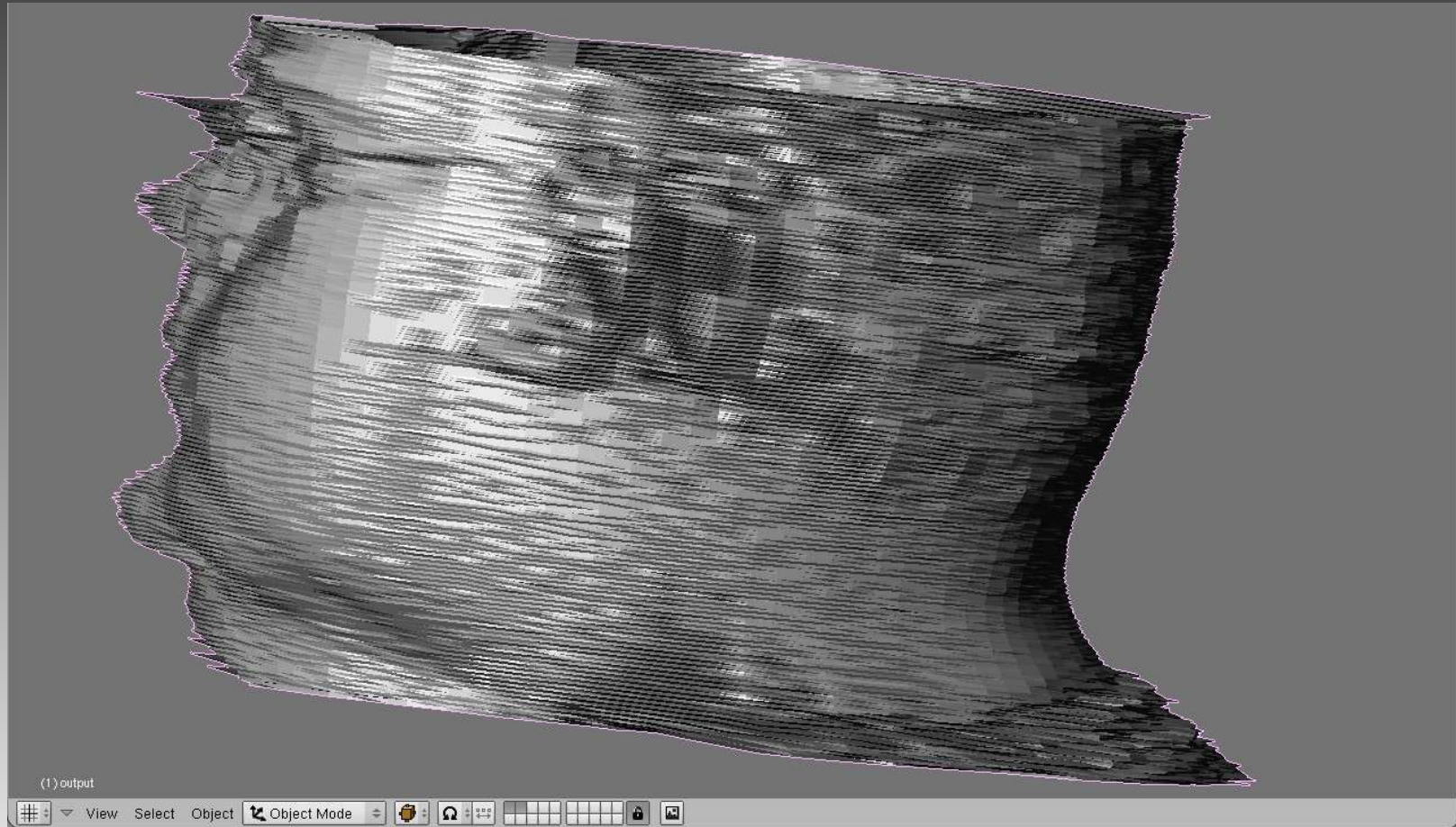


Why Most 3D Scanners are Bad

A slightly disturbed view of 3D scanning



Andrew 'splineboy' Lewis

Why Most 3D Scanners are Bad

- What sort of things are out there?



Why Most 3D Scanners are Bad

- Why Most 3D Scanners Suck
 - Assume that processed $[x,y,z]$ data is random
 - Require finely calibrated machinery
 - Only process data once (with some exceptions)
- Why Lasers Suck
 - Laser speckle is the devil's lava-lamp
 - You only have 2 Eyes, so only 1 mistake allowed
 - Expensive
 - Need focussing, lenses, smooth +ve – blah, blah, blah.

Why Most 3D Scanners are Bad

- Why Motion Control Sucks
 - Needs opto-isolator (Feedback = Dead TTY ports)
 - Mechanical problems (Inertia, Momentum, Gearing)
 - More code, less fun.
- Why Monolithic Applications Suck
 - Code Swamps
 - Lack Flexibility for General User
 - Can't incorporate new technology easily
 - Dependencies, cross compiling, etc....

</RANT>
<IMPROVE>

Making a Better 3D Scanner

- Kick out Motion Control
 - Use *Visual Cues* in the image to detect frames
 - Use slow rpm turntables
 - Get software to calculate the rpm from cues
- Kick out Lasers
 - Use barn-door lights & a 'light-painter' to get neat lines
 - Luxeon-V LED
 - Super-bright
 - Cheaper than laser
 - Variety of Colours

Making a Better 3D Scanner

- Make it open and use modular 'pathways'
 - Several Applications in a chain:
 - Streamer -> PPM2BMP -> Scan.exe -> Blender
 - Open & modular = easy to change
- Can be linked by script:

```
streamer -o /raw/face0000.ppm g-c /dev/video0 -r 25 -s 320x240  
-t 150 -n pal -i 1
```

```
for capfile in /raw/*.ppm
```

```
do
```

```
    ppmtobmp -windows -bpp 24 $capfile > $capfile.bmp
```

```
done
```

```
wine /scanner/scan.exe
```

```
blender
```

Making a Better 3D Scanner

Reusing existing raw sensor data

- **The 4Pass Splinescan system**
 - **Pass #1: Luma Pass**
 - Captures the Laser Line
 - Uses Threshold Brightness Level
 - Can be Colour Range Specific
 - **Pass #2: Chroma Pass**
 - Capture the Texture map
 - Trace Behind the Laser Line
 - Allows for Object Distortion

Making a Better 3D Scanner

Reusing existing raw sensor data

– Keying Pass

- Capture the Background
- Creates a 'Negative Object'
- Boolean with Luma Model to Punch Holes

– Blacklight Pass

- Capture the UV Map
- Like Chroma Pass, but uses UV Light

THE BIT WITH THE MATHS IN

Don't Panic!

It's not that bad. I hate maths, too.

WHAT'S IMPORTANT?

- Determining object shape
 - Trig
 - Basic algebra
 - The cartesian co-ordinate system
- Smoothing raw data
 - Averaging
 - Some set theory

WHAT IS IMPORTANT?

- Reconstructing missing data
 - Trig
 - Basic Algebra

- Joining up the dots
 - NURBS
 - Too much to cover in this lecture, see “Rasters vs. Ducks in Spline Cages” for more info

How Does it Work?

Determining object shape

- The scanner projects a line onto a rotating object.
 - The first step is to take the 2D line data and revolve it around the y-axis to generate a 'cage'
 - Basic 3D Rotation around Y axis:
 - $Y = y$
 - $X = z(\sin(r)) + x(\cos(r))$
 - $Z = z(\cos(r)) - x(\sin(r))$
- (r is the angle of rotation around the y-axis.)

How Does it Work?

Determining object shape

- We can do something quite clever here.
 - The scanner always starts with a 2D image
 - So, the z value is always 0
 - The formula gets even more basic:
 - $X = 0(\sin(r)) + x(\cos(r)) \quad \text{--->} \quad X = x(\cos(r))$
 - $Z = 0(\cos(r)) - x(\sin(r)) \quad \text{--->} \quad Z = -x(\sin(r))$

How Does it Work?

Smoothing with Proximity Weighted Surface Average

Relevance of each point decreases as distance from original point increases. You perform this for every point on the data cloud.

Consider these sets:

$$\mathbf{p} = [x, y]$$

$$\mathbf{a} = \mathbf{p} + [x-1, y : x+1, y : x, y+1 : x, y-1]$$

$$\mathbf{b} = \mathbf{p} + \mathbf{a} + [x-1, y+1 : x-1, y-1 : x+1, y+1 : x+1, y-1]$$

$$\mathbf{c} = \mathbf{p} + \mathbf{a} + \mathbf{b} + [x-2, y : x+2, y : x, y+2 : x, y-2]$$

This results in: $Z_{[x,y]} = \text{Average}(\mathbf{c})$

