Face Modeling Magic (Pre-Project Exploration)

Data Import

Data was exported from the creative camera into a simple csv of xyz coodinates.
Ideally the exported data would be a binary, but for the simple case an ASCII format was chosen.
Each Frame recieved its own sequentially numbered file.

It is possible and desirable to transfer the data to a write through buffer in the future. But for all practical purposes,
the data would not normally be written to disk with out a reason.

In[23]:=
```
FileNameSetter[Dynamic[dir], "Directory"]
```

```
Select directory and store it in a var named dir
```

Out[23]= Browse...

Some Light Pre-processing

Get rid of background data, and store them in a mathematica binary for quick loading.

```
GetFiles[filter_] := Module[{files = {}, fileName = "", temp = {}, index},
    SetDirectory[dir];
    files = FileNames["*.csv"];
    For[index = 1, index < Length[files], index++,
     temp = Import[StringJoin[dir, files[[index]]]];
     temp = Select[temp, #[[3]] < filter &];
     Export[StringJoin[dir, "frame", ToString[index], ".mx"], temp];
    ];
   ];
```

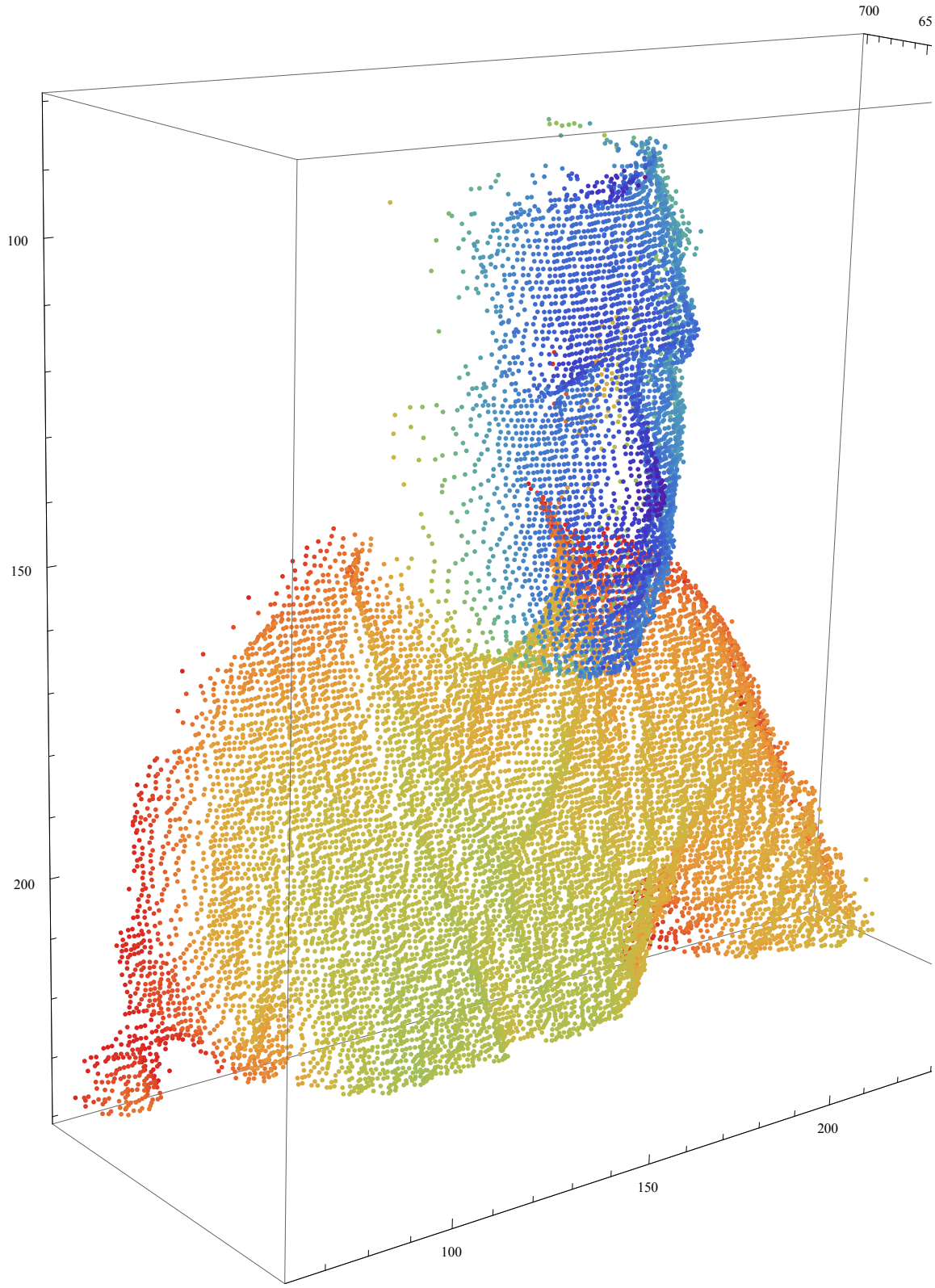
In[49]:=
```
GetFiles[700]
```

A Simple Viewer

In[72]:=

```
files = FileNames["*.mx"];
vdat = Import[StringJoin[dir, files[[60]]]];
ListPointPlot3D[vdat, ColorFunction → "Rainbow"]
```

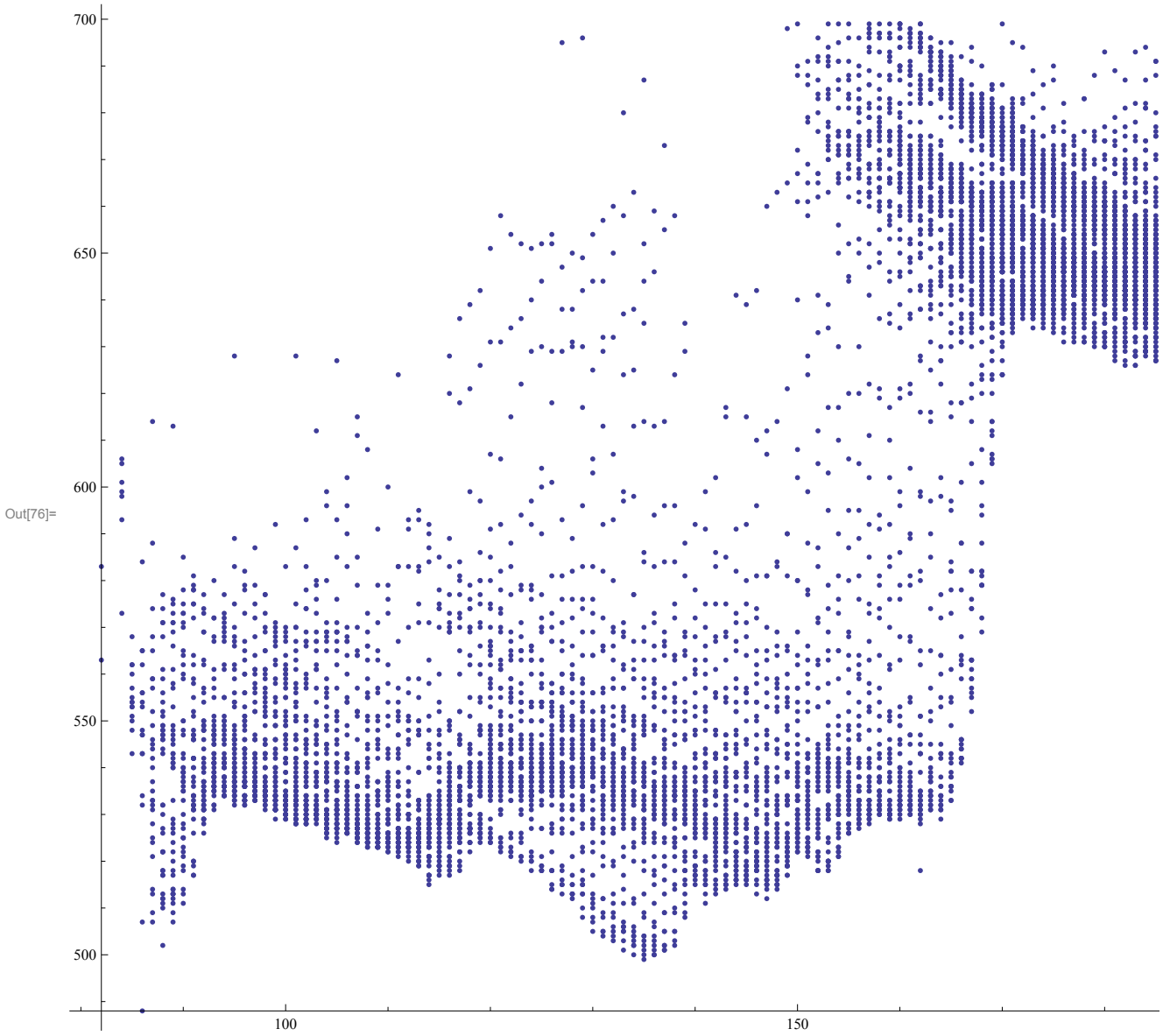
Out[74]=



2D Side Profile

In[76]:=
```
ListPlot[vdat[[1 ;;, 2 ;; 3]]]
```

Out[76]=



Selecting a fixed point to put at the Origin

In this case the tip of the nose

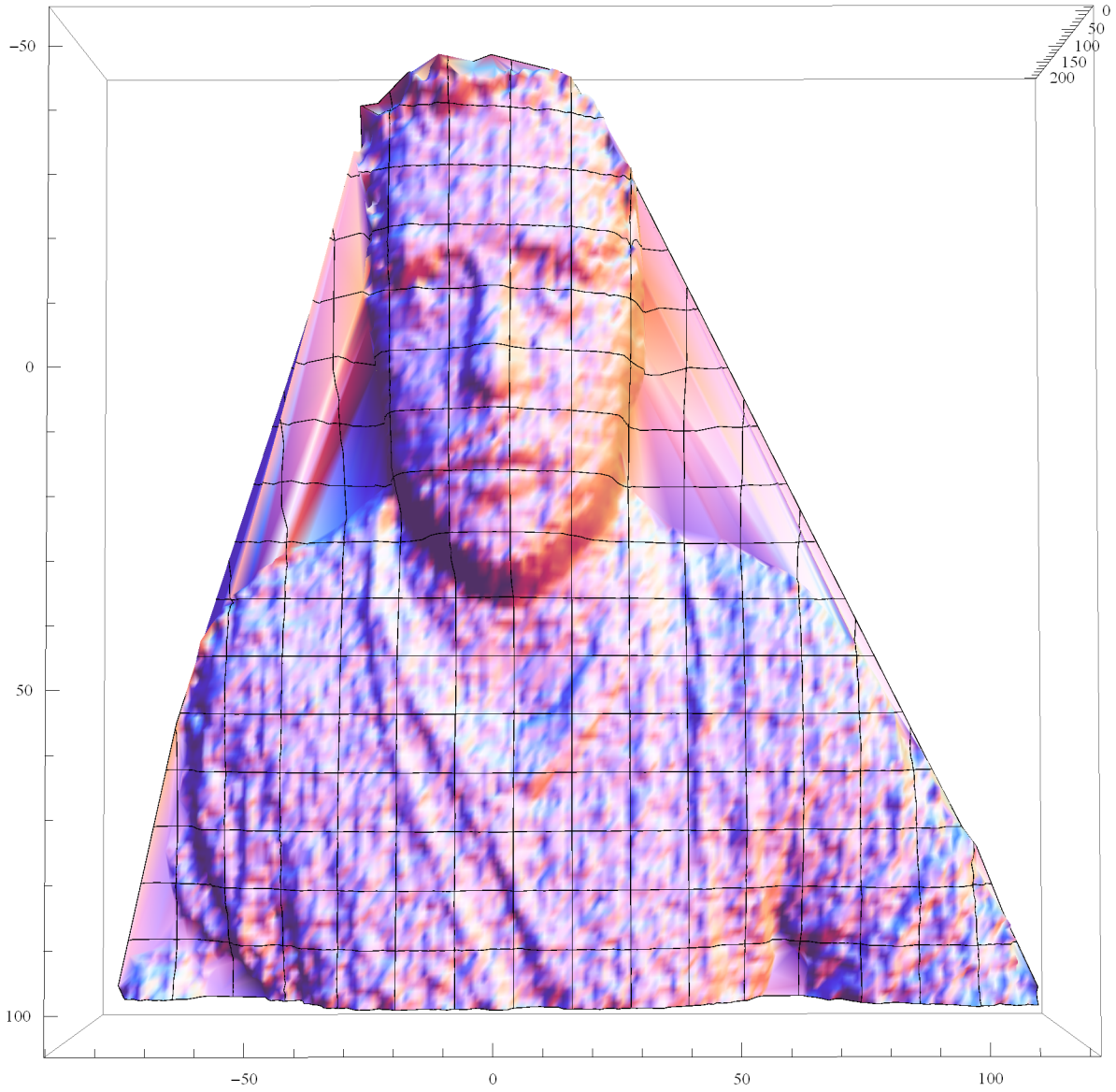
In[86]:=
```
Position[vdat[[1 ;;, 2 ;; 3]], {135, 499}]
```

Out[86]=
```
{{2757}}
```

In[89]:=
```
ovdat = vdat;
ovdat[[1 ;;, 1]] = ovdat[[1 ;;, 1]] - vdat[[2757, 1]];
ovdat[[1 ;;, 2]] = ovdat[[1 ;;, 2]] - vdat[[2757, 2]];
ovdat[[1 ;;, 3]] = ovdat[[1 ;;, 3]] - vdat[[2757, 3]];
```

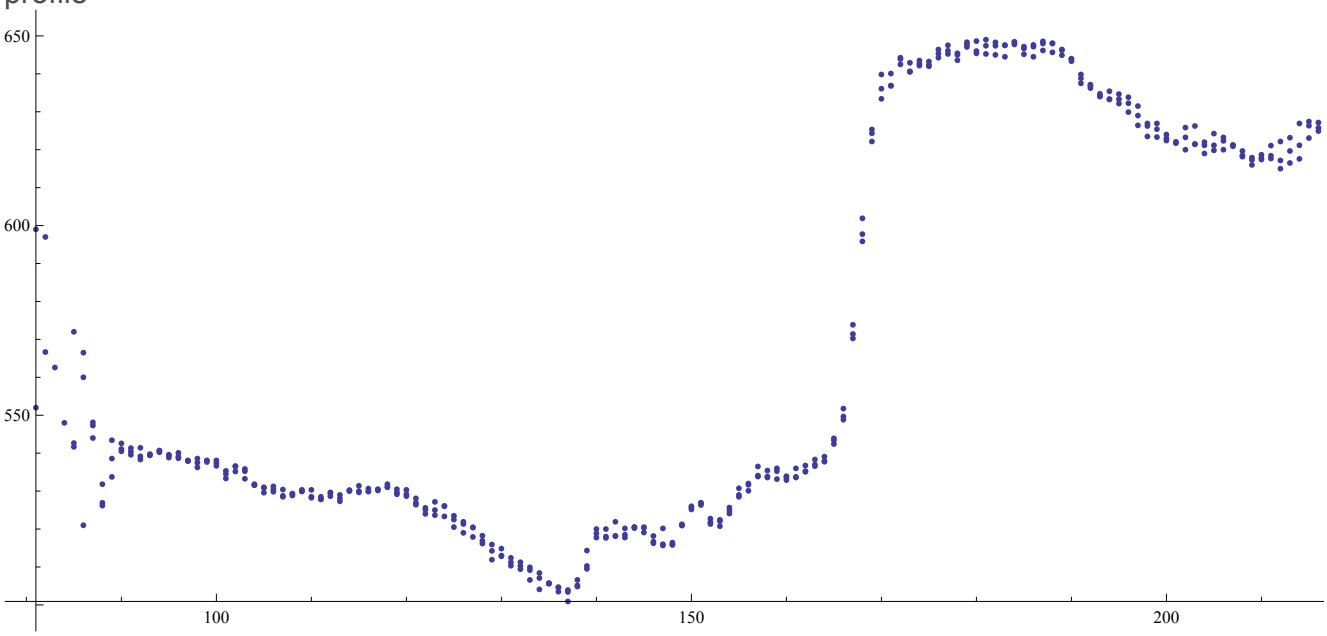
In[96]:=
```
ListPlot3D[ovdat, PerformanceGoal → "Quality"]
```

Out[96]=

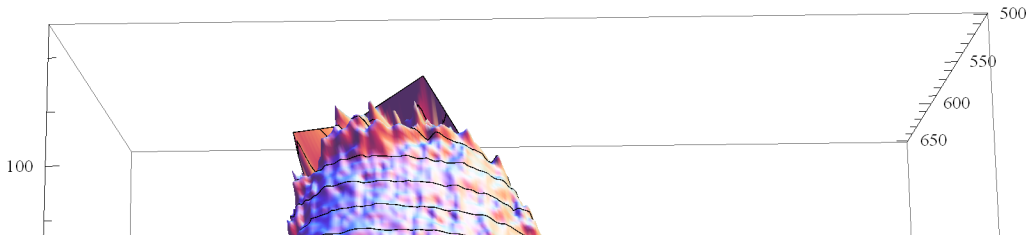


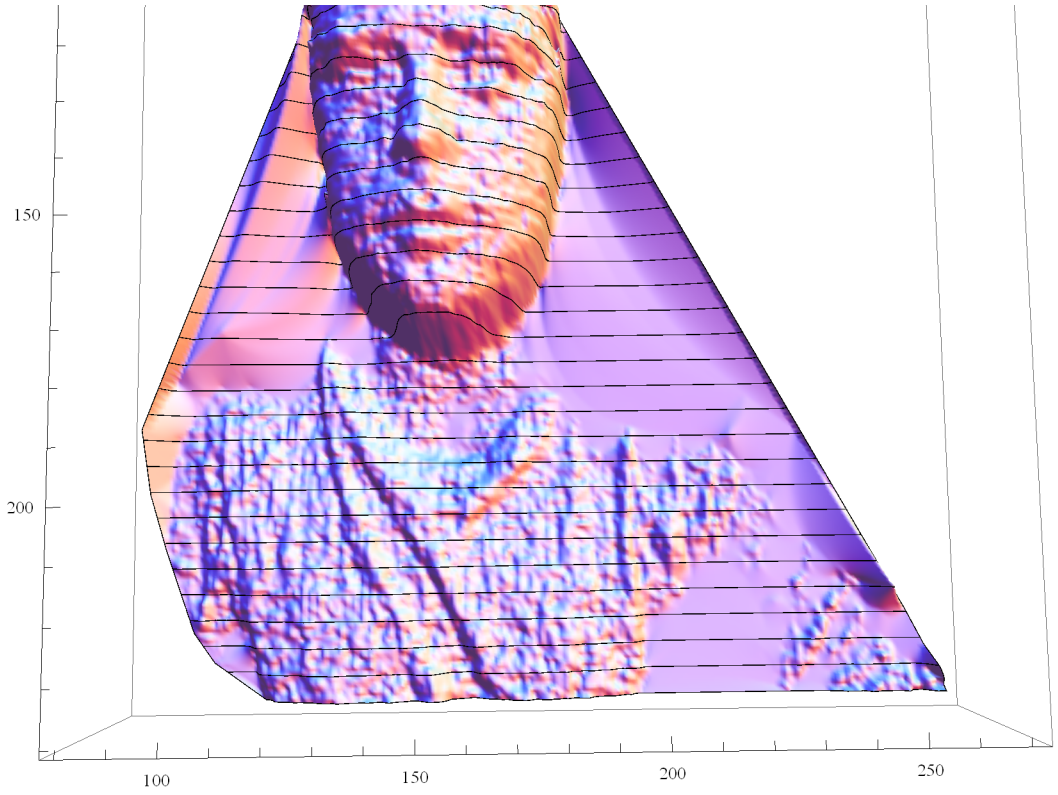
TODO:: Find another 1 or two fixed points to align to an axis, The SDK has some built in features that pick out certain points. Such as the corners of the eyes and mouth.

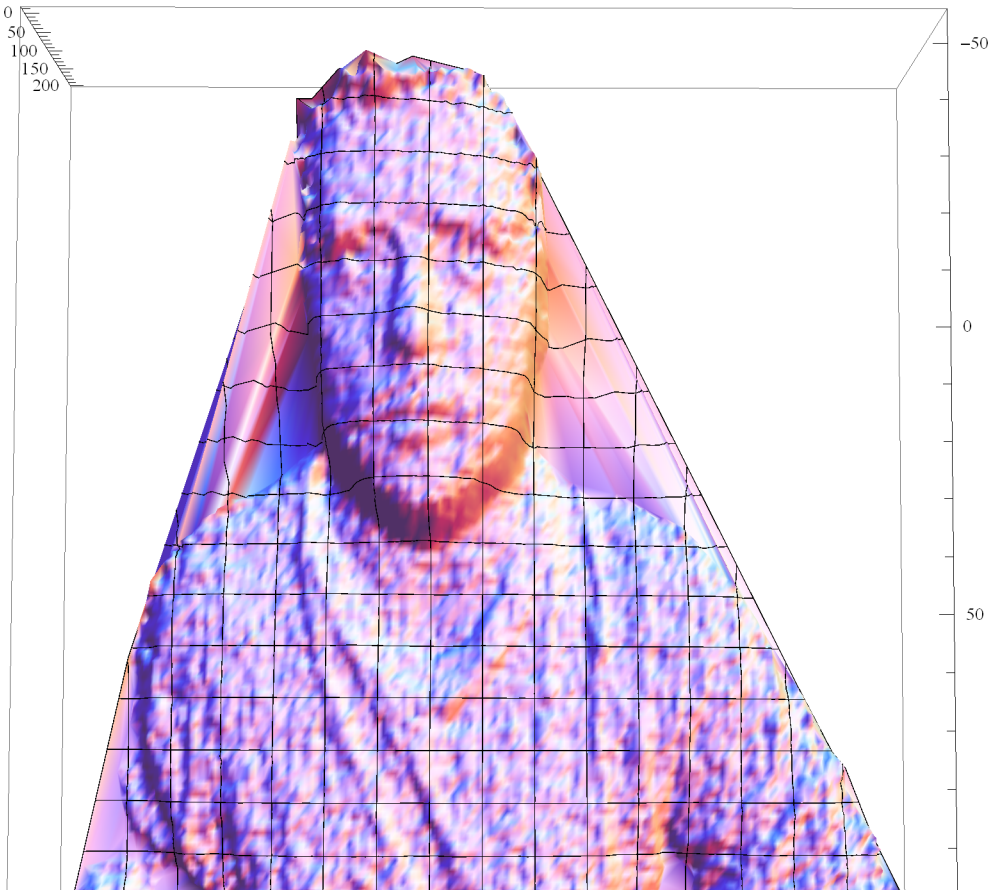
Looks good for contour matching based on a 3 px wide profile

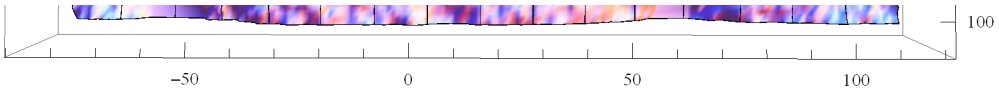


Example of me holding still (That is ‘human still’) and doing pixel for pixel averaging based on 10 Frames.
Without Correction for position.Next to a single frame with no correction. Mind you that the resolution point size is 340 pixels wide
for the whole frame, of which we are about 75 pixels for the face. But with accurate pixel orientation based off points collected
from the 720p image. We can increase the effective number of pixels through multiple frames. Which could take these models and make them much higher quality. Then adding smoothing.




150
200
100
150
200
250


0
50
100
150
200
−50
0
50



The apperant shroud arround the figures is based on the convex hull of the data and can be ignored.
This is just for fast prototyping.

So the first pass generic alogorith that I see based on contour is fairly straight forward.

Make a 3D model of the client

This includes scanning multiple profiles (basically turning the head)
Orienting these profiles to a common orientation based on fixed points.
Combining these to make a model (Effectively increases point density, and more points due to different angles of view.
Some simple smoothing to eliminate jitter due to equipment/resolution.

Scan the subject

Shove the data live data feed into a buffer.
Determine the orientation of the face.
Orient the data based on a simple transform.
Slice the face into Horizontal and vertical slices. (A few pixels wide, and only as many as neccesary)
Compare these slices with the model. (Could be done in parallel).
Come up with some rudementary correlation characteristics.
Give a proxy of goodness of fit.

Report Results

Other nifty things that we will need to develop is some GUI stuff,
Live view, Model View, Debug Terminal, Client name information, Data Structures... etc.

Most of these are prebuilt into the SDK so they are easy.

Above all I think we should use all available tools in the SDK and other libraries as possible
Instead of spending precious time reinventing the wheel.