
//rotate([0,90,180]){
//translate([LUMEN_DEPTH/2,BATH_LENGTH-
1.5*BATH_FIXTURES,0])
//{

//#SPipe(RADIUS=TUBE_LUMEN/2,HEIGHT=LUMEN_DEPTH/2,OUT=BATH_
FIXTURES);
//}
//}

rotate([180,90,0]){
translate([LUMEN_DEPTH/2,BATH_LENGTH-BATH_FIXTURES-
(BATH_FIXTURES/2-BATH_WALL),0])
{
ZPipe(RADIUS=2000,HEIGHT=20000,OUT=20000,ANGLE=45);

//Lower Angle Entries
//translate([5000-sin(45)*2000/2,-2000-sin(45)*2000,0])
//{ZPipe(RADIUS=2000,HEIGHT=10000,OUT=20000,ANGLE=45);}
//
//translate([10000-sin(45)*2000/2,-2000,0])
//{ZPipe(RADIUS=2000,HEIGHT=0,OUT=20000,ANGLE=45);}
}
}
}

//TOP INLET
translate([-BATH_WIDTH/5,-BATH_LENGTH/2,-LUMEN_DEPTH/4])
```

```
rotate([-90,0,0])
#cylinder(r=TUBE_LUMEN/2,h=BATH_LENGTH*2);

translate([00000,0,0]){
//GAS INLET v1
/* translate([-(BATH_WIDTH-BATH_WALL*2)/2,TUBE_LUMEN,-
LUMEN_DEPTH+TUBE_LUMEN])
rotate([0,90,0])
#cylinder(r=TUBE_LUMEN/2,h=BATH_WIDTH-BATH_WALL*2);

 //translate([-(BATH_WIDTH-BATH_WALL*2)/2,-
TUBE_LUMEN+BATH_LENGTH-BATH_FIXTURES*2,-
LUMEN_DEPTH+TUBE_LUMEN])
//rotate([0,90,0])
//#cylinder(r=TUBE_LUMEN/2,h=BATH_WIDTH-BATH_WALL*2);

translate([-BATH_WIDTH/2.6,0,-LUMEN_DEPTH+BATH_WALL*2])
{
for(p=[2:1:13]){
translate([0,LUMEN_RADIUS/2+5000*p,0])
rotate([0,75,0])
#cylinder(r=TUBE_LUMEN/5, h=BATH_WIDTH/2);
}

rotate([90,90,-90])
#JPipe(OR=TUBE_LUMEN/2,IR=0,HEIGHT=0,OUT=BATH_LENGTH-
BATH_FIXTURES*2.5);
}

mirror([[1,0,0]])
{
translate([-BATH_WIDTH/2.6,0,-LUMEN_DEPTH+BATH_WALL*2])
{
for(p=[4:1:11]){
translate([0,LUMEN_RADIUS/2+5000*p,0])
rotate([0,75,0])
#cylinder(r=TUBE_LUMEN/5, h=BATH_WIDTH/2);
}

rotate([90,90,-90])
#JPipe(OR=TUBE_LUMEN/2,IR=0,HEIGHT=LUMEN_DEPTH*2,OUT=BATH_L
ENGTH-BATH_FIXTURES*3);
}
} */
```

```
}

//VISIBILITY CUT
translate([0,-LUMEN_RADIUS,LUMEN_DEPTH])
{
    //cube([LUMEN_RADIUS,LUMEN_RADIUS,LUMEN_DEPTH]);
}

//EDGE
/* translate([BATH_WIDTH/2-4000,-BATH_FIXTURES+1000,-
LUMEN_DEPTH-BATH_FLOOR])
difference()
{
cube([4000,BATH_LENGTH+2000,4000]);
rotate([90,0,0])
translate([00,4000,-BATH_LENGTH-1500])
cylinder(r=4000,h=BATH_LENGTH+3000);
}

mirror([1,0,0])
{
translate([BATH_WIDTH/2-4000,-BATH_FIXTURES+1000,-
LUMEN_DEPTH-BATH_FLOOR])
difference()
{
cube([4000,BATH_LENGTH+2000,4000]);
rotate([90,0,0])
translate([00,4000,-BATH_LENGTH-1500])
cylinder(r=4000,h=BATH_LENGTH+3000);
}
} */

//GAS INLET V2 - HOLE


color("orange")
translate([-BATH_WIDTH/3-BATH_WALL/4,0,-
LUMEN_DEPTH/4+1000])
translate([0,0,-10000+500])
rotate([0,-90,0])
ZPipe(RADIUS=2000,HEIGHT=20000,OUT=BATH_LENGTH/3,ANGLE=45);
```

```
mirror([0,1,0]) {
color("orange")
translate([-BATH_WIDTH/3-BATH_WALL/4,-
BATH_LENGTH+BATH_FIXTURES*1.8,-LUMEN_DEPTH/4+1000])
translate([0,0,-10000+500])
rotate([0,-90,0])
ZPipe(RADIUS=2000,HEIGHT=20000,OUT=BATH_LENGTH/3,ANGLE=45);

}

}

//GAS INLET V2 - PIPE

difference(){
color("orange")
translate([-BATH_WIDTH/3-BATH_WALL/4,0,-
LUMEN_DEPTH/4+1000])
translate([0,0,-10000+500])
rotate([0,-90,0])
ZPipe(OR=2600,HEIGHT=20000,OUTB=BATH_LENGTH/2,OUTA=300,ANGL
E=45,IR=2000);

//Multiple Holes
/* for (x=[0:2500:50000])
{
translate([-BATH_WIDTH/3-
BATH_WALL/4,BATH_FIXTURES*1+BATH_WALL/2+x,-
2*LUMEN_DEPTH/3])
rotate([0,150,0])
#cylinder(r=500, h=26000);
} */

//Tube Gap
translate([-
BATH_WIDTH/3+BATH_WALL,BATH_FIXTURES*1+BATH_WALL/2,-
2*LUMEN_DEPTH/2.8])
rotate([-90,0,0])
#cylinder(r=3500, h=50000);

}
```

```
mirror([0,1,0]) {
color("orange")
translate([-BATH_WIDTH/3-BATH_WALL/4,-
BATH_LENGTH+BATH_FIXTURES*1.8,-LUMEN_DEPTH/4+1000])
translate([0,0,-10000+500])
rotate([0,-90,0])
ZPipe(OR=2600,HEIGHT=20000,OUTB=BATH_FIXTURES/4,OUTA=0,ANGL
E=45,IR=2000);

}

//RULER
rotate([0,10,0])
{
color("pink")
{
translate([-LUMEN_RADIUS/2,LUMEN_RADIUS+BATH_WALL*2,-
LUMEN_DEPTH+BATH_FLOOR+0000])
{
    for(l=[0:1000:50000])
    {

translate([0,l,0]){minkowski(){translate([0,150,0]){sphere(
r=200);} cube([5500,150,BATH_FLOOR+700]);}}
    }

    for(l=[0:10000:50000])
    {

translate([0,l,0]){minkowski(){translate([0,150,0]){sphere(
r=250);} cube([7500,150,BATH_FLOOR+700]);}}
    }
}
}
}

//VARIABLE HEIGHT OUTLET
difference(){
translate([LUMEN_RADIUS-
TUBE_LUMEN/2,LUMEN_RADIUS+TUBE_LUMEN,-LUMEN_DEPTH])
{
    cylinder(r=TUBE_WALL+TUBE_LUMEN/2, h=LUMEN_DEPTH);
}
```

```
translate([LUMEN_RADIUS-
TUBE_LUMEN/2,LUMEN_RADIUS+TUBE_LUMEN,-LUMEN_DEPTH])
{
    cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*2);
}

for(lh=[0:TUBE_LUMEN:LUMEN_DEPTH]){
translate([LUMEN_RADIUS-
TUBE_LUMEN*1,LUMEN_RADIUS+TUBE_LUMEN,lh-LUMEN_DEPTH])
    {#sphere(r=TUBE_LUMEN/2);}

translate([LUMEN_RADIUS-
TUBE_WALL,LUMEN_RADIUS+TUBE_LUMEN*1.5,lh-LUMEN_DEPTH-
TUBE_LUMEN/2])
    {#sphere(r=TUBE_LUMEN/2);}
}
translate([LUMEN_RADIUS-
TUBE_WALL,LUMEN_RADIUS+TUBE_LUMEN*0.25,-
LUMEN_DEPTH+TUBE_LUMEN/2])
    {
    #sphere(r=TUBE_LUMEN/2);
    }

    //Inner Chamber Hole
    translate([BATH_WIDTH/4+2*BATH_WALL,BATH_LENGTH-
BATH_FIXTURES*1.5,-LUMEN_DEPTH/1.1])
rotate([90,0,0])
cylinder(r=TUBE_LUMEN/2, h=BATH_LENGTH-BATH_FIXTURES);
}

difference(){
translate([LUMEN_RADIUS-TUBE_LUMEN/2,LUMEN_CENTER,-
LUMEN_DEPTH])
{
    cylinder(r=TUBE_WALL+TUBE_LUMEN/2, h=LUMEN_DEPTH);
}

translate([LUMEN_RADIUS-TUBE_LUMEN/2,LUMEN_CENTER,-
LUMEN_DEPTH])
{
    cylinder(r=TUBE_LUMEN/2, h=LUMEN_DEPTH*2);
}

for(lh=[0:TUBE_LUMEN:LUMEN_DEPTH]){
```

```
translate([LUMEN_RADIUS-TUBE_LUMEN*1,LUMEN_CENTER,lh-
LUMEN_DEPTH])
    {#sphere(r=TUBE_LUMEN/2);}

translate([LUMEN_RADIUS-TUBE_WALL,LUMEN_CENTER-
TUBE_LUMEN*0.5,lh-LUMEN_DEPTH-TUBE_LUMEN/2])
    {#sphere(r=TUBE_LUMEN/2);}
}
translate([LUMEN_RADIUS-
TUBE_WALL,LUMEN_RADIUS+TUBE_LUMEN*1.25,-
LUMEN_DEPTH+TUBE_LUMEN/2])
    {
    #sphere(r=TUBE_LUMEN/2);
    }

    //Inner Chamber Hole
    translate([BATH_WIDTH/4+2*BATH_WALL,BATH_LENGTH-
BATH_FIXTURES*1.5,-LUMEN_DEPTH/1.1])
rotate([90,0,0])
cylinder(r=TUBE_LUMEN/2, h=BATH_LENGTH-BATH_FIXTURES);
}
///

//2 cm Marker
translate([0,BATH_LENGTH/2-LUMEN_RADIUS-BATH_WALL,-
LUMEN_DEPTH+BATH_FLOOR])
{

        color("blue"){

        translate([10000,0,0]){
        cylinder(r=800,h=9000);
        }

        translate([-10000,0,0]){
        cylinder(r=800,h=9000);
            }
        }

}
```

Appendix 3 – Manual for STMaps Analysis Program and Code

Manual

Example Command Use:

[MapH,MapV,MapVLin,MapDyDt,Vars,Title,f,fw,fm]=STMaps('Video.wmv','Dimensions',[62 62*0.75],'MapArea',[Vars(1) Vars(2) Vars(3) Vars(4)],'MapHDim',[6000 0],'bgMax',0.999,'time',[9*60 (19*60)+10],'DyDt',[10000 3000],'LinearRange',1,'Group','Colon 1','Experiment','Chemical 10 mM','Timestamp','10 16 17');

Example Output:

[mapH,mapV,mapVLin,DyDtImgRaw,Vars,figTitle,f,fw,fm] = STMaps('vidFile')

mapH,mapV,mapVLin: Raw [H]orizontal/[V]ertical/[V]ertical [Lin]ear map data not in a graph or chart.

The variables on the left side of the equal will always be in this order, regardless of whether the variable name is changed or the data was suppressed by an argument.

DyDtImgRaw: Raw change in size over time horizontal map data.

Vars: Certain variables required for consistent analysis of other videos.

Vars=[cx(1),cy(1),cx(2),cy(2),lx(1),lx(2),mmWidth,mmHeight,bgMax,bgTol,tz(1),tz(2),bx(1),by(1),bx(2),by(2),mHMin,mHMax,mVMin,mVMax,DyDt,DyDtRange(1),DyDtRange(2)];

Vars(1:4) - See MapArea

Vars(5:6) - See LinearAnalysis [LeftLimit RightLimit]

Vars(7:8) - See Dimensions

Vars(9) - See BGMax

Vars(10) - See BgTolerance

Vars(11:12) - See Time

Vars(13:16) - See BGArea

Vars(17:20) - See MapHDim,MapVDim

Vars(21:23) - See DyDt

figTitle: Figure titles (and file names) without the specific indicators of what the data is (Horizontal, Vertical, DyDt, etc). See arguments 'Group', 'Experiment', 'Timestamp'.

f: Last video frame analyzed in grayscale.

fw: Last video frame analyzed as it was converted to maps (with arguments considered).

fm: Mask created for irregular shaped analysis.

Example Input Arguments

'vidFile': First argument is always the video file analyzed. This is required. All other arguments either require a declaration of what comes after it, or is a singular optional argument.

'suppressAutoContrast': Prevents black and white conversion of video frames using MATLAB's imbinarize function, used raw grayscale image to produce ST Maps.

'SuppressContrastDefect': By default, STMaps Uses MATLAB's imfill('holes') function to fill empty spaces in binarized image. Used for contrast errors in colon. This arguments suppresses that.

'Polygon': Select an irregular polygonal region to analyze, masking areas outside the region. Can be outside the initial rectangle for analysis but only areas of the mask inside the rectangle where analysis was set will be used to create ST maps. Click the first point with the last point to complete polynomial, and right-click and "create mask" to finish.

'Curve', [mm]: Select 3 points to create a quadratic curve (2nd degree polynomial) that will be the center of the mask, the two outer points will define the edges of the mask. "mm" specifies the diameter of the mask in mm. Equivalent to 'PolyCurve',[mm 3 2].

'PolyCurve',[mm points poly]: Like Curve, except you specify the number of points to select and the degree of polynomial to fit.

'LineMask, [mm]: Specify a 2 point line for the mask. May not be absolutely vertical.

'PiecewiseCurve', [mm points]: Specify a point-to-point line of points specified and mm width to analyze. Must be a function (two lines may not share the same vertical space).

'DefineMask',fm: Use a saved mask.

'colormap',[colormap]: Specify colormap for ST Map. Default is inverted grayscale or equivalent to matlab default colormaps flipud((gray()) (thin is light gray, wide is black). For list of built-in matlab colormaps, see https://www.mathworks.com/help/matlab/ref/colormap.html#buc3wsn-1-map

To flip a default colormap upsidedown, use flipud(), ex: flipud(jet()).

'LinearPlot': Makes a linear plot of the entire length of the image (averaging the thickness to a single point). Default is to select just a single point.

'suppressMapH','suppressMapV','suppressMapL': Suppresses the creation of [H]orizontal and/or [V]ertical ST Map, and/or the [L]inear plot showing the degree averaged side to side movement.

'MapHDim',[Min Max],'MapVDim',[Min Max]: Define the minimum and maximum width or length values on the colormap for the [H]orizontal or [V]ertical ST Map.

'Dimensions',[mmWidth mmHeight]: Define the width and height of the video in mm. Unspecified, the default is 60 mm x 45 mm.

'bgMax', [bgMax]: Specifies the sensitivity of separating the foreground from background. Default is 0.999 (extremely sensitive). If not set, program asks for area to determine background.

'bgTolerance', [bgTol]: Specifies the sensitivity of separating the foreground from background different from setting bgMax. Default is 1 (exactly to bgMax or as set by BGArea).

'BGArea',[TopLeftX TopLeftY BottomRightX BottomRightY]: Defines rectangle where background is determined.

'MapArea',[TopLeftX TopLeftY BottomRightX BottomRightY]: Defines rectangle in frames where ST Map is to be created.

'Time',[StartSecond StopSecond]: Defines start and end time in video for ST Map to be generated. Hint: to start at 45 seconds, and end at 20 minutes, 56 seconds, put 'Time',[45 (60*20)+56].

'LinearAnalysis',[Location] OR [LeftLimit RightLimit]: Specify the location (or range) where to look at the average vertical movement.

'LinearAnalysisRange',[Range]: Specify the distance in pixels from the location where average vertical movement is measured to measure linear analysis.

'DyDt',[DTime] OR [DTime ScaleRange] OR [DTime TopRange BottomRange]: Specify whether to do an ST Change Map and over what time the change is measured. The 2nd are 3rd numbers specify either +/- or a max and min for the color scale of the ST Change Map.

'Group','Group Name','Experiment','Experiment Name','Timestamp','Date': Adds the names and date to the ST Maps titles and files generated (for organization purposes, does not affect calculations). You don't have to follow the suggested format or specify either or all, but it will be added in the order of Group - Experiment - Timestamp.

Code

```
function
[mapH,mapV,mapVLin,DyDtImgRaw,Vars,LabelsLoc,figTitle,f,fw,
fwMask] = STMaps(varargin)
%STMAPS
%ST Map Generator by Adam Blakeney - 2017-2018
%Version 1.0
%For manual with full explanation for output and input
arguments, use command without arguments i.e. STMaps();
%Or see 'STMaps MATLAB Manual.txt' in the function
directory, or comments at the end of m file.
%Example Command Use:
%[MapH,MapV,MapVLin,MapDyDt,Vars,LabelsLoc,Title,f,fw]=STMa
ps('Video.wmv','Dimensions',[62 62*0.75],'MapArea',[Vars(1)
Vars(2) Vars(3) Vars(4)],'MapHDim',[0
6000],'bgMax',0.999,'time',[9*60 (19*60)+10],'DyDt',[10000
3000],'LinearRange',1,'Group','Colon
1','Experiment','Chemical 10 mM','Timestamp','10 16 17');

appname = 'ST Map Generator';
appversion = '1.0';
appnamever = [appname ' ' appversion];

if(nargin==0)
    STMapAbout(appnamever);
    return
end

dimText = 'Warning! Camera Dimensions not set. Defaulting
to 60 mm width, 45 mm height.';

na=nargin;

varargin = [varargin 'end'];

%Key word or character marker to ignore argument
argx = 'Ignore';
argcx = '$';

bgTol=1;
cx(1:2)=0;
bx(1:2)=0;
cy(1:2)=0;
by(1:2)=0;
```

```
lx(1:2)=0;
tz(1:2)=0;

rotAngle=0;

bgMax=0.999;
lxRange=0;
Linex = 0;

mapH=zeros(2,2);
mapV=zeros(2,2);
mapVLin=zeros(2,2);
DyDtImgRaw=zeros(2,2);

%mapVLinOrigin=zeros(2,2);

suppressTwoTone = 0;
FillContrastDefect = 1;
suppressMapH = 0;
suppressMapV = 0;
suppressMapL = 0;
bgMaxSet = 0;

mHSet = 0;
mHMax = 0;
mHMin = 0;

mVSet = 0;
mVMax = 0;
mVMin = 0;

mmWidth = 60;
mmHeight = 45;
dimWarning = 1;

colmap = [1:-1/63:0;1:-1/63:0;1:-1/63:0]';

vidTitle = '';
vidGroup = '';
vidExperiment = '';
vidTimestamp = '';
spaceT = {' '};
spaceG = '';
spaceE = '';
```

```
xAxLabel = '';
xAxUnits = '(mm)';
yAxLabel = '';
yAxUnits = '(Seconds)';
colLabel = '';
colUnits = '(\muM)';

vidFile = varargin{1};

STfigs=null(3);

VNorm = 0;

DyDt = 0;
DyDtRange(1:2) = 0;
DyDtRangeSet = 0;

LinearPlot = 0;

useMask = 0;
DefinedMask = 0;
RectMask = 0;
EllipseMask = 0;
PolyMask = 0;
CurveMask = 0;
FreehandMask = 0;
Piecewise = 0;
SetRotation = 0;
points=0;
poly=0;
mmMaskHt = 0;
xc=zeros(1,2);
yc=zeros(1,2);
endMask = 2;


%CheckDataImage Variable Defaults
AddLabels = 0;
NumLabs = 0;
LabelSet=0;
LabelsPredetermined = 0;
TimeLabels = 0;
HorizontalLabels = 0;
```

```
VerticalLabels = 0;
CheckLabels = '';
AxSet='xxxxxxxxxxxx';
LabelsLoc=zeros(2,2);
LabelLocations = 0;
LabelColors = [[1 0 0];[0 1 0];[0 0 1];[1 1 0];[1 0 1];[0 1
1];[1 0 0];[0 1 0];[0 0 1];[1 1 0];[1 0 1];[0 1 1]];
LabelWidths = [0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75
0.75 0.75 0.75];
LabelStyles = {'--','-.','--','-.','--','-.','--','-.','--
','-.','--','-.','--','-.','--','-.','--'};
Start = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
%%%

for(x = 1:na)
    if (ischar(varargin{x}) &&
~strcmpi(varargin{x}(1),argcx) &&
min(~strcmpi(varargin{x+1},argx)))
        if (strcmpi(varargin{x},'suppressAutoContrast'))
            suppressTwoTone = 1;
        end

        if (strcmpi(varargin{x},'SuppressContrastDefect')
|| strcmpi(varargin{x},'NoFillDefects'))
            FillContrastDefect = 0;
        end

        if (strcmpi(varargin{x},'colormap'))
            colmap = varargin{x+1};
        end

        if (strcmpi(varargin{x},'suppressMapH') ||
strcmpi(varargin{x},'suppressHMap'))
            suppressMapH = 1;
        end

        if (strcmpi(varargin{x},'suppressMapV') ||
strcmpi(varargin{x},'suppressVMap'))
            suppressMapV = 1;
        end

        if (strcmpi(varargin{x},'suppressMapL') ||
strcmpi(varargin{x},'suppressLMap'))
            suppressMapL = 1;
```

```
        end

        if (strcmpi(varargin{x},'Polygon'))
            useMask = 1;
            PolyMask = 1;
        end

        if (strcmpi(varargin{x},'Curve') ||
strcmpi(varargin{x},'CurveMask'))
            useMask = 1;
            CurveMask = 1;
            points=3;
            poly=2;
            mmMaskHt = varargin{x+1};
        end

        if (strcmpi(varargin{x},'LineMask') ||
strcmpi(varargin{x},'Line'))
            useMask = 1;
            CurveMask = 1;
            points=2;
            poly=1;
            mmMaskHt = varargin{x+1};
        end

        if (strcmpi(varargin{x},'PolyCurve') ||
strcmpi(varargin{x},'PolyCurveMask'))
            useMask = 1;
            CurveMask = 1;
            mmMaskHt = varargin{x+1}(1);
            points = varargin{x+1}(2);
            poly = varargin{x+1}(3);
        end

        if (strcmpi(varargin{x},'PiecewiseCurve') ||
strcmpi(varargin{x},'PiecewiseCurveMask') ||
strcmpi(varargin{x},'Piecewise'))
            useMask = 1;
            CurveMask = 1;
            Piecewise = 1;
            mmMaskHt = varargin{x+1}(1);
            points = varargin{x+1}(2);
            poly = 1;
        end
```

```
        if (strcmpi(varargin{x},'DefineMask'))
            useMask = 1;
            DefinedMask = 1;
            Dmask = varargin{x+1};
        end

        if (strcmpi(varargin{x},'LinearPlot') ||
strcmpi(varargin{x},'LinearMap'))
            LinearPlot = 1;
        end

        if (strcmpi(varargin{x},'MapHDim'))
            mHSet = 1;
            mHMin = (varargin{x+1}(1));
            mHMax = (varargin{x+1}(2));
        end

        if (strcmpi(varargin{x},'MapVDim'))
            mVSet = 1;
            mVMin = (varargin{x+1}(1));
            mVMax = (varargin{x+1}(2));
        end

        if (strcmpi(varargin{x},'dimensions'))
            mmWidth = varargin{x+1}(1);
            mmHeight = varargin{x+1}(2);
            dimWarning = 0;
        end

        if (strcmpi(varargin{x},'DimAspect'))
            mmWidth = varargin{x+1}(1);
            mmHeight = mmWidth*varargin{x+1}(2);
            dimWarning = 0;
        end

        if (strcmpi(varargin{x},'bgTolerance'))
            bgTol = varargin{x+1}(1);
        end

        if (strcmpi(varargin{x},'bgMax'))
            bgMax = varargin{x+1}(1);
            bgMaxSet = 1;
        end
```

```
        if (strcmpi(varargin{x},'MapArea'))
            if(length(varargin{x+1})==4)
                cx(1) = varargin{x+1}(1);
                cy(1) = varargin{x+1}(2);
                cx(2) = varargin{x+1}(3);
                cy(2) = varargin{x+1}(4);
            else
                error('Invalid ST Map Area Length. Specify
x1 y1 x2 y2 top left-hand and bottom right-hand corners:
`[],');
            end %if
        end

        if (strcmpi(varargin{x},'BGArea'))
            if(length(varargin{x+1})==4)
                bx(1) = varargin{x+1}(1);
                by(1) = varargin{x+1}(2);
                bx(2) = varargin{x+1}(3);
                by(2) = varargin{x+1}(4);
            else
                error('Invalid ST Map BG Area Length.
Specify x1 y1 x2 y2 top left-hand and bottom right-hand
corners: `[],');
            end %if
        end

        if (strcmpi(varargin{x},'Rotate'))
            if(isnumeric(varargin{x+1}))
                rotAngle = varargin{x+1};
            else
                SetRotation = 1;
            end
        end

        if (strcmpi(varargin{x},'time'))
            if(length(varargin{x+1})==2)
                tz(1) = varargin{x+1}(1);
                tz(2) = varargin{x+1}(2);
            elseif(length(varargin{x+1})==1)
                tz(2) = varargin{x+1};
            else
```

```
                error('Invalid time specification. Specify
either the start and finish second, or the seconds after
start.');
            end %if
        end

        if (strcmpi(varargin{x},'LinearAnalysisRange') ||
strcmpi(varargin{x},'LinearRange'))
            lxRange = varargin{x+1}(1);

            if (Linex ~= 0)
                lx(1) = Linex-lxRange;
                lx(2) = Linex+lxRange;
            end %if
        end

        if (strcmpi(varargin{x},'LinearAnalysis'))
            if(length(varargin{x+1})==1)
                Linex = varargin{x+1}(1);
                lx(1) = Linex-lxRange;
                lx(2) = Linex+lxRange;

            elseif(length(varargin{x+1})==2)
                lx(1) = varargin{x+1}(1);
                lx(2) = varargin{x+1}(2);
                Linex = round(mean([lx(1) lx(2)]));
            else
                error('Invalid Linear Analysis Length.
Specify a single column to analyze or two columns to
measure between.');
            end %if
        end

        if (strcmpi(varargin{x},'DyDt') ||
strcmpi(varargin{x},'FindChange'))
            if(length(varargin{x+1})==1)
                DyDt = varargin{x+1}(1);

            elseif(length(varargin{x+1})==2)
                DyDt = varargin{x+1}(1);
                DyDtRange(1) = varargin{x+1}(2);
                DyDtRange(2) = -1*varargin{x+1}(2);
                DyDtRangeSet = 1;
```

```
            elseif(length(varargin{x+1})==3)
                DyDt = varargin{x+1}(1);
                DyDtRange(1) = varargin{x+1}(2);
                DyDtRange(2) = varargin{x+1}(3);
                DyDtRangeSet = 1;

            else
                error('Invalid DyDt/FindChange
Specification. Specify at least a time in ms over which
changes are measured. A second number is +/- a range for
scale, and 3rd number is the - number in the range.');
            end %if
        end

        if (strcmpi(varargin{x},'Title'))
            vidTitle = varargin{x+1};
            spaceT = {' '};
        end

        if (strcmpi(varargin{x},'Group'))
            vidGroup = varargin{x+1};
            spaceG = {' '};
        end

        if (strcmpi(varargin{x},'Experiment'))
            vidExperiment = varargin{x+1};
            spaceE = {' '};
        end

        if (strcmpi(varargin{x},'Timestamp'))
            vidTimestamp = varargin{x+1};
        end

        if (strcmpi(varargin{x},'XAxisLabel'))
            xAxLabel = varargin{x+1};
        end

        if (strcmpi(varargin{x},'XAxisLabel'))
            xAxUnits = varargin{x+1};
        end

        if (strcmpi(varargin{x},'YAxisLabel'))
            yAxLabel = varargin{x+1};
        end
```

```
        if (strcmpi(varargin{x},'XAxisLabel'))
            yAxUnits = varargin{x+1};
        end

        if (strcmpi(varargin{x},'ColorLabel'))
            colLabel = varargin{x+1};
        end

        if (strcmpi(varargin{x},'ColorUnits'))
            colUnits = varargin{x+1};
        end

        if (strcmpi(varargin{x},'LabelVertical'))
            VerticalLabels = 1;
            AddLabels = 1;
            if(~exist('CheckLabels','var'))
                CheckLabels = (varargin{x+1});
            else
                CheckLabels =
[CheckLabels;(varargin{x+1})];
            end
        end

        if (strcmpi(varargin{x},'LabelHorizontal'))
            HorizontalLabels = 1;
            AddLabels = 1;
            if(~exist('CheckLabels','var'))
                CheckLabels = (varargin{x+1});
            else
                CheckLabels =
[CheckLabels;(varargin{x+1})];
            end
        end

        if (strcmpi(varargin{x},'LabelTimes') ||
strcmpi(varargin{x},'TimeLabels') ||
strcmpi(varargin{x},'LabelTime') ||
strcmpi(varargin{x},'TimeLabel'))
            TimeLabels = 1;

            LabelsLoc = [(varargin{x+1})]';

        end
```

```
        %CheckDataImage
        if (strcmpi(varargin{x},'SetAxis') ||
strcmpi(varargin{x},'AxisSet'))
            AxSet = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'CheckLabel') ||
strcmpi(varargin{x},'CheckLabels') ||
strcmpi(varargin{x},'AddLabels'))
            AddLabels = 1;
            HorizontalLabels = 1;
            VerticalLabels = 1;
            CheckLabels = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'LabelLocation') ||
strcmpi(varargin{x},'LabelLocations'))
            %LabelsPredetermined = 1;
            LabelsLoc = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'LabelColor') ||
strcmpi(varargin{x},'LabelColors'))
            LabelColors = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'LabelWidth') ||
strcmpi(varargin{x},'LabelWidths'))
            LabelWidths = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'LabelStyle') ||
strcmpi(varargin{x},'LabelStyles'))
            LabelStyles = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'Start'))
            Start = (varargin{x+1});
        end

        %%%

    end %ischar
```

```
end %for

%Initialize Video
vidObj = VideoReader(vidFile);
vid_Beginning = vidObj;
vidObj.CurrentTime = tz(1);

vidWidth = vidObj.Width;
vidHeight = vidObj.Height;

mmPixWid = mmWidth/vidWidth;
mmPixHt = mmHeight/vidHeight;

if(SetRotation == 1)
    FirstFrame = readFrame(vidObj);
    FirstFrame = rgb2gray(FirstFrame);
    figure;
    colormap(gray);
    imagesc(FirstFrame);
    title('Draw Line for New Baseline and Rotate Video');
    yticks([0:10/mmPixHt:vidHeight]);
    yticklabels({[0:1:mmHeight/10]});
    ylabel(['Height (cm)']);
    xticks([0:10/mmPixWid:vidWidth]);
    xticklabels({[0:1:mmWidth/10]});
    xlabel(['Length (cm)']);
    il=imline;
    ilp=getPosition(il);
    rotAngleRaw = 90+atand((ilp(1,1)-ilp(2,1))/(ilp(1,2)-
ilp(2,2)));

     if(rotAngleRaw > 90)
         rotAngle = -(rotAngleRaw-180);
     else
         rotAngle = -rotAngleRaw;
     end
end

if(rotAngle ~= 0 && abs(rotAngle) <= 45)
    mmWidthRot = mmWidth*cosd(double(rotAngle));
    mmHeightRot = mmHeight*sind(double(rotAngle));
end
```

```
if(rotAngle ~= 0 && abs(rotAngle) > 45 && abs(rotAngle) <
90)
    mmHeightRot = mmWidth*sind(abs(double(rotAngle))-45);
    mmWidthRot = mmHeight*cosd(abs(double(rotAngle))-45);
end

if(rotAngle == 90)
    mmHeightRot = mmWidth;
    mmWidthRot = mmHeight;
end

if(rotAngle == 0)
    mmHeightRot = mmHeight;
    mmWidthRot = mmWidth;
end

if(dimWarning == 1)
    disp(dimText);
else
    dimText = '';
end %if dimWarningl

figTitle =
strcat(vidTitle,spaceT,vidGroup,spaceG,vidExperiment,spaceE
,vidTimestamp);


NumLabs = length(CheckLabels);
LabelSet=zeros(1,NumLabs);


for(labels=1:NumLabs)
    if(LabelsLoc(labels,1) > 0 || LabelsLoc(labels,2) > 0)
        LabelSet(labels) = 1;
    end %end if LabelsLoc > 0
end %for labels

CreateStruct.Interpreter = 'tex';
CreateStruct.WindowStyle = 'non-modal';
STMapMsg=msgbox({
    appnamever;
    dimText; %Blank Space or Dimension Warning
    '[O] Reading Video - Please Wait';
```

```
    ['[!!] Select Comment Location ',int2str(labels),' of
',int2str(NumLabs),' for'];
    [AxSet(labels),'-Axis comment
',char(CheckLabels(labels))];
    '[ ] Select Top Lefthand Corner of Video for Map';
    '[ ] Select Bottom Righthand Corner of Video for Map';
    '[ ] Select Top Lefthand Corner of Video for
Background';
    '[ ] Select Bottom Righthand Corner of Video for
Background';
    '[ ] Select Cross-Section for Lateral Movement
Analysis';
    '[ ] Generating Maps'
    }, 'ST Map Generator',CreateStruct);
set(findobj(STMapMsg,'style','pushbutton'),'Visible','off')
;
set(STMapMsg,'Position',[50 400 250 200]);

pause(0.1);

vidFPS = vidObj.FrameRate;
vidSec = vidObj.Duration;

if(tz(2) == 0)
    tz(2) = vidSec;
end

vidFramesApprox = vidFPS*(tz(2)-tz(1))-1;

vidWidthRot = vidObj.Width;
vidHeightRot = vidObj.Height;

mmPixWid = mmWidthRot/vidWidthRot;
mmPixHt = mmHeightRot/vidHeightRot;


%Create Struct for individual frames
s =
struct('cdata',zeros(vidHeight,vidWidth,3,'uint16'),'colorm
ap',[]);

%Initialize Map Read Start
vidObj.CurrentTime = tz(1);
```

```
%Check Map Finish
if tz(2) == 0
    tz(2) = vidObj.Duration;
end

%Initialize Start for Map Times
if tz(1) ~= 0
    MapStart = tz(1)*vidFPS;
else
    MapStart = 1;
end

fw=zeros(vidHeight,vidWidth);
fwFilled=zeros(vidHeight,vidWidth,3);
fwRaw=zeros(vidHeight,vidWidth,3);
fwFill=zeros(vidHeight,vidWidth,3);
fwBG=zeros(vidHeight,vidWidth,3);
fwMask = ones(vidHeight,vidWidth);

if(rotAngle ~= 0)
   fw = imrotate_white(fw,double(rotAngle));
   fwFilled = imrotate_white(fwFilled,double(rotAngle));
   fwRaw = imrotate_white(fwRaw,double(rotAngle));
   fwFill = imrotate_white(fwFill,double(rotAngle));
   fwBG = imrotate_white(fwBG,double(rotAngle));
   fwMask = imrotate_white(fwMask,double(rotAngle));

   [vidHeightRot,vidWidthRot] = size(fw);
end


progressbar('Progress','Reading and Converting
Video','Video Contrast','Horizontal Map','Vertical
Map','Linear Plot');
k = 1;

while (vidObj.CurrentTime <= tz(2) && hasFrame(vidObj))
    s(k).cdata = readFrame(vidObj);
    if(rotAngle ~= 0)
        s(k).cdata =
imrotate_white(s(k).cdata,double(rotAngle));
    end
    if(k==1)
        f(:,:) = rgb2gray(s(k).cdata);
```

```
        set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
            appnamever;
            dimText; %Blank Space or Dimension Warning
            '[\surd] Reading Video';
            ['[ ] Select Comment Location
',int2str(labels),' of ',int2str(NumLabs),' for'];
            [AxSet(labels),'-Axis comment
',char(CheckLabels(labels))];
            '[ ] Select Top Lefthand Corner of Video for
Map';
            '[ ] Select Bottom Righthand Corner of Video
for Map';
            '[ ] Select Top Lefthand Corner of Video for
Background';
            '[ ] Select Bottom Righthand Corner of Video
for Background';
            '[ ] Select Cross-Section for Lateral Movement
Analysis';
            '[ ] Generating Maps'
            });

        %Input Map Area
        STfigs(1)=figure;
        colormap(gray);
        imagesc(f(:,:));
        title(figTitle);
        yticks([0:10/mmPixHt:vidHeightRot]);
        yticklabels({[0:1:mmHeightRot/10]});
        ylabel(['Height (cm)']);

        xticks([0:10/mmPixWid:vidWidthRot]);
        xticklabels({[0:1:mmWidthRot/10]});
        xlabel(['Length (cm)']);

        hold on;

        if(AddLabels == 1)
            NumLabs = length(CheckLabels);
            %AxSet(NumLabs+1) = 'x';

            for(labels=1:NumLabs)

set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
                    appnamever;
```

```
                    dimText; %Blank Space or Dimension
Warning
                    '[\surd] Reading Video';
                    ['[!!] Select Comment Location
',int2str(labels),' of ',int2str(NumLabs),' for'];
                    [AxSet(labels),'-Axis comment
',char(CheckLabels(labels))];
                    '[ ] Select Top Lefthand Corner of
Video for Map';
                    '[ ] Select Bottom Righthand Corner of
Video for Map';
                    '[ ] Select Top Lefthand Corner of
Video for Background';
                    '[ ] Select Bottom Righthand Corner of
Video for Background';
                    '[ ] Select Cross-Section for Lateral
Movement Analysis';
                    '[ ] Generating Maps'
                    });

                if(AxSet(labels) ~= 't')

                    if(LabelSet(labels) == 0)

[LabelsLoc(labels,1),LabelsLoc(labels,2)] = ginput(1);
                    end %if not LabelSet

                    if(AxSet(labels) == 'y')
                        line([0
vidWidth],[LabelsLoc(labels,2) LabelsLoc(labels,2)],
'Color',LabelColors(labels,:),'LineStyle',char(LabelStyles(
labels)),'LineWidth',LabelWidths(labels));
                    else
                        line([LabelsLoc(labels,1)
LabelsLoc(labels,1)],[0 vidHeight],
'Color',LabelColors(labels,:),'LineStyle',char(LabelStyles(
labels)),'LineWidth',LabelWidths(labels));
                    end
                end %if not time
            end %for labels
        end %if AddLabels

        set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
            appnamever;
```

```
            dimText; %Blank Space or Dimension Warning
            '[\surd] Reading Video';
            ['[\surd] Select Comment Location
',int2str(labels),' of ',int2str(NumLabs),' for'];
            [AxSet(labels),'-Axis comment
',char(CheckLabels(labels))];
            '[!!] Select Top Lefthand Corner of Video for
Map';
            '[ ] Select Bottom Righthand Corner of Video
for Map';
            '[ ] Select Top Lefthand Corner of Video for
Background';
            '[ ] Select Bottom Righthand Corner of Video
for Background';
            '[ ] Select Cross-Section for Lateral Movement
Analysis';
            '[ ] Generating Maps'
            });

        if(cx(1) == 0 && cy(1) == 0)
            [cx(1),cy(1)] = ginput(1);
        end %if

        line([cx(1) cx(1)],[0 vidHeightRot],
'Color','blue');
        line([0 vidWidthRot],[cy(1) cy(1)],
'Color','blue');

        set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
            appnamever;
            dimWarning; %Blank Space or Dimension Warning
            '[\surd] Reading Video';
            '[\surd] Select Top Lefthand Corner of Video
for Map';
            '[!!] Select Bottom Righthand Corner of Video
for Map';
            '[ ] Select Top Lefthand Corner of Video for
Background';
            '[ ] Select Bottom Righthand Corner of Video
for Background';
            '[ ] Select Cross-Section for Lateral Movement
Analysis';
            '[ ] Generating Maps'
            });
```

```
        if(cx(2) == 0 && cy(2) == 0)
            [cx(2),cy(2)] = ginput(1);
        end %if

        line([cx(2) cx(2)],[0 vidHeightRot],
'Color','blue');
        line([vidWidthRot 0],[cy(2) cy(2)],
'Color','blue');
        hold off;
        cx = uint16(cx);
        cy = uint16(cy);

        %Mask
        if(PolyMask == 1)


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
            appnamever;
            dimWarning; %Blank Space or Dimension Warning
            '[\surd] Reading Video';
            '[\surd] Select Top Lefthand Corner of Video
for Map';
            '[\surd] Select Bottom Righthand Corner of
Video for Map';
            '[!!] Select Points for Polygon Mask';
            '[ ] Select Top Lefthand Corner of Video for
Background';
            '[ ] Select Bottom Righthand Corner of Video
for Background';
            '[ ] Select Cross-Section for Lateral Movement
Analysis';
            '[ ] Generating Maps'
            });

            STfigs(1);
            fwMask = roipoly;
            endMask = 2;
            for (n=cx(1):1:cx(2))
                if (max(fwMask(:,n)) == 1)
                    xc(1)=round(n);
                    break
                end
            end %for
```

```
            for (n=cx(2):-1:cx(1))
                if (max(fwMask(:,n)) == 1)
                    xc(2)=round(n);
                    break
                end
            end %for
        end

        if(CurveMask == 1)


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
            appnamever;
            dimWarning; %Blank Space or Dimension Warning
            '[\surd] Reading Video';
            '[\surd] Select Top Lefthand Corner of Video
for Map';
            '[\surd] Select Bottom Righthand Corner of
Video for Map';
            '[!!] Select Points for Curve Mask';
            '[ ] Select Top Lefthand Corner of Video for
Background';
            '[ ] Select Bottom Righthand Corner of Video
for Background';
            '[ ] Select Cross-Section for Lateral Movement
Analysis';
            '[ ] Generating Maps'
            });

            STfigs(1);
            hold on;

            endMask=points;

            mmPixMaskHt = round(mmMaskHt/mmPixHt);

            xc=zeros(1,points);
            yc=zeros(1,points);
            for(n=1:points)
                [xc(n),yc(n)]=ginput(1);
                xc(n)=round(xc(n));
                yc(n)=round(yc(n));
                plot(xc(n),yc(n),'ro');
```

```
            end %for n=1:points

            if(Piecewise == 0)
                polyc = polyfit(xc,yc,poly);
                polyv =
polyval(polyc,round(xc(1)):1:round(xc(points)));
                plot(xc(1):1:xc(points),polyv,'k');
            end %piecewise == 0

            if(Piecewise == 1)
                polyv = double.empty(1,0);
                for(pw=1:1:points-1)
                    polyc =
polyfit(xc(pw:pw+1),yc(pw:pw+1),poly);
                    polyvs =
polyval(polyc,round(xc(pw)):1:round(xc(pw+1)));
                    polyv = [polyv polyvs];
                    plot(xc(pw):1:xc(pw+1),polyvs,'k');
                end %for

            end %Piecewise Curve



            fwMask = zeros(vidHeightRot,vidWidthRot);

            yn=1;
            for(xn=xc(1):1:xc(points))
                fwMask(round(polyv(yn)-
0.5*mmPixMaskHt):1:round(polyv(yn)+0.5*mmPixMaskHt),xn) =
1;
                yn=yn+1;
            end %for xn
        end

        if(DefinedMask == 1)
            fwMask = Dmask;
            endMask=2;

            for (n=cx(1):1:cx(2))
                if (max(fwMask(:,n)) == 1)
                    xc(1)=round(n);
                    break
                end
```

```
            end %for

            for (n=cx(2):-1:cx(1))
                if (max(fwMask(:,n)) == 1)
                    xc(2)=round(n);
                    break
                end
            end %for
        end

        %Array for length y of section
        yLengthArray = double(1:(cy(2)-cy(1)+1));

        %Input Sample Background Area

        %if(suppressTwoTone == 0)


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
                appnamever;
                dimText; %Blank Space or Dimension Warning
                '[\surd] Reading Video';
                '[\surd] Select Top Lefthand Corner of
Video for Map';
                '[\surd] Select Bottom Righthand Corner of
Video for Map';
                '[!!] Select Top Lefthand Corner of Video
for Background';
                '[ ] Select Bottom Righthand Corner of
Video for Background';
                '[ ] Select Cross-Section for Lateral
Movement Analysis';
                '[ ] Generating Maps'
                });

            pause(0.1);

            if(bx(1) == 0 && by(1) == 0 && bgMaxSet == 0)
                [bx(1),by(1)] = ginput(1);
            end %if

            hold on;
            line([bx(1) bx(1)],[0 vidHeightRot],
'Color','red');
```

```
            line([0 vidWidthRot],[by(1) by(1)],
'Color','red');


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
                appnamever;
                dimText; %Blank Space or Dimension Warning
                '[\surd] Reading Video';
                '[\surd] Select Top Lefthand Corner of
Video for Map';
                '[\surd] Select Bottom Righthand Corner of
Video for Map';
                '[\surd] Select Top Lefthand Corner of
Video for Background';
                '[!!] Select Bottom Righthand Corner of
Video for Background';
                '[ ] Select Cross-Section for Lateral
Movement Analysis';
                '[ ] Generating Maps'
                });

            if(bx(2) == 0 && by(2) == 0 && bgMaxSet == 0)
                [bx(2),by(2)] = ginput(1);
            end %if

            line([bx(2) bx(2)],[0 vidHeightRot],
'Color','red');
            line([vidWidthRot 0],[by(2) by(2)],
'Color','red');
            hold off;
            bx = uint16(bx);
            by = uint16(by);

            if(bgMaxSet == 0)
                bgMax =
max(max(f(by(1):by(2),bx(1):bx(1))));
            end %if

            %Sample Background Coloring
            %Convert Frame to Grayscale 0 to 1

                f(:,:) = rgb2gray(s(k).cdata);
                f = double(f)/255;
```

```
                %Create 2 Tone Image
            if (suppressTwoTone == 0)
                fwFilled(:,:,1) =
imfill(imcomplement(imbinarize(f,bgMax*bgTol)),'holes');
                fwRaw(:,:,3) =
imcomplement(imbinarize(f,bgMax*bgTol));
                fwFill(:,:,1) = fwFilled(:,:,1)-
fwRaw(:,:,3);
                fwBG(:,:,2) =
imcomplement(fwFilled(:,:,1));
            end %endif
                hold on;
                imagesc(fwFill,'AlphaData',0.2);
                imagesc(fwRaw,'AlphaData',0.2);
                imagesc(fwBG,'AlphaData',0.2);
                if(useMask == 1)
                    imagesc(fwMask*255,'AlphaData',0.2)
                end

            %%%

            %Lateral Movement Analysis


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
                appnamever;
                dimText; %Blank Space or Dimension Warning
                '[\surd] Reading Video';
                '[\surd] Select Top Lefthand Corner of
Video for Map';
                '[\surd] Select Bottom Righthand Corner of
Video for Map';
                '[\surd] Select Top Lefthand Corner of
Video for Background';
                '[\surd] Select Bottom Righthand Corner of
Video for Background';
                '[!!] Select Cross-Section for Lateral
Movement Analysis';
                '[ ] Generating Maps'
                });

            if(lx(1) == 0 && lx(2) == 0 && LinearPlot == 0)
                [Linex,~] = ginput(1);
                lx(1)=Linex-lxRange;
```

```
                lx(2)=Linex+lxRange;
            end %if

            hold on;
            line([Linex Linex],[0 vidHeightRot],
'Color','green');
            line([lx(1) lx(1)],[0 vidHeightRot],
'Color','green','LineStyle','-.');
            line([lx(2) lx(2)],[0 vidHeightRot],
'Color','green','LineStyle','-.');
            hold off;
            lx = uint16(lx);


set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
                appnamever;
                dimText; %Blank Space or Dimension Warning
                '[\surd] Reading Video';
                '[\surd] Select Top Lefthand Corner of
Video for Map';
                '[\surd] Select Bottom Righthand Corner of
Video for Map';
                '[\surd] Select Top Lefthand Corner of
Video for Background';
                '[\surd] Select Bottom Righthand Corner of
Video for Background';
                '[\surd] Select Cross-Section for Lateral
Movement Analysis';
                '[O] Generating Maps - Please Wait';
                });

            pause(0.1);

        %Initialize Array Size
        if (lx(2) == 0 && lx(1) > 0)
            lx(2) = lx(1);
        end %if LinearPlot

        %CreateMask Dimensions
        if (useMask == 1)
           cx(1) = xc(1);
           cx(2) = xc(endMask);
        end
```

```
        if LinearPlot == 1
            lx = cx;
        end %if LinearPlot

        mapH=zeros(round(k-tz(1)*vidFPS+1),cx(2)-cx(1)+1);
        mapV=zeros(cy(2)-cy(1)+1,round(k-tz(1)*vidFPS+1));
        mapVLin = zeros(lx(2)-lx(1)+1,round(vidFPS*(tz(2)-
tz(1))+1));

        %Scale Colors
        maxHt = mmPixHt*(abs(cy(2)-cy(1)));
        maxWid = mmPixWid*(abs(cx(2)-cx(1)));
    end %if k==1

    if(vidObj.CurrentTime <= tz(2))

        %Convert Frame to Grayscale 0 to 1
        f(:,:) = rgb2gray(s(k).cdata);
        f = double(f)/255;

        %Create 2 Tone Image
        if (suppressTwoTone == 1)
            fw=f;
        else
            if (FillContrastDefect == 1)
                fw =
imfill(imcomplement(imbinarize(f,bgMax*bgTol)),'holes');
            else
                fw =
imcomplement(imbinarize(f,bgMax*bgTol));
            end %FillContrastDefect
        end %endif

        %CreateMask
        if (useMask == 1)
           fw = fw.*fwMask;
        end

        fw=uint16(fw*65535);

        %Create Horizontal Map
        if suppressMapH == 0
            for (x=cx(1):cx(2))
```

```
                mapH(k,x-cx(1)+1) =
1000*maxHt*(mean(fw(cy(1):cy(2),x))/65535);
            end %endfor1
        end %SuppressHMap

        %Create Vertical Map
        if suppressMapV == 0
          for (y=cy(1):cy(2))
                mapV(y-cy(1)+1,k) =
1000*maxWid*(mean(fw(y,cx(1):cx(2)))/65535);
            end %endfor1
        end %SuppressVMap

        %Create Linear Plot

        if suppressMapL == 0
            for  (xl=lx(1):lx(2))
                yValsArray = fw(cy(1):cy(2),xl)/65535;
                mapVLin((xl-lx(1))+1,k) =
(yLengthArray*double(yValsArray))/double(sum(yValsArray));
            end
        end %if suppressMapL
    end %if k time
    k = k+1;
        progressbar(3/10,k/vidFramesApprox,[],[],[],[]);
end  %while vidObj.CurrentTime <= tz(2)
progressbar(7/10,[],1,1,1,1);
%Initialize Map End Time
MapEnd = k;

%Normalize Vertical Graph Position
VNorm = round(mean(mapVLin(1,:)));

if max(isnan(VNorm))==1
    disp("Error: Vertical graph normalization has an NaN
value. Averaging max and min.")
    VNorm = (max(mapVLin(1,:)) + min(mapVLin(1,:)))/2;
end %NaN

%Generate Figure Titles

if strcmp(figTitle,'')
    figTitle = 'ST Map';
end
```

```
set(findobj(STMapMsg,'Tag','MessageBox'),'String',{
    appnamever;
    dimText; %Blank Space or Dimension Warning
    '[\surd] Reading Video';
    '[\surd] Select Top Lefthand Corner of Video for Map';
    '[\surd] Select Bottom Righthand Corner of Video for
Map';
    '[\surd] Select Top Lefthand Corner of Video for
Background';
    '[\surd] Select Bottom Righthand Corner of Video for
Background';
    '[O] Generating Maps - Please Wait';
    });

pause(0.1);

%Set Label Locations

for(labels = 1:NumLabs)
    if (AxSet(labels) == 't')
        LabelLocations(labels) = round((LabelsLoc(labels)-
tz(1))*vidFPS);
    elseif (AxSet(labels) == 'y')
        LabelLocations(labels) = LabelsLoc(labels,1)-cy(1);
    else
        LabelLocations(labels) = LabelsLoc(labels,1)-cx(1);
    end
end

%Scale Maps
[Vy,Vx] = size(mapV);
[Hy,Hx] = size(mapH);

%Create Graphs

SecTickUnits = 10;
if (k > 1000)
    SecTickUnits=100;
elseif (k >= 500)
    SecTickUnits=50;
elseif (k >= 200)
    SecTickUnits=20;
elseif (k < 200)
```

```
    SecTickUnits=10;
end


if (suppressMapH == 0)

    if (mHSet == 0)
        mHMax = max(max(mapH));
        mHMin = min(min(mapH));
    end %if

    STfigs(2)=figure; imagesc(mapH,[mHMin mHMax]);
colormap(colmap);
    cH = colorbar;
    %cH.Label.String = 'Colon Diameter (\muM)';
    cH.Label.String = [colLabel ' ' colUnits];
    yticks([0:SecTickUnits*vidFPS:k]);
    yticklabels({[0:SecTickUnits*vidFPS:k]/vidFPS});
    ylabel([yAxLabel ' ' yAxUnits]);
    HMapTitle = strcat(figTitle,' Horizontal');
    set(gcf,'Position',[50 50, 550 700])
    title(HMapTitle);

    mmTickUnits = 10;
    if (Hx > 1000)
        mmTickUnits=100;
    elseif (Hx >= 800)
        mmTickUnits=50;
    elseif (Hx >= 400)
        mmTickUnits=20;
    elseif (Hx < 400)
        mmTickUnits=10;
    end

    xticks([0:mmTickUnits/mmPixWid:Hx]);
    xticklabels({[0:mmTickUnits/mmPixWid:Hx]*(mmPixWid)});
    xlabel([xAxLabel ' ' xAxUnits]);

    if (HorizontalLabels == 1)
        LabLocHor(1:NumLabs)=0;
        for(labels = 1:NumLabs)
            if (AxSet(labels) == 'y')
                LabLocHor(labels) = 0;
            elseif (AxSet(labels) == 't')
```

```
                LabLocHor(labels) = LabelLocations(labels);
            else
                LabLocHor(labels) = LabelLocations(labels);
            end
        end

        AxSetHor = strrep(AxSet,'t','y');

CheckDataImage(STfigs(2),'CheckLabels',CheckLabels,'SetAxis
',AxSetHor,'LabelLocations',LabLocHor,'LabelColors',LabelCo
lors,'LabelWidths',LabelWidths,'LabelStyles',LabelStyles,'S
tart',Start);
    end

    progressbar(8/10,[],[],[],[],[]);

    HMapPNG = strcat(HMapTitle,'.png');

    HMapEMF = strcat(HMapTitle,'.emf');

    try
        HMapEMFNS = strtrim(HMapEMF);
        saveas(gcf,string(HMapEMFNS));
    catch
        disp('An error occured durring the saving of the
Horizontal ST Map');
        disp(HMapEMFNS);
        disp('Saving as HMap.emf');
        try
            saveas(gcf,'HMap.emf');
        catch
            disp('An error occured durring the saving of
the Horizontal ST Map');
            disp('HMap.emf');
        end
    end

    try
        HMapPNGNS = strtrim(HMapPNG);
        saveas(gcf,string(HMapPNGNS));
    catch
        disp('An error occured durring the saving of the
Horizontal ST Map');
        disp(HMapPNGNS);
```

```
        disp('Saving as HMap.png');
        try
            saveas(gcf,'HMap.png');
        catch
            disp('An error occured durring the saving of
the Horizontal ST Map');
            disp('HMap.png');
        end
    end

end %if suppressMapH

if (suppressMapV == 0)
    if (mVSet == 0)
        mVMax = max(max(mapV));
        mVMin = min(min(mapV));
    end %if

    STfigs(3)=figure; imagesc(mapV-VNorm,[mVMin-VNorm
mVMax-VNorm]); colormap(colmap);
    yticks([0:(mmTickUnits/10)/mmPixHt:round(Vx)]);
    yticklabels({round((-
VNorm:(mmTickUnits/10)/mmPixHt:round(Vx-
VNorm))*(mmPixHt))});
    ylabel(['Vertical ' xAxLabel ' ' xAxUnits]);

    xticks([0:SecTickUnits*vidFPS:k]);
    xticklabels({[0:SecTickUnits*vidFPS:k]/vidFPS});
    xlabel([yAxLabel ' ' yAxUnits]);
    set(gcf,'Position',[70 70, 800 450])

    VMapTitle = strcat(figTitle,' Vertical');
    title(VMapTitle);

    if (VerticalLabels == 1)
        LabLocVer(1:NumLabs)=0;
        for(labels = 1:NumLabs)
            if (AxSet(labels) == 'x')
                LabLocVer(labels) = 0;
            elseif (AxSet(labels) == 't')
                LabLocVer(labels) = LabelLocations(labels);
            else
                LabLocVer(labels) = LabelLocations(labels)-
VNorm;
```

```
            end
        end

        AxSetVer = strrep(AxSet,'t','x');

CheckDataImage(STfigs(3),'CheckLabels',CheckLabels,'SetAxis
',AxSetVer,'LabelLocations',LabLocVer,'LabelColors',LabelCo
lors,'LabelWidths',LabelWidths,'LabelStyles',LabelStyles,'S
tart',Start);
    end

    VMapPNG = strcat(VMapTitle,'.png');

    VMapEMF = strcat(VMapTitle,'.emf');

    try
        VMapPNGNS = strtrim(VMapPNG);
        saveas(gcf,string(VMapPNGNS));
    catch
        disp('An error occured durring the saving of the
Vertical ST Map');
        disp(VMapPNGNS);
        disp('Saving as VMap.png');
        try
            saveas(gcf,'VMap.png');
        catch
            disp('An error occured durring the saving of
the Vertical ST Map');
            disp('VMap.png');
        end
    end

    try
        VMapEMFNS = strtrim(VMapEMF);
        saveas(gcf,string(VMapEMFNS));
    catch
        disp('An error occured durring the saving of the
Vertical ST Map');
        disp(VMapEMFNS);
        disp('Saving as VMap.emf');
        try
            saveas(gcf,'VMap.emf');
        catch
```

```
            disp('An error occured durring the saving of
the Vertical ST Map');
            disp('VMap.emf');
        end
    end

end %if suppressMapV

if (suppressMapL == 0)

    mapVLinDim = size(mapVLin);

    STfigs(4) = SliderDataPlot([mapVLin]'-
VNorm,'YAxisLabel','Amplitude','XAxisLabel','Time','Positio
n','Title',figTitle,round(mapVLinDim(1)/2));
    figure(STfigs(4));
    yticks([0:(mmTickUnits/10)/mmPixHt:round(Vx)]);
    yticklabels({round((-
VNorm:(mmTickUnits/10)/mmPixHt:round(Vx-
VNorm))*(mmPixHt))});
    ylabel(['Displacement ' xAxUnits]);

    xticks([0:SecTickUnits*vidFPS:k]);
    xticklabels({[0:SecTickUnits*vidFPS:k]/vidFPS});
    xlabel([yAxLabel ' ' yAxUnits]);

    LMapTitle = strcat(figTitle,' Amplitude');
    title(LMapTitle);

    LMapPNG = strcat(LMapTitle,'.png');

    LMapEMF = strcat(LMapTitle,'.emf');

    try
        LMapPNGNS = strtrim(LMapPNG);
        saveas(gcf,string(LMapPNGNS));
    catch
        disp('An error occured durring the saving of the
Amplitude Map');
        disp(LMapPNGNS);
        disp('Saving as LMap.png');
        try
            saveas(gcf,'LMap.png');
        catch
```

```
            disp('An error occured durring the saving of
the Amplitude Map');
            disp('LMap.png');
        end
    end

    try
        LMapEMFNS = strtrim(LMapEMF);
        saveas(gcf,string(LMapEMFNS));
    catch
        disp('An error occured durring the saving of the
Amplitude Map');
        disp(LMapEMFNS);
        disp('Saving as LMap.emf');
        try
            saveas(gcf,'LMap.emf');
        catch
            disp('An error occured durring the saving of
the Amplitude Map');
            disp('LMap.emf');
        end
    end
end %if suppressMapL

%DyDt Horizontal Map

if(DyDt > 0)
    DyDtImgRaw = DyDtImg(mapH,round(k/vidFPS),DyDt);

    if (DyDtRangeSet == 0)
        DyDtRange(1) = max(max(DyDtImgRaw));
        DyDtRange(2) = min(min(DyDtImgRaw));
    end %if

    STfigs(5)=figure; imagesc(DyDtImgRaw,[DyDtRange(2)
DyDtRange(1)]); colormap(hot);
    cHDyDt = colorbar;
    cHDyDt.Label.String = ['Change in ' colLabel ' '
colUnits];
    yticks([0:SecTickUnits*vidFPS:k]);
    yticklabels({[0:SecTickUnits*vidFPS:k]/vidFPS});
    ylabel([yAxLabel ' ' yAxUnits]);
    HDyDtMapTitle = strcat(figTitle,' Horizontal Dy/Dt');
    set(gcf,'Position',[55 55, 550 700])
```

```
    title(HDyDtMapTitle);

    if (suppressMapH == 1)
        mmTickUnits = 10;
        if (Hx > 1000)
            mmTickUnits=100;
        elseif (Hx >= 800)
            mmTickUnits=50;
        elseif (Hx >= 400)
            mmTickUnits=20;
        elseif (Hx < 400)
            mmTickUnits=10;
        end
    end %if suppressMapH

    xticks([0:mmTickUnits/mmPixWid:Hx]);
    xticklabels({[0:mmTickUnits/mmPixWid:Hx]*(mmPixWid)});
    xlabel([xAxLabel ' ' xAxUnits]);

    if (HorizontalLabels == 1)
        LabLocHor(1:NumLabs)=0;
        for(labels = 1:NumLabs)
            if (AxSet(labels) == 'y')
                LabLocHor(labels) = 0;
            elseif (AxSet(labels) == 't')
                LabLocHor(labels) = LabelLocations(labels);
            else
                LabLocHor(labels) = LabelLocations(labels);
            end
        end

        AxSetHor = strrep(AxSet,'t','y');

CheckDataImage(STfigs(5),'CheckLabels',CheckLabels,'SetAxis
',AxSetHor,'LabelLocations',LabLocHor,'LabelColors',LabelCo
lors,'LabelWidths',LabelWidths,'LabelStyles',LabelStyles,'S
tart',Start);
    end


    HDyDtMapFile = strcat(figTitle,' Horizontal DyDt');

    HDyDtMapPNG = strcat(HDyDtMapFile,'.png');
```

```
    HDyDtMapEMF = strcat(HDyDtMapFile,'.emf');

    try
        HDyDtMapEMFNS = strtrim(HDyDtMapEMF);
        saveas(gcf,string(HDyDtMapEMFNS));
    catch
        disp('An error occured durring the saving of the
Horizontal DyDt ST Map');
        disp(HDyDtMapEMFNS);
        disp('Saving as HDyDtMap.emf');
        try
            saveas(gcf,'HDyDtMap.emf');
        catch
            disp('An error occured durring the saving of
the Horizontal DyDt ST Map');
            disp('HDyDtMap.emf');
        end
    end

    try
        HDyDtMapPNGNS = strtrim(HDyDtMapPNG);
        saveas(gcf,string(HDyDtMapPNGNS));
    catch
        disp('An error occured durring the saving of the
Horizontal DyDt ST Map');
        disp(HDyDtMapPNGNS);
        disp('Saving as HDyDtMap.png');
        try
            saveas(gcf,'HDyDtMap.png');
        catch
            disp('An error occured durring the saving of
the Horizontal DyDt ST Map');
            disp('HDyDtMap.png');
        end
    end

end %if DyDt > 0

pause(0.1);

try
    figTitleSav = strcat(strtrim(figTitle),'.fig');

    savefig(STfigs,string(figTitleSav));
```

```
catch
    disp('An error occured durring the saving of the ST Map
figures');
    disp(figTitleSav);
    disp('Saving as figTitle.fig');
    try
        savefig(STfigs,'figTitle.fig');
    catch
        disp('An error occured durring the saving of the ST
Map figures');
        disp('figTitle.fig');
    end
end

pause(0.1);
%progressbar('Progress','Reading and Converting
Video','Creating Video Contrast','Creating Horizontal
Map','Creating Vertical Map','Creating Linear Plot');
progressbar(9/10,[],[],[],[],[]);
Vars=[cx(1),cy(1),cx(2),cy(2),lx(1),lx(2),mmWidth,mmHeight,
bgMax,bgTol,tz(1),tz(2),rotAngle,bx(1),by(1),bx(2),by(2)];

clear s STfigs appname appversion cH cHDyDt CreateStruct
dimText dimWarning HMap* LMap* VMap* MatSave space*;
delete(STMapMsg);
%Save MATLAB Variables File

try
    MatSaveTitle=strtrim(figTitle);
    MatSave=strcat(MatSaveTitle,'.mat');
    save(MatSave{1});
catch
    disp('An error occured durring the saving of the ST Map
MATLAB Variables');
    disp(MatSave);
    disp('Saving as MatSave.mat');
    try
        save('MatSave.mat');
    catch
        disp('An error occured durring the saving of the ST
Map MATLAB Variables');
        disp('MatSave.mat');
    end
end
```

```
progressbar(1);

%End Sound
Buzzer(0.9,1024*4,0.1,1000);pause(0.2);Buzzer(0.9,1024*4,0.
1,500);
end %main function

function [DyDtImgRaw]=DyDtImg (imgfile,MapTime,ThreshTime)


  [MapRowDim,MapColDim]=size(imgfile);
  EdgeImg = zeros(MapRowDim,MapColDim,3);

  mSecPerPx=(1000*MapTime)/MapRowDim;

  PxTimeThresh = round(ThreshTime/mSecPerPx);

  DyDtImgRaw = zeros(MapRowDim-PxTimeThresh,MapColDim);

  progressbar('Calculating DyDt');
  for y=1:1:MapRowDim-PxTimeThresh
    for x=1:1:MapColDim
      DyDtImgRaw(y,x) = imgfile(y,x)-
imgfile(y+PxTimeThresh,x);
    end %for
    progressbar(y/(MapRowDim-PxTimeThresh));
  end %for

end %function

function Buzzer(amp,fs,duration,freq)
values=0:1/fs:duration;
a=amp*sin(2*pi*freq*values);
sound(a, fs)
end

function STMapAbout(anv)
hlp = fileread('STMaps MATLAB Manual.txt');
hlp(hlp == char(10)) = '';
disp(anv);
disp(hlp);
end
```

```
function [wMap,wPlot,Titles,gx,gy,wCount,pm] =
WaveMap(varargin)

fNames = varargin{1};
ExpName = 'Wave Maps';
scale = 1;
mapscale = 0;
unitscale = 1;
plotscale = 0;
wCountSamps = 1;
wCount = 0;
SuppressFigures = 0;
PeaksOnly = 0;
wFigs = 0;
PeakSecs = 1;
PeakProm = 1;
PeakWidth=100;

%Colormap
R=[[ones(1,16)] [1-1/16:-1/16:0] [zeros(1,16)]
[1/16:1/16:1]]';
G=[[zeros(1,16)] [zeros(1,16)] [0:1/16:1-1/16]
[ones(1,16)]]';
B=[[1:-1/16:1/16] [zeros(1,16)] [zeros(1,16)]
[1/16:1/16:1]]';

cmap = [R G B];

varargin = [varargin 'end'];
argcx = '$';
argx = 'Ignore';
k=1;

for(x = 1:nargin)
    if (ischar(varargin{x}) &&
~strcmpi(varargin{x}(1),argcx) &&
min(~strcmpi(varargin{x+1},argx)))
        if (strcmpi(varargin{x},'SuppressFigures'))
            SuppressFigures = 1;
        end

        if (strcmpi(varargin{x},'PeaksOnly'))
            PeaksOnly = 1;
        end
```

```
        if (strcmpi(varargin{x},'MinPeakSeconds'))
            PeakSecs = varargin{x+1};
        end
        if (strcmpi(varargin{x},'MinPeakProminence'))
            PeakProm = varargin{x+1};
        end
        if (strcmpi(varargin{x},'MinPeakWidth'))
            PeakWidth = varargin{x+1};
        end

        if (strcmpi(varargin{x},'Peaks'))
            PeakSecs = varargin{x+1}(1);
            PeakProm = varargin{x+1}(2);
            PeakWidth = varargin{x+1}(3);
        end

        if (strcmpi(varargin{x},'ScaleData'))
            scale = varargin{x+1};
        end

        if (strcmpi(varargin{x},'MapScale'))
            mapscale = varargin{x+1};
        end

        if (strcmpi(varargin{x},'PlotScale'))
            plotscale = varargin{x+1};
        end

        if (strcmpi(varargin{x},'UnitScale'))
            unitscale = varargin{x+1};
        end

        if (strcmpi(varargin{x},'IntensifyData'))
            scale = 1/varargin{x+1};
        end

        if (strcmpi(varargin{x},'PeristalsisSamples') ||
strcmpi(varargin{x},'WaveSamples'))
            wCountSamps = varargin{x+1};
        end

        if (strcmpi(varargin{x},'ExperimentName'))
            ExpName = varargin{x+1};
```

```
        end

        if (strcmpi(varargin{x},'ColorMap'))
            cmap = varargin{x+1};
        end

        if (strcmpi(varargin{x},'GroupSizes'))
            GroupSizes = horzcat(0,varargin{x+1});
        end

        if (strcmpi(varargin{x},'GroupNames'))
            GroupNames = varargin{x+1};
        end

    end
end

fileload = 1;

if(fileload == 1)
    vnames =
{"LabLocHor";"mapH";"DyDtImgRaw";"mmPixWid";"vidFPS";"figTi
tle";"CheckLabels";"SecTickUnits";"mmTickUnits"};
    Samples=LoadVars(fNames,vnames);

end

for(n = 1:length(Samples))
    if(PeaksOnly == 0)
    [gx{n} gy{n}] =
gradient(imresize((Samples(n).DyDtImgRaw.DyDtImgRaw)/unitsc
ale,scale));

    %wMap = imresize(-gx{n} .* gy{n},1/scale);
    wMap{n} =
(sqrt(gx{n}.^2+gy{n}.^2).*(gx{n}.*gy{n})./abs(gx{n}.*gy{n})
)*-Samples(n).mmPixWid.mmPixWid*Samples(n).vidFPS.vidFPS;
    wMap{n} = wMap{n} * (scale^2);
    wMap{n}(isnan(wMap{n})) = 0;
    wMap{n} = imresize(wMap{n}, 1/scale);
    wPlot{n} = mean(wMap{n});

    if(SuppressFigures == 0)
        wFigs(k)=figure;
```

```
        imagesc(wMap{n});
        if(mapscale ~= 0)
            caxis(mapscale);
        end
        colormap(cmap);
        title([Samples(n).figTitle.figTitle 'Wave Direction
Map']);

        cH = colorbar;
        cH.Label.String = ['Relative Wave Velocity'];
        set(gcf,'Position',[50 50, 550 700])

yticks([0:Samples(n).SecTickUnits.SecTickUnits*Samples(n).v
idFPS.vidFPS:size(wMap{n},1)]);

yticklabels({[0:Samples(n).SecTickUnits.SecTickUnits*Sample
s(n).vidFPS.vidFPS:size(wMap{n},1)]/Samples(n).vidFPS.vidFP
S});
        ylabel('Time (Seconds)');


xticks([0:Samples(n).mmTickUnits.mmTickUnits/Samples(n).mmP
ixWid.mmPixWid:size(wMap{n},2)]);

xticklabels({[0:Samples(n).mmTickUnits.mmTickUnits/Samples(
n).mmPixWid.mmPixWid:size(wMap{n},2)]*(Samples(n).mmPixWid.
mmPixWid)});
        xlabel('Length (mm)');

    end
    end

    Titles{n} = Samples(n).figTitle.figTitle(1);

    if(SuppressFigures == 0 && PeaksOnly == 0)
        k=k+1;
        wFigs(k)=figure;
        plot(wPlot{n});
        if(plotscale ~= 0)
            ylim(plotscale);
        end
        title([Samples(n).figTitle.figTitle 'Relative
Average Velocity']);
        ylabel('Relative Average Velocity');
```

```
xticks([0:Samples(n).mmTickUnits.mmTickUnits/Samples(n).mmP
ixWid.mmPixWid:size(wMap,2)]);

xticklabels({[0:Samples(n).mmTickUnits.mmTickUnits/Samples(
n).mmPixWid.mmPixWid:size(wMap,2)]*(Samples(n).mmPixWid.mmP
ixWid)});
        xlabel('Length (mm)');
    end
    k=k+1;



pCountDiv=round(size(Samples(n).DyDtImgRaw.DyDtImgRaw,2)/(w
CountSamps+1));
    k=1;

for(x=1:pCountDiv:size(Samples(n).DyDtImgRaw.DyDtImgRaw,2)-
pCountDiv)
        pm =
movmean(abs(diff(Samples(n).DyDtImgRaw.DyDtImgRaw(:,x),2)),
500);
        [pmp
pml]=findpeaks(pm,'MinPeakProminence',PeakProm,'MinPeakWidt
h',PeakWidth,'MinPeakDistance',Samples(n).vidFPS.vidFPS*Pea
kSecs);
        pmc(k) = length(pmp);
        k=k+1;
    end

    wCount(n,1) = mean(pmc);
    wCount(n,2) = median(pmc);
    wCount(n,3) = mode(round(pmc));
    pmall =
movmean(abs(diff(mean(Samples(n).DyDtImgRaw.DyDtImgRaw,2)))
,500);
    wCount(n,4) =
size(findpeaks(pmall,'MinPeakProminence',0.5,'MinPeakWidth'
,200),1);

end %Samples

if(SuppressFigures == 0 && PeaksOnly == 0)
    figTitleSav = strcat(strtrim(ExpName),'.fig');
```

```
    savefig(wFigs,string(figTitleSav));
end


if(PeaksOnly == 1)
wMap = wCount;
wPlot = pm;
end

end
```

```
function
[mHm,mHmNorm,mHmGD,mHmGDNorm,xi,DyDtm,CoMNormDyDt,pctAboveV
al,mnDyDt,xiAll,DyDtmM,xiDyDtTh,NZDyDtTh,DyDtThSumI] =
PlotGroups(varargin)

fNames=varargin{1};
numFiles=length(fNames);

interpWid = 0;
OriginLoc = 1;
OriginName = 'Origin';
GroupSizes = [0 0];
GroupNames = {'Group 1','Group 2','Group 3','Group
4','Group 5','Group 6','Group 7','Group 8','Group 9','Group
10'};
Normalize = 0;
Pairwise = 0;
ExpName = 'Experiment';
WriteTable = 0;
DyDtm = {0};
CoMNormDyDt = 0;
regions = 0;
mnDyDt = 0;
DyDt=0;
Waves = 0;
PosValsOnly = 0;
NegValsOnly = 0;
ThreshVal = 0;
pctMean = 1;
ColorOrder = [[1 0 0];[0 1 0];[0 0 1];[1 1 0];[1 0 1];[0 1
1];[1 0 0];[0 1 0];[0 0 1];[1 1 0];[1 0 1];[0 1 1]];
Colors =
[ColorOrder;ColorOrder*0.65;ColorOrder*0.25;ColorOrder*0.15
];
GroupColors = Colors;
Widths = ones(1,numFiles+1)*0.75;
StyleOrder = {'-','--','-.'};
Styles = repelem(StyleOrder,numFiles+1);

for(x=1:nargin)
    if(ischar(varargin{x}))
        if (strcmpi(varargin{x},'Color') ||
strcmpi(varargin{x},'Colors'))
            Colors = varargin{x+1};
```

```
        end
        if (strcmpi(varargin{x},'GroupColor') ||
strcmpi(varargin{x},'GroupColors'))
            GroupColors = varargin{x+1};
        end
        if (strcmpi(varargin{x},'Width') ||
strcmpi(varargin{x},'Widths'))
            Widths = (varargin{x+1});
        end

        if (strcmpi(varargin{x},'Style') ||
strcmpi(varargin{x},'Styles'))
            Styles = (varargin{x+1});
        end
        if (strcmpi(varargin{x},'Normalize'))
            Normalize = 1;
        end
        if (strcmpi(varargin{x},'OriginName'))
            OriginName = varargin{x+1};
        end
        if (strcmpi(varargin{x},'InterpolationWidth'))
            interpWid = varargin{x+1};
        end
        if (strcmpi(varargin{x},'GroupSizes'))
            GroupSizes = horzcat(0,varargin{x+1});
        end
        if (strcmpi(varargin{x},'GroupNames'))
            GroupNames = varargin{x+1};
        end
        if (strcmpi(varargin{x},'ExperimentName'))
            ExpName = varargin{x+1};
        end
        if (strcmpi(varargin{x},'Pairwise'))
            Pairwise = 1;
        end
        if (strcmpi(varargin{x},'WriteTable'))
            WriteTable = 1;
        end
        if (strcmpi(varargin{x},'Regions'))
            regions = varargin{x+1};
        end
        if (strcmpi(varargin{x},'DyDt'))
            DyDt = 1;
        end
```

```
        if (strcmpi(varargin{x},'PositiveValuesOnly'))
            PosValsOnly = 1;
        end
        if (strcmpi(varargin{x},'NegativeValuesOnly'))
            NegValsOnly = 1;
        end
        if (strcmpi(varargin{x},'Waves'))
            Waves = 1;
        end
        if (strcmpi(varargin{x},'AllFigures'))
            DyDt = -1;
        end
        if (strcmpi(varargin{x},'ThresholdValue'))
            ThreshVal = varargin{x+1};
        end

        if (strcmpi(varargin{x},'PercentMean'))
            pctMean = varargin{x+1};
        end

    end %is char
end

vnames =
{"LabLocHor";"mapH";"DyDtImgRaw";"mmPixWid";"vidFPS";"figTi
tle";"CheckLabels"};
Samples=LoadVars(fNames,vnames);
[~,NumSamples] = size(Samples);
[~,NumGroups] = size(GroupSizes);
NumGroups=NumGroups-1;

GNVs = matlab.lang.makeValidName(GroupNames);
NumGST=length(GroupSizes)-1;

if(Waves == 1)
    [pMap] = WaveMap(fNames,'SuppressFigures');
    for(n=1:NumSamples)
        Samples(n).DyDtImgRaw.DyDtImgRaw=pMap{n};
    end
end

if(PosValsOnly == 1)
    for(n=1:NumSamples)
```

```
Samples(n).DyDtImgRaw.DyDtImgRaw=Samples(n).DyDtImgRaw.DyDt
ImgRaw.*(Samples(n).DyDtImgRaw.DyDtImgRaw>0);
    end
end

if(NegValsOnly == 1)
    for(n=1:NumSamples)

Samples(n).DyDtImgRaw.DyDtImgRaw=Samples(n).DyDtImgRaw.DyDt
ImgRaw.*(Samples(n).DyDtImgRaw.DyDtImgRaw<0);
    end
end

for(n=1:NumSamples)

    if(~strcmpi(OriginName,'None'))

[~,c]=find(strcmpi(OriginName,Samples(n).CheckLabels.CheckL
abels),1);
        OriginLoc = Samples(n).LabLocHor.LabLocHor(c);
    end

    if(strcmpi(OriginName,'None'))
        OriginLoc = 1;
    end

    %if(DyDt<1)

[mHm{n},xi{n},NZ{n}]=mapHmean(Samples(n).mapH.mapH,Samples(
n).mmPixWid.mmPixWid,interpWid,OriginLoc*Samples(n).mmPixWi
d.mmPixWid);

        if isempty(NZ{n})
            NZ{n} = 0;
        end

        xiZ{n}=round(xi{n}-NZ{n},6);
    %end

    if(abs(DyDt)>0)

[DyDtm{n},xiDyDt{n},NZDyDt{n}]=mapHmean(Samples(n).DyDtImgR
```

```
aw.DyDtImgRaw,Samples(n).mmPixWid.mmPixWid,interpWid,Origin
Loc*Samples(n).mmPixWid.mmPixWid);

        if isempty(NZDyDt{n})
            NZDyDt{n} = 0;
        end

        xiDyDtZ{n}=round(xiDyDt{n}-NZDyDt{n},6);
    end
end

if(interpWid~=0)
if(DyDt<1) %Diameter Group Means
if(GroupSizes(2) > 0)

[mHmGD,xiGD,GDmN,GDmS]=AverageGroups(mHm,xiZ,GroupSizes,int
erpWid);

    %Plot Diameter Groups
    figure;
    hold on;
    xlabel('Length (mm)');
    ylabel('Diameter (\mum)');

    for(n=1:NumGroups)
        mHmGDNorm{n} = mHmGD{n}-nanmean(mHmGD{n});
        plot(xiGD,mHmGD{n}-
(Normalize*nanmean(mHmGD{n})),'Color',GroupColors(n,:),'Lin
eWidth',Widths(n),'LineStyle',char(Styles(n)),'DisplayName'
,char(GroupNames(n)));
    end
    legend();
    line([0 0],[max(max(mHmGD{1}))*1.10
(min(min(mHmGD{1}))-
Normalize*nanmean(mHmGD{n}))*1.10],'DisplayName',OriginName
,'Color',[0 0 0]);

    %GroupData
    em=1;
    for(x=xiGD(1):(xiGD(2)-xiGD(1)):xiGD(length(xiGD)))
        for (NGS = 1:NumGST)
            grpstart = 1+sum(GroupSizes(1:NGS));
            grpfin = sum(GroupSizes(1:NGS+1));
```

```
            el=1;
            for(n=grpstart:grpfin)
                if(isfinite(mHm{n}(round(xi{n}-
NZ{n},6)==round(x,6))))

MeanD.(GNVs{NGS})(el,em)=mHm{n}(round(xi{n}-
NZ{n},6)==round(x,6));
                else
                    MeanD.(GNVs{NGS})(el,em)=NaN;
                end %if isfinite
                el=el+1;
            end %for grp

        end %for NGS
        em=em+1;
    end %for x
    %end GroupData

    %TTest
%     tn=1;
%     tg=1;
%     for(gt=1:NumGST-1)
%         for(oh=gt+1:NumGST)
%
[~,p(tn,:)]=ttest(MeanD.(GNVs{gt}),MeanD.(GNVs{oh}));
%             tcomp{tn} = [GNVs{gt} ' vs. ' GNVs{oh}];
%             tn=tn+1;
%         end
%         if(max(size(regions)) > 1)
%             [regs,~] = size(regions);
%             for(roh = 1:regs)
%
[~,gp(tg)]=ttest(reshape(MeanD.(GNVs{gt})(:,regions(:,1):re
gions(:,2)),[],1),reshape(MeanD.(GNVs{oh})(:,:,regions(:,1)
:regions(:,2)),[],1));
%                 tgcomp{tg} = [GNVs{gt} ' vs. ' GNVs{oh}];
%                 tg=tg+1;
%             end
%         end
%     end

%     for(ent=1:1:tn-1)
%         c=2.10+0.5*ent;
%         if(isfinite(xiGD(p(ent,:)<0.05)))
```

```
%
plot(xiGD(p(ent,:)<0.05),max(max(mHmGD{1}))*c,'+b','Display
Name',tcomp{ent});
%         end
%         if(isfinite(xiGD(p(ent,:)<0.01)))
%
plot(xiGD(p(ent,:)<0.01),max(max(mHmGD{1}))*c,'xm');
%         end
%         if(isfinite(xiGD(p(ent,:)<0.001)))
%
plot(xiGD(p(ent,:)<0.001),max(max(mHmGD{1}))*c,'*r');
%         end
%     end

%end TTest

end %if Group Means Diameter Processing
end %if Diameter


if(abs(DyDt)>0) %DyDt
if(GroupSizes(2) > 0)

[DyDtmM,xiAll,DyDtmN,DyDtmS]=AverageGroups(DyDtm,xiDyDtZ,Gr
oupSizes,interpWid);

    %Plot Groups
    figure;
    %GroupFigs(1)=figure;
    hold on;
    xlabel('Length (mm)');
    ylabel('Change Diameter (\mum)');

    if(Waves == 1)
        ylabel('Relative Velocity');
    end

    for(n=1:NumGroups)

plot(xiAll,DyDtmM{n},'Color',GroupColors(n,:),'LineWidth',W
idths(n),'LineStyle',char(Styles(n)),'DisplayName',char(Gro
upNames(n)));
    end
    legend();
```

```
    line([0 0],[max(max(DyDtmM{1}))*1.10
min(min(DyDtmM{1}))*1.10],'DisplayName',OriginName,'Color',
[0 0 0]);

    %GroupData
    dem=1;
    for(x=xiAll(1):(xiAll(2)-
xiAll(1)):xiAll(length(xiAll)))
        for (NGS = 1:NumGST)
            grpstart = 1+sum(GroupSizes(1:NGS));
            grpfin = sum(GroupSizes(1:NGS+1));

            del=1;
            for(n=grpstart:grpfin)
                if(isfinite(DyDtm{n}(round(xiDyDt{n}-
NZDyDt{n},6)==round(x,6))))

MeanDy.(GNVs{NGS})(del,dem)=DyDtm{n}(round(xiDyDt{n}-
NZDyDt{n},6)==round(x,6));
                else
                    MeanDy.(GNVs{NGS})(del,dem)=NaN;
                end %if isfinite
                del=del+1;
            end %for grp

        end %for NGS
        dem=dem+1;
    end %for x
    %end GroupData

    %TTest
    tn=1;
    tg=1;
    for(gt=1:NumGST-1)
        for(oh=gt+1:NumGST)

[~,dp(tn,:)]=ttest(MeanDy.(GNVs{gt}),MeanDy.(GNVs{oh}));
            dtcomp{tn} = [GNVs{gt} ' vs. ' GNVs{oh}];
            tn=tn+1;
        end

        if(max(size(regions)) > 1)
            [regs,~] = size(regions);
            for(roh = gt+1:NumGST)
```

```
                for(regc=1:regs)

[~,gdp(tg)]=ttest(reshape(MeanDy.(GNVs{gt})(:,find(xiAll==r
egions(regc,1)):find(xiAll==regions(regc,2))),[],1),reshape
(MeanDy.(GNVs{roh})(:,find(xiAll==regions(regc,1)):find(xiA
ll==regions(regc,2))),[],1));
                dtgcomp{tg} = [GNVs{gt} ' vs. ' GNVs{roh} '
Region ' int2str(regions(regc,1)) ' to '
int2str(regions(regc,2))];
                tg=tg+1;
                end
            end
        end
    end


end %if Group Means Processing
end %if DyDt Means
end %if interpWid

if(DyDt<1)
    %Diameter Plot
    figure;
    hold on;
    xlabel('Length (mm)');
    ylabel('Diameter (\mum)');

    for(n=1:NumSamples)
        mHmNorm{n} = mHm{n}-nanmean(mHm{n});
        plot(xi{n}-NZ{n},mHm{n}-
(Normalize*nanmean(mHm{n})),'Color',Colors(n,:),'LineWidth'
,Widths(n),'LineStyle',char(Styles(n)),'DisplayName',char(c
ellstr(Samples(n).figTitle.figTitle)));
    end
    legend()
    line([0 0],[max(max(mHm{1}-
(Normalize*nanmean(mHm{1}))))*1.10 min(min(mHm{1}-
(Normalize*nanmean(mHm{1}))))*1.10],'DisplayName',OriginNam
e,'Color',[0 0 0]);

    if(interpWid~=0)
    %GroupData for TTest
    em=1;
    for(x=xiAll(1):interpWid:xiAll(length(xiAll)))
```

```
        for (NGS = 1:NumGST)
            grpstart = 1+sum(GroupSizes(1:NGS));
            grpfin = sum(GroupSizes(1:NGS+1));

            el=1;
            for(n=grpstart:grpfin)
                if(isfinite(mHm{n}(round(xi{n}-
NZ{n},6)==round(x,6))))

MeanD.(GNVs{NGS})(el,em)=mHm{n}(round(xi{n}-
NZ{n},6)==round(x,6));
                else
                    MeanD.(GNVs{NGS})(el,em)=NaN;
                end %if isfinite
                el=el+1;
            end %for grp

        end %for NGS
        em=em+1;
    end %for x

    tn=1;
    tg=1;
    for(gt=1:NumGST-1)
        for(oh=gt+1:NumGST)

[~,p(tn,:)]=ttest(MeanD.(GNVs{gt}),MeanD.(GNVs{oh}));
            tcomp{tn} = [GNVs{gt} ' vs. ' GNVs{oh}];
            tn=tn+1;
        end
        if(max(size(regions)) > 1)
             [regs,~] = size(regions);
            for(roh = gt+1:NumGST)
                for(regc=1:regs)

[~,gp(tg)]=ttest(reshape(MeanD.(GNVs{gt})(:,find(xiAll==reg
ions(regc,1)):find(xiAll==regions(regc,2))),[],1),reshape(M
eanD.(GNVs{oh})(:,find(xiAll==regions(regc,1)):find(xiAll==
regions(regc,2))),[],1));
                tgcomp{tg} = [GNVs{gt} ' vs. ' GNVs{roh} '
Region ' int2str(regions(regc,1)) ' to '
int2str(regions(regc,2))];
                tg=tg+1;
                end
```

```
            end
        end
    end
    end %if interpWid
end %End Diameter Plot

if(abs(DyDt)>0) %DyDt Plot
    figure;
    hold on;
    xlabel('Length (mm)');
    ylabel('Change in Diameter (\mum)');

    if(Waves == 1)
        ylabel('Relative Velocity');
    end

    for(n=1:NumSamples)
        plot(xi{n}-
NZ{n},DyDtm{n},'Color',Colors(n,:),'LineWidth',Widths(n),'L
ineStyle',char(Styles(n)),'DisplayName',char(cellstr(Sample
s(n).figTitle.figTitle)));
    end
    legend()
    line([0 0],[max(max(DyDtm{1}))
min(min(DyDtm{1}))],'DisplayName',OriginName,'color',[0 0
0]);

    %Center of Mass and Above Mean DyDt
    if(ThreshVal > 0)

        for(n=1:NumSamples)

DyDtTh{n}=zeros(size(Samples(n).DyDtImgRaw.DyDtImgRaw));
            DyDtTh{n}(Samples(n).DyDtImgRaw.DyDtImgRaw >
ThreshVal)= 1;
            DyDtThSum{n}=sum(DyDtTh{n});


[DyDtThSumI{n},xiDyDtTh{n},NZDyDtTh{n}]=mapHmean(DyDtThSum{
n},Samples(n).mmPixWid.mmPixWid,interpWid,Samples(n).LabLoc
Hor.LabLocHor*Samples(n).mmPixWid.mmPixWid);

[pctAboveVal(n),rawAboveVal(n),Val(n),pctAboveMean(n),rawAb
oveMean(n),calcMean(n)] = AboveMean (DyDtThSum{n},pctMean);
```

```
[CoMNormDyDt(n),CoMRawDyDt(n),mnDyDt(n),peakDyDt(n),peakloc
DyDt(n)]=CenterOfMass(DyDtThSum{n});

            if(GroupSizes(2) > 0)

            end
        end

        %Center of Movement Plot
        figure;
        hold on;
        for(n=1:NumSamples)
        plot(xiDyDtTh{n}-
NZDyDtTh{n},DyDtThSumI{n},'Color',Colors(n,:),'LineWidth',W
idths(n),'LineStyle',char(Styles(n)),'DisplayName',char(cel
lstr(Samples(n).figTitle.figTitle)));
        end

        title('Center of Movement');
        legend();

        %Center of Movement Bar
        figure;
        barCenterMovement =
bar(CoMNormDyDt,'FaceColor','flat');

        for(n=1:NumSamples)
            barCenterMovement.CData(n,:)= Colors(n,:);
        end %for colors

        title('Center of Movement');

        %Center of Movement BoxPlot
        if(GroupSizes(2) > 0)
            GroupBoxNames =
repelem(GroupNames(1)',GroupSizes(2));
            for(gn=2:NumGroups)
            GroupBoxNames = [GroupBoxNames
repelem(GroupNames(gn)',GroupSizes(gn+1))];
            end

            figure;
```

```
            boxplot(CoMNormDyDt,GroupBoxNames);
            title('Center of Movement');
        end

        %Above Mean Bar
        figure;
        barCenterMovement =
bar(pctAboveVal,'FaceColor','flat');

        for(n=1:NumSamples)
            barCenterMovement.CData(n,:)= Colors(n,:);
        end %for colors
        pctText=round(100*pctMean);
        title(['Percent Above ',int2str(pctText),'% of
Mean']);

        %Above Mean BoxPlot
        if(GroupSizes(2) > 0)
            figure;
            boxplot(pctAboveVal,GroupBoxNames);
            title(['Percent Above ',int2str(pctText),'% of
Mean']);
        end

    end %Thresh Val
end %End DyDt

%Create Data Table

for (NGS = 1:NumGST)
    grpstart = 1+sum(GroupSizes(1:NGS));
    grpfin = sum(GroupSizes(1:NGS+1));
    CoMTable.(GNVs{NGS}) = CoMNormDyDt(grpstart:grpfin)';
    MeanTable.(GNVs{NGS}) = pctAboveVal(grpstart:grpfin)';
end

%Save Data
MatSave=strcat(ExpName,'.mat');
save(MatSave);

if(WriteTable == 1)
    CoMT = struct2table(CoMTable);
    MT = struct2table(MeanTable);
```

```
    writetable(CoMT,['Center of Mean ' ExpName
'.csv'],'Delimiter',',');
    writetable(MT,['Pct Above Mean ' ExpName
'.csv'],'Delimiter',',');


end%WriteTable
end %main function

function [Mg,X,N,S]=AverageGroups(m,xiZ,GS,interpWid)

NumSamples=length(m);
minX=round(min([xiZ{1:NumSamples}]),6);
maxX=round(max([xiZ{1:NumSamples}]),6);
X = round((minX:interpWid:maxX),6);
NumGS=length(GS)-1;

    for(ng=1:NumGS)
        grpstart = 1+sum(GS(1:ng));
        grpfin = sum(GS(1:ng+1));

        S{ng}=zeros(1,length(X));
        N{ng}=zeros(1,length(X));

        for(y=grpstart:grpfin)
            for(x=xiZ{y}(1:length(xiZ{y})))
                AllLoc=find(X==x);
                ZLoc=find(xiZ{y}==x);
                S{ng}(AllLoc) = S{ng}(AllLoc)+m{y}(ZLoc);
                N{ng}(AllLoc) = N{ng}(AllLoc)+1;
            end
        end
        Mg{ng} = S{ng}./N{ng};
    end

end

function [mapHmeanI,xmmi,NewZero] =
mapHmean(mapH,mmPixWid,interpWid,ZeroPoint)

mapHmean=mean(mapH,1);

if (interpWid==0)
    interpWid = mmPixWid;
```

```
end

NewZero=round(ZeroPoint/interpWid)*interpWid;

xmm = 0:mmPixWid:(size(mapHmean,2)-1)*mmPixWid;

xmmi = 0:interpWid:(size(mapHmean,2)-1)*mmPixWid;

if(interpWid==0)
    mapHmeanI = mapHmean;
else
    mapHmeanI=interp1(xmm,mapHmean,xmmi);
end

end
```