An Interactive Tool Suite for Embossing 2D Images

George Antonopoulos
Department of Computer Science,
University of Ioannina
Ioannina, Greece
email: gantonop@cs.uoi.gr

Ioannis Fudos
Department of Computer Science
University of Ioannina
Ioannina, Greece
email: fudos@cs.uoi.gr

***Abstract* -- We report on the development of a tool suite for rapidly creating high quality 3D embossed portraits using a single 2D snapshot. Our goal is to establish a mapping from 2-D to 3D and then interactively edit the embossed 3D surface and correct any artifacts that have occurred. To establish the initial mapping, we use an enhanced grayscale representation of the image and then employ techniques such as edge detection, region growing and interactive-iterative enhancement to obtain an flawless 3D embossing surface. Finally, we present an evaluation of the usability of our embossing tool suite.**

Keywords -- embossing, usability, lightness, luminance, region selection, region growing, interactive

I. INTRODUCTION

Rapid embossing of portraits and patterns is a process central to jewelry design. Usually, it is not possible to extract accurate information for a 3D representation with our only data source being a 2D snapshot. Nevertheless, a 2D image contains different types of information which we exploit to achieve our goal. The information of the image that we use to obtain an initial embossing surface is: a) the XY-plane distribution b) the luminance, c) the edges, d) the value of the pixel colors. For each of these information types, we adapt several methods for their extraction and interpolation and we describe how they can be used to produce a 3D embossing surface.

There are several commercial applications which perform the task of embossing. All of them provide some initial result but inadequate or no tools for further model refinement. The most renowned one is ArtCam [11] which yields very good results, has some refinement tools, but does not provide the fast detailed local control tools as our approach does. Another commercial product is MeshCam [10] which follows a more traditional CAD/CAM design process and is fit for designing sophisticated patterns capable of being embossed by accurate CNC machines.

Other products are, 3DMaker [12], VS3D [13] and Geomagic Studio [14] which employ a global conversion method with options for improving the final embossing surface but lack the tools for further interactive editing and improvement. Finally, classical embossing techniques in the context of image processing create an anaglyph of the edges of an image and therefore cannot be compared to our tool suite.

Holoimages [15], an elaborate technique, has also been proposed for the embossing task. It involves the capturing in a picture of the model to be embossed, along with a light interference pattern projected on it. The processing of the resulting image uses the pattern projected on the object to extract its depth and provide us with an embossing. This method uses extra information derived from the deformation of the projected light pattern on the object. Our method uses a single 2D snapshot of the figure that we wish to emboss.

The rest of this paper is organized as follows. Section 2 presents the initial transition from 2D to 3D using a straightforward mesh creation based on the image size, the luminance and the lightness attributes. Section 3 introduces our interactive region selection algorithms, while Section 4 describes the employment of smoothing algorithms for eliminating artifacts and improving the quality of the overall embossing surface. Section 5 presents an evaluation of the usability of our technique and finally Section 6 offers conclusions.

II. OBTAINING THE INITIAL 3D MAPPING

The first step in the creation of our embossing is the transformation of the picture from 2D to 3D. In order to achieve that we a) extract 3D points from the picture, b) triangulate these points.

This is accomplished by considering the image as a uniform point distribution in 3D with the Z coordinate (depth) pinned to 0. Thus each pixel is mapped to a point in 3D.

As for the triangulation, it is straightforward to obtain a triangulation with congruent triangles: every 4 pixels form an Undirected Cyclic Graph containing 2 triangular facets. Fig. 1 depicts this concept:

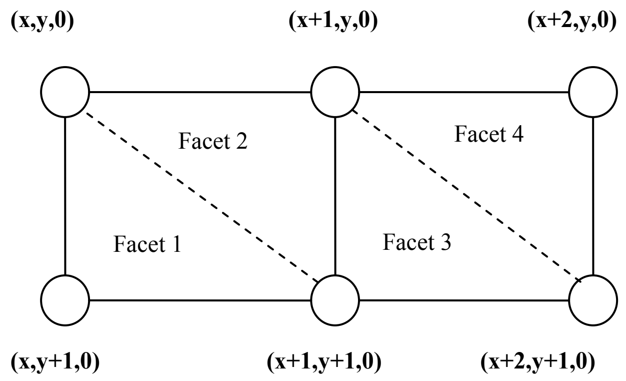


Figure 1. Triangulation

For an image of width w and height h, we get:

$$2(w-1)(h-1) \tag{1}$$

triangular facets.

The basic step of the whole process is based on the, seemingly rational, observation that those areas of a portrait positioned higher should be more intensively lit than those at lower height. What we need is a mean of calculating the quantity of light that a pixel carries. That mean is “luminance” and, the process of deriving it is “lightness”. In the rest of this section we describe how we use these measures to produce an initial approach of our embossing and how we can refine the results of this approach.

A. Lightness on Grayscale

The color of an image pixel is a composition of the three basic colors (R) red, (G) green, (B) blue, with each one ranging from 0 (minimum intensity) to 255 (maximum intensity). Thus, the color attribute is a vector in the 3D RGB space.

We use the “Generalized Lightness Hue and Saturation (GLHS)” [1] model where the luminance Y of a pixel C springs from the following formula [2]:

$$Y=0.2125R+0.7154G+0.0721B \tag{2}$$

Y is linear on the RGB color elements of light and the coefficients are determined experimentally from the real-world sensitivity of the human eye to each element of light. We use luminance in order to obtain lightness. According to CIE, Lightness (L^*) of a pixel is given by the following formula [8][9]:

$$L^* = 116 f(Y/Y_n) - 16 \tag{3}$$

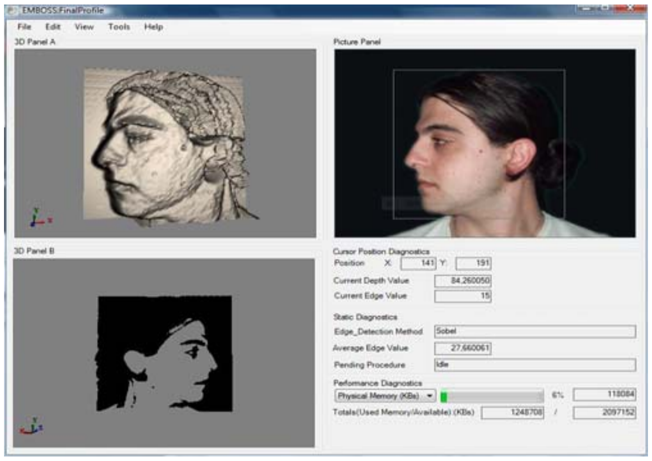
where

$$f(\frac{Y}{Y_n}) = (\frac{Y}{Y_n})^{\frac{1}{3}},\ (\frac{Y}{Y_n}) > (\frac{24}{116})^3$$
$$f(\frac{Y}{Y_n}) = (\frac{841}{108})(\frac{Y}{Y_n}) + \frac{16}{116},\ (\frac{Y}{Y_n}) \le (\frac{24}{116})^3 \tag{4}$$

Y_n is Luminance quantity which serves as a reference to which the Luminance Y of the current pixel is compared. The Luminance of white serves as a reference in our calculations which means that we set R=G=B=255 in equation (2). The above method yields a grayscale representation of our picture, where the color value of each pixel is equal to its Lightness.

By using equation (3) we have computed the quantity which intuitively would represent the depth of an image in 3D. The result of this process is shown in Fig. 2. There we can see the initial approach of our method to the given picture where Lightness is used to provide the depth of the corresponding pixel.

At the top-right is the image used for embossing. The top left panel illustrates the resulting 3D representation.



Cursor Position Diagnostics
Position X: 141 Y: 191
Current Depth Value 84.260050
Current Edge Value 15

Static Diagnostics
Edge_Detection Method Sobel
Average Edge Value 27.660061
Pending Procedure Idle

Pefomance Diagnostics
Physical Memory (KBs) 6% 118084
Totals(Used Memory/Available) (KBs) 1248708 / 2097152

Figure 2. Initial approach

B. Refinement

The results of our initial approach are often unintuitive and undesirable. Notice large distance between the highest point and the XY plane (Depth=0) and the fact that neighboring points may exhibit large depth variations.

To this end, we perform a Z axis scaling so as to reduce pixel depths without affecting the image ratio. This is realized by dividing all depths by the same number. Fig. 3 shows the results of the scaling for various scaling.

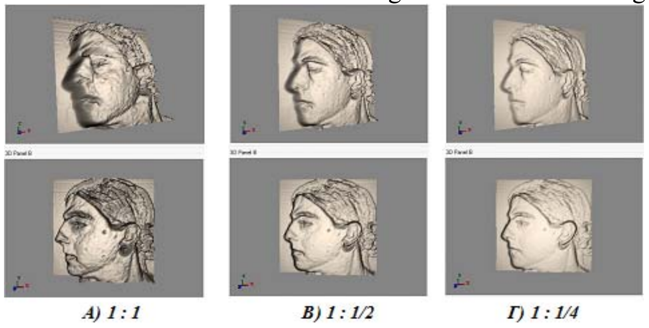


Figure 3. Scaling Trials. a) Division by 1 (initial), b) Division by 2, c) Division by 4

As we can see scaling considerably improves the initial 3D mapping. Note that for very small scaling factors the 3D mapping will become almost flat.

III. REGION SELECTION TOOLS

The initial 3D embossing surface may contains serious imperfections and artifacts. These mostly appear in areas with low lightness, which is a result of their color (spots, moles, hairy parts etc). Therefore, we need tools for determining such areas and then interactively editing their morphology. We exploit to two more types of information that an image provides: a) edges, and b) pixel color. We shall also provide a method for free sketching to give the user the ability to manually define a region for editing.

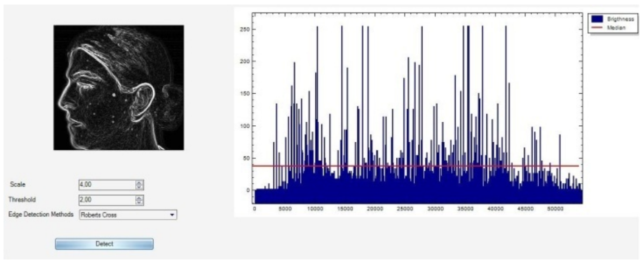
A. *Edges and Edge Detection*

Edges can be used to spatially define areas which, due to their color, yielded a false initial approach. To detect which pixels fall within edge regions, we use the following common Edge Detection Algorithms: a) Robert's Cross Edge Detection [3], b) Sobel Edge Detection [3][6] and c) Canny Edge Detection [3][4].

1) *Robert's Cross:* Robert's Cross is an algorithm which uses 2x2 convolution kernels (Fig. 4). This choice of kernels makes Robert's Cross a fast algorithm but sensitive to noise. Fig. 5 shows a result of this algorithm to an image, as well as a histogram of edge-value distribution.

$$\underset{Gx}{\begin{array}{|c|c|}\hline +1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}} \qquad \underset{Gy}{\begin{array}{|c|c|}\hline 0 & +1 \\ \hline -1 & 0 \\ \hline \end{array}}$$

Figure 4. Robert's Cross's Convolution Kernels



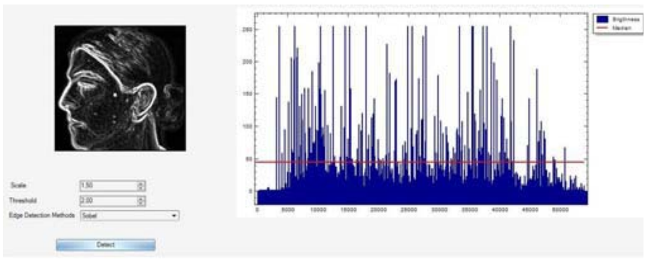
Scale 4.00
Threshold 2.00
Edge Detection Methods Roberts Cross
Detect

Figure 5. Edge Detection by Robert's Cross Algorithm

2) *Sobel:* Sobel is a widely used high quality algorithm. It uses 3x3 convolution kernels (Fig. 6) which increase its execution time but reduce its susceptibility to noise. Fig. 7 shows a result of applying this algorithm to a picture, as well as a histogram of edge-value distribution.

$$\underset{Gx}{\begin{array}{|c|c|c|}\hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array}} \qquad \underset{Gy}{\begin{array}{|c|c|c|}\hline +1 & +2 & +1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}}$$

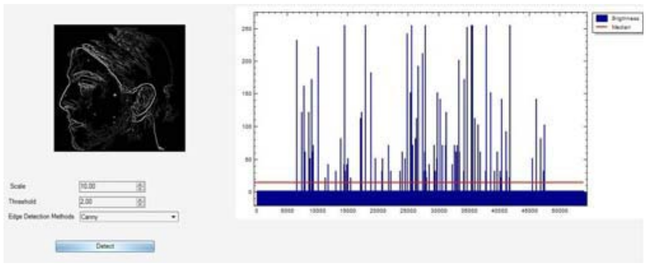
Figure 6. Sobel's Convolution Kernels



Scale 1.50
Threshold 2.00
Edge Detection Methods Sobel
Detect

Figure 7. Edge Detection by Sobel Technique

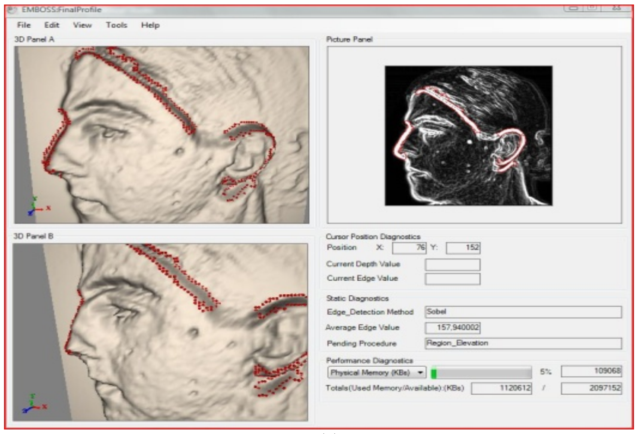
3) *Canny:* Canny's algorithm recognizes true edges but ignores some which might prove useful. Fig. 8 shows a result of applying this algorithm to a picture as well as a histogram of edge-value distribution.



Scale 10.00
Threshold 2.00
Edge Detection Methods Canny
Detect

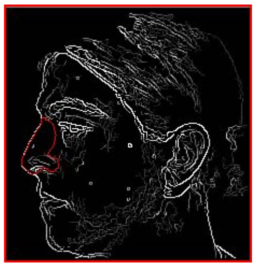
Figure 8. Edge Detection by Canny's Method

Fig. 9 shows two examples of Edge Detection usage.



Cursor Position Diagnostics
Position X: 76 Y: 152
Current Depth Value
Current Edge Value
Static Diagnostics
Edge_Detection Method Sobel
Average Edge Value 157,940002
Pending Procedure Region_Elevation
Performance Diagnostics
Physical Memory (KBs) 5% 109068
Totals(Used Memory/Available):(KBs) 1120612 / 2097152

(a)



(b)

Figure 9. a) Usage of Sobel's method, b) results of Canny's method

B. *Pixel Color*

We use the initial pixel color information to define chromatic areas on the face and perform modifications to the corresponding areas of the 3D representation. Fig. 10 shows an example of such an area definition.

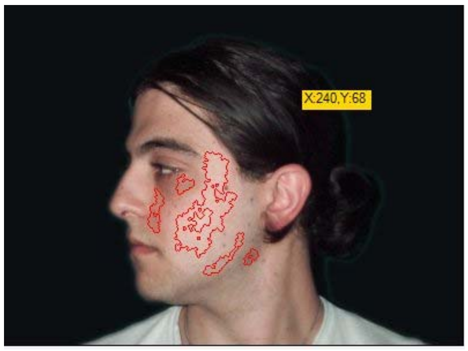


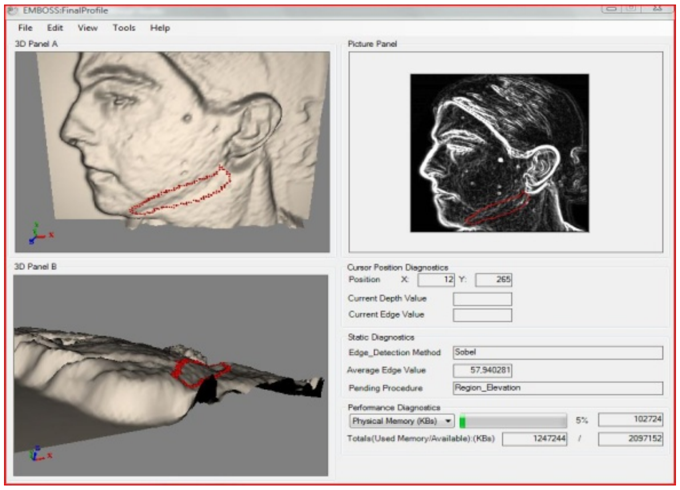
Figure 10. Finding of Chromatic Regions

To detect these areas, we used a variation of a simple region growing algorithm [7]:

For each pixel around a seed pixel, we check it's 8 neighboring (8-connected) ones and we name a pixel as a region limit if either (a) each RGB component of its color deviates from the corresponding component value of the seed by a threshold that is set by the user or (b) the pixel's spatial Euclidean distance from the seed is larger than a given limit set adaptively by the tool.

C. *Free Sketching*

There is often a need for free area definition. Especially when we need to perform smoothing on regions where significant localized depth variations occur which cannot be caught by Edge Detection. Fig. 11 shows such a case. Non-Convex areas are discarded while gaps between the starting and ending point are interpolated linearly by an efficient algorithm (e.g. Bresenham's [5] algorithm).



Cursor Position Diagnostics
Position X: 12 Y: 265
Current Depth Value
Current Edge Value
Static Diagnostics
Edge_Detection Method: Sobel
Average Edge Value: 57.940281
Pending Procedure: Region_Elevation
Performance Diagnostics
Physical Memory (KBs) 5% 102724
Totals(Used Memory/Available):(KBs) 1247244 / 2097152

Figure 11. Free Sketching

IV. *SURFACE ENHANCEMENT TOOLS*

In this section we describe the development of methods to edit the morphology of the selected areas, to enhance the quality of our embossing surface. We use three methods which exploit the algorithms we have developed for area selection: i) Depth modification of a specific point (seed) along with a specified area around it, ii) Depth modification of an area, selected via our area selection algorithms, iii) Smoothing of a selected area using various smoothing filters.

A. *Depth Modification of Specific Point*

This algorithm serves the purpose of helping the user to intervene into the morphology of small areas. Such areas could be spots or moles or defects produced by noise. In a sense this is a region selection algorithm too with the difference that the region cannot exceed certain limits. The algorithm could be useful even for larger areas provided that more than one point will be selected for modification. The basic characteristic of the algorithm is that, while the seed's depth gets modified, so is a specific area around it, using linear interpolation. In the following section we will describe the steps of the algorithm.

1) Finding Seed's Neighboring Points and Seed's Neighborhood Border Points: The borders of the neighborhood around a seed point are either points belonging to edges or points whose Euclidian distance from the seed is greater than a predefined threshold. In order to find both the neighborhood and border points we perform the following variation of a Region Growing algorithm:

- If borders are defined by edges (computed by some edge detection algorithm) then we consider as border those points of the neighborhood that, after edge detection, correspond to pixels that fall within edge areas or their Euclidian distance from the seed exceeds a given threshold.
- If borders are defined by the Euclidian distance from the seed then we consider as border those points of the neighborhood whose Euclidian distance from the seed is greater than a given threshold.

At the same time we define as inner neighborhood points those points contained within the borders. As far as editing is concerned, only the inner neighborhood points are modified along with the seed.

Fig. 12 gives us an example of selecting a region around a specific point with borders selected by Euclidian distance. Border points are the highlighted in red, while inner neighborhood points are highlighted white. Seed is shown in cyan.

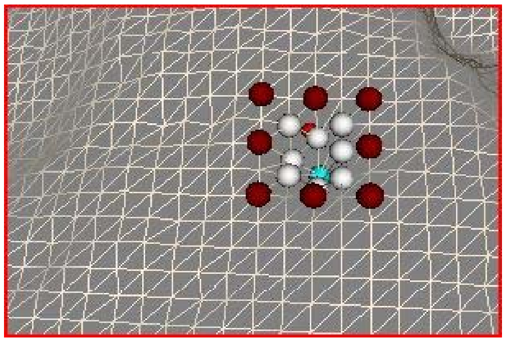


Figure 12. Region Selection around a Point Based on Euclidian Distance

2) Associating Inner Points to Border Points: To maintain the surface morphology during editing we associate each inner point with a border point. As described later, the topology maintained is that of the distance between the seed and the inner point to be modified and the distance between the inner point and its associated border.

The mapping is performed as follows. We project all neighborhood points to the XY plane. We compute the equation of the line that passes through the seed and the inner point to be mapped (denoted by P). We denote this line by SP (seed-P).

We then project the border points on that line and we associate the inner point to the border whose distance from the line is the smallest and its projection point lies on the ray of SP from seed to P.

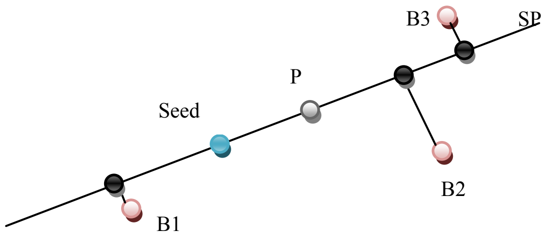


Figure 13: Associating an Inner Point to a Border Point.

In Fig. 13, the winning border should be B3 since its distance from SP, while not the smallest (B1's is), is the smallest of a border whose projection is on the ray from seed towards P (B1's projection on the line is before seed).

Each border point may have more than one inner points associated with it. Borders that at the end of the mapping process have no inner points attached to them will be removed from the editing area.

3) Adjusting Inner Point Modification According to Seed Modification: Seed point is chosen explicitly by the user, who also modifies its depth. As the depth of the seed is modified, the inner points of the neighborhood around the seed should also undergo a depth adjustment in a way that preserves the local topology. To compute the adjustment we apply the following procedure which is also illustrated in Fig. 14.

Given the fact that point modification takes place along the Z-axis, we project the points to the YZ plane. Every inner point P has a fixed Y coordinate and resides on the line y=Py. Note that only inner neighborhood points are modified.

Prior to seed modification, we compute the intersection of the y line with the line defined by the seed point and the associated border point of P. We denote as PA this intersection. We then perform the desired modification of the Seed point and compute the point of the intersection again. We name that point PB. The modification that should be applied to P is equal to the distance *d(PA, PB)* between the two intersections.

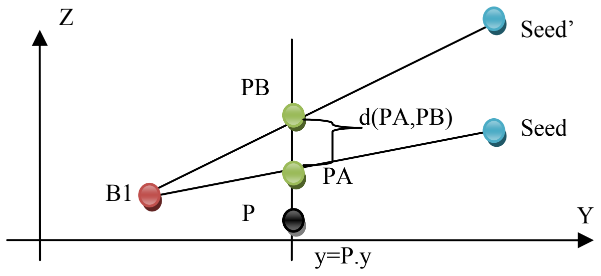


Figure 14. Computing Modification Quantity

The above procedure ensures that all 3D point modifications are combined with neighboring point adjustments that maintain the surface topology. It is now possible to intervene to various points of an area and modify their depth appropriately. Fig. 15 gives an example of such an editing operation.

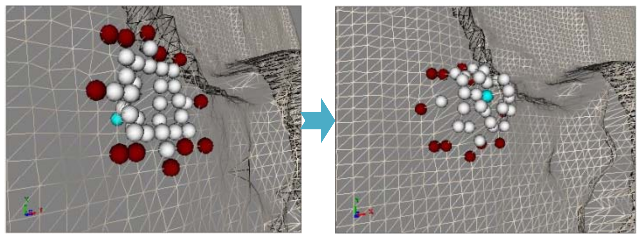


Figure 15: Point Modification Example

It becomes apparent from the top pictures of Fig. 17 that if the seed is positioned lower along the Z-axis than its neighbors, depth increase followed by simultaneous depth increase of the neighbors yields in an undesirable result. To cope with this case, we provide the ability to impose a restriction to the height of the neighboring points that have the right to be modified. More specifically, no neighboring point is gets modified before the seed reaches its height. Fig. 16 depicts an example of applying that restriction.

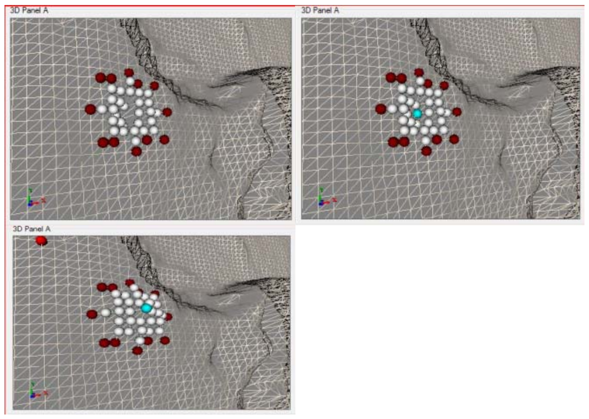


Figure 16. Restricted Modification of Neighboring Points

B. *Selected Area Modification*

This method allows us to adjust the depth of entire areas. It is particularly useful for large areas that share some attribute (e.g. hair). Fig. 17 shows an operation of editing the area that corresponds to the hair of a profile snapshot.

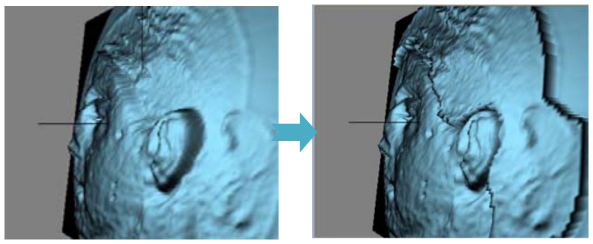


Figure 17. Area Depth Modification

C. *Selected Area Smoothing Using Smoothing Filters*

Every method discussed so far modifies the morphology of our 3D model without guarantying that the result will be uniform enough to be acceptable. Given that our 3D representation is based on 2D it is safe to assume that rapid local depth variations are actually noise. To eliminate that noise we provide the user with smoothing tools:

1) Mean Filter: For each point of an area we compute the average of its neighboring points depth (Lightness), with the seed point included. The size of the neighborhood is defined by the user

2) Median Filter: The difference between the Mean and the Median filter is that the dimensions of the kernel define the area around the seed from which we gather depth values, position them in ascending order and choose as new value of the seed the depth that is positioned in the middle.

3) Gaussian Filter: In this method the convolution kernel is selected such that it represents a Gaussian cavity (bell shaped). The Gaussian distribution is given by the following formula:

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5}$$

which it turns out to be described by the convolution kernel depicted on Fig. 18.

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Figure 18. Gaussian Filter Kernel

4) Conservative Filter: This is a variation o the Median Filter. The seed's value is compared to the maximum and minimum of the ordering. If it falls between it remains unchanged. If it is bigger than the maximum or smaller than the minimum it is replaced by the corresponding one. Fig. 19 illustrates the result of the area smoothing using Mean Filter after applied to the whole surface.

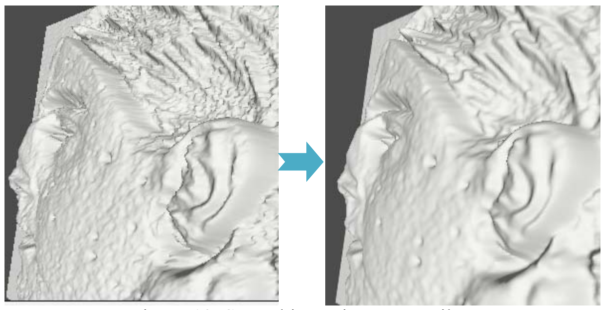


Figure 19. Smoothing using Mean Filter

V. *APPLICATION EVALUATION*

The evaluation of our application was conducted through a training procedure. Several users after having undergone supervised training on how to use the program, have produced an embossing with the guidance of the developer, and then they were left to repeat the procedure by themselves, having only the manual as a guide. In the end they were asked to fill in a questionnaire, so as to evaluate the software by grading several aspects from 0 (weak) to 7 (strong). They were also asked to emphasize on some weak and strong features of the program. The questions about the program revolved around two of the major characteristics of software design, Learnability and Robustness and the three standards of Satisfaction, Efficiency and Effectiveness. The results are shown in Fig. 20. Note that Lernability consists of Predictability, Familiarity, Generalizability and Consistency, while Robustness consists of Observability, Recoverability and Task Conformance.

Effectiveness metrics reveal the ability of our tool suite to perform the embossing task with adequate preciseness. The general notion among the users was that the application provides enough tools to perform the task and that they could revert to various combinations of tools for performing this task. Finally, the majority of the users pointed out that, although they had to put in a certain amount of effort to produce a final outcome, the results were overall satisfactory.

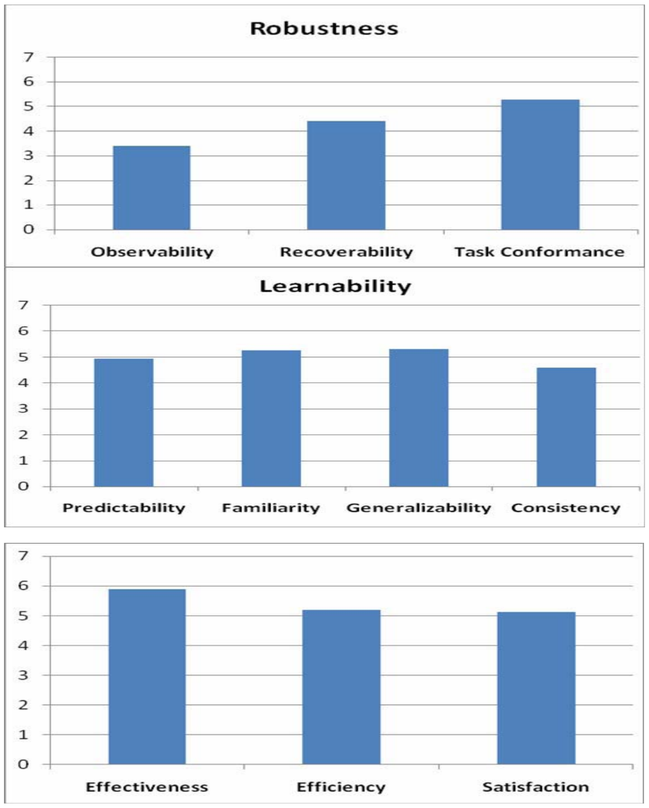


Figure 20. Evaluation Results

VI. RESULTS AND CONCLUSIONS

To obtain better results faster, the user should set certain lighting conditions as well as an appropriate camera angle. The lighting of the picture should allow the face to have shades. An ambient lighting for example would result in weak shading, rendering our method inadequate. A directional light (e.g. sunlight) would also give poor results. The ideal light condition appear to be under a Point Light such as a lamp, in which every light ray has different angle from the others illuminating differently as they hit the face. Additionally the light source should be as close to the camera eye as possible.

We provide results of our method in various face positions. The images used are shown in Fig 21. It is important to place the object to be embossed in a uniform background (ideally dark). This would help our area selection methods to select and modify the depth of the background. Nevertheless it is possible that some preprocessing of an image is required.

Fig. 22 shows the 3D representations (initial approaches) of the pictures after the applying of scaling. As it turns out, working on the model, results in defects around areas of rapid curvature change which can be smoothed via some filter. For this reason it is recommended to choose a small scaling factor. This would result in a model with lots of noise but in the same time should provide means for smoothing without worrying about flattening completely the smoothed area.

Fig. 23 illustrates the final results of our method. As it turned out in practice, the most troublesome areas are those of hair growth, and particularly the eyebrows. Also eyes prove to be an extremely difficult part to represent due to the difference in color between the eye bulb and the pupil. Finally, ears prove are quite troublesome because of their complex morphology.

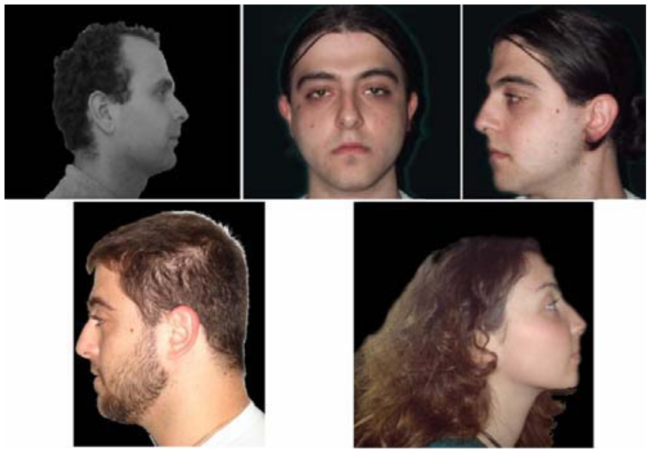


Figure 21. Working Pictures

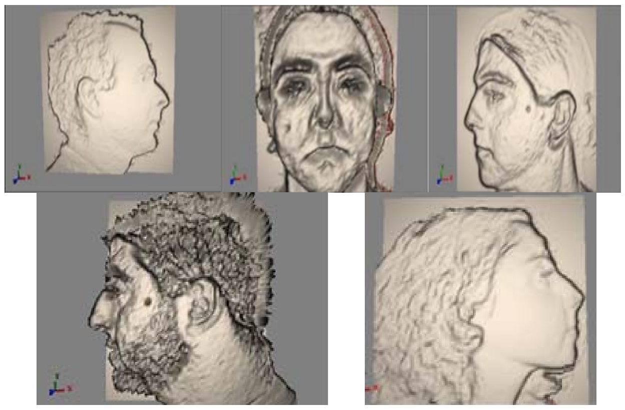


Figure 22. Initial Approaches

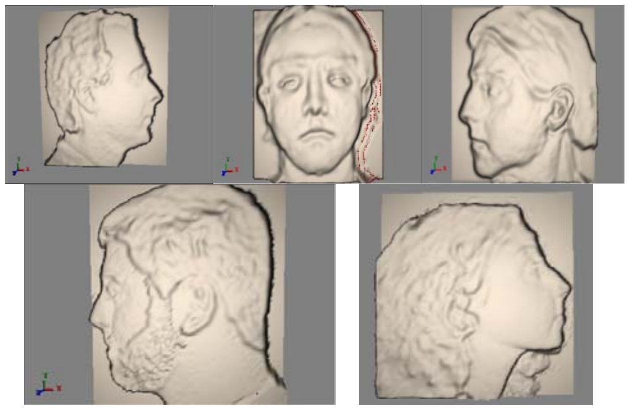


Figure 23. Final Results

REFERENCES

[1] H. Levkowitz and G. T. Herman: “*GLHS: A generalized lightness, hue and saturation color model. CVGIP: Graphical Models and Image Processing”, 55(4):271–285, 1993.*

[2] A. Hanbury: “*The taming of the hue, saturation and brightness colour space”, In Proceedings of the 7th CVWW, Bad Aussee, Austria, 2002.*

[3] HIPR2: “*Image Processing and Learning Resources”,* http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

[4] J. Canny: “*A Computational Approach to Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence”, 8:679-714, 1986.*

[5] J. E. Bresenham, *"Algorithm for computer control of a digital plotter"*, *IBM Systems Journal*, Vol. 4, No.1, January 1965, pp. 25–30

[6] I. Sobel, G. Feldman: *"A 3x3 Isotropic Gradient Operator for Image Processing", presented at a talk at the Stanford Artificial Project in 1968, unpublished but often cited, orig. in Pattern Classification and Scene Analysis, R. Duda. and P. Hart, John Wiley and Sons,'73, pp271-2.*

[7] J. D. Foley and A. van Dam (1982)*: “Fundamentals of Interactive Computer Graphics”, Boston, MA, USA: Addison-Wesley.*

[8] CIE Standard S 014-4/E:2006: *Colorimetry - Part 4: “CIE 1976 L*a*b* Colour Spaces”*

[9] C. Poynton: ”*Frequently Asked Questions about Color*”, http://www.poynton.com

[10] MeshCAM: *Powerful CAM Software for non-machinists.* http://www.meshcam.com

[11] ArtCAM: *Artistic CAD/CAM Software. http://www.artcam.com*

[12] 3DMaker: *Software for Making Anaglyphs and Stereograms, http://www.tabberer.com/sandyknoll/more/3dmaker/3dmaker.html*

[13] VS3D:*Virtual Sculpture CAD/CAM Software.* http://www.designscomputed.com/vs3d/

[14] Geomagic Studio: *3D Software for Creating 3D Models from 3D Scanner Data.* http://www.geomagic.com/en/products/studio/index.shtml

[15] X. Gu, S. Zhang, L. Zhang, R. Martin, P. Huang and S. Yau: *“Holoimages”*, Proceedings of the 2006 ACM symposium on Solid and physical modeling, 129-138.