Role of Threshold Value and CSF to Simplify and Render an Image

Prof. Vishal Dahiya
Indus Institute of Technology & Engineering,
Ahmedabad.

Dr. Priti Srinivas Sajja
Sardar Patel University,
Vallabh Vidya Nagar.

ABSTRACT
In this paper, we present new efficient algorithms that simplify and render an image effectively on the screen. Simplification is required to reduce the complexity of an image and facilitate efficient rendering. First algorithm is based upon the threshold value simplification that is if there will be minor changes in the threshold value produces different percentage of simplification in the same image. The threshold value used here is based on the pixel values of an image. Second algorithm is based on Contrast Sensitivity Function (CSF), where the CSF is determined using luminance value of the image. Both these algorithms produce a simplified image which can be analyzed, processed and communicated efficiently and result in reduced cost of operation based on the images. The article concludes with the comparative results of both the algorithms.

Keywords:
CSF, simplification, Rendering, Threshold value.

1. INTRODUCTION
Object simplification algorithms are used in graphics to create new objects that are visually similar to the original object but that are less complex and thus more economical to render. In other words, the goal is to minimize the perceptual deviation between the renderings of the two models. However, a very common method for measuring the similarity of simplified models to the original has been to use geometry-based metrics. These approaches minimize geometric deviation between models.
Most of the previous simplification methods are based on number of assumptions such as pre-lit color assign to vertex, good contrast value assigned to boundary, contrast of front facing region is assumed or given etc. No algorithm today excels at simplifying all models. Some approaches best suit curved, organic forms, while others work best at preserving mechanical objects with sharp corners, flat faces, and regular curves. Many models, such as radiositized scenes or scientific visualization data sets, have pre computed colors or lighting that must be considered. Some scenes, such as terrain data sets and volumetric iso-surfaces from medical or scientific visualization, comprise a few large, high-complexity, individual objects.
Simplification is required to eliminate redundant geometry, reduce model size, and improve runtime performance by managing level of details. In the proposed algorithms, we have considered most important factor such as geometric accuracy, visual fidelity, reduces preprocess time and radical simplified image.

This work is very useful in rendering a given image very fast by reducing the size and then that reduced image can be stored in database so that further all the computation can be done on that reduced image.
This work offers a new approach to the problem of object simplification. The method simplifies the underlying object form before addressing an object's representation. A matrix is a collection of values arranged into a fixed number of rows and columns. Every image is a collection of pixels. So the image matrix consists of values of all the pixels. Pixel is having [x, y] coordinates and corresponding to that coordinate [Rx,y, Gx,y, Bx,y] values will be there. Number of rows and column of the image is dependent on the image. The matrix which represents the perceptual importance of every point on a model, for all values in the matrix it implements simplification iteration and it identifies the point that is least important visually.
The simplification transforms itself also uses perceptual criteria to dictate how to simplify the object in a way that is least damaging perceptually. In this work, we also demonstrate that there are qualitatively different results produced by applying CSF (Critical Sensitivity Function) and threshold values and comparison between these two when applying to an object. The rendering of the object is fast in both the methods. The simplification in both the cases will be different. Every method includes matrix for ensuring that the simplifying operations maintain as much similarity as possible to the original object and matrix for measuring the reduction in object complexity.
The goal of this research article is to explore the simplification methods to reduce the size of the object without destroying any useful information from it. So the object simplification is the driving problem behind this work. For determining that simplified objects are less complex and thus more rapidly rendered, the hypothesis is that reducing the visual complexity of an object can lead to a lower representation cost and therefore more efficient rendering.

2. PREVIOUS WORK
A common theme in simplification is the use of either an explicit or an implied geometric distance measure between the original and the simplified model to guide the simplification process. Several simplification approaches rely on successive vertex removal and local re-triangulation to create models with fewer geometric primitives. [1] It use a vertex removal method that is guided by how far a given vertex is to a plane that is fit to neighboring points of the candidate vertex. This approach has more recently been augmented to allow changes to the topology of the surface [2]. They further improved the method by allowing edge flips during simplification to better match the original surface. In [3, 4] they build an internal and an external envelope around the given model that avoids self-intersection and that is no further than from the original surface. Each vertex is tested by

attempting to remove it and re-triangulate the hole in such a manner that the resulting surface does not intersect either envelope, rather than removing one vertex at a time, [5] cluster all the vertices of a model into the cells of a grid of cubes and then replace all of the vertices that fall into a single cell by a single representative vertex. In this manner, they too are able to maintain an upper bound on the Hausdorff distance between the two models, where it is the length of the diagonal of a cell. Another local operation used for simplification is edge collapse: two vertices that share an edge are removed and replaced by a single vertex. [6] Hoppe and co-workers use edge collapse, edge swap, and edge split operations to simplify models. These operations are guided by an energy function that measures the distance from the current simplified model to a set of points in 3D that sample the original model.

Some researchers use successive edge collapse operations that are guided by a distance measure. Each vertex of the original model has associated with it a list of the planes of the adjacent faces to the vertex [8]. The position of the new vertex from a potential edge collapse is chosen to be one of the two original vertices of that edge. The cost of performing the edge collapse is determined by the maximum distance to the set of planes associated with the edge's vertices, and a priority queue is used to order the edge collapse operations. The vertex associated with an edge collapse acquires the lists of planes from both vertices of the edge. In article [9] they convert a set of implicit plane equations into a symmetric 4 by 4 matrix that allows the rapid computation of the sum of the square of the distances between a given point and the planes. They use this new distance measure to guide a series of vertex pair contractions, a generalization of edge collapse that allows topological changes to a model.

Most of the simplification methods that have been published produce static models from an off-line simplification algorithm. A few researchers have built a sequence of progressively simplified meshes and have then used a run-time algorithm to selectively refine the model based on the observer's viewpoint. Hoppe performed a careful analysis of edge-collapse dependencies to create a view-dependent algorithm for displaying surfaces [7] his approach exploits frame coherence and moves vertex positions over several frames to eliminate popping artifacts. Luebke and Erikson perform view-dependent simplification based on a hierarchical data structure that can be built off-line using vertex clustering, envelopes or progressive meshes [10]. All of the above view-dependent methods use geometric distance measures to perform their off-line simplification preprocessing. In particular, the preprocessing is typically done in a *view-independent* manner, and it is the responsibility of the *run-time* system to construct a view-dependent model by choosing from a set of predefined simplification moves (e.g. edge collapse, vertex removal, etc.) which, if not constructed carefully, are not likely to optimally preserve model appearance[11]. While motivated by the same goal, their approach is quite different from ours in that they rely on converting the model to an alternate representation. In order to display such an augmented model, specialized hardware or software algorithms are needed.

All available simplification algorithms, however, are to speed up rendering for visualization of complex databases. For this purpose, the most important measure of fidelity isn't geometric but perceptual: Does the simplification look like the original? To date, only Cohen's appearance-preserving simplification [14] and Lindstrom's image-driven simplification [15] attempt to address this question. Perceptual metrics and perceptually driven simplification seem like crucial topics for research.

Luebke-Hallen [13] approach is to evaluate local simplification operations according to the *worst-case contrast* and *worst-case spatial frequency* of features they could induce in the image. This provides a principled way to reason about the perceptibility of the resulting simplification. We extend these concepts to a more general and practical framework for simplification.

3. IMAGE BASED SIMPLIFICATION

3.1 Simplification Plan

Preliminary requirement for designing simplification algorithms is to make a decision that how to select a location for all iteration on which the simplification operations have to perform. This decision making process defines the schedule of operations. Taking a perceptual approach to the problem suggests that we want to simplify at the location on the object that will least affect the object's perception. Which points on the object will allow modification to itself and its surrounding region with the minimal change in our perception of the object? So we have to find out the points which are having very low intensity value. First Step is to calculate the intensity values of all the pixel of an image and then create the object matrix for that image. A threshold value will be selected manually and based on that threshold value the pixel having less intensity value as compared to the threshold value can be removed and create a new object matrix for the same image. Render the new object matrix to display the image.

3.2 Visual Complexity and Similarity

Another important choice in designing a simplification method is to find ways to measure the degree of simplification and for comparing the visual similarity with the original. The number of primitives in an object's representation is used almost universally to measure the degree of simplification. In this case, the simplification process directly simplifies an object's representation. The hypothesis is that reducing an object's perceptual complexity will lead to representations that can be rendered more quickly. Perceptually, the size and representation of an object is a measure of its complexity. Objects with many tentacles of projection and indentation have more visual complexity than more circular, blobby objects.

3.3 Simplification Operation

First operation is to find the intensity value of each and every pixel of an image and then creating an object matrix that is collection of rows and column of the pixel intensity values. Each pixel is having three values that is [Rx,y, Gx,y, Bx,y] where RGB stands for Red, Green, Blue value and [x,y] is the position of the pixel. Number of operations has to be performed on the same matrix and the rendering technique is used to render that matrix on the screen. Comparison between the actual and simplified is to be done to get the reduction in the size of the simplified image. So finding out the pixels which are having least visual significance values and then applying the operations on these pixels to simplify the object is the main concept of our algorithm.

4. PROPOSED SIMPLIFICATION ALGORITHMS

4.1. Algorithm 1

This is a novel approach of object simplification which is implemented in this research work. The key parameter in the thresholding process is the choice of the threshold value. Several different methods for choosing a threshold exist; users can manually choose a threshold value, or a thresholding algorithm can compute a value automatically, which is known as automatic thresholding. A simple method would be to choose the mean or median value, the rationale being that if the object pixels are brighter than the background, they should also be brighter than the average. In this algorithm, we are assuming the Threshold Value (TV). The simplification of the object is directly proportional to the selected threshold value. In RGB color scheme the comparison will be three times that is with R coordinate, G coordinates and B coordinates. Figure 1 is describing the flow diagram of algorithm 1.

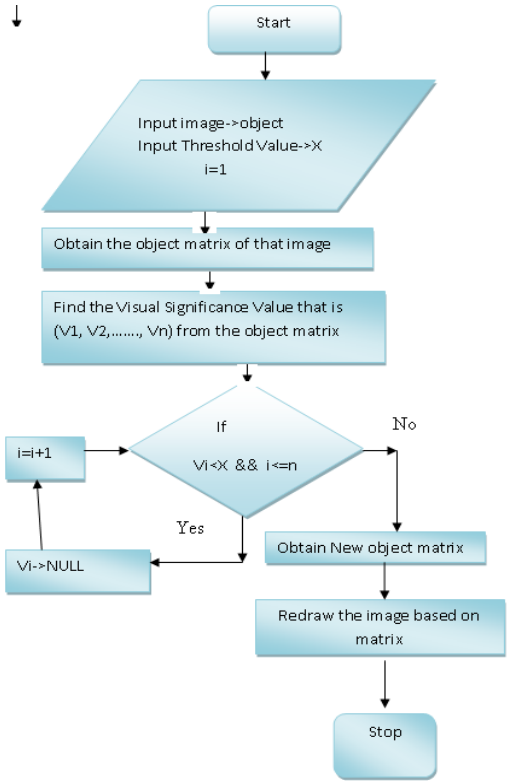

Figure 1: Flow Diagram of Algorithm1

4.2. Algorithm 2:

The spatial performance of the visual system is closely linked with the temporal performance. The contrast value of a person eye is directly proportional to the threshold level and the perception criteria. The criterion is basically the function of the geometry of the object stimulus. This algorithm uses Contrast Sensitivity Function (CSF) in which we calculate the value of contrast based on equation (1) to simplify an object. Figure 2 is showing the flow diagram of algorithm 2.

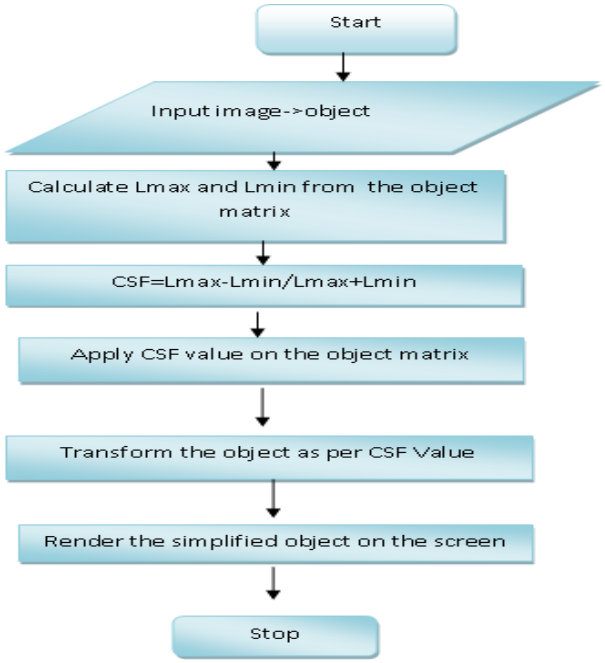

Figure 2: Flow Diagram of Algorithm 2

This algorithm has been showing a large change in the object. It is simplified to the large extent. So the simplification of the image is directly proportional to the eye of the observer.

So both the algorithms are giving very tempting results when applied on the object and the object is simplified to large extent. In the next Para, implementation of these algorithm and its results has been discussed.

5. SAMPLE CODES OF THE PROPOSED ALGORITHMS

These algorithms, we have implemented with the help of MATLAB 7.6.0 (R2008a) [12].The image dimension taken for the result analysis is 320*240.

5.1. Program Code for Algorithm 1

```
Pinfo = imfinfo(path);
 W = Pinfo.Width;
 H = Pinfo.Height;
 ori_size = Pinfo.FileSize;
 Type = Pinfo.ColorType;
if(strcmpi(Type,'grayscale'))
   'grayscale'
   i=1;
   while(i <= H)
            j=1;
           while( j <= W)
                        intensity = I2(i,j);
                         if intensity < 0.0099
                                    I2(i,j) = 0.0000;
                         end
            j=j+1;
      end
      i = i+1;
```

```
    end

elseif(strcmpi(Type,'truecolor'))
    'truecolor'
    i=1;
    while(i <= H)
        j=1;
        while( j <= W)
            R = I2(i,j,1);
            G = I2(i,j,2);
            B = I2(i,j,3);
            if R < 0.0099
                I2(i,j,1) = 0;
            end
            if G < 0.0099
                I2(i,j,2) = 0;
            end
            if B < 0.0099
                I2(i,j,3) = 0;
            end
            j=j+1;
        end
        i = i+1;
    end
end
```

5.2. Program Code for Algorithm 2

```
Pinfo = imfinfo(P);
W = Pinfo.Width;
H = Pinfo.Height;
iSize = Pinfo.FileSize;
F = Pinfo.Format;
I2 = I;
imtool(I)
lmax = max(I2(:));
lmin = min(I2(:));
  contrast = (lmax - lmin)/(lmax+lmin);
newmat = contrast .* I2;
  imtool(newmat)
imwrite(newmat,P1);
 Pinfo1 = imfinfo(P1);
 W1 = Pinfo1.Width;
 H1 = Pinfo1.Height;
 iSize1 = Pinfo1.FileSize;
 sprintf('Simplified image size is %d Bytes',iSize1)
```

The above code is just a snap shot of the algorithm 2 which is implemented successfully. Results have been discussed in next section.

6. RESULTS OF THE ALGORITHMS

Both the algorithms are tested on similar images and different results have been given by them. The dimension of the image has been taken 320*240. First the original image has been displayed, after that simplified image with the help of Algorithm-1 and then with Algorithm-2 has been displayed.

The Figure 3 is the original image which is having file format as Joint Photographic expert Group.

Image 1: (Image.jpg)

Figure 3- Original Image
Original image size is 17733 bytes.

Figure 4: Simplified image size is 17417 bytes using algorithm1.

If a person looks at both the figures, Figure-3 and Figure-4 hardly find out any difference but if we see the file size then there is difference. So the simplified image can be stored in database for further processing. So it will consume less time to retrieve and also consume less space.

Figure 5: Simplified image size is 17401 bytes using algorithm 2.

So using the algorithm 2, we will get more simplified image.

Image 2: Lilly.png (Portable Network Graphics)

Figure 6: Original size is 188078 byte

Figure 7: Simplified image size is 113600 bytes using algorithm 1.

Figure 8: Simplified image size is 113600 bytes using algorithm 2.

So for png file, both algorithms show same results. The resultant file will be <filename>.jpeg.

Image 3: sunset.gif (Graphic Interchange Format)

Figure 9: Original image size is 26786 bytes

Figure 10: Simplified image size is 1265 bytes using algorithm 1.

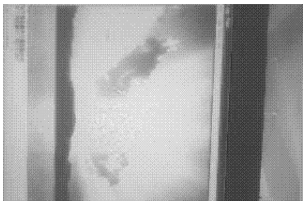

Figure 11: Simplified image size is 26786 bytes using algorithm 2.

Both the algorithms are not giving good results on GIF files. Algorithm1 is not applicable on GIF files.

Image 4: Light.tif (Tagged Image File Format)

Figure 12: Original image size is 5760054 bytes.

Figure 13: Simplified image size is 10846 bytes using algorithm 1.

Figure 14: Simplified image size is 10846 bytes using algorithm 2.

So algorithm 1 and algorithm 2 gives different results for the different type of images, the level of simplification in both is also different with respect to the same image. Table 1 is showing the results of both the algorithm after implementation.

TABLE 1

Results of Algorithm 1 and Algorithm 2 for different type of images

Sr. No	Image Type	Original Image Size (in Bytes)	Algorithm-1 Result (in Bytes)	Algorithm-2 Result (in Bytes)
1	JPG	17733	17417	17401
2	PNG	188078	113600	113600
3	GIF	26786	1265	26786
4	TIFF	5760054	10846	10846

Table 2 is showing the results of algorithm 1 when implemented with different threshold values, same image can be reduced to large extent by changing the threshold value. So algorithm 1 will be very effective if we see its results for image simplification.

TABLE 2

Image Size vs. Threshold Value (TV)

TV	Original Image Size (in bytes)	Rendered image Size (in bytes)	Difference in size (in bytes)
0.0999	17733	17304	429
0.0899	17733	17315	418
0.0799	17733	17283	450
0.0699	17733	17305	428
0.0599	17733	17303	427
0.0499	17733	17330	403
0.0399	17733	17332	401

As we change the value of TV rendered image size is changing and reducing to large extent. The reduction in file size without any distortion in the original image will accelerate the rendering of that image whenever it is used by any process. The simplified image can be stored in the database for further processing so automatically the retrieval time will be very less. So we can say the Algorithm 1 is very effective though the Algorithm 2 is also effective if we see the results in Table 1.

Figure 15 is showing the relationship between the size of the images and threshold values.

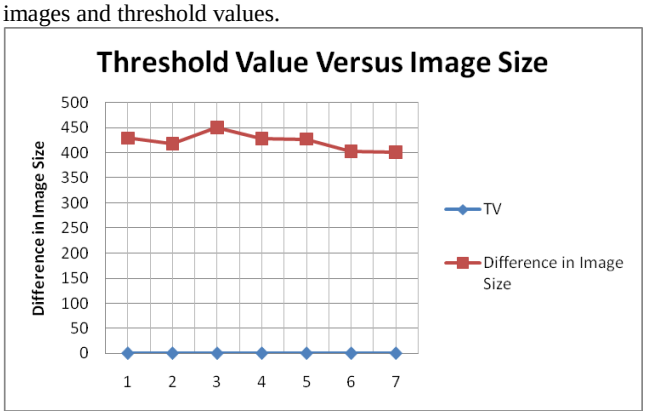

Figure 15: Effect of TV on the Image Size

7. CONCLUSIONS

Simplification and rendering are the two basic operation of image processing which are efficiently achieved with the help of the algorithms which are discussed in this research paper. After implementation algorithms are giving good results but they are working efficiently on some types of image file. Based on the type of image file the simplification level and time taken to render that image has been changing. Rendering of the image has been accelerated to a large extent. This is a novel concept in the field of image processing. The proposed algorithms provide simplification which is not present in existing algorithms in all regions. The main drawback of the algorithms that the pixel value which is lost is lost forever, but that is not affecting the image visibility.

We would like to extend our work to include these two algorithms in the area of finding how an image is perceived by human visual system and then simplify the perceived image to save data storage and also render it fast on the display screen.

8. REFERNCES

[1] Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. "*Decimation of Triangle Meshes"*. Proceedings of SIGGRAPH 92. In Computer Graphics 26(2) (July 1992), pp. 65–70.

[2] Moenning, C., and Dodgson, N. A. 2003. *"A new point cloud simplification algorithm"*. Proc. 3rd Int. Conf. on Visualization, Imaging and Image Processing, 1027–1033.

[3] Ciampalini, A., Cignoni, P., Montani, C., and Scopigno, R. "*Multiresolution Decimation Based on Global Error*". The Visual Computer, Springer International, 13(5), 1997, pp. 228–246.

[4] Cohen, J., Varshney, A.,Manocha, D., Turk, G.,Weber, H., Agarwal, P., Brooks, F., and WRIGHT, W. "*Simplification Envelopes*". Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 119–128.

[5] Peyr´e, G., and Cohen, L. 2003. "*Geodesic re-meshing and parameterization using front propagation*". In Proc. 2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision.

[6] Hoppe, H., Derose, T., Duchamp, T., Mcdonald, J., and Stuetzle, W. "*Mesh Optimization*". Proceedings of SIGGRAPH 93. In Computer Graphics Proceedings, Annual Conference Series,1993, ACM SIGGRAPH, pp. 19–26.

[7] Hoppe, H. *"View-Dependent Refinement of Progressive Meshes"*. Proceedings of SIGGRAPH 97. In Computer Graphics Proceedings, Annual Conference Series, 1997, ACMSIGGRAPH, pp. 189–198.

[8] Ronfard, R. and Rossignac, J. "*Full-Range Approximation of Triangulated Polyhedra"*. Proceedings of Eurographics 96. In Computer Graphics Forum, 15(3), August 1996, pp. 67–76.

[9] Garland, M. and Heckbert, P. S. "*Surface Simplification using Quadric Error Metrics"*. Proceedings of SIGGRAPH 97. In Computer Graphics Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 209–216.

[10] LUEBKE, D. and ERIKSON, C. *"View-Dependent Simplification of Arbitrary Polygonal Environments"*. Proceedings of SIGGRAPH 97. In Computer Graphics Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 199–208.

[11] Nebras N. Hasson, S. A. Aljunid, R. Badishah Ahmad, "*Simplification of Raster Images to Extract Visual Information"* IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.11, November 2006, pp. 49-52.

[12] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, *"Digital Image Processing Using MATLAB"*. Pearson Education, Inc., 2004.pp.511-529.

[13] Luebke, D., and Hallen, B. 2001. *"Perceptually Driven Simplification for Interactive Rendering"*. Proceedings of 2001 Eurographics Rendering Workshop, 223-234.

[14] J. Cohen, M. Olano, and D. Manocha, *"Appearance Preserving Simplification,"* Computer Graphics (Proc. Siggraph 98), vol. 32, ACM Press, New York, 1998, pp. 115-122.

[15] P. Lindstrom and G. Turk, *"Image-Driven Simplification,"* ACM Trans. Graphics. vol. 19, no. 3, July 2000, pp. 204-241.