Image-Based Modeling of Curved Objects

Michael Schuresko

mds2@andrew.cmu.edu

Abstract

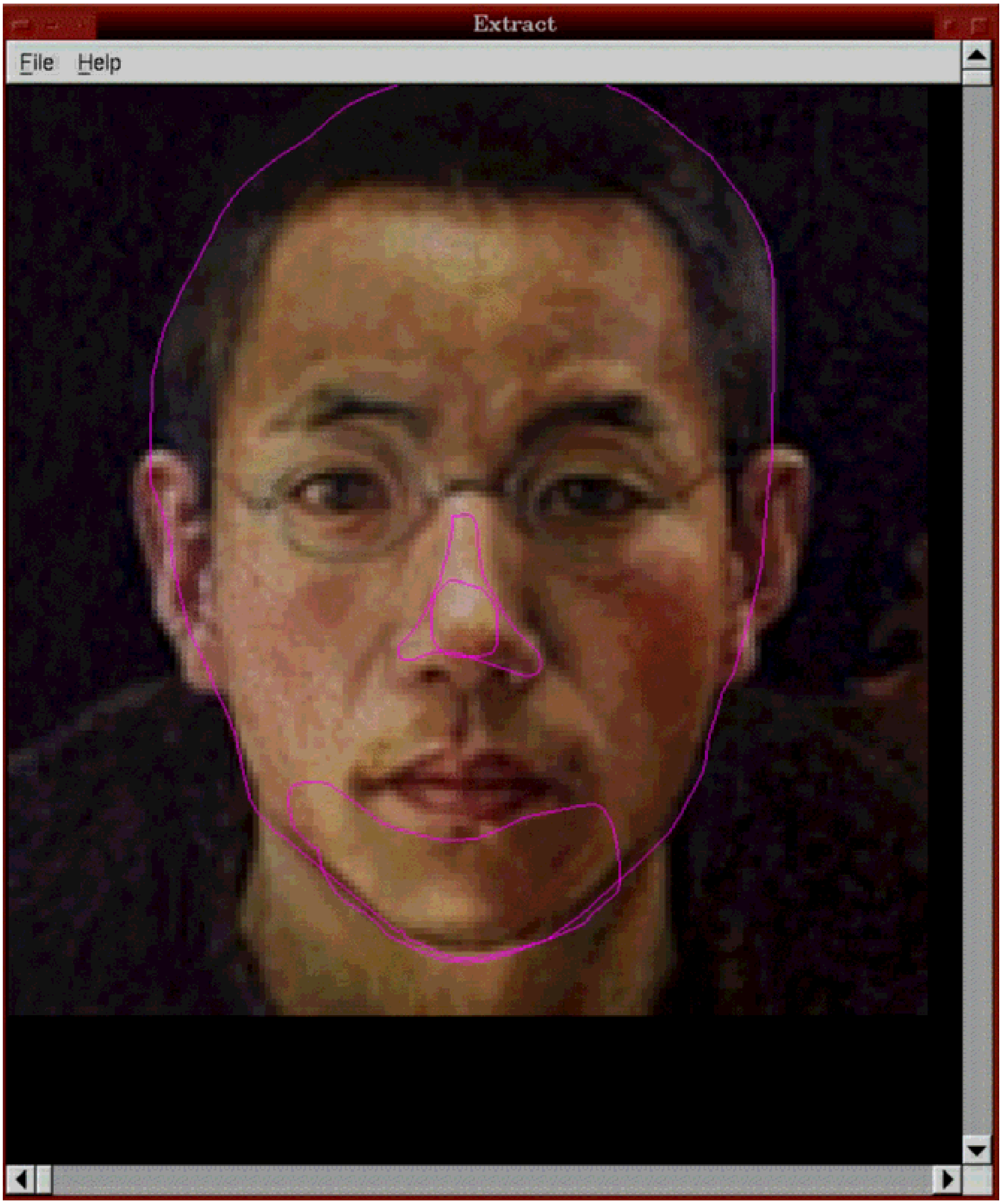
There already exist many single-view modeling programs for architectural scenes with heavy planar geometry. There are also non-view based modeling programs for non-geometric curved objects (particularly relevant to this paper is Takeo Igarashi's work on Teddy [1]). This work combines the two ideas to make an image based modeler for non-geometric curved objects. The program has been tested on images of human faces among other things. While the algorithm is much more general then a face modeler, I find that faces provide a rigorous test of my hypothesis, as a small error in a face image is readily apparent to a human viewer.

Purpose

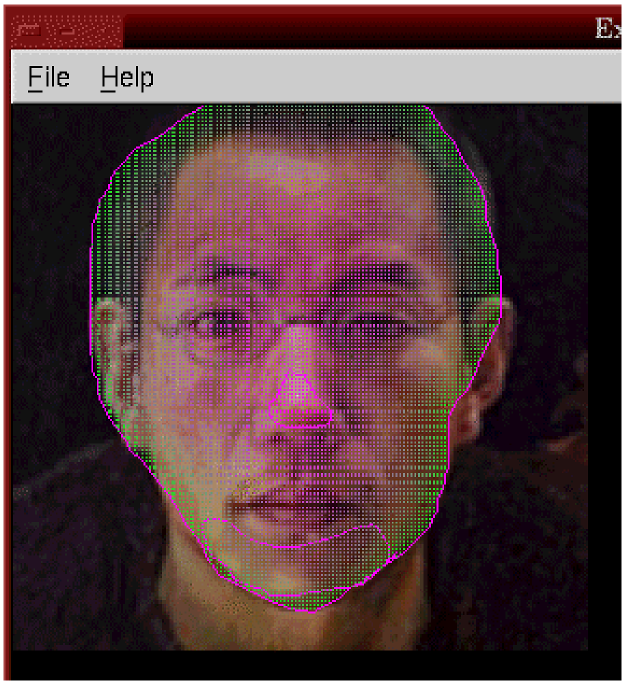
This paper presents a method for modelling simple curved surfaces from a single image. The premise of the work is that the geometry of an object can be radically simplified and still produce convicincing new views, provided the object is textured with a detailed photograph or image. The program described in this paper provides a simple interface to model a small set of shapes, and accurate texture them with an image or painting. The program then allows the user to rotate the resulting model to produce novel views of the image.

Interface

The first portion of the interface is the actual modeler itself. The user uses a mouse to sketch out closed loops, reminiscent of contour lines. The loops do not, however, act as contour lines per se. Rather the area enclosed by each loop is bumped out as on ovloid shape which is represented as a height field on top of the 2d image (this idea was based on Takeo Igarashi's work [1]). The resulting height fields from each loop are summed to form a final height field for the image. A screenshot of sample input is shown below.



The loop around the chin region is necessary, because the human face isn't a perfect oval. The slight "bumping out" of the chin area deforms the oval to be closer to the human face shape. A sample heightfield is shown below. In this heightfield, green represents heights closer to 0 (with respect to the image plane), while purple represents heights closer to the maximum of the heightfield.



The second portion of the interface is the 3d rotation tool, it takes in the photo-with-heightfield from the modeler, and allows the user to interactively rotate the image and save the result as a tiff.

Algorithm

Each loop that the user sketches is made into a bounding polygon. This program differs significantly from Teddy[1] in the algorithmic details. First off, the polygon sides aren't normalized to be the same length. Second, there are no tests for self-intersection, the algorithm used here can use self-intersecting polygons as easily as it can use single-loop polygons. Membership in the polygon is determined by the parity of the number of line segments crossed when drawing a line between <0,0> and the pixel being tested (some additional tests are used to catch the degenerate cases where two line segments meet). The heightfield is initialy set to

*Height(thisPixel due to thisPoly) := isInside(thisPixel, thisPoly) * closestDist(thisPixel, thisPoly)*

This process could be greatly sped up by precomputing the constrained Delaunay triangulation of the polygon (in fact, could be converted to a really simple scanline algorithm). While the heights are being computed, a variable keeps track of the maximum distance of any interior point to the polygon border. This is stored in a variable called maxheight. The heights due to each polygon are transformed by

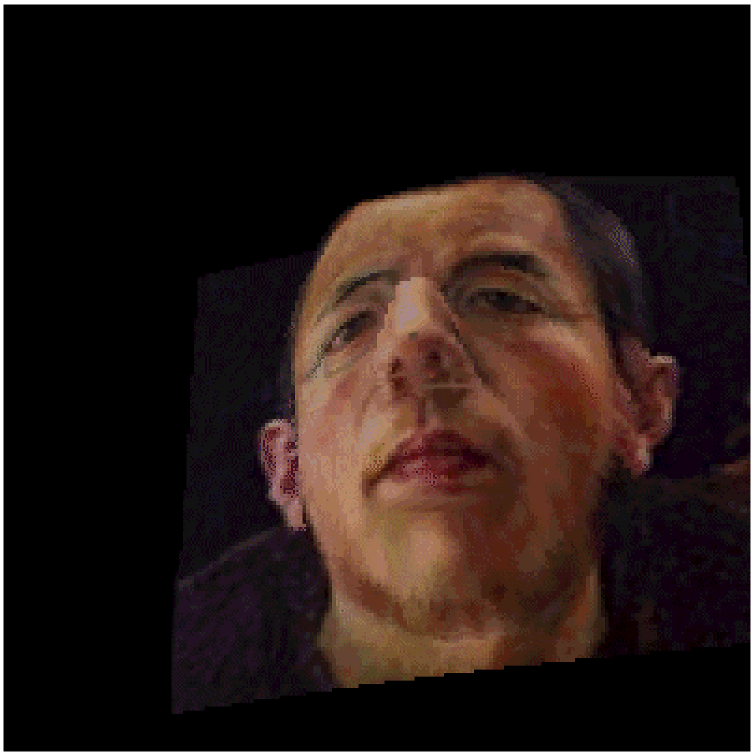
*Height(thisPixel) := sqrt(maxheight*maxheight - (maxheight-Height(thisPixel))*(maxheight-Height(thisPixel)))*

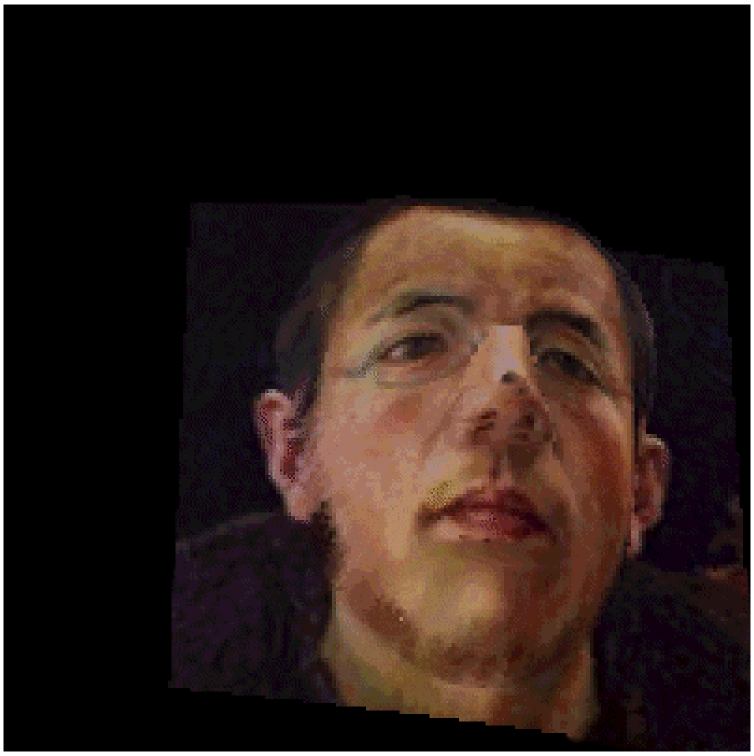
This makes the entire shape into a piecewise mesh of partial spheres, with radius equal to maxheight. The shape is C_0 continuous, and as the number of points on the polygon approaches infinity, the shape approaches C_1 continuity. The shape due to each polygon is then summed toghether (making it easy to get noses, etc.). Note that the sum of these height-fields (which is the final result) is usually not C_1 continuous, but still has garrunteed C_0 continuity.

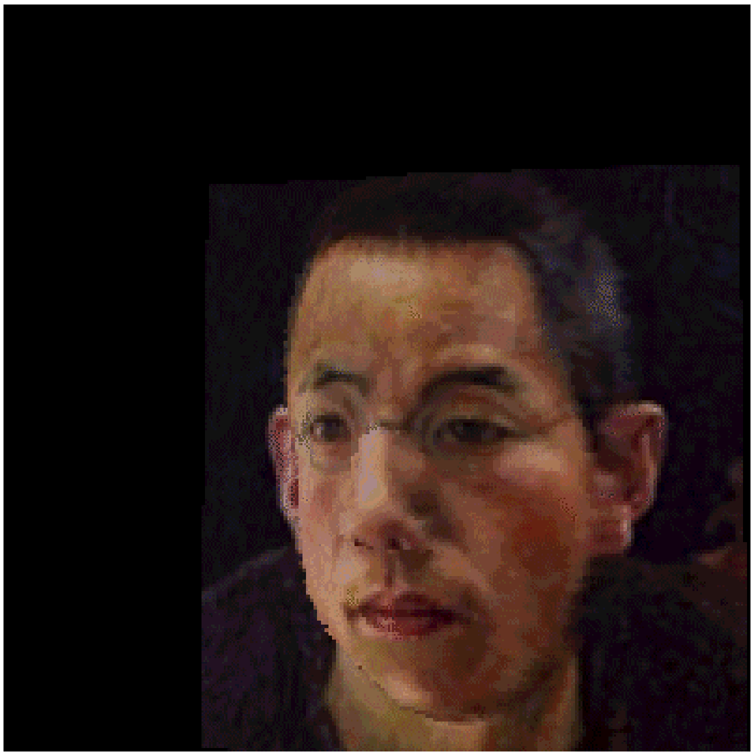
Once the height-field has been computed, the heights are converted to "generalized disparity" values, so that they can be viewed using McMillan's algorithm for 3d viewing using image-warping [2].

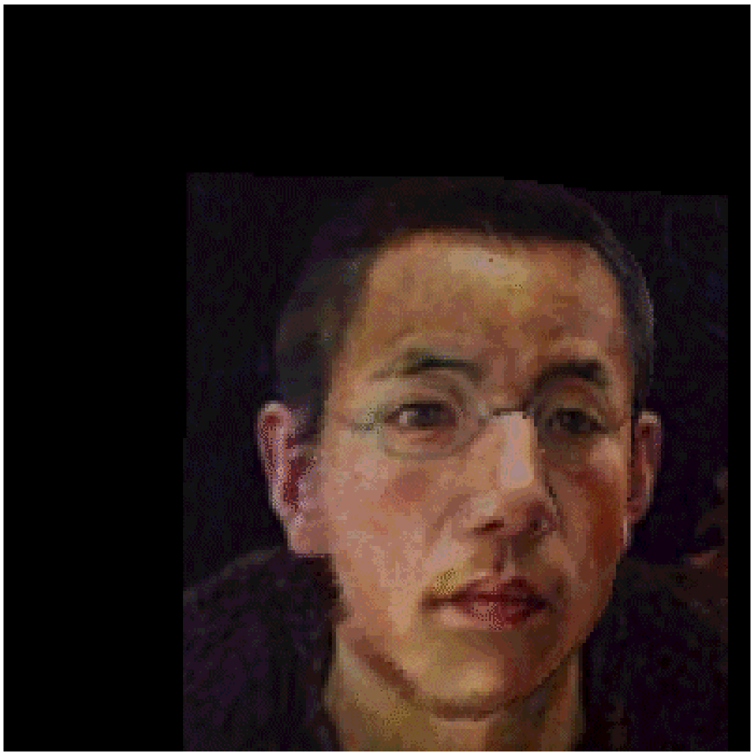
Results

Kohno Michisei Self Portrait

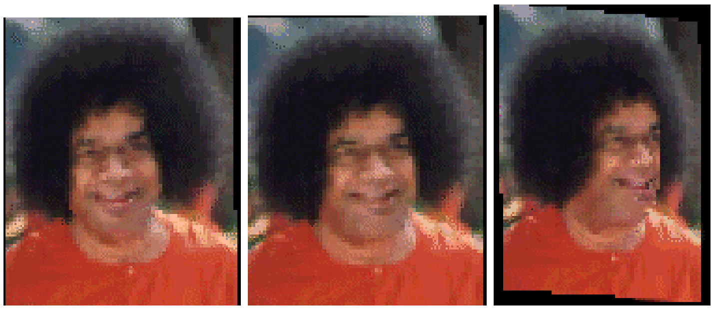






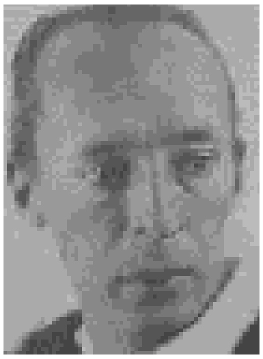
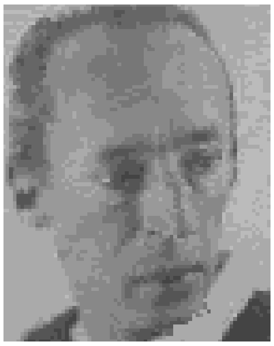


His Holiness Sri Sathya Sai-Baba, A New (Age) Perspective



Found object, Starbucks Coffee Canister

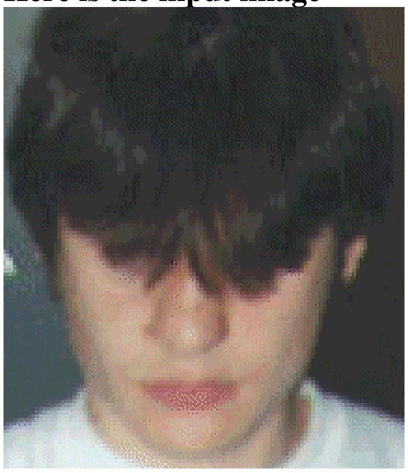
Vladimir Nabokov


Algorithm flaws

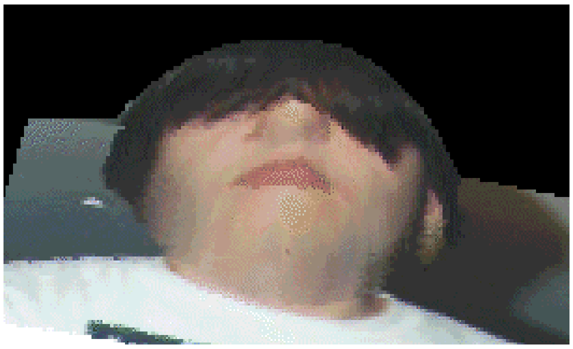
Because the premise of the work is that detailed textures can make up for poor or slightly innacurate shape information, the algorithm does create poor / innacuratte shape information. When too few contours are drawn, or when a shape is rotated to extreme positions, this becomes readily apparent. Here is an example.

Here is the input image



Here is the image, rotated a little bit

Here is a view rotated to extreme positions



Future Work

The first piece of future work to be done would be to speed up the heightfield creation process, by pre-processing polygons into Delaunay triangulations. The other remaining area of interest is texture synthesis for hole-filling. An approach that I did not have time to implement was to find a scalar field that yields the heightfield as an implicit surface, using numerical finite element methods, and then rewarp the surface onto a bounding cube by following the gradient of the scalar field from every point on the object surface, to the bounding cube. Texture synthesis can then be used on the bounding cube and reprojected back onto the 3d object. Also, missing shape detail could be synthesized by extending traditional texture-synthesis methods into 3d, possibly interpolating 3d shape and color as a hypertexture.

This paper is available via http://www.cs.cmu.edu/~ph/869/results/schuresko/final_project/report.html

---

References

1. Takeo Igarashi, Satoshi Matsuoka, Hidehiko Tanaka, **Teddy: A Sketching Interface for 3D Freeform Design**
   available at http://www.mtl.t.u-tokyo.ac.jp/~takeo/teddy/teddy.htm
   Presented at SIGGRAPH 99
2. Leonard McMillan and Gary Bishop, **Head-tracked stereoscopic display using image warping**
   available at http://graphics.lcs.mit.edu/~mcmillan/Publications/spie95.ps
   Stereoscopic Displays and Virtual Reality Systems II,
   SPIE Proceedings 2409, Feb. 1995

Michael D Schuresko
Last modified: Sun Dec 5 21:12:32 EST 1999