Probability Models for Enhancing Calibration of Computer Numerical Control (CNC) Systems

Jacob Seman, Viktor Woldruff, Tyler Nelson

Partial Report

ECEN3810

Introduction to Probability Theory

Table of Contents

I. Executive Summary:

Computer-numerical-control (CNC) printed-circuit-board (PCB) milling is a valuable tool for students in the field of Electrical & Computer Engineering. Milling PCBs requires a well-calibrated CNC mill and a flat PCB surface. Variance exists in the surface of single-sided copper boards, and additional variance is introduced by fixturing methods. This variance must be accounted for, to achieve accurate reproduction of PCB designs, and can be modeled to aid in machine calibration and programmed height offsets, which allows precision machining even when boards are considerably warped by defects or fixturing. Heightmaps of the copper-clad board, where measurements in the Z-axis were taken at regular intervals in the X-Y plane, were produced using the CNC control software Candle[1], and correlations were identified in the resulting dataset. Normalization was required to remove an observed tilt in the fixturing and expose the subtle variation in the heightmaps. MATLAB was used to generate a variance model showing any observed correlation in X- and Y-axes and compared against the measured data.

Commented [JS1]: The boards themselves don't have the calibration property, but variation does exist in their surfaces.

II. Method:

Data was collected for the physical system via CNC machine calibration and the resultant heightmap of 10 single-sided copper-clad boards. This was achieved by the use of the heightmap functionality of the Candle CNC software[2], wherein the software can detect when the metal cutting bit makes electrical contact with the copper-clad board by performing a grid routine of probing contact operations according to user specification[2] [Fig. 1]. All 10 boards were clamped at a specific torque value of 3 NM [Fig. 2], applied to the single fixture clamp bolt using a torque-release hand tool set to this value.

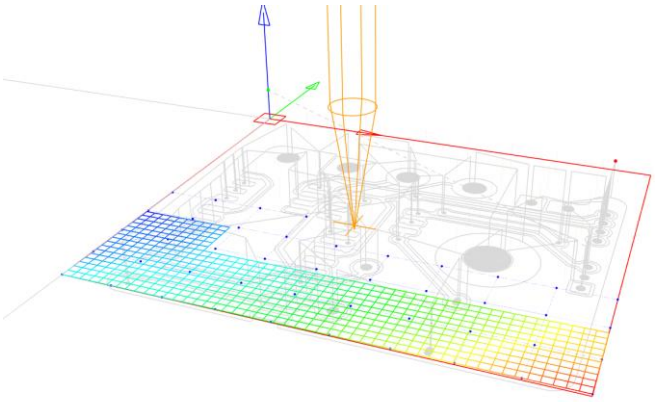

Figure 1 - Candle CNC software performing heightmap operation using electrical contact probing.

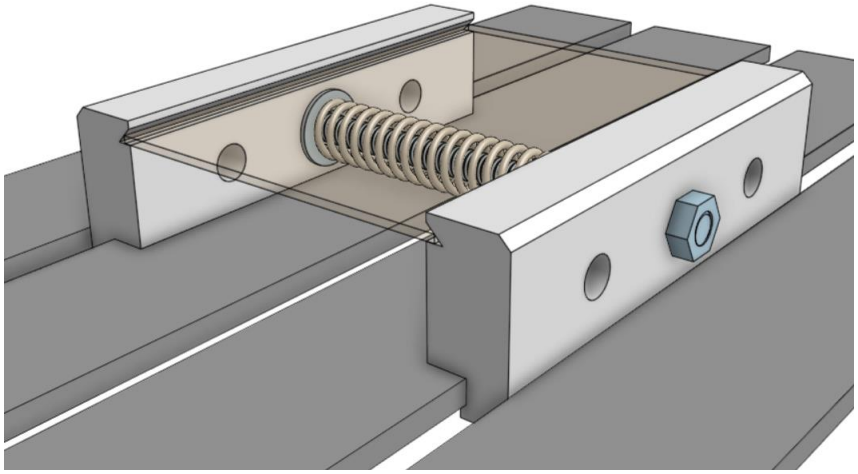

Figure 2 - PCB board fixturing specifically designed for 100x70mm PCBs.

The boards were measured to be 100mm by 70mm, and the heightmap routine was specified within a 95mm by 65mm region, with a resolution of 34 points by 16 points, in X- and Y-axes, respectively. Once the heightmaps were obtained for all 10 copper boards, a MATLAB script was developed to ingest the heightmap output files and plot a mesh for visual observation [Fig. 3]. Additionally, data was normalized by X and Y linear approximation of the quadrant averages and similarly plotted. This allowed improved visualization of the local variance in Z height, over X and Y, in the resultant surface plots [Fig. 4].

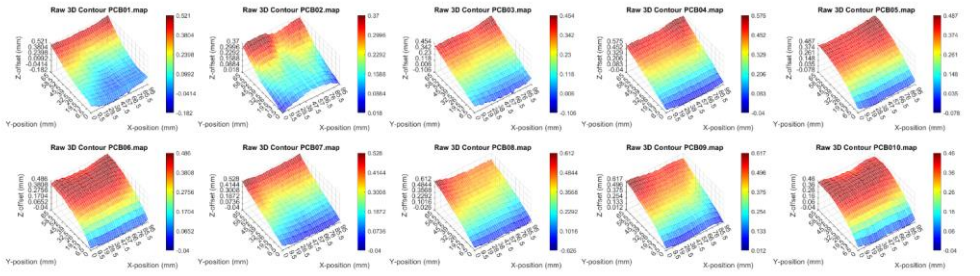

Figure 3 - MATLAB script output showing raw heightmap data.

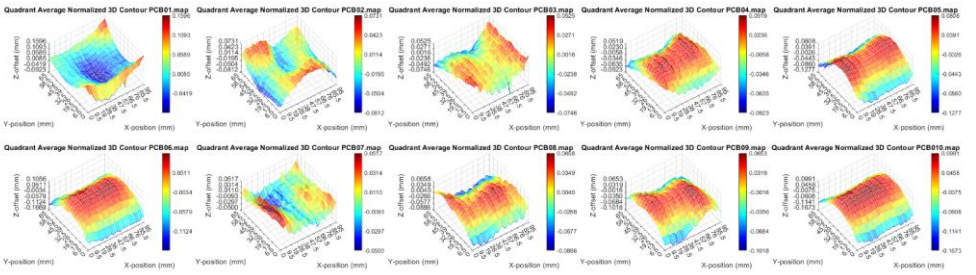

Figure 4 - MATLAB script output showing normalized heightmap data.

III. Analysis:

A tendency for variance to be higher in the center of the Y-axis was noted, with slight reduction in variance at the extremes of the X-axis. Conversely, variance was considerably less at the extremes of the Y-axis, with little or no change along the X-axis at these points. This effect is attributable to the fixturing of the PCBs, where the clamping force imparts some flex, typical convex in the Y-axis. A numerical model with adjustable parameters was developed using GeoGebra[3] that allowed replicating this tendency on a 100mm by 70mm 3D surface [Fig. 5].

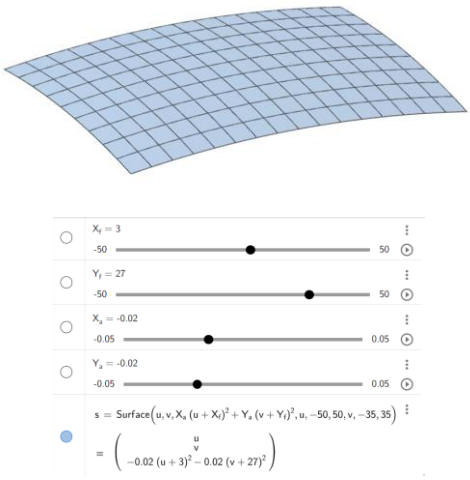

Figure 5 - GeoGebra numerical surface model with adjustable parameters.

A matrix of means for the entire dataset was produced by creating a matrix of sums of variance from the local average for each heightmap and dividing each coordinate sum by the size of the entire dataset. This allowed plotting of a single parabolic surface which represents the mean variance, by coordinate, for the entire dataset of 10 boards. Note that each observed surface imparts some effect on the mean surface. Notably, there is a subtle concavity over the X-axis (with lessened variance just at the X-extremes), and a more dominant convexity over the Y-axis (with a taper toward much lower variance at the Y-extremes) [Fig. 6].

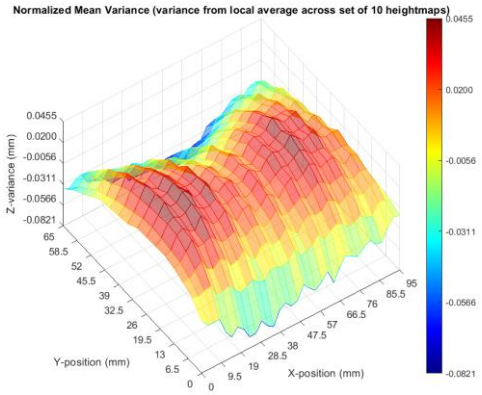

Figure 6 – Normalized mean variance surface, representing the entire dataset across 10 boards.

Adjusting the numerical model in GeoGebra allows approximation of this 'saddle' surface [Fig. 7]. In reality, this numerical model would benefit from greater order functions in X and Y, to capture the lessened variance at either dimension's extremes.

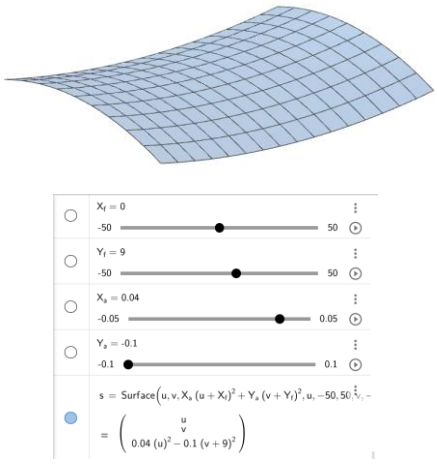

Figure 7 - Numerical representation in GeoGebra of mean variance surface.

Also of note is the consistency of tendencies unique to the X- and Y-axes. A ~10µm ripple along the X-axis was observed, as well as a sweeping parabola along the Y-axis. Creating vectors of the row and column averages allows plotting in 2D to better observe these features [Fig. 8]. It is suspected that the ripple along the X-axis is induced by the CNC machine itself, likely from a bowed lead screw putting strain on the tool head as it rotates to traverse the gantry along the X-axis. This is likely a common trait among inexpensive desktop CNC machines. Performing a higher resolution heightmap routine would likely reveal that this ripple has a period matching the travel in X per lead screw rotation, which was measured to be roughly 4.3mm. Note that the observed interval is twice this, most likely due to the selected X-resolution of the heightmap coincidentally being near a harmonic of the lead screw rotation interval.

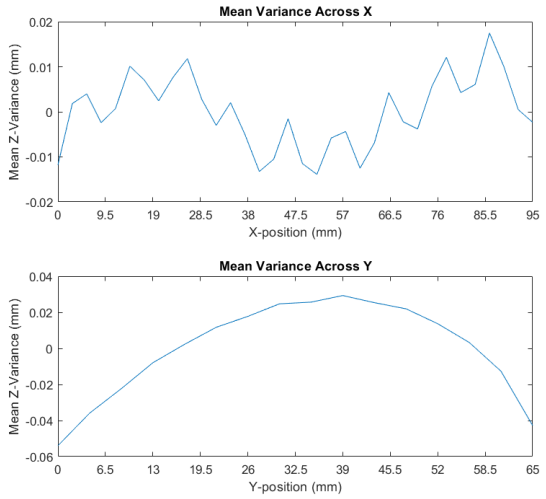

Figure 8 - MATLAB plots of 2D X- and Y-axes mean vectors.

While the normalized mean dataset provides an excellent visual comparison of local variations such as the lead-screw effect described above, the raw mean data is most immediately valuable to real-world applications and is a better model of the actual physical PCB when clamped in the fixture [Fig. 9].

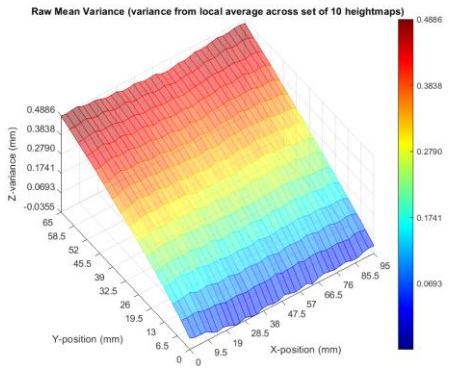

Figure 9 – Raw mean variance surface, representing the entire dataset across 10 boards.

Most immediately noticeable is the strong tendency for one side of the fixture to be nearly 0.5mm higher than the other, with few other features represented. This slope is significant and obfuscates any other local variance features observed in the normalized data. While heightmaps created in the Candle software are accurate enough to account for the subtle variation of the lead screw or other local variance in the PCB, the sloped heightmap observed in the raw mean plot may be applied to any PCB clamped in this fixture without the need for creating a heightmap unique to that specific PCB. This is further explored by creating standard deviation plots from the dataset [Fig. 10 & Fig. 11].

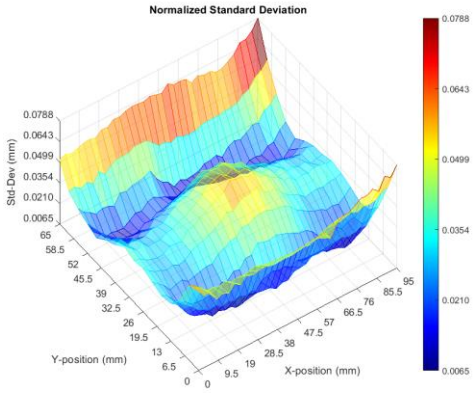

Figure 10 – Normalized standard deviation surface, representing the entire dataset across 10 boards.

The normalized standard deviation surface shows the effect of the quadrant averaging normalization, which presents itself as near-zero deviation in the center of each quadrant. This method skews the consistency of the fixture clamping surface, causing high deviation at these locations. The raw standard deviation provides more useful information [Fig. 10].

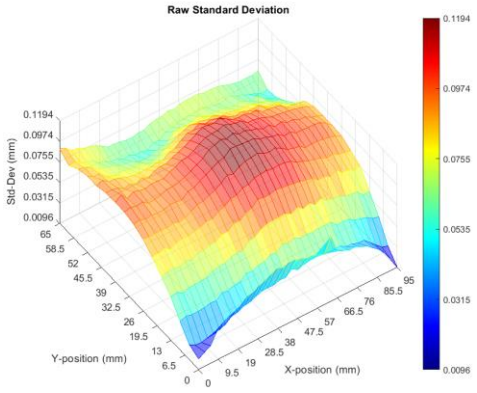

Figure 11 – Raw standard deviation surface, representing the entire dataset across 10 boards.

Standard deviation in the raw data set behaves as expected – with much less deviation at the clamping surfaces. The center of the PCB varies as much as 0.12mm, showing the effect of PCB flexure as the fixture is clamped at consistent torque values. This informs best practices when applying the mean heightmap to an unknown PCB – milling results will be most consistent when fitting smaller designs closer to the lower end of the fixture due to the increased likelihood of the actual PCB position being "on-model" with the mean heightmap. Error percentage was also plotted, wherein each board sample was compared to the mean data [Fig. 12]. This showed areas of higher error per each board. The mean of the error data set is the same as the raw standard deviation, but in terms of percentage error [Fig. 13].

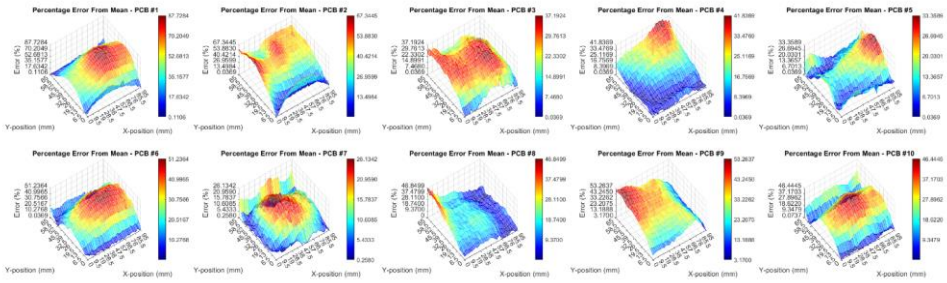

Figure 12 - Percentage error per board, showing areas of greatest deviation from the mean.

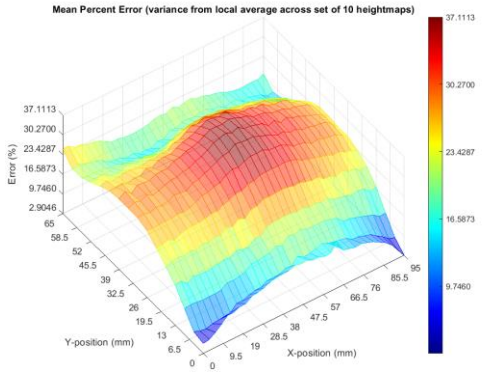

Figure 13 – Mean percent error, representing the entire dataset across 10 boards. (Equivalent to std. deviation in percent)

The small size of the data set must be addressed. Only ten boards were acquired for testing, and from a single source, limiting the viability of this model. The most prominent feature of the mean data was noted to be induced by the fixture design, where mean error was as high as 37% [Fig. 13]. "Bootstrapping" of the data set, or resampling via random sampling with replacement, was applied to judge the limit of confidence in the mean data by taking the mean of a larger, generated data set [Fig. 14]. 1000 samples were generated for the bootstrapped data set. Each of these 1000 samples consisted of the mean of 10 random samples from the original data set, chosen with replacement.

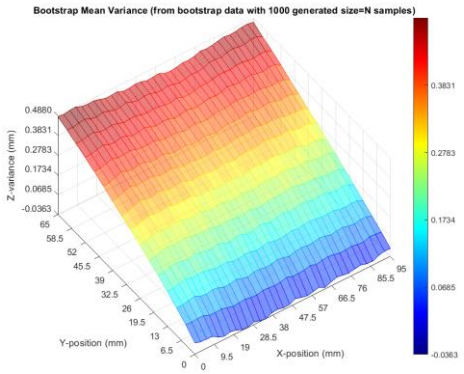

Figure 14 – Bootstrapped data set mean. (1000 generated size N sample means)

The mean of the bootstrap data bears a strong resemblance to the original data set. The difference between the original mean and the bootstrap mean was also determined. The bootstrap mean was observed to be at most 1.2 microns higher than the original mean in the center of the board, and at most 0.9 microns lower along the upper edge [Fig. 15].

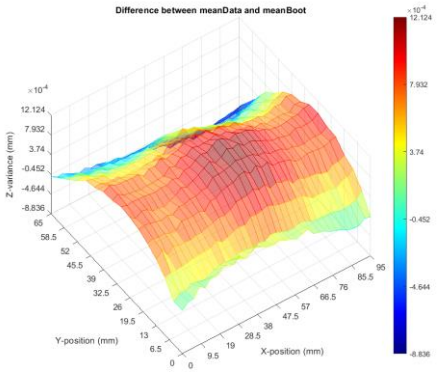

Figure 15 – Difference between bootstrapped data set mean and the original data set mean.

The standard deviation present in the bootstrapped data was also nearly identical to the original data set's standard deviation in terms of relative deviation by coordinate position [Fig. 16]. However, it is important to note that the range of deviation is significantly less than the original data set. The bootstrap scheme imparts this reduction by being significantly larger, and the fact that the overall shape is nearly identical improves confidence in this model. A new map file was generated for use with the Candle CNC software using the bootstrap mean data based on this confidence.

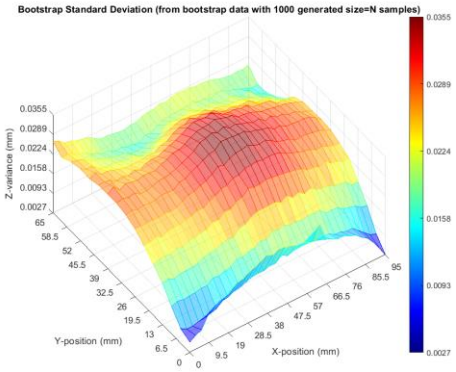

Figure 16 – Standard deviation of bootstrapped data set.

IV. Results:

In the context of CNC system calibration, the key concepts in probability modeling that are explored are variance and covariance. Variance, as observed in the heightmaps of copper-clad boards, signifies the local deviation in Z-height [Fig. 6]. Meanwhile, covariance, specifically modeled as a parabolic function, highlights the interdependence between variables, emphasizing its significance in understanding the nuanced relationships within the CNC calibration process [Fig. 8].

Furthermore, while alternative probability models like Markov chains and Monte Carlo simulation find relevance in various engineering contexts, the specificity of CNC calibration demands a nuanced approach centered around variance and covariance modeling. Confidence in the resultant mean variance models [Fig. 9 & 14] is supported by the real sample set's standard deviation [Fig. 11] as well as that of the simulated sample set [Fig. 16].

Heightmap data was found to show a consistent pattern in variance depending on X and Y coordinate position. The edges of the board that were in contact with the fixturing typically did not vary as much, but closer to the center of the board a more pronounced variance was observed. This variance was correlated in the X- and Y-axes and became most pronounced in the center of either axis range. The mean represents the expected value of the board height for each X and Y coordinate, and so this was used to generate a model heightmap file which may be tested in the Confluence Hall labs during PCB production.

Lastly, all of the data, scripts, and figures used in this project were compiled into a Github repository[6] for use in the Confluence Hall electronics lab. It is hoped that continued contributions will improve the data set and resultant model to benefit all who use these facilities.

V. MATLAB Code Appendix

Data set, raw and normalized means, raw and normalized standard deviations:

```
%% Reset
close all;
clear all;
clc;

%% Colormap
map=jet;

%% Plot two figures with east colorbar legend
figure(1)
tiledlayout(2,5);
%meanVal(10)=zeros; not needed due to normalization-this value is ~zero
meanPlot(16,34)=zeros;
meanValRaw(10)=zeros;
meanPlotRaw(16,34)=zeros;
stdDevData(16,34,10)=zeros;
stdDevDataRaw(16,34,10)=zeros;
stdDevPlot(16,34)=zeros;

for i=1:1:10
    %% Filename selections
    map_num="PCB0"+i+".map";

    %% Import *.map file and get mesh parameters for parsing and plotting
    data=dlmread(map_num,';');
    xSize=data(1,3);
    ySize=data(1,4);
    xRes=data(2,1);
    yRes=data(2,2);
    xStep=xSize/(xRes-1);
    yStep=ySize/(yRes-1);

    %% Built plot matrix and parameters
    plotData=data;
    for j=1:3
        plotData(1,:)=[];
    end
    xPlot=0:xStep:xSize;
    xMin=min(xPlot,[],'all');
    xMax=max(xPlot,[],'all');
    xRange=[xMin xMax];
    xSteps=(xMax-xMin)/10;
    yPlot=0:yStep:ySize;
    yMin=min(yPlot,[],'all');
    yMax=max(yPlot,[],'all');
    yRange=[yMin yMax];
    ySteps=(yMax-yMin)/10;

    %% Calculate linear approximations from quadrant average and normalize
    % quadrant averages & linear approximation in x:
    P1=mean(plotData(1:yRes/2,1:xRes/2),'all');
    P2=mean(plotData(1:yRes/2,(xRes/2)+1:xRes),'all');
    P3=mean(plotData((yRes/2)+1:yRes,1:xRes/2),'all');
    P4=mean(plotData((yRes/2)+1:yRes,(xRes/2)+1:xRes),'all');
    Mx_1=(P2-P1)/(xMax/2);
    Mx_2=(P4-P3)/(xMax/2);
    Mx=(Mx_1+Mx_2)/2; % avg slope in x
    % apply normalization in x
    for j=1:xRes
        for k=1:yRes
            plotData(k,j)=plotData(k,j)-Mx*(((j-1)*xStep)-xSize/2);
        end
    end
    % quadrant averages & linear approximation in y:
```

```
    P1=mean(plotData(1:yRes/2,1:xRes/2),'all');
    P2=mean(plotData(1:yRes/2,(xRes/2)+1:xRes),'all');
    P3=mean(plotData((yRes/2)+1:yRes,1:xRes/2),'all');
    P4=mean(plotData((yRes/2)+1:yRes,(xRes/2)+1:xRes),'all');
    My_1=(P3-P1)/(yMax/2);
    My_2=(P4-P2)/(yMax/2);
    My=(My_1+My_2)/2; % avg slope in y
    % apply normalization in y
    for j=1:xRes
        for k=1:yRes
            plotData(k,j)=plotData(k,j)-My*(((k-1)*yStep)-ySize/2);
        end
    end
    % apply normalization in z
    zOffs=mean(plotData,'all');
    for j=1:xRes
        for k=1:yRes
            plotData(k,j)=plotData(k,j)-zOffs;
        end
    end

    %% Get normalized mean of whole matrix
    %meanVal(i)=mean(plotData,'all');
    for j=1:xRes
        for k=1:yRes
            meanPlot(k,j)=meanPlot(k,j)+(plotData(k,j));%-meanVal(i));
            stdDevData(k,j,i)=plotData(k,j);%-meanVal(i);
        end
    end

    %% Calculate z-parameters
    zMin=min(plotData,[],'all');
    zMax=max(plotData,[],'all');
    zRange=[zMin zMax];
    zSteps=(zMax-zMin)/5;

    % first tile
    nexttile
    s=mesh(xPlot,yPlot,plotData,'FaceAlpha','0.5');
    colormap(map);
    title('Quadrant Average Normalized 3D Contour '+map_num);
    xlabel('X-position (mm)')
    ylabel('Y-position (mm)')
    zlabel('Z-offset (mm)')
    xlim(xRange);
    xticks(xMin:xSteps:xMax);
    ylim(yRange);
    yticks(yMin:ySteps:yMax);
    zlim(zRange);
    zticks(zMin:zSteps:zMax);
    s.FaceColor = 'flat';
    rotate(s, [0 0 1], 180)
    view(325,60);
    cb=colorbar('Ticks',zMin:zSteps:zMax);
    %cb.Layout.Tile='east';
end

figure(2)
tiledlayout(2,5);
for i=1:1:10
    %% Filename selections
    map_num="PCB0"+i+".map";

    %% Import *.map file and get mesh parameters for parsing and plotting
    data=dlmread(map_num,';');
    xSize=data(1,3);
    ySize=data(1,4);
    xRes=data(2,1);
```

```
    yRes=data(2,2);
    xStep=xSize/(xRes-1);
    yStep=ySize/(yRes-1);

    %% Plot second map for comparison
    data=dlmread(map_num,';');
    xSize=data(1,3);
    ySize=data(1,4);
    xRes=data(2,1);
    yRes=data(2,2);
    xStep=xSize/(xRes-1);
    yStep=ySize/(yRes-1);

    % Build plot matrix and plot parameters
    plotData=data;
    for j=1:3
        plotData(1,:)=[];
    end
    xPlot=0:xStep:xSize;
    xMin=min(xPlot,[],'all');
    xMax=max(xPlot,[],'all');
    xRange=[xMin xMax];
    xSteps=(xMax-xMin)/10;
    yPlot=0:yStep:ySize;
    yMin=min(yPlot,[],'all');
    yMax=max(yPlot,[],'all');
    yRange=[yMin yMax];
    ySteps=(yMax-yMin)/10;
    zMin=min(plotData,[],'all');
    zMax=max(plotData,[],'all');
    zRange=[zMin zMax];
    zSteps=(zMax-zMin)/5;

    %% Get raw mean of whole matrix
    meanValRaw(i)=mean(plotData,'all');
    for j=1:xRes
        for k=1:yRes
            meanPlotRaw(k,j)=meanPlotRaw(k,j)+(plotData(k,j));%-meanValRaw(i));
            stdDevDataRaw(k,j,i)=plotData(k,j);%-meanVal(i);
        end
    end

    nexttile
    s=mesh(xPlot,yPlot,plotData,'FaceAlpha','0.5');
    colormap(map);
    title('Raw 3D Contour '+map_num);
    xlabel('X-position (mm)')
    ylabel('Y-position (mm)')
    zlabel('Z-offset (mm)')
    xlim(xRange);
    xticks(xMin:xSteps:xMax);
    ylim(yRange);
    yticks(yMin:ySteps:yMax);
    zlim(zRange);
    zticks(zMin:zSteps:zMax);
    s.FaceColor = 'flat';
    rotate(s, [0 0 1], 180)
    view(325,60);
    cb=colorbar('Ticks',zMin:zSteps:zMax);
    %cb.Layout.Tile='east';
end

%% Calculate point means
for j=1:xRes
    for k=1:yRes
        meanPlot(k,j)=meanPlot(k,j)/10;
        meanPlotRaw(k,j)=meanPlotRaw(k,j)/10;
    end
```

```
end

%% Plot normalized mean set
% calculate z-parameters
zMin=min(meanPlot,[],'all');
zMax=max(meanPlot,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
figure(3)
s=mesh(xPlot,yPlot,meanPlot,'FaceAlpha','0.5');
colormap(map);
title('Normalized Mean Variance (variance from local average across set of 10 heightmaps)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Plot raw mean set
% recalculate z-parameters
zMin=min(meanPlotRaw,[],'all');
zMax=max(meanPlotRaw,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;

figure(4)
s=mesh(xPlot,yPlot,meanPlotRaw,'FaceAlpha','0.5');
colormap(map);
title('Raw Mean Variance (variance from local average across set of 10 heightmaps)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Plot normalized 2D plot of mean in X and Y
meanNVarX=mean(meanPlot,1);
meanNVarY=mean(meanPlot,2)';
figure(5)
tiledlayout(2,1)
nexttile
plot(linspace(0,xMax,xRes),meanNVarX)
title('Mean Variance Across X');
xlim(xRange);
xticks(xMin:xSteps:xMax);
xlabel('X-position (mm)')
ylabel('Mean Z-Variance (mm)')
nexttile
plot(linspace(0,yMax,yRes),meanNVarY)
title('Mean Variance Across Y');
xlim(yRange);
xticks(yMin:ySteps:yMax);
```

```
xlabel('Y-position (mm)')
ylabel('Mean Z-Variance (mm)')

%% X-Y Std Deviation
% get per-coordinate deviations
stdDevPlot=std(stdDevData,0,3);
figure(5)
% plot std deviation result
% calculate z-parameters
zMin=min(stdDevPlot,[],'all');
zMax=max(stdDevPlot,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;

figure(6)
s=mesh(xPlot,yPlot,stdDevPlot,'FaceAlpha','0.5');
colormap(map);
title('Normalized Standard Deviation');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Std-Dev (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Plot Raw Std Deviation
% get per-coordinate deviations
stdDevPlotRaw=std(stdDevDataRaw,0,3);
figure(7)
% recalculate z-parameters
zMin=min(stdDevPlotRaw,[],'all');
zMax=max(stdDevPlotRaw,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;

s=mesh(xPlot,yPlot,stdDevPlotRaw,'FaceAlpha','0.5');
colormap(map);
title('Raw Standard Deviation');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Std-Dev (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Plot 2D plot of std dev in X and Y
stdDevX=mean(stdDevPlot,1);
stdDevY=mean(stdDevPlot,2)';
figure(8)
tiledlayout(2,1)
nexttile
plot(linspace(0,xMax,xRes),stdDevX)
title('Standard Deviation Across X');
xlim(xRange);
```

```
xticks(xMin:xSteps:xMax);
xlabel('X-position (mm)')
ylabel('X-Std-Dev (mm)')
nexttile
plot(linspace(0,yMax,yRes),stdDevY)
title('Standard Deviation Across Y');
xlim(yRange);
xticks(yMin:ySteps:yMax);
xlabel('Y-position (mm)')
ylabel('Y-Std-Dev (mm)')
```

Published with MATLAB® R2023a

Fixture re-clamp and difference compare:

```
%% Reset
close all;
clear all;
clc;

%% Filename selections
map1="PCB07.map";
map2="PCB07b.map";

%% Colormap
map='jet';

%% Import *.map file and get mesh parameters for parsing and plotting
data=dlmread(map1,';');
xSize=data(1,3);
ySize=data(1,4);
xRes=data(2,1);
yRes=data(2,2);
xStep=xSize/(xRes-1);
yStep=ySize/(yRes-1);

%% Build plot matrix and plot parameters
plotData1=data;
for i=1:3
    plotData1(1,:)=[];
end
xPlot=0:xStep:xSize;
xMin=min(xPlot,[],'all');
xMax=max(xPlot,[],'all');
xRange=[xMin xMax];
xSteps=(xMax-xMin)/10;
yPlot=0:yStep:ySize;
yMin=min(yPlot,[],'all');
yMax=max(yPlot,[],'all');
yRange=[yMin yMax];
ySteps=(yMax-yMin)/10;
zMin=min(plotData1,[],'all');
zMax=max(plotData1,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/10;

%% Plot two figures with east colorbar legend
tiledlayout(2,2);
% first figure
nexttile
s=mesh(xPlot,yPlot,plotData1,'FaceAlpha','0.5');
colormap(map);
title('3D Contour '+map1);
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);
cb.Layout.Tile='east';
% second figure
nexttile
s=mesh(xPlot,yPlot,plotData1,'FaceAlpha','0.5');
title('2D Contour '+map1);
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
```

```
yticks(yMin:ySteps:yMax);
zlim(zRange);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(2);
clim(zRange);

%% Plot second map for comparison
data=dlmread(map2,';');
xSize=data(1,3);
ySize=data(1,4);
xRes=data(2,1);
yRes=data(2,2);
xStep=xSize/(xRes-1);
yStep=ySize/(yRes-1);

% Build plot matrix and plot parameters
plotData2=data;
for i=1:3
    plotData2(1,:)=[];
end
xPlot=0:xStep:xSize;
xMin=min(xPlot,[],'all');
xMax=max(xPlot,[],'all');
xRange=[xMin xMax];
xSteps=(xMax-xMin)/10;
yPlot=0:yStep:ySize;
yMin=min(yPlot,[],'all');
yMax=max(yPlot,[],'all');
yRange=[yMin yMax];
ySteps=(yMax-yMin)/10;
zMin=min(plotData2,[],'all');
zMax=max(plotData2,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/10;

% first figure
nexttile
s=mesh(xPlot,yPlot,plotData2,'FaceAlpha','0.5');
colormap(map);
title('3D Contour '+map2);
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);
cb.Layout.Tile='east';
% second figure
nexttile
s=mesh(xPlot,yPlot,plotData2,'FaceAlpha','0.5');
title('2D Contour '+map2);
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(2);
clim(zRange);

%% compare board differences
figure(2)
```

```
tiledlayout(1,2)
nexttile
diffData(yRes,xRes,2)=zeros;
diffData(:,:,1)=plotData1;
diffData(:,:,2)=plotData2;
diffData=std(diffData,0,3);
zMin=min(diffData,[],'all');
zMax=max(diffData,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/10;
s=mesh(xPlot,yPlot,diffData,'FaceAlpha','0.5');
colormap(map);
title('3D Difference Contour - Remove and Reclamp');
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);
cb.Layout.Tile='east';
nexttile
s=mesh(xPlot,yPlot,diffData,'FaceAlpha','0.5');
title('2D Difference Contour - Remove and Reclamp');
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(2);
clim(zRange);
```

Published with MATLAB® R2023a

Bootstrap data generation and heightmap model file generation:

```
%% Reset
close all;
clear all;
clc;

%% Colormap
colormap(jet);

%% Init data and parameters
% Import *.map file and get mesh parameters for parsing and plotting
data(:,:)=dlmread("PCB01.map",';');
xSize=data(1,3);
ySize=data(1,4);
xRes=data(2,1);
yRes=data(2,2);
xStep=xSize/(xRes-1);
yStep=ySize/(yRes-1);
xPlot=0:xStep:xSize;
xMin=min(xPlot,[],'all');
xMax=max(xPlot,[],'all');
xRange=[xMin xMax];
xSteps=(xMax-xMin)/10;
yPlot=0:yStep:ySize;
yMin=min(yPlot,[],'all');
yMax=max(yPlot,[],'all');
yRange=[yMin yMax];
ySteps=(yMax-yMin)/10;
data(19,34)=zeros;
for i=1:1:10
    % Filename selections
    map_num="PCB0"+i+".map";
    % Import *.map file and get mesh parameters for parsing and plotting
    data(:,:,i)=dlmread(map_num,';');
end
% Correct matrix
for j=1:1:3
    data(1,:,:)=[];
end

%% Per-coordinate means
meanData=mean(data,3);
figure(1)
% recalculate z-parameters
zMin=min(meanData,[],'all');
zMax=max(meanData,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
s=mesh(xPlot,yPlot,meanData,'FaceAlpha','0.5');
title('Raw Mean Variance (variance from local average across set of 10 heightmaps)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Per-board %-error from mean
figure(2)
errorBoard(16,34,10)=zeros;
```

```
colormap(jet);
tiledlayout(2,5)
for i=1:1:10
    nexttile
    errorBoard(:,:,i)=(abs(data(:,:,i)-meanData)./mean(meanData,'all')).*100;
    % recalculate z-parameters
    zMin=min(errorBoard(:,:,i),[],'all');
    zMax=max(errorBoard(:,:,i),[],'all');
    zRange=[zMin zMax];
    zSteps=(zMax-zMin)/5;
    s=mesh(xPlot,yPlot,errorBoard(:,:,i),'FaceAlpha','0.5');
    title("Percentage Error From Mean - PCB #"+i);
    xlabel('X-position (mm)')
    ylabel('Y-position (mm)')
    zlabel('Error (%)')
    xlim(xRange);
    xticks(xMin:xSteps:xMax);
    ylim(yRange);
    yticks(yMin:ySteps:yMax);
    zlim(zRange);
    zticks(zMin:zSteps:zMax);
    s.FaceColor = 'flat';
    rotate(s, [0 0 1], 180)
    view(325,60);
    cb=colorbar('Ticks',zMin:zSteps:zMax);
end

%% Mean percent error, note: this is the same as standard deviation
figure(3)
meanError=mean(errorBoard,3);
colormap(jet);
% recalculate z-parameters
zMin=min(meanError,[],'all');
zMax=max(meanError,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
s=mesh(xPlot,yPlot,meanError,'FaceAlpha','0.5');
title('Mean Percent Error (variance from local average across set of 10 heightmaps)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Error (%)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Create additional samples using bootstrap method
bootData(16,34,1000)=zeros;
for i=1:1:1000
    for j=1:1:10
        % sample randomly from real dataset, sum
        bootData(:,:,i)=bootData(:,:,i)+data(:,:,randi([1 10]));
    end
    bootData(:,:,i)=bootData(:,:,i)/10;
end

%% Plot the mean of the bootstrap data
figure(4)
meanBoot=mean(bootData,3);
colormap(jet);
% recalculate z-parameters
zMin=min(meanBoot,[],'all');
```

```
zMax=max(meanBoot,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
s=mesh(xPlot,yPlot,meanBoot,'FaceAlpha','0.5');
title('Bootstrap Mean Variance (from bootstrap data with 1000 generated size=N samples)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Plot the std dev of the bootstrap data
figure(5)
stdDevBoot=std(bootData,0,3);
colormap(jet);
% recalculate z-parameters
zMin=min(stdDevBoot,[],'all');
zMax=max(stdDevBoot,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
s=mesh(xPlot,yPlot,stdDevBoot,'FaceAlpha','0.5');
title('Bootstrap Standard Deviation (from bootstrap data with 1000 generated size=N
samples)');
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Different between meanData and meanBoot
figure(6)
diffBoot=meanBoot-meanData;
colormap(jet);
% recalculate z-parameters
zMin=min(diffBoot,[],'all');
zMax=max(diffBoot,[],'all');
zRange=[zMin zMax];
zSteps=(zMax-zMin)/5;
s=mesh(xPlot,yPlot,diffBoot,'FaceAlpha','0.5');
title("Difference between meanData and meanBoot");
xlabel('X-position (mm)')
ylabel('Y-position (mm)')
zlabel('Z-variance (mm)')
xlim(xRange);
xticks(xMin:xSteps:xMax);
ylim(yRange);
yticks(yMin:ySteps:yMax);
zlim(zRange);
zticks(zMin:zSteps:zMax);
s.FaceColor = 'flat';
rotate(s, [0 0 1], 180)
view(325,60);
```

```
cb=colorbar('Ticks',zMin:zSteps:zMax);

%% Write out bootstrap model
% get map parameters from another map
bootModel=dlmread("PCB01.map",';');
for i=1:1:16
    for j=1:1:34
        bootModel(i+3,j)=bootData(i,j);
    end
end
% write to file using ';' delimiter
% header:
fid=fopen('bootModel.map','w'); % filename, write mode
fprintf(fid,"%d;%d;%d;%d\n",bootModel(1,1),bootModel(1,2),bootModel(1,3),bootModel(1,4));
fprintf(fid,"%d;%d;%.2f;%.2f\n",bootModel(2,1),bootModel(2,2),bootModel(2,3),bootModel(2,4));
fprintf(fid,"%d;%d;%d\n",bootModel(3,1),bootModel(3,2),bootModel(3,3));
% data:
for i=4:1:19
    for j=1:1:33
        fprintf(fid,"%.16f;",bootModel(i,j));
    end
    fprintf(fid,"%.16f\n",bootModel(i,34));
end
fclose(fid);
```

Published with MATLAB® R2023a

VI. References:

1. Denvi. (n.d.). Denvi/Candle: GRBL controller application with G-code visualizer written in Qt. GitHub. https://github.com/Denvi/Candle
2. George. (2023, March 10). How to utilize height mapping in candle. SainSmart Resource Center. https://docs.sainsmart.com/article/kj4xzak19j-how-to-utilize-height-mapping-in-candle
3. Seman, Jacob. "PCB Mesh Model - 3D Calculator." *GeoGebra*, www.geogebra.org/3d/gc2grpqr. Accessed 4 Dec. 2023.
4. Claudet, A., Tran, H., & Su, J.-C. (2008). Quantification of uncertainty in machining operations for on-machine acceptance. Office of Scientific and Technical Information (OSTI). https://doi.org/10.2172/945903
5. Ignacio Lira, George Cargill. Uncertainty analysis of positional deviations of CNC machine tools, Precision Engineering, Volume 28, Issue 2, 2004, Pages 232-239, ISSN 0141-6359, https://doi.org/10.1016/j.precisioneng.2003.06.001
6. Seman, Jacob, PCB_MFG, (2023), GitHub repository, https://github.com/Jbsco/PCB_MFG/reports/