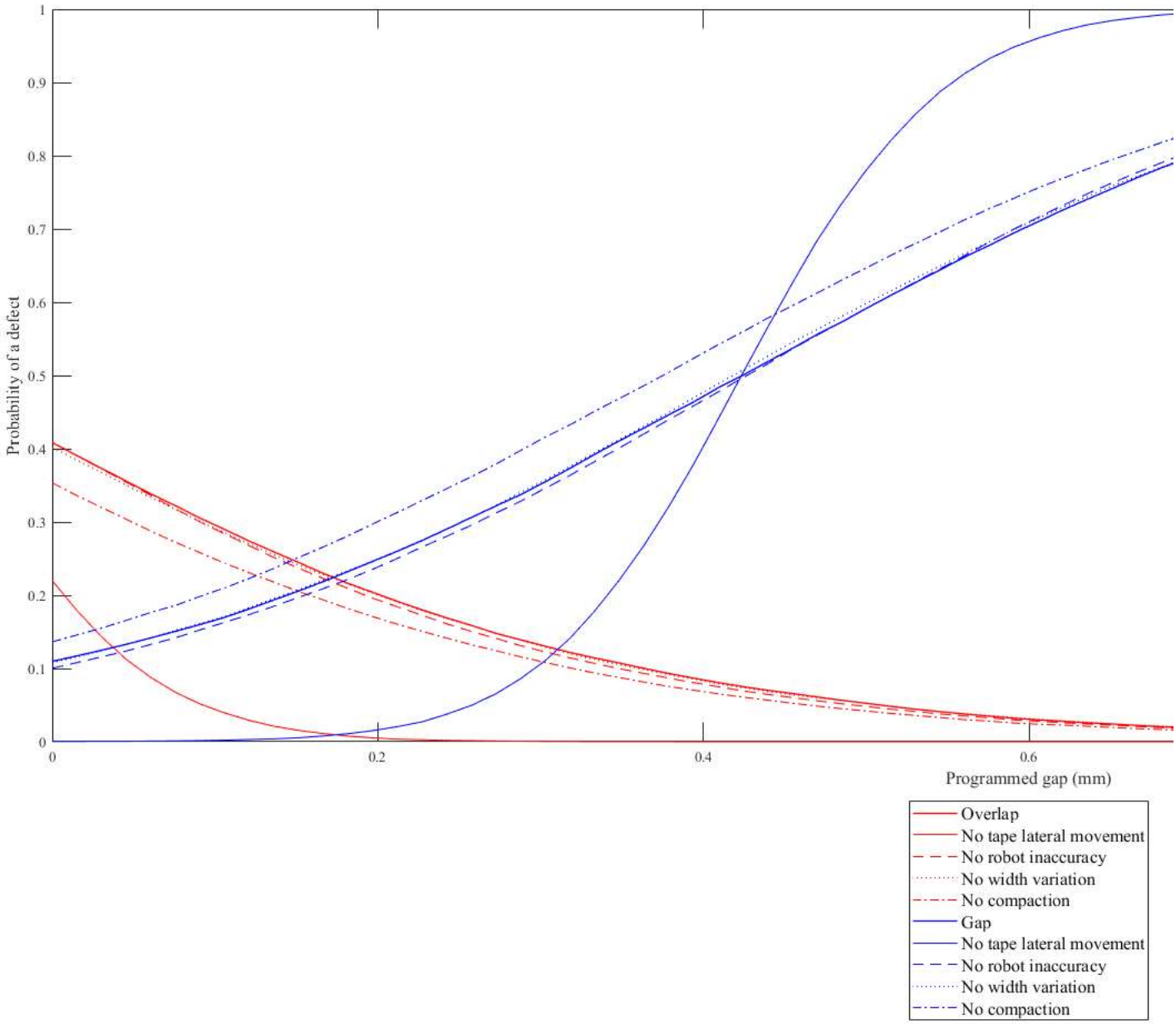
Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Code for global predictions of gaps and overlaps
% Author : Siddharth Pantoji @ TU Delft
% Description :
% This script is to use fused data from all 4 sensors and do statistical
% processing on Laminate level predictions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

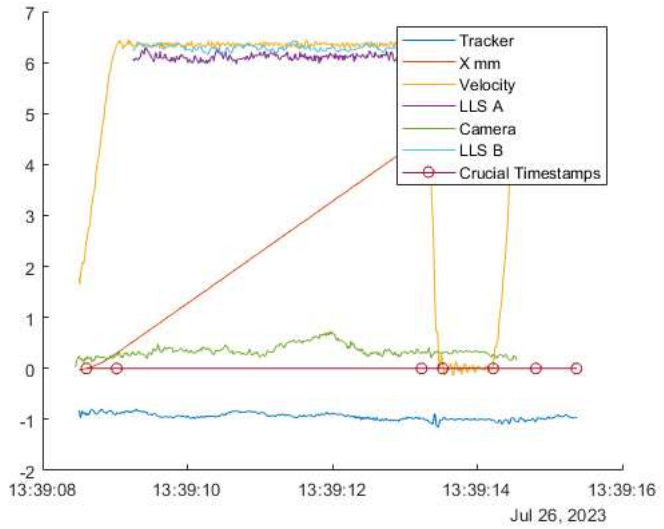
load neccesary data

```
clc
clear all
close all

% Get to storage folder for 4 sensors data
cd 'O:\Siddharth Experiment 31 straight lines 26th July\Manipulated timetables\Struct with all data'

% load all mats.
mat = dir('*.mat');
for q = 1:length(mat)
    load(mat(q).name);
end
clear q mat Data
```



Timestamps visualisation for a single run

```
%dont need everytime
i = 14;
figure(1);
hold on
plot(DataTimeTables(i).Tracker.TimeStamp,DataTimeTables(i).Tracker.Y_normalised);
plot(DataTimeTables(i).Tracker.TimeStamp,(DataTimeTables(i).Tracker.X_mm_/200));% mag reduced to visualise
plot(DataTimeTables(i).Tracker.TimeStamp,(DataTimeTables(i).Tracker.Velocity/31.5));% mag reduced to visualise
plot(DataTimeTables(i).LLS_A.Time,DataTimeTables(i).LLS_A.TapeWidth_Arc1);
plot(DataTimeTables(i).Camera.TimeStamps,DataTimeTables(i).Camera.TapeCenterLine);
plot(DataTimeTables(i).LLS_B.Time,DataTimeTables(i).LLS_B.TapeWidthAfterCompaction_Rotated);

x(1) = RunCrucialTimeStamps.RunStartTime(i);
x(2) = RunCrucialTimeStamps.PlatueauStartTime(i);
x(3) = RunCrucialTimeStamps.PlatueauEndTime(i);
x(4) = RunCrucialTimeStamps.CutOffStartTime(i);
x(5) = RunCrucialTimeStamps.CutOffEndTime(i);
x(6) = RunCrucialTimeStamps.Run1mEndTime(i);
x(7) = RunCrucialTimeStamps.RunEndTime(i);
plot(x,[0 0 0 0 0 0 0],'-o');

legend('Tracker','X mm','Velocity','LLS A','Camera','LLS B','Crucial Timestamps')

clear x
```



Adding velocity based weights to all tracker timetables

```
for i = 1:31
    clear weights smallNoIndex

    weights = DataTimeTables(i).Tracker.Velocity/200; %velocity/max velocity
    smallNoIndex = weights < 0.05; % removing very small and negative weights
    weights(smallNoIndex) = 0; %

    DataTimeTables(i).Tracker.Weights = weights;

end

clear weights smallNoIndex
```

Create joined tables for each run for using velocity based weights for all variables

```
for i =1:31

    clear a b c d
    a = DataTimeTables(i).Tracker;
    b = DataTimeTables(i).Camera;
    c = DataTimeTables(i).LLS_A;
    d = DataTimeTables(i).LLS_B;

    DataStreamsJoinedInterpolated(i).Run = synchronize(a,b,c,d,'union','linear');
    DataStreamsJoined(i).Run = synchronize(a,b,c,d,'union');
    DataStreamsJoined(i).Run.Weights = DataStreamsJoinedInterpolated(i).Run.Weights;
    DataStreamsJoined(i).Run.X_mm_ = DataStreamsJoinedInterpolated(i).Run.X_mm_;

    clear DataStreamsJoinedInterpolated a b c d i

end
```

Collecting 4 data vectors and respective weights while ignoring Nan lines

camera

```
for i = 1:31

    lineno = 1;
    clear TapeCenterLine Weights Time X_mm

    for j = 1 : height(DataStreamsJoined(i).Run)

        if isnan(DataStreamsJoined(i).Run.TapeCenterLine(j)) == 0;

        TapeCenterLine(lineno) = DataStreamsJoined(i).Run.TapeCenterLine(j);
        Weights(lineno) = DataStreamsJoined(i).Run.Weights(j);
        X_mm(lineno) = DataStreamsJoined(i).Run.X_mm_(j);
        Time(lineno) = convertTo(DataStreamsJoined(i).Run.TimeStamp(j),'excel');

        lineno = lineno + 1;
        end

    end

    Time = datetime(Time, 'ConvertFrom', 'excel', 'Format', 'MM/dd/yyyy HH:mm:ss.SSS');
```

```
    Camera_Tapelateralmovement = timetable(Time',X_mm',TapeCenterLine',Weights');
    Camera_Tapelateralmovement = renamevars(Camera_Tapelateralmovement,"Var1","X_mm");
    Camera_Tapelateralmovement = renamevars(Camera_Tapelateralmovement,"Var2","TapeCenterLine");
    Camera_Tapelateralmovement = renamevars(Camera_Tapelateralmovement,"Var3","Weights");

Camera_distilled(i).Run = Camera_Tapelateralmovement;

end

clear Camera_Tapelateralmovement i j lineno Time X_mm Weights TapeCenterLine
```

tracker

```
for i = 1:31

    lineno = 1;
    clear Y_normalised Weights Time X_mm

    for j = 1 : height(DataStreamsJoined(i).Run)

        if isnan(DataStreamsJoined(i).Run.Y_normalised(j)) == 0;

        Y_normalised(lineno) = DataStreamsJoined(i).Run.Y_normalised(j);
        Weights(lineno) = DataStreamsJoined(i).Run.Weights(j);
        X_mm(lineno) = DataStreamsJoined(i).Run.X_mm_(j);
        Time(lineno) = convertTo(DataStreamsJoined(i).Run.TimeStamp(j),'excel');

        lineno = lineno + 1;
        end

    end

    Time = datetime(Time, 'ConvertFrom', 'excel', 'Format', 'MM/dd/yyyy HH:mm:ss.SSS');
    Tracker_Y = timetable(Time',X_mm',Y_normalised',Weights');
    Tracker_Y = renamevars(Tracker_Y,"Var1","X_mm");
    Tracker_Y = renamevars(Tracker_Y,"Var2","Y_normalised");
    Tracker_Y = renamevars(Tracker_Y,"Var3","Weights");

Tracker_distilled(i).Run = Tracker_Y;

end

clear Tracker_Y i j lineno Time X_mm Weights Y_normalised
```

LLS A

```
for i = 1:31

    lineno = 1;
    clear TapeWidth1 TapeWidth_Arc1 Weights Time X_mm

    for j = 1 : height(DataStreamsJoined(i).Run)

        if isnan(DataStreamsJoined(i).Run.TapeWidth1(j)) == 0;

        TapeWidth1(lineno) = DataStreamsJoined(i).Run.TapeWidth1(j);
        TapeWidth_Arc1(lineno) = DataStreamsJoined(i).Run.TapeWidth_Arc1(j);
        Weights(lineno) = DataStreamsJoined(i).Run.Weights(j);
        X_mm(lineno) = DataStreamsJoined(i).Run.X_mm_(j);
        Time(lineno) = convertTo(DataStreamsJoined(i).Run.TimeStamp(j),'excel');

        lineno = lineno + 1;
        end

    end

    Time = datetime(Time, 'ConvertFrom', 'excel', 'Format', 'MM/dd/yyyy HH:mm:ss.SSS');
    LLSAWidth = timetable(Time',X_mm',TapeWidth_Arc1',Weights',TapeWidth1');
    LLSAWidth = renamevars(LLSAWidth,"Var1","X_mm");
    LLSAWidth = renamevars(LLSAWidth,"Var2","TapeWidth_Arc1");
    LLSAWidth = renamevars(LLSAWidth,"Var3","Weights");
    LLSAWidth = renamevars(LLSAWidth,"Var4","TapeWidth1");

LLSA_distilled(i).Run = LLSAWidth;

end

clear LLSAWidth i j lineno Time X_mm Weights TapeWidth1 TapeWidth_Arc1
```

LLS B

```
for i = 1:31
```

```
    lineno = 1;
    clear TapeWidthAfterCompaction TapeWidthAfterCompaction_Rotated Weights Time X_mm

    for j = 1 : height(DataStreamsJoined(i).Run)

        if isnan(DataStreamsJoined(i).Run.TapeWidthAfterCompaction(j)) == 0;

        TapeWidthAfterCompaction(lineno) = DataStreamsJoined(i).Run.TapeWidthAfterCompaction(j);
        TapeWidthAfterCompaction_Rotated(lineno) = DataStreamsJoined(i).Run.TapeWidthAfterCompaction_Rotated(j);
        Weights(lineno) = DataStreamsJoined(i).Run.Weights(j);
        X_mm(lineno) = DataStreamsJoined(i).Run.X_mm_(j);
        Time(lineno) = convertTo(DataStreamsJoined(i).Run.TimeStamp(j),'excel');

        lineno = lineno + 1;
        end

    end

    Time = datetime(Time, 'ConvertFrom', 'excel', 'Format', 'MM/dd/yyyy HH:mm:ss.SSS');
    LLSBWidth = timetable(Time',X_mm',TapeWidthAfterCompaction_Rotated',Weights',TapeWidthAfterCompaction');
    LLSBWidth = renamevars(LLSBWidth,"Var1","X_mm");
    LLSBWidth = renamevars(LLSBWidth,"Var2","TapeWidthAfterCompaction_Rotated");
    LLSBWidth = renamevars(LLSBWidth,"Var3","Weights");
    LLSBWidth = renamevars(LLSBWidth,"Var4","TapeWidthAfterCompaction");

LLSB_distilled(i).Run = LLSBWidth;

end

clear LLSBWidth i j lineno Time X_mm Weights TapeWidthAfterCompaction TapeWidthAfterCompaction_Rotated
```

For joined tables, indices have changed (Done once no need to redo)

finding more accurate index for 0 and 1000 mm X For LLS B though the run start is 1m start Ans Run end is run 1 m end

```
% for i = 1 : 31;
%     flag = 0;
%     clear RunX
%         RunX = DataStreamsJoined(i).Run.X_mm_;
%         for j = 1 : length(RunX)
%              if RunX(j) >= 0 && flag == 0;
% %                 Run1mStartIndex_JoinedTT(i) = j;
%                 Run1mStartTime_JoinedTT(i) = DataStreamsJoined(i).Run.TimeStamp(j);
%                 flag = 1;
%              end
%         end
% end
%
% for i = 1 : 31;
%     flag = 0;
%     clear RunX
%         RunX = DataStreamsJoined(i).Run.X_mm_;
%         for j = 1 : length(RunX)
%              if RunX(j) >= 1000 && flag == 0;
% %                 Run1mEndIndex_JoinedTT(i) = j;
%                 Run1mEndTime_JoinedTT(i) = DataStreamsJoined(i).Run.TimeStamp(j);
%                 flag = 1;
%                 j
%              end
%         end
% end
%
% RunCrucialTimeStamps.RunStartTime = Run1mStartTime_JoinedTT';
% RunCrucialTimeStamps.Run1mEndTime = Run1mEndTime_JoinedTT';

% save RunCrucialTimeStamps.mat RunCrucialTimeStamps
```

Creating 4 arrays (unbiased along length)

Sampling without replacement (observation is selected at most once. Never reselected)

```
% tracker
% (sampling by weights based on velocity)

Tracker31joined =[];

for i = 1:31;

    clear temp temptable
    OneM_timerange = timerange...
        (RunCrucialTimeStamps.RunStartTime(i),RunCrucialTimeStamps.Run1mEndTime(i));
    temptable = Tracker_distilled(i).Run(OneM_timerange,:);
    samplesize = 0.87 * height(temptable); %avg velocity related metric
    weights = Tracker_distilled(i).Run.Weights(OneM_timerange,:);
```

```
    [Sample_from_temp,ids] = datasample(temptable,round(samplesize),'Replace',false,'Weights',weights);
    Sample_from_temp = sortrows(Sample_from_temp);
    Tracker31joined = [Tracker31joined;Sample_from_temp];

end

% figure;plot(Tracker31joined.Y_normalised,'o');
% figure;histogram(Tracker31joined.Y_normalised,100);
% figure;plot(Sample_from_temp.Time,Sample_from_temp.Weights,'o')

clear OneM_timerange weights i ids samplesize Sample_from_temp temptable
```

camera

(sampling by weights based on velocity)

```
Camera31joined =[];

for i = 1:31;

    clear temp temptable
    OneM_timerange = timerange...
        (RunCrucialTimeStamps.RunStartTime(i),RunCrucialTimeStamps.Run1mEndTime(i));
    temptable = Camera_distilled(i).Run(OneM_timerange,:);
    samplesize = 0.87 * height(temptable); %avg velocity related metric
    weights = Camera_distilled(i).Run.Weights(OneM_timerange,:);
    [Sample_from_temp,ids] = datasample(temptable,round(samplesize),'Replace',false,'Weights',weights);
    Sample_from_temp = sortrows(Sample_from_temp);
    Camera31joined = [Camera31joined;Sample_from_temp];

end

% figure;plot(Camera31joined.TapeCenterLine,'o');
% figure;histogram(Camera31joined.TapeCenterLine,100);
% figure;plot(Sample_from_temp.Time,Sample_from_temp.Weights,'o')

clear OneM_timerange weights i ids samplesize Sample_from_temp temptable
```

LLS A

(sampling by weights based on velocity)

```
LLS_A31joined =[];

for i = 1:31;

    clear temp temptable
    OneM_timerange = timerange...
        (RunCrucialTimeStamps.RunStartTime(i),RunCrucialTimeStamps.Run1mEndTime(i));
    temptable = LLSA_distilled(i).Run(OneM_timerange,:);
    samplesize = 0.86 * height(temptable); %avg velocity related metric
    weights = LLSA_distilled(i).Run.Weights(OneM_timerange,:);
    [Sample_from_temp,ids] = datasample(temptable,round(samplesize),'Replace',false,'Weights',weights);
    Sample_from_temp = sortrows(Sample_from_temp);
    LLS_A31joined = [LLS_A31joined;Sample_from_temp];

end

% figure;plot(LLS_A31joined.TapeWidth_Arc1,'o');
% figure;histogram(LLS_A31joined.TapeWidth_Arc1);
% figure;plot(Sample_from_temp.Time,Sample_from_temp.Weights,'o')

clear OneM_timerange weights i ids samplesize Sample_from_temp temptable
```

LLS B

For LLS B the tracker starts recording profiles at X ~= 100 and continues to record till X ~= 1100

```
%Thus the timestamps for range are runstart and run end (not run1Mend) %***

LLS_B31joined =[];

for i = 2:31; % Run 1 had defects...twists, tear?

    clear temp temptable
    OneM_timerange = timerange...
        (RunCrucialTimeStamps.RunStartTime(i),RunCrucialTimeStamps.RunEndTime(i));%***run end (not run1Mend)
    temptable = LLSB_distilled(i).Run(OneM_timerange,:);
    samplesize = 0.86 * height(temptable); %avg velocity related metric
    weights = LLSB_distilled(i).Run.Weights(OneM_timerange,:);
    [Sample_from_temp,ids] = datasample(temptable,round(samplesize),'Replace',false,'Weights',weights);
    Sample_from_temp = sortrows(Sample_from_temp);
    LLS_B31joined = [LLS_B31joined;Sample_from_temp];
```

```
end

% figure;plot(LLS_B31joined.TapeWidthAfterCompaction_Rotated);
% figure;histogram(LLS_B31joined.TapeWidthAfterCompaction_Rotated);
% figure;plot(Sample_from_temp.Time,Sample_from_temp.Weights,'o')

clear OneM_timerange weights i ids samplesize Sample_from_temp temptable
```

Create measured gaps histogram

Programmed gap is 6.15 mm

```
MeasuredGap_30joined = [];

for i = 1:30
    MeasuredGap_30joined = vertcat(MeasuredGap_30joined,MeasuredGapData(i).Gap.Gap_Rotated);
end

% %sample equal to no of simulations
% histogram(MeasuredGap_30joined,100);
% figure;plot(MeasuredGap_30joined,'o')
```

Saving reduced data

```
save MeasuredGap_30joined.mat MeasuredGap_30joined % 46384
save Tracker31joined.mat Tracker31joined % 16717
save Camera31joined.mat Camera31joined % 9838
save LLS_A31joined.mat LLS_A31joined % 11618
save LLS_B31joined.mat LLS_B31joined % 12536

mean(LLS_A31joined.TapeWidth_Arc1);
mean(LLS_B31joined.TapeWidthAfterCompaction_Rotated);
% diff in mean tape width 0.1564

% for matlab distribution fitting tool

M = MeasuredGap_30joined;
T = Tracker31joined.Y_normalised;
C = Camera31joined.TapeCenterLine;
A = LLS_A31joined.TapeWidth_Arc1;
% change to keep consistent width measuremnt algorithm
% A = LLS_A31joined.TapeWidth1;
B = LLS_B31joined.TapeWidthAfterCompaction_Rotated;

% getting range min max mean std

maxval = max(T)
minval = min(T)
rangval = max(T) - min(T)
meanval = mean(T)
stddevval = std(T)
```

```
maxval =

    -0.66113


minval =

     -1.1979


rangval =

      0.53681


meanval =

    -0.92461


stddevval =

    0.065793
```

Gap predictor / Tape simulator (FROM ACTUAL MEASURED DATA)

Do many simulations Number of simulations (currently chosen as the min number of distinct data points available for the data stream with the least datapoints ie the camera)

```
Num_of_Simulations = 9838;

Tracker_Sample1 = datasample(T,Num_of_Simulations,'Replace',false);
Camera_Sample1 = datasample(C,Num_of_Simulations,'Replace',false);
LLSA_Sample1 = datasample(A,Num_of_Simulations,'Replace',false);
LLSB_Sample1 = datasample(B,Num_of_Simulations,'Replace',false);

Tracker_Sample2 = datasample(T,Num_of_Simulations,'Replace',false);
Camera_Sample2 = datasample(C,Num_of_Simulations,'Replace',false);
LLSA_Sample2 = datasample(A,Num_of_Simulations,'Replace',false);
LLSB_Sample2 = datasample(B,Num_of_Simulations,'Replace',false);

% Figures to compare width before and after compaction
% figure;hold on
% plot(LLS_A31joined.TapeWidth_Arc1)
% plot(LLS_B31joined.TapeWidthAfterCompaction_Rotated)
```

Sim run section

```
for i = 1 : Num_of_Simulations

% Tape 1 XXXXXXXXXXXXXXXXXXXX
% Get tape position (robot)
tape_position_tracker = Tracker_Sample1(i);

% Get tape position from lateral movement (check if reference system is
% same as robot)                                                    ***NEED TO CHECK
tape_position_camera = Camera_Sample1(i);

% net position
Resultant_tape_position = tape_position_tracker + tape_position_camera;

% Get tape width LLS A sample = w_A
tape_width_LLSA = LLSA_Sample1(i);
halfwidthA = (tape_width_LLSA/2);
tape1_edge_left = Resultant_tape_position - halfwidthA;
tape1_edge_right = Resultant_tape_position + halfwidthA;

% Get width increase due to compaction = ( w_B - w_A ) such that w_B >= w_A
tape_width_LLSB = LLSB_Sample1(i);
% if not resample.
j = 1;
while tape_width_LLSB <= tape_width_LLSA
     if (i+j) > height(LLSB_Sample1)
         break
     end
    tape_width_LLSB = LLSB_Sample1(i+j);
    j = j + 1;
end

while tape_width_LLSB <= tape_width_LLSA
     if (i-j) < 0
         break
     end
    tape_width_LLSB = LLSB_Sample1(i-j);
    j = j - 1;
end

width_increase_compaction = tape_width_LLSB - tape_width_LLSA;
half_width_increase_compaction = width_increase_compaction/2;

%update tape edge
tape1_edge_left = tape1_edge_left - half_width_increase_compaction;
tape1_edge_right = tape1_edge_right + half_width_increase_compaction;

% Save edge data in PredictedGaps(SimulationNumber).Tape 1
PredictedGaps(i).Tape1.LeftEdge = tape1_edge_left;
PredictedGaps(i).Tape1.RightEdge = tape1_edge_right;

% Tape 2  XXXXXXXXXXXXXXXXXXXX
% Get tape position (robot)
tape_position_tracker = Tracker_Sample2(i);

% Get tape position from lateral movement (check if reference system is
% same as robot)                                                    ***NEED TO CHECK
tape_position_camera = Camera_Sample2(i);

% net position
Resultant_tape_position = tape_position_tracker + tape_position_camera;

% Add 12.5 for adjacent tape
Resultant_tape_position = Resultant_tape_position + 12.5; %programmed gap

% Get tape width LLS A sample = w_A
tape_width_LLSA = LLSA_Sample2(i);
```

```
halfwidthA = (tape_width_LLSA/2);
tape2_edge_left = Resultant_tape_position - halfwidthA;
tape2_edge_right = Resultant_tape_position + halfwidthA;

% Get width increase due to compaction = ( w_B - w_A ) such that w_B >= w_A
tape_width_LLSB = LLSB_Sample2(i);
% if not resample.
j = 1;
while tape_width_LLSB <= tape_width_LLSA
     if (i+j) > height(LLSB_Sample2)
         break
     end
    tape_width_LLSB = LLSB_Sample2(i+j);
    j = j + 1;
end

while tape_width_LLSB <= tape_width_LLSA
     if (i-j) < 0
         break
     end
    tape_width_LLSB = LLSB_Sample2(i-j);
    j = j - 1;
end

width_increase_compaction = tape_width_LLSB - tape_width_LLSA;
half_width_increase_compaction = width_increase_compaction/2;

%update tape edge
tape2_edge_left = tape2_edge_left - half_width_increase_compaction;
tape2_edge_right = tape2_edge_right + half_width_increase_compaction;

% Save edge data in PredictedGaps(SimulationNumber).Tape 2
PredictedGaps(i).Tape2.LeftEdge = tape2_edge_left;
PredictedGaps(i).Tape2.RightEdge = tape2_edge_right;

% Find gap for simulation
% Edge to edge distance
% Save PredictedGaps(SimulationNumber).Gap
PredictedGaps(i).Gap = tape2_edge_left - tape1_edge_right;

end

PredictedGaps_FromData = PredictedGaps;
save PredictedGaps_FromData.mat PredictedGaps_FromData

% for i = 1 : length(PredictedGaps_From_Data)
% PG_From_Data(i) = PredictedGaps(i).Gap;
% end
%
% % histogram of predicted gaps
% histogram(PG_From_Data)
% figure;plot([PredictedGaps_From_Data.Gap],'o')
```

Tuning no of bins

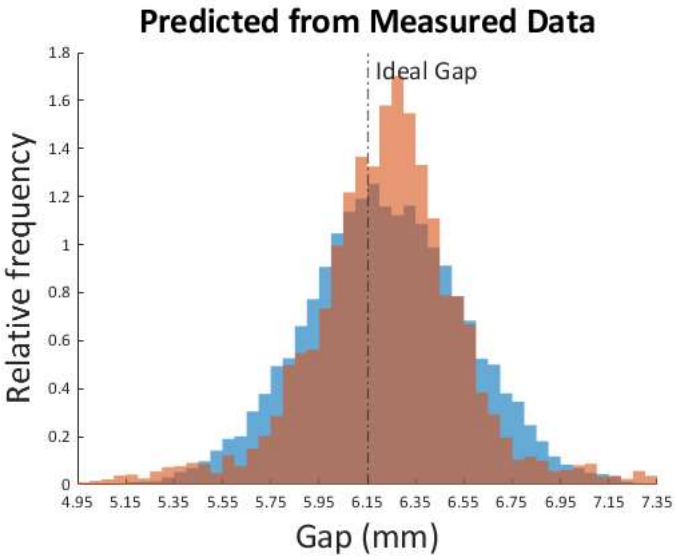
```
MeasuredGap_30joined_sample = datasample(M,Num_of_Simulations,'Replace',false);

for i = 1 : length(PredictedGaps_FromData)
    PredictedGap_Simulator_FromData(i) = PredictedGaps_FromData(i).Gap;
end
```

Measured vs Predicted - Comparision figure FROM DATA

```
%labeltextsize = 13;
labeltextsize = 20;
smallfontsize = 15;
ticklabelfontsize = 15;

%font = 'times'
font = 'calibri';

% NormalisationMethod = 'pdf';
% nbins = 100;

figure;
hold on;
title('Predicted from Measured Data','fontsize',labeltextsize)
histogram(PredictedGap_Simulator_FromData,'Normalization','pdf','EdgeColor','none')
histogram(MeasuredGap_30joined_sample,'Normalization','pdf','EdgeColor','none')
FontSize = ticklabelfontsize
xlabel('Gap (mm)','fontsize',labeltextsize);
ylabel('Relative frequency','fontsize',labeltextsize);
ax = gca;
ax.XLim = ([4.95 7.35]);
grid off
```

```
get(gca,'fontname');
set(gca,'fontname',font);
xline(6.15,'-.','Ideal Gap','LineWidth',1,'LabelOrientation','horizontal','fontname',font,'FontSize',smallfontsize);
Color = [0 0 0];
xticks([4.95:0.2:7.35])
hold off
```

```
FontSize =

    15
```



Gap predictor / Tape simulator (FROM FITS)

Create samples from fits of ditrsibutions

```
% Fit_T_SkewNormal
% Fit_C_SkewNormal
% save '4 Fits.mat' Fit_C_Extremevalue Fit_T_Logistic Fit_A_Normal Fit_B_Normal

% Fit_C_Extremevalue
% Fit_T_Logistic
% Fit_A_Normal
% Fit_B_Normal

Num_of_Simulations = 46384; %9838;

for i = 1 : Num_of_Simulations
    Tracker_Sample1_FromFit(i) = random(Fit_T_Logistic);
    Camera_Sample1_FromFit(i) = random(Fit_C_Extremevalue);
    LLSA_Sample1_FromFit(i) = random(Fit_A_Normal);
    LLSB_Sample1_FromFit(i) = random(Fit_B_Normal);
end

for i = 1 : Num_of_Simulations
    Tracker_Sample2_FromFit(i) = random(Fit_T_Logistic);
    Camera_Sample2_FromFit(i) = random(Fit_C_Extremevalue);
    LLSA_Sample2_FromFit(i) = random(Fit_A_Normal);
    LLSB_Sample2_FromFit(i) = random(Fit_B_Normal);
end

% find when width of B is smaller than width at A (problematic for finding contribution of compaction)

for i = 1:Num_of_Simulations
    if LLSA_Sample1_FromFit(i) >= LLSB_Sample1_FromFit(i)
        Problem_Sample1(i) = 1;
    else
        Problem_Sample1(i) = 0;
    end
end

for i = 1:Num_of_Simulations
    if LLSA_Sample2_FromFit(i) >= LLSB_Sample2_FromFit(i)
        Problem_Sample2(i) = 1;
    else
        Problem_Sample2(i) = 0;
```

```
    end
end
```

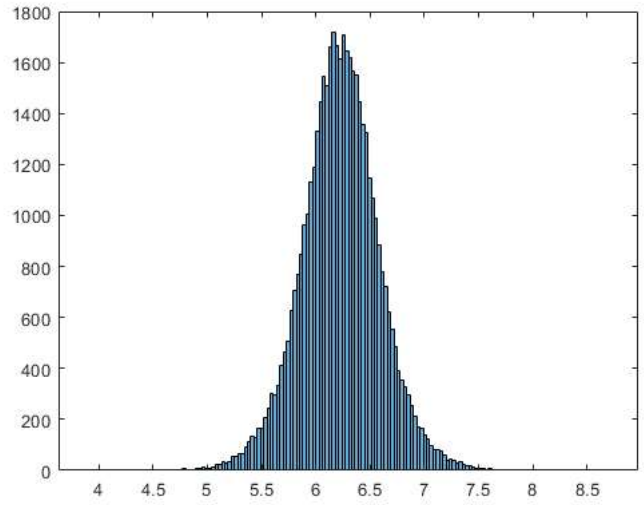
Sim run section

```
for i = 1 : Num_of_Simulations

% Tape 1 XXXXXXXXXXXXXXXXXXXX
% Get tape position (robot)
tape_position_tracker = Tracker_Sample1_FromFit(i);

% Get tape position from lateral movement (check if reference system is
% same as robot)                                                          ***NEED TO CHECK
tape_position_camera = Camera_Sample1_FromFit(i);

% net position
Resultant_tape_position = tape_position_tracker + tape_position_camera;

% Get tape width LLS A sample = w_A
tape_width_LLSA = LLSA_Sample1_FromFit(i);
halfwidthA = (tape_width_LLSA/2);
tape1_edge_left = Resultant_tape_position - halfwidthA;
tape1_edge_right = Resultant_tape_position + halfwidthA;

% Get width increase due to compaction = ( w_B - w_A ) such that w_B >= w_A
tape_width_LLSB = LLSB_Sample1_FromFit(i);

% if problematic than resample.
if Problem_Sample1(i) == 1;
    while tape_width_LLSB <= tape_width_LLSA
        j = randperm(Num_of_Simulations,1);
        tape_width_LLSB = LLSB_Sample1_FromFit(j);
    end
end

% j = 1;
% while tape_width_LLSB <= tape_width_LLSA
%      if (i+j) > height(LLSB_Sample1_FromFit)
%          break
%      end
%     tape_width_LLSB = LLSB_Sample1_FromFit(i+j);
%     j = j + 1;
% end
%
% while tape_width_LLSB <= tape_width_LLSA
%      if (i-j) < 0
%          break
%      end
%     tape_width_LLSB = LLSB_Sample1_FromFit(i-j);
%     j = j - 1;
% end

width_increase_compaction = tape_width_LLSB - tape_width_LLSA;
half_width_increase_compaction = width_increase_compaction/2;

%update tape edge
tape1_edge_left = tape1_edge_left - half_width_increase_compaction;
tape1_edge_right = tape1_edge_right + half_width_increase_compaction;

% Save edge data in PredictedGaps(SimulationNumber).Tape 1
PredictedGaps_FromFit(i).Tape1.LeftEdge = tape1_edge_left;
PredictedGaps_FromFit(i).Tape1.RightEdge = tape1_edge_right;

% Tape 2  XXXXXXXXXXXXXXXXXXXX
% Get tape position (robot)
tape_position_tracker = Tracker_Sample2_FromFit(i);

% Get tape position from lateral movement (check if reference system is
% same as robot)                                                          ***NEED TO CHECK
tape_position_camera = Camera_Sample2_FromFit(i);

% net position
Resultant_tape_position = tape_position_tracker + tape_position_camera;

% Add 12.5 for adjacent tape
Resultant_tape_position = Resultant_tape_position + 12.5; %programmed gap

% Get tape width LLS A sample = w_A
tape_width_LLSA = LLSA_Sample2_FromFit(i);
halfwidthA = (tape_width_LLSA/2);
tape2_edge_left = Resultant_tape_position - halfwidthA;
tape2_edge_right = Resultant_tape_position + halfwidthA;

% Get width increase due to compaction = ( w_B - w_A ) such that w_B >= w_A
tape_width_LLSB = LLSB_Sample2_FromFit(i);% if problematic than resample.
% if problematic than resample.
```

```
if Problem_Sample2(i) == 1;
    while tape_width_LLSB <= tape_width_LLSA
        j = randperm(Num_of_Simulations,1);
        tape_width_LLSB = LLSB_Sample2_FromFit(j);
    end
end


% j = 1;
% while tape_width_LLSB <= tape_width_LLSA
%      if (i+j) > height(LLSB_Sample2_FromFit)
%          break
%      end
%     tape_width_LLSB = LLSB_Sample2_FromFit(i+j);
%     j = j + 1;
% end
%
% while tape_width_LLSB <= tape_width_LLSA
%      if (i-j) < 0
%          break
%      end
%     tape_width_LLSB = LLSB_Sample2_FromFit(i-j);
%     j = j - 1;
% end

width_increase_compaction = tape_width_LLSB - tape_width_LLSA;
half_width_increase_compaction = width_increase_compaction/2;

%update tape edge
tape2_edge_left = tape2_edge_left - half_width_increase_compaction;
tape2_edge_right = tape2_edge_right + half_width_increase_compaction;

% Save edge data in PredictedGaps(SimulationNumber).Tape 2
PredictedGaps_FromFit(i).Tape2.LeftEdge = tape2_edge_left;
PredictedGaps_FromFit(i).Tape2.RightEdge = tape2_edge_right;

% Find gap for simulation
% Edge to edge distance
% Save PredictedGaps(SimulationNumber).Gap
PredictedGaps_FromFit(i).Gap = tape2_edge_left - tape1_edge_right;

end

save PredictedGap_FromFit.mat PredictedGaps_FromFit

% for i = 1 : length(PredictedGaps_FromFit)
% PG_FromFit(i) = PredictedGaps_FromFit(i).Gap;
% end
%
% % histogram of predicted gaps
% histogram(PG_FromFit)
% figure;plot([PredictedGaps_FromFit.Gap],'o')
```

Tuning no of bins

```
MeasuredGap_30joined_sample = datasample(MeasuredGap_30joined,Num_of_Simulations,'Replace',false);

for i = 1 : length(PredictedGaps_FromFit)
    PredictedGap_Simulator_FromFit(i) = PredictedGaps_FromFit(i).Gap;
end

histogram(PredictedGap_Simulator_FromFit)
```

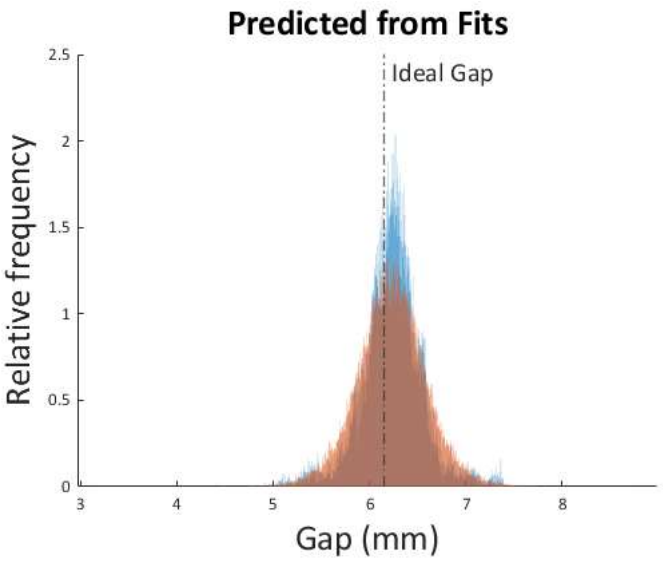


Measured vs Predicted - Comparision figure FROM FITS

```
% %labeltextsize = 13;
% labeltextsize = 20;
% smallfontsize = 15;
% ticklabelfontsize = 15;
% %font = 'times'
% font = 'calibri';

% NormalisationMethod = 'probability';
nbins = 1000;

figure;
hold on;
title('Predicted from Fits','fontsize',labeltextsize)

h200 = histogram(MeasuredGap_30joined_sample,nbins+300,'Normalization','pdf','EdgeColor','none')
h100 = histogram(PredictedGap_Simulator_FromFit,nbins,'Normalization','pdf','EdgeColor','none')
FontSize = ticklabelfontsize
xlabel('Gap (mm)','fontsize',labeltextsize);
ylabel('Relative frequency','fontsize',labeltextsize);
ax = gca;
% ax.XLim = ([4.95 7.35]);
grid off
get(gca,'fontname');
set(gca,'fontname',font);
xline(6.15,'-.','Ideal Gap','LineWidth',1,'LabelOrientation','horizontal','fontname',font,'FontSize',smallfontsize);
Color = [0 0 0];
% xticks([4.95:0.2:7.35])
hold off;
```

```
h200 =

  Histogram with properties:

            Data: [46384×1 double]
          Values: [0.0052932 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … ] (1×1300 double)
         NumBins: 1300
        BinEdges: [3.244 3.2481 3.2521 3.2562 3.2603 … ] (1×1301 double)
        BinWidth: 0.004073
       BinLimits: [3.244 8.5389]
   Normalization: 'pdf'
       FaceColor: 'auto'
       EdgeColor: 'none'

  Use GET to show all properties


h100 =

  Histogram with properties:

            Data: [6.383 6.0229 6.0244 6.0284 5.5516 … ] (1×46384 double)
          Values: [0.0044608 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … ] (1×1000 double)
         NumBins: 1000
        BinEdges: [3.88 3.8848 3.8897 3.8945 3.8993 … ] (1×1001 double)
        BinWidth: 0.004833
       BinLimits: [3.88 8.713]
   Normalization: 'pdf'
```

```
        FaceColor: 'auto'
        EdgeColor: 'none'

   Use GET to show all properties


FontSize =

    15
```



Code graveyard / Appendix

% ids shows which values were selected data = DataTimeTables(20).Tracker.Y_normalised; samplesize = 0.8129 * height(data); %avg veloity related metric weights = DataTimeTables(20).Tracker.Velocity/200; %velocity/max velocity smallNoIndex = weights < 0.05; %removing small, non-negative and non-NaN weights weights(smallNoIndex) = 0; % [Sample,ids] = datasample(data,round(samplesize),'Replace',false,'Weights',weights);

i = 1; plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); LLS_A31joined = DataTimeTables(i).LLS_A(plateautimerange,:);

for i = 2:31 clear temp plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); temp = DataTimeTables(i).LLS_A(plateautimerange,:); LLS_A31joined = [LLS_A31joined;temp]; end

figure;plot(LLS_A31joined.TapeWidth_Arc1); figure;histogram(LLS_A31joined.TapeWidth_Arc1);

Creating 4 distributions (plateau region)

% tracker (by platuea time stamps) i = 1; plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); Tracker31joined = DataTimeTables(i).Tracker(plateautimerange,:);

for i = 2:31 clear temp plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); temp = DataTimeTables(i).Tracker(plateautimerange,:); Tracker31joined = [Tracker31joined;temp]; end

figure;plot(Tracker31joined.Y_normalised); figure;plot(Tracker31joined.Velocity); figure;histogram(Tracker31joined.Y_normalised);

% camera (by platuea time stamps) i = 1; plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); Camera31joined = DataTimeTables(i).Camera(plateautimerange,:);

for i = 2:31 clear temp plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); temp = DataTimeTables(i).Camera(plateautimerange,:); Camera31joined = [Camera31joined;temp]; end

figure;plot(Camera31joined.TapeCenterLine); figure;histogram(Camera31joined.TapeCenterLine);

% LLS A (by platuea time stamps) % cos tape runs at same speed under LLS A as the layup speed) i = 1; plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); LLS_A31joined = DataTimeTables(i).LLS_A(plateautimerange,:);

for i = 2:31 clear temp plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); temp = DataTimeTables(i).LLS_A(plateautimerange,:); LLS_A31joined = [LLS_A31joined;temp]; end

figure;plot(LLS_A31joined.TapeWidth_Arc1); figure;histogram(LLS_A31joined.TapeWidth_Arc1);

% Timestamps visualisation for a single run hold on plot(DataTimeTables(15).Tracker.TimeStamp,DataTimeTables(15).Tracker.Y_normalised); plot(DataTimeTables(15).Tracker.TimeStamp, (DataTimeTables(15).Tracker.X_mm_/200));% mag reduced to visualise plot(DataTimeTables(15).Tracker.TimeStamp,(DataTimeTables(15).Tracker.Velocity/31.5));% mag reduced to visualise plot(DataTimeTables(15).LLS_A.Time,DataTimeTables(15).LLS_A.TapeWidth_Arc1); plot(DataTimeTables(15).Camera.TimeStamps,DataTimeTables(15).Camera.TapeCenterLine); plot(DataTimeTables(15).LLS_B.Time,DataTimeTables(15).LLS_B.TapeWidthAfterCompaction_Rotated);

x(1) = RunCrucialTimeStamps.RunStartTime(15); x(2) = RunCrucialTimeStamps.PlatueauStartTime(15); x(3) = RunCrucialTimeStamps.PlatueauEndTime(15); x(4) = RunCrucialTimeStamps.CutOffStartTime(15); x(5) = RunCrucialTimeStamps.CutOffEndTime(15); x(6) = RunCrucialTimeStamps.Run1mEndTime(15); x(7) = RunCrucialTimeStamps.RunEndTime(15); plot(x,[0 0 0 0 0 0 0],'-o');

```
legend('Tracker','X mm','Velocity','LLS A','Camera','LLS B','Crucial Timestamps')

% LLS B (cannot use platuea time stamps as LLS B measurement point is at a distance from TCP) % (LLS B values start and end with detection) % For LLS B stream at true time platueau region starts at ~X = 103. Therefore plateau velocity % region ends at ~X = 903 + 103???

% Sampling without replacement (observation is selected at most once. Never reselected) % ids shows which values were selected data = DataTimeTables(20).Tracker.Y_normalised; samplesize = 0.8129 * height(data); %avg veloity related metric weights = DataTimeTables(20).Tracker.Velocity/200; %velocity/max velocity smallNoIndex = weights < 0.05; %removing small, non-negative and non-NaN weights weights(smallNoIndex) = 0; % [Sample,ids] = datasample(data,round(samplesize),'Replace',false,'Weights',weights);

i = 1; plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); LLS_A31joined = DataTimeTables(i).LLS_A(plateautimerange,:);

for i = 2:31 clear temp plateautimerange = timerange... (RunCrucialTimeStamps.PlatueauStartTime(i),RunCrucialTimeStamps.PlatueauEndTime(i)); temp = DataTimeTables(i).LLS_A(plateautimerange,:); LLS_A31joined = [LLS_A31joined;temp]; end

figure;plot(LLS_A31joined.TapeWidth_Arc1); figure;histogram(LLS_A31joined.TapeWidth_Arc1);
```

Creating 4 distributions (biased across lengths) %%%%%TRASH

```
i = 1; %initialise Camera_31joined = Data(i).Camera; LLSA_31joined = Data(i).LLS_A; LLSB_31joined = Data(i).LLS_B; Tracker_31joined = Data(i).Tracker;

for i = 2:31 Camera_31joined = vertcat(Camera_31joined,Data(i).Camera) LLSA_31joined = vertcat(LLSA_31joined,Data(i).LLS_A) LLSB_31joined = vertcat(LLSB_31joined,Data(i).LLS_B) Tracker_31joined = vertcat(Tracker_31joined,Data(i).Tracker) end
```

Published with MATLAB® R2024a