May 2017

Analysis of Bas-Relief Generation Techniques

Zachary Salim Benzaid
University of Wisconsin-Milwaukee

ANALYSIS OF BAS-RELIEF GENERATION TECHNIQUES

by

Zachary Benzaid

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

May 2017

ABSTRACT

ANALYSIS OF BAS-RELIEF GENERATION TECHNIQUES

by

Zachary Benzaid

The University of Wisconsin-Milwaukee, 2017
Under the Supervision of Professor Zeyun Yu

Simplifying the process of generating relief sculptures has been an interesting topic of research in the past decade. A relief is a type of sculpture that does not entirely extend into three-dimensional space. Instead, it has details that are carved into a flat surface, like wood or stone, such that there are slight elevations from the flat plane that define the subject of the sculpture. When viewed orthogonally straight on, a relief can look like a full sculpture or statue in the respect that a full sense of depth from the subject can be perceived. Creating such a model manually is a tedious and difficult process, akin to the challenges a painter may face when designing a convincing painting.

Like with painting, certain digital tools (3D modeling programs most commonly) can make the process a little easier, but can still take a lot of time to obtain sufficient details. To further simplify the process of relief generation, a sizable amount of research has gone into developing semi-automated processes of creating reliefs based on different types of models. These methods can vary in many ways, including the type of input used, the computational time required, and the quality of the resulting model. The performance typically depends on the type of operations applied to the input model, and usually user-specified parameters to modify

its appearance.

In this thesis, we try to accomplish a few related topics. First, we analyze previous work in the field and briefly summarize the procedures to emphasize a variety of ways to solve the problem. We then look at specific algorithms for generating reliefs from 2D and 3D models. After explaining two of each type, a “basic” approach, and a more sophisticated one, we compare the algorithms based on their difficulty to implement, the quality of the results, and the time to process. The final section will include some more sample results of the previous algorithms, and will suggest possible ideas to enhance their results, which could be applied in continuing research on the topic.

To

My family (Mom, Dad, Sami, Kayla, and Blackjack),

Kristy,

And to all the friends who have helped me along the way.

Hopefully someday I can return the favor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF BASIC TERMINOLOGY

There are other terms to consider, but many of the important ones are explained in the thesis where relevant. These are terms not explicitly explained elsewhere.

Height/Depth field: Terms used when referring to a 3D model with respect to a 2D plane. The plane is measured in x and y coordinates, and heights refer to the z-coordinate, the distance between the given point from the plane.

Intensity: The measure of brightness of an image. In most applications, measured between 0 (dark) and 255 (bright). Often manipulated to influence values of the height field.

Gradient Domain: Gradient is the generalized term for the derivative in multiple dimensions. In Image Processing, the gradient domain is represented as an image of gradients, changes in intensity values between pixels.

Attenuation/Compression: A process to decrease the total range between a data set. Linear compression will reduce all nonzero values equally, while a non-linear compression will weigh the amount of compression applied based on the current magnitudes of the points.

Filter: In Image Processing, a filter is a constant set of data used to modify and enhance an image. In the spatial domain, they are typically represented as a kernel, which is applied in convolution with the data of the image. Common operations for filters are smoothing and sharpening images.

Normalization: An adjustment of the range of a data set to a new range. Commonly, it will be used to constrain data values between 0 and 1.

Mesh: A digital representation of a three-dimensional model. Most of these algorithms aim to generate a mesh as their result. Data representation can vary, but the easiest interpretation is a set of points with a position (x,y) and a height (z).

ACKNOWLEDGEMENTS

I would like to extend my gratitude to many of the UWM Faculty. First I would like to thank professor Zeyun Yu for acting as my advisor for this thesis. He was also responsible for helping me develop much of my graphics and programming skills required for this and many more projects.

I would also like to thank professor John Boyland for both his patience and offering me a chance to work as a Teaching Assistant during my time here. This was of great assistance in terms of my financial capability to continue my studies.

I am also grateful to professor Seyed H. Hosseini and professor Ichiro Suzuki, who agreed to act on my evaluation committee with a rather short amount of notice. I apologize.

I also extend gratitude to Meili Wang, whose previous research in the field was of great help to me during my survey, and serves as the primary motivation for the gradient domain 2D image algorithm implemented for this thesis.

Also thanks to Kevin Davis, whose concurrent capstone project provided useful information on the feature line algorithm, as well as someone to have discussions with on the topic.

Finally, I will also offer my thanks to Priya Vashistha, who was a close friend during my time at UW-Milwaukee and spurred me on to try my best within the past year.

CHAPTER 1

INTRODUCTION

1.1 Overview of Reliefs:

The practice of three-dimensional modeling of objects has been a significant topic of research in a vast range of disciplines. In medicine for instance, a reconstruction of a skeletal structure or of various organs has vastly improved the understanding of the human body and in turn advanced the success and usefulness of the entire field. In engineering and construction, three dimensional models can be useful as a more detailed blueprint for design and development. Even in artistic endeavors, models tend to stand out more than paintings when viewed at a library, a museum, or virtually any other public place.

Figure 1.1: An example of a full 3D sculpture. By McKay Savage from London, UK - China - Beijing 12 - lion outside the Tibetan Monastery, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=23469380

Since modeling can have so many purposes and be approached in many ways, it is difficult to derive a universal procedure with which to create all types of models. Terms of depth, accuracy of details, and the effort to create them can all variate with different types. This paper will focus on a specific type of model, known commonly as a relief.

A relief is a peculiar type of model. The initial feature that is noticed when considering them is that, for being three dimensional models, they are relatively flat in appearance. This observation is obvious when one is viewed from the side, but what about when looking directly at it?

Sometimes, a relief will be described as a hybrid between a painting and a sculpture. Although it is technically a three-dimensional model in terms of physical form, to create a proper one requires a fair degree of understanding of how flat paintings generally work.

Figure 1.2: An example of a painting. Landscape paintings especially utilize visual techniques to suggest depth within the image. {{PD 1923}} from Wikimedia Commons.

The most significant part is human depth perception. When a painting is viewed, despite trivially knowing that the contents of it are not 3D, it is possible to give a convincing impression

that they are a view of a 3D object, with cues like shading and utilizing properties of focal perspective. A relief can take this further, with using minute differences in the physical depth of the sculpture to convey the perception of a three-dimensional object when viewed face on.

When explaining this phenomenon to other people, I generally attempt to pull a coin out from my pocket as an example. The image on a coin is small, and obviously not completely three- dimensional, but the image of the face or architecture viewed upon it still gives a reasonable amount of depth information for the user to perceive that object as if it were a full sculpture.

Figure 1.3: A common example of a bas-relief: Art on the backside of a Sacagawea dollar. {{PD}} Wikimedia commons.

The challenges of a relief:

In terms of sculpture types, a relief and its properties aren't anything new. Such structures have been created by humans since ancient times, the example often provided is early carvings on cave walls. At that time, they were primarily used as elements for storytelling or illustration. Now reliefs can have a wide variety of purposes, including art, packaging, manufacturing, and

minting. As such it is useful to find relief generating strategies that can account for the wide varieties of medium they can be made from.

In most cases, reliefs are divided into two smaller categories: the high-relief and the bas-relief. The primary difference between them is the constraint of physical depth. High-reliefs may have a much wider depth range, making them more like full sculptures. Bas reliefs are much more compressed in terms of their dimensions, which makes them more akin to the paintings that were described earlier. In this paper, we will focus on bas-relief generation.

In either case, creating a relief from scratch is difficult, and making a high quality one makes that an understatement. By hand, modeling a relief that gives a true illusion of 3D depth requires significant planning, a lot of effort in understanding human visual perception, and an exorbitant amount of time.

Problems that require a lot of time to complete are also the type where it is natural to attempt them with the use of computing. While there are nice modeling programs available that allow custom construction, even doing it by hand with those tools can be challenging. To this end, there is strong interest in the ability to at least semi-automate the process of relief generation, particularly when provided some model as its basis.

Since the problem of generating convincing relief requires knowledge of visualization techniques, the concepts of Image Processing lend themselves well to the process. To allow for the process to be at least semi-automated, it is important to have a model as an input. Research so far has been primarily done on using 3D model data as input for generation, although some other methods may only require a single 2D image to work. The output of most

of these methods generates a 3D mesh file which can later be post-processed or used directly to model the physical equivalent in the medium of interest.

1.2 The Focus of This Thesis

Relief generation has proven to be an interesting problem because of the many possible optimizations that can be made to enhance the result. The immediate problem posed with relief generation is the depth constraint. Ideally, a relief should be nearly flat, yet retain some sense of the depth information from the original subject. A basic compression operation may come to mind as a solution, but there can be many caveats.

The most significant one would be a loss of details in the result. By nature of the problem, it is extremely difficult to not lose any of the detail in the relief. If there is too much compression, we risk losing visual cues in the change of depth within the object the relief is based on. In addition, extremely small details within the subject can be flattened out entirely.

In that sense, studying relief generation comes down to determining intermediate procedures that can robustly handle solving those two overarching problems with the direct approach. Many of the operations studied have other utility in areas of Computer Graphics and Image Processing.

The primary purpose of this thesis will be to illustrate the significant overlap between this problem and the related fields. This will be accomplished by analyzing many of the previously proposed algorithms of relief generation. Each algorithm in turn will find itself more

suited to certain types of model input. Specifically, in this paper we focus on proposed methods that either utilize 3D or 2D digital inputs.

1.3 Summary of the Following Sections

The remainder of the thesis is divided into four chapters:

Chapter 2 summarizes a sizable amount of the research done in the area up to this point. We will focus on the algorithms and results of the papers each summary is focused on. The content is divided between algorithms that work with 3D input and those that use 2D input. Some detail is provided about the ideas behind each of the steps, and the strengths and weaknesses of each algorithm in terms of quality of results, computing efficiency, and difficulty of implementation are discussed.

The next two chapters serve as in-depth comparisons between a pair of relief generation algorithms. One algorithm is considered as “basic”, indicating an approach that is quickly and easily implemented for effective use. The second algorithm is deemed “advanced”, wherein the computations and concepts to implement the program can be tougher to implement, but notable improvements in results may be achieved.

Chapter 3 compares two algorithms that work based on a single 2D image as input. In addition to more details regarding these types of approach, the procedures and implementation are thoroughly explained for a gradient domain based algorithm based on Meili Wang’s thesis. This

algorithm is in turn compared with one based on “Region-based bas-relief from a single image” in the same criteria as those in Chapter 2.

Chapter 4 focuses on algorithms that work based on a 3D model as input. The simple approach here is a novel one based on taking a depth map of a 3D scene. In contrast, this is compared to one of the standard algorithms proposed by Weyrich in the paper “Digital Bas-Relief from 3D Scenes”. Discussion about using 3D models as a base is also provided here.

Chapter 5 is the conclusion to the thesis. It acts as a summary of the previous discussions, in terms of considering the resulting reliefs. Some additional sample outputs from the algorithms implemented in the earlier chapters can be found here so the original chapter doesn’t get too cluttered. Finally, I describe some of my experience in working with these algorithms, including limitations, and suggest alternative ideas for how they could be implemented.

(a)   (b)

Figure 1.4: (a) An example of a high-relief (image taken by I. Sailko) and (b) An example of a bas-relief (image taken by Jastrow) Note the difference in the constraint of depth between the two examples.

CHAPTER 2

BACKGROUND SURVEY

2.1 Overview of Previous Research

Research in bas-relief generation has taken many different directions within the past decade. In most contexts, the goal has been to generate a reproducible mesh, which can later be used and enhanced in more advanced graphical manipulation programs. As a result, specific colors and textures for the reliefs are usually not considered. The advantage with this limitation is that the required data for the relief can be summed up as a height field. We specify the height value at each coordinate on a two-dimensional plane, and can map this set of points to the resulting mesh.

Obtaining a height field itself is not especially difficult, but making one that satisfies the properties accepted for a relief directly is another matter. The process differs considerably based on whether the subject used is 2D or 3D. Currently, three-dimensional models are the more thoroughly studied, so discussion of them will be first.

2.2 Summaries of 3D Algorithms

Starting with a three-dimensional model for relief generation is useful for a few reasons. The first is that when utilized, one can immediately obtain an extremely accurate height field to specify each point with for a new sculpture. When handled properly, this can provide our algorithm with "true" values that when compressed can maintain relative distances between elements in the scene. Conversely, with no processing on the data the height field will usually be insufficient for a relief, being too wide of a range.

Since there is already access to a realistic set of depth data, the goal of most 3D algorithms entail at least two tasks; a compression scheme of some sort to bring the model into acceptable relief height levels, and a rescaling factor to ensure that not too much depth information is lost. A balance between these two operations is sufficient to make a decent looking relief (as we'll see in Chapter 4), but there is a great degree of improvement that can be achieved using additional pre and post-processing techniques to further adjust the data. This section is divided into algorithms specified for each of the listed papers.

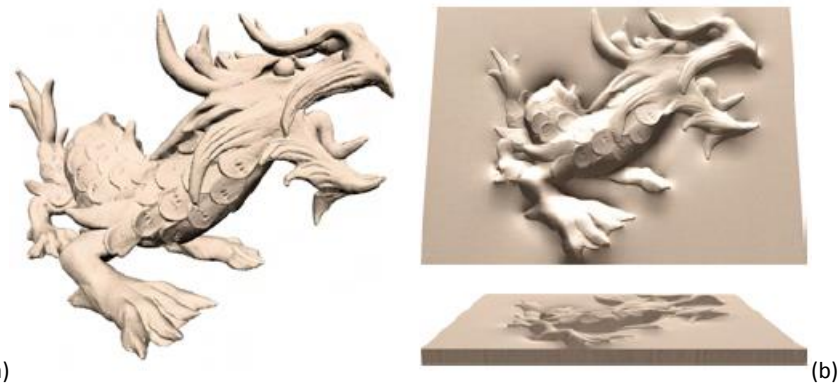
"Digital Bas-Relief from 3D Scenes" (Weyrich et al)

In "Digital Bas Relief from 3D Scenes", a semi-automatic relief creation algorithm is discussed. The input is a 3D model with an orthogonal camera view. The key to the approach is preserving the surface gradients of each of the points in the height field. The gradients help to define the specific type of shape that said point has, and are later used to help preserve finer features.

The height field is differentiated over, and depth compression is done in the gradient domain with their magnitudes, while the direction of each gradient is maintained. Finally, the resulting heights are recovered through an integration with the modified gradients.

In more detail, the technique uses a gradient compression scheme not unlike HDR compression in image processing (Fattal et al, 2003) . The mapping of the gradient magnitudes has a height limit, and those that exceed the limit are set to 0. Symbolically, this height limit is to help suppress the silhouettes and edges of the image, which will in most cases have gradient values that are orders of magnitude higher than others in the scene. The new height field is determined by a minimization with the least square difference. For implementation, they recommend treating the minimization process as a discrete Poisson system. The Laplacian of the resulting field is set to the divergence of the modified gradients. This rather difficult calculation can be expedited with a computer solver.

This paper also describes some possible editing operations for the artist using it, including feature scaling, and adjustments for differing material types. Changes that were suggested by the authors are to attempt parametrizing the relief over arbitrary shapes, which would likely improve shape accuracy. Further analysis on how the integration step is affected by different surface types was also discussed. Later in this thesis, we compare this algorithm to a more basic, direct approach to emphasize the benefits of more intermediate processing.



(a) (b)

Figure 2.1: Weyrich et al's dragon example (a) The 3D model used as input (b) The resulting relief from two differing angles

"Relief Generation Using Adaptive Histogram Equalization" (Sun et al)

In "Relief Generation Using Adaptive Histogram Equalization", another algorithm is provided that is based on a 3D model input. This method uses principles from Histogram Equalization in Image Processing. Each of the components of the given height field is sorted by a bin sort and mapped to new values for uniform distribution. The number of bins selected for use controls how many artifacts will remain: more bins means less artifacts and a clearer image, but longer computing time.

The precision of the equalization is also dependent on how large the neighborhood size used for sample points is. Larger values will lose some precision, while too small of values will lose data about the image. The equalization itself doesn't retain shape information, so the gradients are also weighted to maintain their information. The influence of neighboring points in the field is calculated with a Gaussian weighting function detailed in the paper.

The whole algorithm itself looks as follows: There's an initial input of a height field. The maximum height is found and then scaled down. A gradient attenuation function is then used,

which zeroes out any boundary points. Then histogram equalization is utilized, continuing until irrelevant heights are completely clipped from the data and redistributed properly.

Comparison to other methods shows that the histogram equalization is effective at preserving very fine features in the scene without blurring them out. Further, it generates a stronger outline on the background, giving it more of a sense of 3D in most cases. However, there is some apparent resulting noise and planar distortions, which can be further amplified depending on the set parameters. Another limitation is the ability to produce negative heights. While typically these can have an adverse effect on the mesh design, the authors mention that they can be useful for certain applications. Likewise, the planar distortions can sometimes add to the depth effect. Regardless, the method is stated to be somewhat unpolished and takes longer computation time. They mention some ways to improve the speed would be using sampling and interpolation for the data points. A suggested bucket sort approach could also reduce the order of complexity from $O(n^2)$ to an $O(n)$ operation.

"Feature Sensitive Bas-Relief Generation" (Kerber et al)

In "Feature Sensitive Bas Relief Generation", a filtering method is used from the input of the 3D scene. A binary mask is used on the data, and input is scaled by normalization (by setting the smallest foreground value to the new background value). The gradient of is calculated and then checked along a threshold, and outlier detection is done by looking for vast differences in gradient values.

Next, an adaptive attenuation function is used to compress the heights relative to their initial values. A bilateral filtering is also used with a 1D Gaussian kernel. The next step is a decomposition of the gradient into “coarse” and “fine” values. Coarse values correspond to larger details in the scene while the fine values specify smaller details. In modifying the relation between them, a broader range of all the details can be preserved.

The new height field is then calculated by computing the first order Laplacian on the new gradient values. After some post-processing, we get result. Notes by the author indicate that the method gives a somewhat elevated silhouette, but this can be adjusted with further Gaussian smoothing. The method typically only does one decomposition, but using more of them can distinguish between fine details and noise better. This method is also appealing because it only requires fewer input parameters specified by the user, compared to other methods where the user may have to guess the optimal values for the compression. This filtering procedure is effective at maintaining geometric detail in complex scenes. For further reading, some artistic applications of the algorithm are described in further detail, an example of which being a bas-relief with a cubist style.

“Real-time Generation of Digital Bas-Reliefs” (Kerber et al)

In “Real-time Generation of Digital Bas-Reliefs”, two generation procedures are described. One is referred to as the Range Domain Approach. Initially, a binary mask for finding background points is used, then decomposition and enhancement begins. This is done with Bilateral Filtering and a Laplacian diffusion, which helps mitigate possible noise added to the process.

Lastly, it uses an Unsharp masking procedure to boost the appearance of the finer details. For this approach, a post-processing rescaling operation is also utilized to fit size requirements.

The other approach is referred to as Gradient Domain Approach. The pre and post processing steps are the same as the Range Domain Approach. First large gradients (that indicate outlines) are adjusted to 0 gradient. This generates a continuous gradient image, which Unsharp masking is directly applied to. With the modified gradients, a new height field is constructed with solving a Poisson system like in Kerber's earlier approach.

Performance wise, the Range Domain Approach is fast to process and gives good sharp results at preserving coarse structures. However, this approach is less consistent with smaller details of the scene. For each procedure, the computation time is proportional to the complexity of the scene. The Gradient Domain Approach preserves finer features better, but takes more time.

For further improvement, bilateral filters compatible with the Gaussian transform can also increase processing speed.

"Representations of 3D Scenes in a Limited Depth Range" (Arpa)

In "Representations of 3D Scenes in a Limited Depth Range", The bas-relief generation technique involving the Poisson equations is mentioned, but the paper mainly focuses on a high-relief generation method. Rather than using the entire original scene, It uses a set of selected attenuation points, and calculates where all other points should be placed relative to them. To preserve the shape more effectively, differential coordinates are used rather than

cartesian, since they can also maintain information about the shape and curvature of the points. It is a concise, effective method at retaining good depths and features, but is limited to use with connected meshes as input. The authors indicate the method could be more generalized to generate any type of relief. For further modification, continuous surfaces, material composition, and plane deformations are briefly described in the conclusion.

"Computer Assisted Relief Generation – A Survey" (Kerber et al)

The survey of Computer Assisted Relief Generation by Kerber covers the 3 possible methods of input mentioned at the beginning of this paper (manual design, 2D models, 3D models). Benefits and downsides of each approach are considered. For Manual Digital Construction, designs are easier to make than in real life because you can undo mistakes and have greater control over the production. However, this approach requires a lot of time, and the user to be skilled at creating reliefs to begin with.

Figuring out how to create a relief from a given image can be difficult, and requires more specific information about features of the image to come up with a unique solution. However, since said image already exists, no design skill is required. Numerous methods for starting with 3D models have already been mentioned. It's generally an approach with high interest since much of the work about details is already provided in a model. Of course, this is the approach that is contingent on how depth compression of the scene is handled, so it can require a lot of computational work.

Another aspect the author mentions is that most research done on generating reliefs has remained with planar surfaces. Further work could be done on curved surfaces as well.

Also, there's currently no research on non-uniform material reliefs. The author believes that some combination of the techniques from the different existing approaches could create a breakthrough in the area.

Each of the 3D model algorithms described above uses differing techniques to approach the problem, but some ideas remain consistent. A non-linear depth compression scheme is almost always preferred, to act as a compromise between the height constraints and not removing to much depth information from the subject. Some work in the gradient domain of the scene is also commonplace, since the gradient is especially useful in edge detection, a procedure that has far more applications than just relief generation.

In the next section, we discuss some of the reading on 2D algorithms.

2.3 Summaries of 2D Algorithms

In image applications, the conversion of a 2D object into 3D is often referred to as an "ill-posed" problem. Usually if this process is necessary, it is ideal to have multiple images of the same scene from different perspectives, to better evaluate the true depths of the subject of an image. With just one image, there can be multiple interpretations, and consequently multiple solutions.

There is still some value to researching 2D algorithms however. The immediate motivation is that access to pictures and photos is far more common than having 3D models on hand. To get a reasonable result, some more data from that image must be considered. A standard approach, like for 3D algorithms, is to consider the gradient domain of an image. Information on edges and changes in intensity levels can help suggest what would be a reasonable interpretation, both for focal depth and relative ordering of objects if the picture is of an entire scene. Some of the more prominent papers about the topic are summarized here.

"Making Bas-Reliefs from Photographs of Human Faces" (Wu et al)

"Making Bas-reliefs from Photographs of Human Faces" begins to examine if the input provided is a 2D image. The outline of the process starts with two different lightings of the image. These are function fitted, and then SFS (Shape from Shading) is applied to them. Finally, the two separate relief surfaces from each image are combined to create the final relief.

Some methods used by earlier groups were also mentioned, mostly pertaining to 3D scenes. Many of the methods referenced were already described in the previous section, in addition to the earliest method; a perspective transformation of height fields for a 3D scene. While this allowed the creation of bas-reliefs, many detailed features were lost in the process. The most relevant algorithm mentioned for this paper is the one utilizing Histogram Equalization. This procedure ends up being used for this paper's method for the learning process.

Multiple experiments were made on the image lighting procedure. Modifying the scaling factor in saliency map calculation affects the level of detail the result has (higher values give

more detail, but also more noise). In considering the geometry of the resulting reliefs, the consensus is that the relief surfaces are smooth and detailed, but the lips are generally inaccurate due to limitations of the lighting learning procedure. But for this same reason, the detail of the eyebrows is preserved very effectively.

Human faces are a common subject to represent in images, and reliefs are no different, so the method described (based on neural networks, image relighting, and shape from shading) is shown to have some initial success. There are some improvements recommended by the authors. One possibility is improving the algorithm for point to point correspondence. Another is to use multiple training images of the same face.

"Bas-Relief Modeling from Normal Images with Intuitive Styles" (Ji et al)

In "Bas-Relief Modeling from Normal Images with Intuitive Styles", a method is described that includes no depth discontinuity, stylization (between "round" and "flat" reliefs), and is based in the normal image space. In further detail, it begins with the normal image of a scene. In this way, depth discontinuities can be ignored. Specifically, the input image is a surface normal image. The goals of this process are to restore geometry from the normal values. These contain fine details as well as no discontinuities. The process also squashes foreground objects against the background (making the flattened image) and preserving fine details in the process. From the normal values, the height field can also be calculated. This involves the Laplacians of the normal.

The modeling method also makes use of image layers. This is done with 4 layers including an alpha channel. The layers are linearly combined and blended Laplacians are integrated over. In camera space, normal images tend to look blue based on the normal spectrums. Height field recovery for this method occurs from the normal field. Not only does it keep the image free of depth discontinuity, but it also allows for gradients to be calculated directly from the normal (rather than requiring the neighboring pixels in calculations).

There is also a useful threshold parameter that determines balance of details and heights. This allows for the creation of both round-type bas-reliefs and flat bas-reliefs. The algorithm is observed to be useful for getting smooth relief results compared to other methods. This also preserves detailed features effectively. Normal images can be created from general images, as well as 3D models, so retrieving input is not too difficult.

A bilateral filter is used on the normal to help smooth the image and not lose edge features. The resulting models from the method can also be easily moved into other models or scenes. One noted problem with the algorithm was that based on its freeing of depth discontinuities, some special cases of subjects may not work. The example provided is a cube from an orthogonal view. The vertical edges would require a light sloping in order to be used. Another possible area of research suggested is working with grayscale images as input. The normal values could be calculated in this image by shading and silhouettes implicit in the image.

"Region-based bas-relief generation from a single image" (Zeng et al)

In "Region-based bas-relief generation from a single image", it is noted that there are usually very few 3D models readily usable for a bas-relief generation algorithm. As such, more commonly available 2D images become the focus. Shape from Shading is briefly discussed, as being mainly applicable to images of human faces, and even then, quite inconsistent with potential illumination problems.

The method proposed here attempts to simulate a sculptor's approach, is region-based, and applicable to many different types of input images (not just faces). Off the bat, the problem of generating the model from the image looks difficult, mainly since calculating depths isn't immediately obvious. Previous approaches described include a gradient and intensity based method, but it has difficulty keeping front to back relations of different image regions. Shape from shading is also described again, but still has its limited scope. Finally, sketch based methods are described, which this paper falls under.

The first variation uses simple line drawings, which while efficient loses a lot of surface detail. A line labeling method is also described, can help preserve some depth information, but only at a basic level. The algorithm in this paper goes for an approach similar to an artist sculpting something directly from an image.

It focuses on 3D geometric info implicit to the input image. Specifically, it indicates front-to-back relationships, salient features, and avoiding extraneous details. The two steps used are a "region based layer determination" and a "relief construction". For layers, the line image is utilized. The algorithm described in the paper "Coherent Line Drawing" (Kang et al) is

applied to obtain the line drawings, which involves working with the Edge Tangent Flow of an image. Seed points are taken from the produced lines, and they are used to determine specific connectivity between the regions of the image. With these connections, different regions are extracted and sorted by depth, and height values are determined. Then in the reconstruction phase, height values are used to build the base surface, and detail is added by using gradient and grayscale data from the input image. Another check is also done to assure front-to-back relations are retained.

The results of the method are of good quality, though it has problems in certain situations. Complicated shading hinders its effectiveness, and the procedure generally requires more user intervention to ensure accuracy. The amount of user modification required is proportional to the geometric complexity of the image, though it's still generally minimal. The approach is akin to how a sculptor may approach the same problem; starting with thicker outlines to indicate height differences and incrementally adding more of the important tinier details. Further discussion of this algorithm will be in Chapter 3, contrasting it to a more standard gradient domain approach.

"3D Digital Relief Generation" (Wang)

Finally, we look at "3D Digital Relief Generation" by Meili Wang. It discusses a few new algorithms for relief generation; one using 2D images as the basis, and another using 3D models. It is also unique in that it describes techniques to create "sunken-reliefs", where details are carved into the surface rather than extending out. The proposed methods are applicable to

three types of reliefs (high, low, and sunken). High relief is defined as more than half depth extending from the background, while bas relief has less than half depth extended. Bas relief creation is typically contingent on non-linear compression to maintain detail without shape distortion. High relief doesn't require as strict of standards.

The 2D image method presented by the paper does the following: converts input to grey level image, calculate image gradients, attenuate gradients for smoothing, Unsharp masking to enhance fine features, create modified image solving Poisson equation, apply gamma correction, and translate the result to a triangulated mesh to represent the final relief's shape. The results of the approach are fairly accurate, and the algorithm is quick as well. An implementation of this method and its performance is thoroughly examined as part of Chapter 3 in this thesis.

The 3D model method in the paper takes these steps: 3D Unsharp masking is applied to the model by means of subtracting Laplacian smoothing from the original mesh. This results in the mesh details, which are scaled by a set factor and added back to the original to boost its detail. At that point, the standard proportional nonlinear scaling scheme is used. The algorithm can be used to create both high relief and bas relief sculptures. It gives results that are comparable to the standard algorithms of its kind (like Kerber or Weyrich) and is also easy and efficient to implement.

The thesis then touches on sunken reliefs, which are mentioned in few other papers. They emphasize the importance that lines play in this type of relief. Many ancient reliefs were of sunken-relief type; where they were defined by thickness and depth of the lines that were

made in them. 3D models are the basis of the input for these two procedures. For better modeling, a combination of inputs including a line drawing image, a Lambertian image with proper light direction, and a depth image for height data.

The first algorithm focuses on line drawing, distinguishing between contour lines and suggestive contour lines. Then image processing techniques can be applied to the line sets to enhance quality. Before extracting the lines from a 3D model, some mesh smoothing may be required.

One unsolved problem in using these procedures is transforming the line information into the 3D carving and maintaining smooth height transition simultaneously. This is important to giving an authentic 3D feel to the image. The 3D object is divided into a depth image, a rendered image, and a line drawing. The depth input is compressed, the Lambertian image gives the reflection radiance, and the line drawing details engravings. After these adjustments, the three separate images are then composed together again for the final relief. The method with the three input parts generates smooth height information for the relief, which makes it a bit more convincing than the other method, though requiring more effort.

A few considerations for future research discussed include stylization, animation, and spatial arrangement. Deformation techniques could be applied to stylize certain reliefs. Continued improvement of techniques could help certain problems of relief production like overlapping objects, foreshortening, coloring, and lighting.

The bodies of work summarized above cover a good breadth of the research in simplifying the process of relief generation. In regards of the goal of this thesis, we will examine pairs of algorithms in close detail in terms of performance and ease of use. For each chapter, we describe our implementation of a basic method of each input type (2D and 3D) and then compare its results to a more sophisticated algorithm of that version (in terms of operations). Metrics will include the amount of time required for usage, an analysis of the types of results each algorithm provides, and the practicality of reproducing the results. After these comparisons, we conclude the thesis by showing some more of the results generated from each algorithm, and consider possible directions to take for improving results and ease of use.

CHAPTER 3

2D IMPLEMENTATIONS

3.1 Walkthrough of Basic 2D Algorithm

In this chapter, we discuss in further detail two different algorithms with the purpose of generating bas-relief from a 2D image input. There are several factors to be considered specifically for the framing of the two-dimensional problem. The list below describes some of the core important factors in the process:

(1) **Depth information**: As described in most of the papers on 2D image algorithms, the problem that manifests itself immediately is that there is no complete knowledge of the depth within a single image. A reasonable interpretation must be determined in some manner. Manually, a person can use their own experience with the lighting, shadows, and other visual cues to come up with a reasonable model, but mimicking this capability in an automated process is very difficult without some artificial intelligence approaches. One example of this was mentioned with the learning algorithm for faces, but even this required multiple images with different lightings and additional time.

(2) **Silhouettes and Edges**: More generally, they can be referred to as outlines. In most images, these provide visual information in distinguishing between separate objects and the background. For 2D images especially, it is useful to generate a reasonable contrast between the foreground objects and the background as a starting establishment for interpreting heights.

(3) **Fine Details**: Many images, often photographic ones, will contain many small features and details within the subject of the image. Examples are features like the texture of a backpack, the shininess of an apple, or the claw of the Sphinx example. In the process of converting 2D data into heights, it is preferable to maintain, some information about these features to keep them in the result and not lose too much information about the subject.

The initial idea that may come to mind resolving these issues is to translate intensity values of a given image into heights. This can make a three-dimensional result, but often it will distort the appearance of the initial subject. One technique that is commonly used to aid in shape preservation is to also work in the gradient domain of an image. The benefits of this will be expanded upon in discussion of the following algorithm.

Relief Generation on a 2D Image with the Gradient Domain

The first approach discussed, based on Meili Wang's proposed method for reliefs from 2D images, utilizes the gradient domain to retain fine details of the image. It also uses a non-linear attenuation function to maintain a foreshortening distance appearance.

The algorithm follows these steps from the 2D input image:

1. Translate the image into grayscale
2. Calculate the gradient domain of the image
3. Perform attenuation to compress the image
4. Use Unsharp masking to extract the fine features from the gradient

5. Reconstruct the image based on the Unsharp masked image to use as a height field
6. Apply a gamma correction scheme to mitigate deformations
7. Generate the mesh based on the corrected result

The method progresses linearly through these steps. We now go into further detail of each step in the same order.

Figure 3.1: An image of a hand-drawn dog. This will be the sample image for the following intermediate steps. This algorithm works especially well on images with simpler details.

Gradient domain calculation:

In most environments, it is straightforward to obtain a grayscale version of a given image. My implementation uses the OpenCV library to perform this operation mainly for its robustness. The idea is to convert an image of any common given format into an 8-bit grayscale image, the data of which can be directly manipulated.

With the grayscale image, the first step is to calculate the image gradient. The general definition of the gradient applies here; a first-order derivative vector that indicates the change of a value in each direction. In this application, our interest lies in determining the differences

in intensity between each pixel. For simplification, we work with the x-gradient (gradient values along the x-axis) and the y-gradient separately.

Since we are working with digital images, we can only work with discrete gradients. There are many ways to calculate the image gradient. One common way is to use a filter. A standard example of a gradient filter is the Sobel operator. But for simplicity, Wang proposed using the forward difference at each pixel, which still provides a reasonable estimate for a discrete derivative. The formulation is straightforward:

$$grad_x\big(I(x,y)\big) = I(x+1,y) - I(x,y)$$

$$grad_y\big(I(x,y)\big) = I(x,y+1) - I(x,y)$$

Here, $I(x,y)$ represents our initial gray-scaled image (written as a function value for each point), a standard representation. For a direct implementation of this approach, a small padding step for the image may be considered.

The gradient image is what is operated on in all subsequent steps. The gradient is useful for one main reason; to determine edges of an image. Edge values will either have significantly higher or lower gradients in comparison to other data points within the image. For 2D images, it is especially important to maintain the edges to preserve the shape of the final relief. Calculating the gradient can be considered a linear-order operation, where run-time is dependent on the total number of pixels in the image.


(a)

(b)

Figure 3.2: The gradient domain of the image. (a) x-component gradient (b) y-component gradient

Gradient Attenuation:

The gradient calculation is useful as is for differentiating the heights between objects. But the magnitudes of the gradients can still differ significantly, which can either fail to meet the depth constraint or distort the model's shape if used directly. To keep a reasonable relative distance between features, a compression step is required.

In most relief generation applications, a non-linear attenuation function is preferred as it can accomplish reduction of the overall height range while still maintaining details provided by the smaller heights of the image. This is accomplished by reducing higher gradient values by a larger factor than lower ones.

The attenuation proposed and implemented here is based off Fattal's proposed attenuation function for high dynamic range compression (HDR). The formulation is:

$$A_x(x,y) = \frac{\alpha}{\|grad_x(x,y)\|} * (\frac{\|grad_x(x,y)\|}{\alpha})^{\beta}$$

$$A_y(x,y) = \frac{\alpha}{\|grad_y(x,y)\|} * (\frac{\|grad_y(x,y)\|}{\alpha})^{\beta}$$

Defined are two modifiable parameters, α and β. α determines the lower limit for not changing the gradient value, whereas β, if between 0 and 1, sets the upper limit. Gradients larger than β are attenuated by the function, whereas magnitudes smaller than α are enhanced. The suggestion by Fattal was to set $\alpha = 0.1$ and $\beta = 0.9$ specifically for relief generation. Since we're working with one-dimensional gradient components, the norm of the gradient can be simplified as the magnitude.

(a)   (b)

Figure 3.3: The gradient images after attenuation is applied. (a) x-component (b) y-component The order of applying attenuation is also O(n), where n is the number of pixels in the image.

Unsharp Masking:

Unsharp masking is a common procedure in Image Processing that is used to enhance smaller details within an image. In a high-level context, the procedure can be described as a subtraction between the original image and a blurred version of it, which is then added back into the original image. This leads to a sharpening of the image. Many of the smaller features of the

image are captured as gradient values of high magnitudes in the gradient domain, and applying this boosting procedure highlights them. Formulaically, Unsharp masking can be written as:

$$M_x(x,y) = A_x(x,y) + \delta * (A_x(x,y) - blur(A_x(x,y))$$

$$M_y(x,y) = A_y(x,y) + \delta * (A_y(x,y) - blur(A_y(x,y))$$

The parameter δ is introduced here to allow scaling of the Unsharp masking. Larger δ values will increase the difference in boosting between the higher and lower magnitudes. It is noted that this parameter can be useful in stylizing the result of the algorithm. In Wang's experiments, she found $\delta = 5.0$ to be a reasonable value.

The blur function included is a stand in for any sort of blurring of the image. Wang's paper suggested the usage of Gaussian Smoothing. The implementation of the algorithm for this paper applies a basic Gaussian filter of standard deviation 1 to the image for the smoothing, which can be modified in size to change the impact. The blurring operation, when performed as a filter, grows based on the size of the filter applied to each pixel.

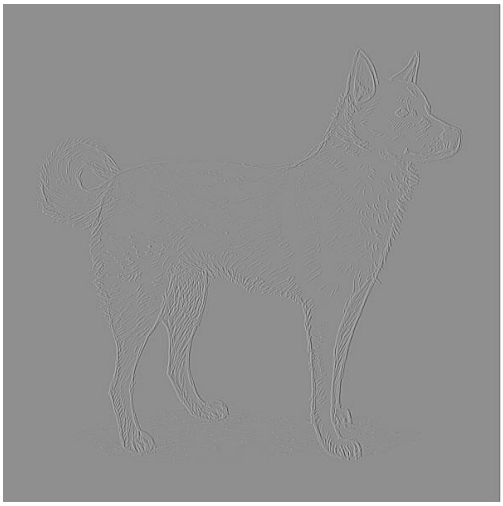
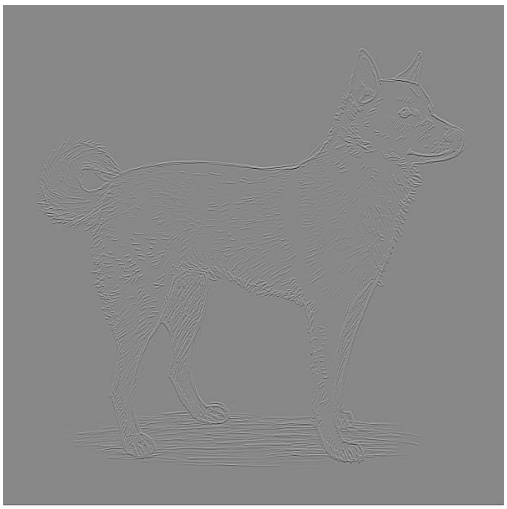
(a)   (b)

Figure 3.4: The result of boosting the gradients with Unsharp masking. (a) x-component (b) y-component (It may be a bit difficult to see because only the finest features remain)

Image Reconstruction:

With the modified gradient values that were calculated from the previous steps, we now want to reconstruct the new image. Specifically, we want an image whose own gradients are a close estimate to the gradient values we obtained from the previous steps (here, M_x and M_y). The approach taken for this type of image reconstruction treats the problem as an integration of the difference of least squares. Equivalently, a discrete Poisson System can be set up to find a solution. In this case, the system would look like:

$$\nabla^2 R(x,y) - div(M(x,y)) = 0$$

With the function R representing the height image to be reconstructed. To solve, we can set up the problem as a system of matrices of the standard form $Ax = b$. The matrix b is the calculated divergence of M. This is written as the sum of the gradients of M, or:

$$b = div\big(M(x,y)\big) = \frac{\partial M_x}{\partial x} + \frac{\partial M_y}{\partial y}$$

The matrix A is the Laplacian matrix applied to that problem. Using a scientific computing tool such as MATLAB allows for abstraction in calculating the Laplacian matrix for the specific system. If it does need to be calculated however, one approach of construction considers the size of the system.

If the given image is of size m*n, a Laplacian matrix of ((m-2)*(n-2)) * ((m-2)*(n-2)) can be formed. Within this matrix, a set of (n-2)*(n-2) square matrices can be fit (m-2) times in both dimensions within the matrix. Using the square Laplacian filter and the Identity matrix on the first and second bands of the matrix respectively will work as a custom built Laplacian matrix.

This system does not account for the boundary points of the image however, so boundary conditions can be set, usually either as the original image pixel values or zeroes, to create an edge tapering effect. The result of this image reconstruction is what will be interpreted as the height field for the bas-relief. The complexity of solving the system grows cubically with the size of the Laplacian matrix, making this the most expensive operation in the algorithm.

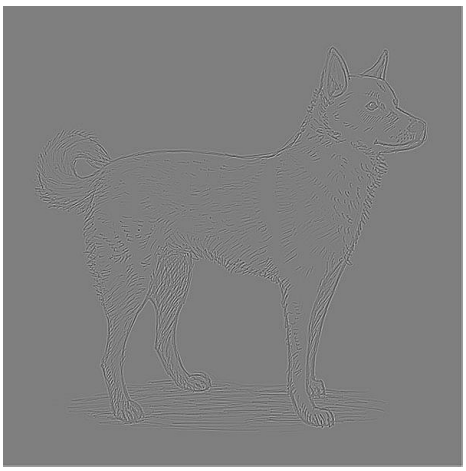
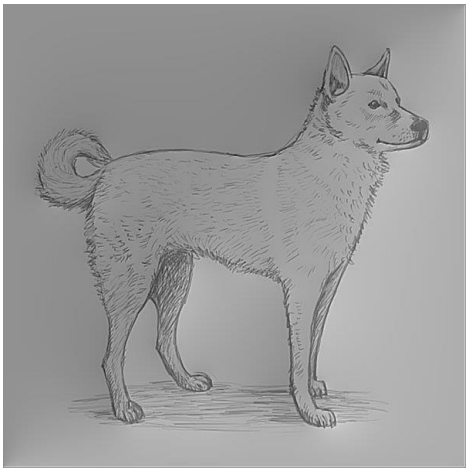
(a)   (b)

Figure 3.5: (a) The input image for the reconstruction (the result of divergence of the boosted gradients) and (b) The resulting image from reconstruction with the Poisson System. Used as the base for the relief.

Gamma Correction:

A final post-processing step is recommended before attempting to generate the mesh from the output. In this algorithm, gamma correction is used to reduce image distortion and deformation that could have propagated through the previous steps. By utilizing a gamma power function, the final image can have its contrast enhanced to further differentiate between the heights of its features while retaining a sufficient level of compression. The correction function used here is:

$$I'(x,y) = \left(\frac{R(x,y) - \min\big(R(x,y)\big)}{\max\big(R(x,y)\big)}\right)^{1/\gamma}$$

γ is used for the image gamma value. A gamma value larger than 1 enhances the contrast in darker regions. For the current output, gamma is set to 2.5. The function also normalizes the values. As expected, the minimum and maximum values of the reconstructed image are calculated prior.

At this point the data is sufficient for forming a relief, although any number of post-processing options could be used. In this implementation, the output values are scaled between 0 and 1, which can be difficult to see in terms of a resulting mesh with no other attributes, so some additional scaling is used to emphasize the outlines.

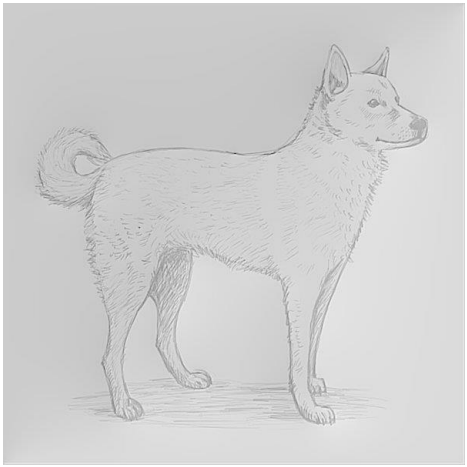


Figure 3.6: The image after applying the gamma correction function. This result is used as the height field for generating the mesh.

Result and Discussion

As figures in this section are a few sample reliefs generated from the implementation of this algorithm for the thesis and the images they are initially based on. For this analysis, the procedure was CPU implemented in C++ with usage of the OpenCV library for its useful

conversion features, and with the Eigen library for improved computational efficiency working with matrices.

The running time of the algorithm is reasonable. For smaller images, like the sphinx model based on Wang's example, the process finishes in a few seconds. Larger images that were tested (roughly 600x600) can take upwards of two minutes, though testing even larger than that would increase the time based on the current implementation. The bottleneck for the procedure is without question the Poisson System solver. Both generating the Laplacian matrix suitable for the given image (which grows quadratically with the image size) and solving the system are major components. However, since the Laplacian matrix is a Sparse matrix (with roughly 5 values per row at most) sparse matrix operations were used to slightly improve performance. Alternative solution methods could also be considered, such as an iterative method like the Fast Fourier Transform.

(a)   (b)

(c)   (d)

Figure 3.7: A comparison between (a) This output image (b) This output mesh (c) Wang's output image (d) Wang's output mesh based on the algorithm. Notably, the input image size for Wang's result was much larger (640x480) as opposed to the one used for this test (125x101) resulting in additional artifacts.

There is a fair variation in the results, as the paper originally suggested. The algorithm is best suited for images with simpler details and less complexity in shading. The intermediate steps for the sphinx model when running the algorithm approximates Wang's results, but the image used was about a quarter of the size of her sample image. This results in a coarser mesh with some jagged edges if left unprocessed. Simpler and slightly larger images, like the dog and Pikachu™ drawings look more reasonably smooth and retain their finer details quite effectively.

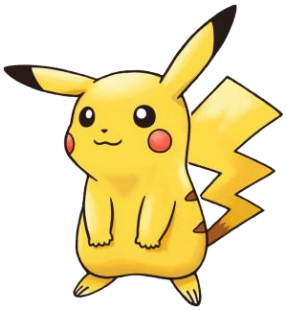
A few notable issues when working in the gradient domain are stability and contrast. In Scientific Computing terminology, the gradient among other derivative operations are deemed unstable in comparison to the spatial domain. Since they are more sensitive to changes in the data, they are also more sensitive to noise introduced in intermediate steps. If not handled properly, such propagated noise can greatly disrupt the appearance of the final relief, a phenomenon that occurred frequently during earlier testing. Extra processing to search for unexpected outliers in data can help stabilize the algorithm increasing its robustness. A light smoothing operation (such as a Laplacian or Taubin) on the final relief can also subdue more of the jagged, inconsistent edges of the scene.

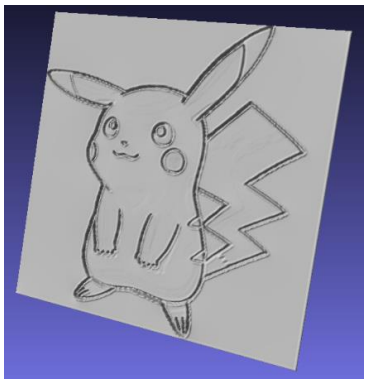
Contrast in the final image is also important in retrieving sufficient depth information. Since the edges of the image can have significantly higher magnitudes than all other data points, a compression of the more intermediate values can become apparent in the image, causing more uniform intensity that in turn reduces the depth range of the subject. The final

gamma correction step helps to some extent here, but further contrast enhancement could be applied in post-processing. To further differentiate the focus of the relief from the background, a background subtraction operation, often used in video processing, was considered, and could be effectively applied here to give a further rising effect. Overall, the method is effective for its simplicity to write.


(a)


(b)


(c)


(d)


(e)

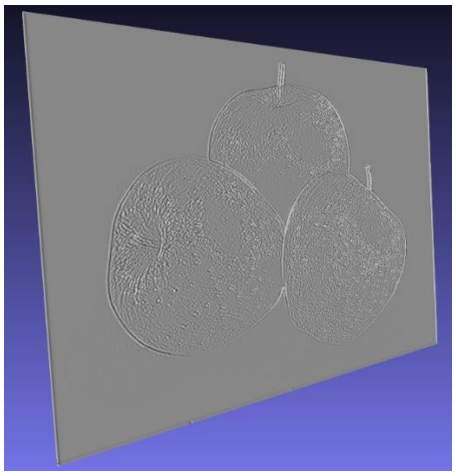

(f)

Figure 3.8: More sample mesh results shown adjacent with their input image:

(a)-(b) Line Drawn Dog (c)-(d) Pikachu™ (e)-(f) Apples

The results fall in line with expectations, where simpler images more easily retain their features.

3.2 Walkthrough of an Advanced 2D algorithm

Next, we describe another 2D algorithm that was implemented by a fellow assistant. This approach is based on Zeng's paper "Region-based bas-relief generation from a single image". The general approach was summarized in Chapter 2 among other methods, but here the outlined steps are considered in further detail. Rather than mainly relying on the gradient, this procedure focuses on dividing the image into separate regions that can be used to determine relative ordering of the scene. Our steps of focus can be categorized as follows:

1. Conversion of the input image to grayscale
2. Feature Line Extraction from the input image
3. Region layers determined by feature lines
4. Image gradient calculation
5. Model base surface based on regions
6. Generate depth map with gradient and surface information

Feature Line Extraction:

We once again assume that the reader can find a reasonable way to convert to a grayscale image without too much difficulty. With our input image, we want to focus on feature lines. There are a few ways to consider this problem, but the algorithm describes the approach

proposed by Kang et al in “Coherent Line Drawing”. This is another extensive procedure in and of itself, so we just summarize the paper for this step.

The two primary steps discussed in this paper are calculating the Edge Tangent Flow of the image and performing a Flow-based Difference-of-Gaussians. The ETF is calculated by use of a filter that uses spatial weight, magnitude weight, direction weight, and the tangent vector to the point. An iterative procedure for Flow Difference of Gaussians with a filter is also described. The idea is to measure a region around the point to determine genuine edges as opposed to tiny, spurious ones that would be unhelpful for the result.

The implemented method for this step leaves some undesired small lines from shading, so an L0 Smoothing process is applied to the feature lines. To further eliminate noise and unnecessary data, a pyramid scheme is implemented to down-sample the image a set amount of times. Then, the line extraction technique is performed iteratively on each size, and as the pyramid is up-scaled again, important feature lines are tracked as ones that consistently appear in each size of the image.

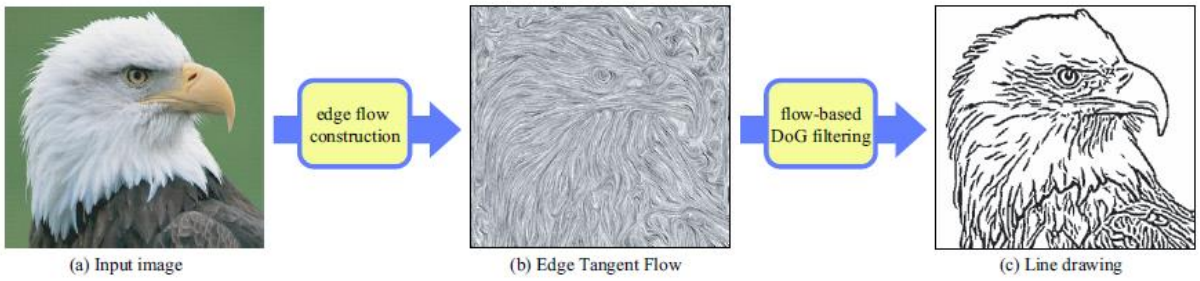


Figure 3.9: The process of generating a feature line image, as illustrated in Kang et al’s “Coherent Line Drawing”

Region Determination:

With the feature line input at this step, we want to create a region map that can later be used to help maintain ordering of the depth in the image. A step to determine 'skeleton pixels' is utilized. These are single pixel wide lines retrieved from the feature lines used for region divisions. With these, seed points are selected, and a heap sort is utilized based on their height distance values. The region growth is stopped at skeleton points or pixels already reached.

It is explained that this process alone separates the image into too many regions, so some are combined based on how many shared pixels they have, assuming there is no skeletal pixel separator.

With the set of regions, the next step is to determine their relative depths to each other. Relationships considered between the regions are referred to as "containing", "neighboring", or "disjoint". To represent these relations, a two-level graph is used, to represent the foreground and background of the scene. Junctions (the connections between the segments of the graph) are used to try determining the relative depth of the region. It is noted that a lack of junctions will require user intervention to properly place that specific region.

Image Gradient Calculation:

Like with most of the 2D image algorithms discussed in research, using the gradient domain image is also of assistance. Here, large gradients represent margins in the image that are orthogonal to the feature lines, and smaller gradients are found between segments.

Base Surface Generation:

With the regions sorted and the heights determined, a base surface for the relief can be structured. Here, the surface is represented as a basic offset function, where the background plane is set to height 0 and features above the background are represented by their distance above it. The height field determined by the regions and the gradient information are the primary sources for the depth measurements.

With this base surface, the relief is then represented as a triangular mesh using a Delaunay triangulation approach.

Results and Discussion:

The algorithm uses a notable amount more information than the gradient domain algorithm previously discussed. Observations noted from our C++ implementation were that the region determination sequence and the initial L0 smoothing of the image require a high degree of computation time. Kevin's CPU implementation of this algorithm took upwards of 12 minutes to process a moderately sized image. However, utilizing a GPU implementation of the same program greatly expedited the results.

With the region operations, the resulting reliefs are dependent on the balance between the base surface structure, the gradient image, and the initial grayscale image. From experimentation and the author's recommendation, precedence should be given to the depth map to best preserve front-to-back relationships of the regions.

This algorithm provides more control and detail in the final relief than the gradient domain approach, but has a few additional difficulties. Since not all the regions can always be properly determined with the image information, more user intervention is required to get an ideal arrangement. With an optimal setup, the result will have less flat regions overall. But like with the gradient, the approach is less reliable with more complicated shading in the image. The user interaction for proper results in the algorithm increases accordingly with the difficulty of the scene.

Figure 3.10: The output of a small flower image based on this algorithm, with added texture. The range between the foreground and background is more obvious than the internal features, a trait typically resulting from 2D algorithms.

Comparing the two previous approaches, we get an illustration of the important aspects specific to the 2D image problem. The largest hurdles involve properly gauging the relative distance of each of the elements in the image, as well as an important necessity of separating the background from the foreground. Common problems with this type of algorithm include

difficulty in handling especially detailed shading, and usually mandatory necessity for the user to intervene in determining the final appearance.

Some of these features are not as significant for the 3D model input problem. We look at that version of the relief-generation problem in the next chapter.

CHAPTER 4

3D IMPLEMENTATIONS

4.1 Walkthrough of a Basic 3D algorithm

Starting from a 3D model to generate a bas-relief for most people is the more intuitive approach. The main reason is that it resolves the largest issue that every 2D approach to the problem has. Namely, real depth values for the scene can be directly obtained to allow for a higher accuracy starting point. Most early research in semi-automated relief generation was based on utilizing full three-dimensional scenes.

With this view, one could see compression as the only required step for producing the relief, but maintaining necessary detail for a coherent scene is difficult without considering some other factors:

(1) Foreshortening: While we are interested in flattening the 3D model to fit the relief, an important element of 3D based bas-reliefs is to retain that perception of depth, as if the object being viewed is projecting back into the surface of the relief. A term commonly associated in art and photography, foreshortening is used to describe the phenomenon that elements of a scene closer to the viewer appear flatter than those in more distant planes. If we just apply a linear compression to the 3D model, this effect would be lost and the result would appear too flat.

(2) Discontinuity: Unlike in a 2D scene, since a 3D scene uses a full range of depth, spatial discontinuities are much more apparent, especially when considering multiple disjoint

objects. Leaving these discontinuities in the scene increases the difficulty of a proper compression, and can ruin the desired effect of the bas-relief by leaving large gaps in the surface. Handling discontinuities in a way that retains an accurate front-to-back perspective of objects is important for a proper looking relief.

(3) **Feature Preservation:** Similarly to 2D based reliefs, attention must be paid to retaining important features that both detail the object of the scene and give further depth cues to the viewer. Any type of compression scheme inherently threatens the saliency of such smaller features, so additional processing operations are useful so information can be re-obtained after the compression.

While the problem based on the 2D and 3D models differ, some operations are applicable in both situations. The gradient domain can still provide useful information about the visible edges of a 3D scene. Smoothing and sharpening techniques like the previously explained Unsharp-masking can help reduce noise from processing and preserve detailed features respectively.

First we look at a straightforward approach to the 3D bas-relief generation problem, and provide specific details on implementation.

Relief Generation from a 3D model based on a depth map:

Physical 3D models that can be used as a basis for bas-relief generation aren't especially common to the public outside of perhaps a hobbyist 3D printer. This method instead works with a digital 3D mesh as its input. It requires three basic steps:

1. Generate a depth map from the 3D model
2. Apply a non-linear compression function on the depth map
3. Apply a linear rescaling of the resulting compression

We discuss each procedure in order:

Depth map generation:

This is arguably the most important step in the process. The practicality of a depth map for a digital mesh is that it can provide depth information about the model in a normalized range. Having an interface to obtain a depth map with is also useful, as it allows the user to choose what angle to view the scene from before generating the relief. With an application such as MeshLab or a basic OpenGL viewer, the depth map can be obtained by reading the z-buffer, as suggested in Weyrich et al's "Digital Bas Relief from 3D Scenes".

This step is the most sensitive in determining the results of the algorithm. The user needs to set the near and far plane such that as much contrast as possible is obtained from the scene. In a program like MeshLab, the contrast can be measured with a depth map filter, where darker values are closer to the camera. When satisfied with the positioning, a 2D snapshot is then taken of the depth map. This approach to retrieving the depth values is convenient, since

the user set perspective already applies foreshortening to the height map. If the depth map needs to be calculated manually, one can utilize a ray-tracing approach at each pixel in the camera view in a linear-order operation.

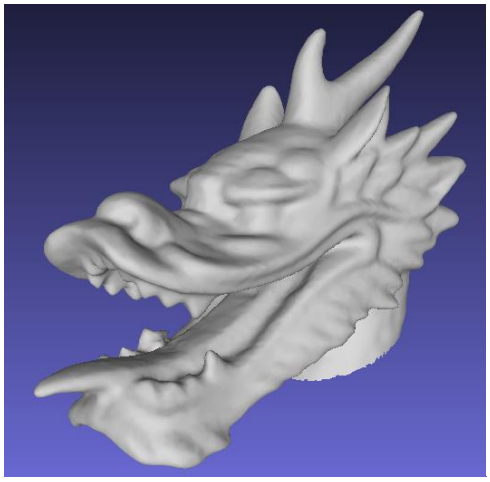
(a)   (b)

Figure 4.1: The dragon head is used as the sample input: (a) The original mesh (b) The generated depth map.

Nonlinear compression:

Like in most other bas-relief generation algorithms, a non-linear compression scheme is selected to prevent suppressing too much detail from lower heights. The one selected here is again based on Fattal et al's logarithmic function. This type of compression function suppresses the larger height values unlike the gamma functions, which are typically used to raise the lower heights to enhance contrast. With this step alone however, there is still too much distance between the subject and the background for the relief. The final step is used to mitigate this issue.

Figure 4.2: The depth map after non-linear compression. The background is still distant in this example. The depth measure is also inverted to fit with the convention of larger intensity representing larger heights.

Linear Rescaling:

Since the depth data we have been working with thus far is between the 0-1 range of the initial depth map, a normalization to a larger range of depths is desired for a better sized relief. This rescaling also re-extends the details from the smaller heights of the image, without overly undoing the natural foreground foreshortening from the initial input. The result of this rescaling is then used to create a 3D mesh for the relief.

Figure 4.3: The depth map after linear rescaling and a background elevation. This is used for the following mesh.

Results and Discussion:

The implementation of this algorithm is especially quick with only two key operations. It is also robust in that a scene from any angle may be used. The CPU run-time is also quick because of this, so it can be a very fast way to at least get a base model for a bas-relief.

There are many limitations, however, from missing some detail preserving procedures of more sophisticated algorithms. The program, while easy to write, is highly dependent on the user in two ways. The first is based on the quality of the depth map as input. While nice for providing real depth values, detail is easily lost depending on the amount of contrast within the scene, and smaller details will be more difficult to preserve in general.

In addition, the effectiveness of compression and scaling schemes used is also dependent on the input, so the user may have to test different combinations and functions to improve the quality of the result. Another useful post-processing technique before writing the mesh is to elevate the background. By default, the background is at 0 height, and so is

unchanged by the functions applied to it. Raising the background to be slightly below the lower data points of the mesh indirectly increases the contrast of the subject further, as well as smoothing the slope between the background and the edges.

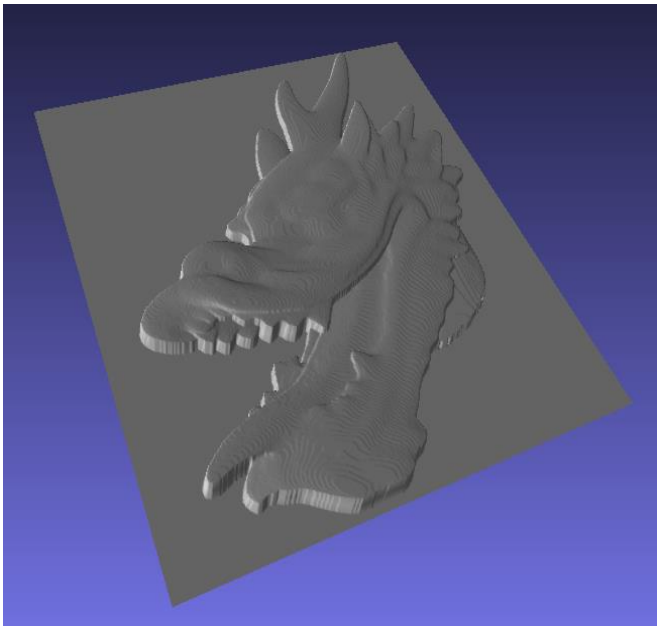


Figure 4.4: The resulting mesh from the dragon depth map. The quality is heavily dependent on the degree of detail that remains in the original map.

Next we look at Weyrich et al's proposed method of relief generation, which utilizes further steps to retain scene information and requires less input from the end user.

4.2 Walkthrough of an advanced 3D algorithm:

In comparison to the basic algorithm, the one described in "Digital Bas-Relief from 3D Scenes" can be initialized with a similar type of input, but also considers many additional factors of the geometry to improve the accuracy of the final mesh. The steps described in detail are:

1. Read in the input of the 3D model/scene
2. Calculate the gradient domain of the scene
3. Maintain the direction vectors of the gradient scene
4. Apply non-linear depth compression to the gradient magnitudes
5. Recombine the gradient directions with magnitude
6. Reconstruct the height field with the updated gradient information

Reading the input:

The procedure for reading in image input is nearly identical to the simplified algorithm. Utilized are the natural foreshortening provided from the camera view, as well as normalized depth values for the entire scene that can later be rescaled.

Gradient Domain:

In algorithms working with the 2D version of the relief generation problem, some work in the gradient domain is almost always compulsory to start evaluating height differences in a scene. With a 3D scene, this process isn't quite as important since we work with realistic height values, but the gradient domain still retains its practicality for edge detection, which can be used to sharpen features and outlines.

For simplicity, the paper proposes gradient calculation by using forward differences, but in our implementation, we utilized central differences. Both are effective approximating the derivative, but can have slightly different results. The central difference can also be described in a straightforward set of equations:

$$G_x(x,y) = (H(x+1,y) - H(x-1,y))/2$$

$$G_y(x,y) = (H(x,y+1) - H(x,y-1))/2$$

$$G_m(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

In scientific computing, subtraction and division operations typically make results of floating point arithmetic less stable under extreme conditions. In context without potentially overflowing data however, the central difference is a more accurate approximation for the discrete derivative than either the forward or backward difference.

In this step, the magnitude of each gradient is also calculated with a standard Cartesian distance formula. A simpler but less accurate approximation alternative is to simply sum the magnitudes of each component.

Maintain gradient direction vectors:

The direction of the gradient contains some geometric information about the scene that can be used to preserve details, so it is saved for further use. We can get the direction vector as follows:

$$\vec{v}(x,y) = \nabla \mathrm{H}(\mathrm{x},\mathrm{y})/\|\nabla \mathrm{H}(\mathrm{x},\mathrm{y})\|$$

The gradient value at each point in the scene is divided by its magnitude to give the normal direction of the vector. Splitting this into the x and y components, we can use the formulation as:

$$\overrightarrow{v_x}(x,y) = G_x(x,y)/\|G_m(x,y)\|$$

$$\overrightarrow{v_y}(x,y) = G_y(x,y)/\|G_m(x,y)\|$$

from our previous work.

Figure 4.5: Visual representation of gradient directions of the scene input. To note are the contours formed by "layers" at the same directions.

Non-linear depth compression:

In this process, rather than manipulating the height field directly, the gradient magnitudes are the input for the depth compression scheme. The advantage of attenuating the gradient magnitudes is that it can smoothen the slopes between points in the final height field. There are a couple of factors to consider when working with the gradient magnitudes.

First, we need to deal with silhouettes in the image of the scene. Silhouettes can be thought of as edges inside or outside of the subject. As expected with gradient calculation, magnitudes in these areas will be vastly larger than any others in the scene. To suppress the

appearance of these silhouettes, as well as allow for more compression range of the other details, we need to account for them specifically in the function used.

The suggestion for addressing this potential issue is to set a silhouette threshold as a parameter. Any magnitude values larger than this silhouette threshold are labeled silhouettes, and will be suppressed.

This algorithm also uses the HDR compression techniques analyzed by Fattal et al. Here, the compression function is represented as:

$$C(x) = \frac{1}{\alpha} * \log(1 + \alpha x)$$

The parameter α controls compression rate, with higher values correlating to more compression.

Combining this attenuation function with the thresholding of silhouette values, the entire function can be described in a piecewise way:

$$G'_m(x,y) = \begin{cases} C(G_m(x,y)), & G_m(x,y) < s \\ 0, & G_m(x,y) \geq s \end{cases}$$

Where s is the defined silhouette threshold. Values that surpass it are brought to the background height of 0, which can help whatever they're outlining stand out further. A suitable silhouette threshold for a given scene usually must be determined by experimentation to provide the most contrast.

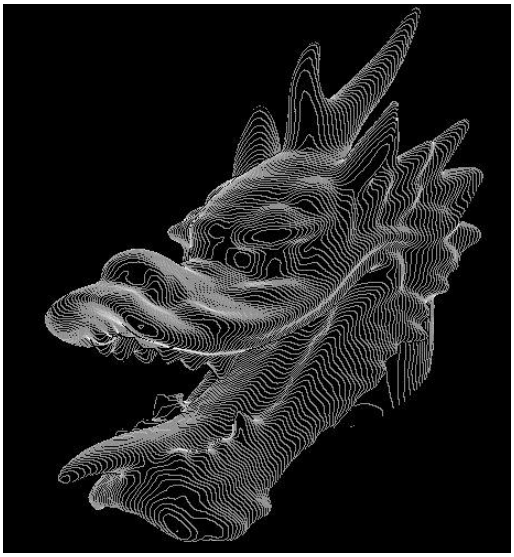


Figure 4.6: A visualization of compression on the scene input. Notably the largest gradient values (silhouettes and the outline) have been zeroed out, but more testing of parameters can be used to mitigate the number of visible contours.

Recombining Gradient information:

With the updated gradient magnitudes, we now want to re-apply their original direction information to aid in the reconstruction process. The procedure is simple:

$$U_x(x,y) = G'_m(x,y) * \overrightarrow{v_x}(x,y)$$

$$U_y(x,y) = G'_m(x,y) * \overrightarrow{v_y}(x,y)$$

Height Field Reconstruction:

The desired height field to use for generating the bas-relief can be obtained by integration over the recombined gradient information calculated in the previous steps:

$$H'(x,y) = \arg min \iint \|G(x,y) - U(x,y)\|^2 \, dxdy$$

Which is a minimization of the least square difference between the original and new gradient fields. Rather than performing this integration directly, we reformulate it into a discrete Poisson System to solve:

$$\nabla^2 H'(x,y) = div(U(x,y))$$

Here we can recombine the x and y gradients through the divergence operator. This system can be solved in a near identical approach to the implementation explained in Chapter 3.

Results and Discussion:

The result of the height reconstruction can then be directly translated into a mesh, though other possible post-processing steps can be used for enhancement. Utilization of the gradient information combined with complete depth data allow for a relief with more detail and depth range than most 2D algorithms, as well as more saliency than the basic approach in the last section.

Run-time for the processing of this algorithm is relatively quick, similarly to the 2D gradient approach. But like that approach, the same issues with the size and complexity of the Poisson System can greatly extend the computation time.

Working with the gradient magnitudes specifically can also create visible contours within the final relief if not accounted for. Since the magnitude is measuring the degree of change in height within the scene, the contour lines can appear if there is still wide variance between non-silhouette gradient values. Depending on the desired appearance of the output, this may be considered a drawback.

There are a few ways to mitigate the contrast. Working with the compression and silhouette parameters can respectively flatten the contrast more and eliminate higher contours, though the determination process would be very dependent on the base model, and higher compression can cost more details and depth. One suggestion provided in the paper is a down-sampling process on the gradients before the reconstruction. A down-sampling or down-scaling could blur the contours together, making the relief look less disjoint in appearance. However, this down-scaling and up-scaling procedure can introduce additional noise to the final scene.

While being conceptually easier than the 2D version of the problem, the 3D bas-relief generation problem still has its own set of challenges for improvement. The time to process and retrieve input information from a 3D scene can take more time and effort, but the payoff is the ability to skip guessing specific heights in any arbitrary image. Designing a good compression function is also more important when working with 3D input, but it is easier to get more use out of the entire depth range.

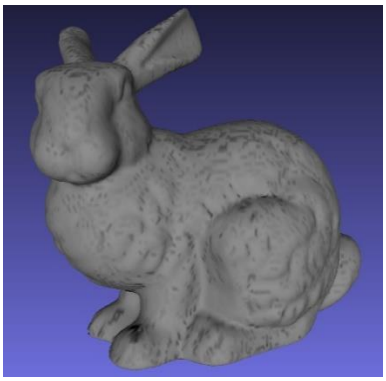
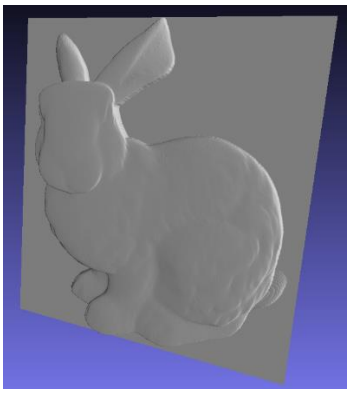
(a)   (b)

Figure 4.7: Another sample result from the algorithm of 4.2, based on the Stanford Bunny.

CHAPTER 5
CONCLUSIONS

5.1 Conclusions and Further Research:

In this thesis, we surveyed a multitude of the current and previous research focused on simplifying the construction of bas-relief structures. Procedures proposed to work on both images and models as initial inputs were discussed. In observing the approaches to this problem, a few common operations from graphics and image processing appear to apply to both types of problems.

The thesis also compared the performance and usability of the methods. Some of them are shown to be basic enough in implementation to be commonly utilized even outside of the area of research. Such methods, though perhaps inadequate compared to those more advanced and computationally intensive, have their own importance in accessibility to get more people interested in the problem, much like the increasing popularity of 3D printing among hobbyists.

However, many of the algorithms can still require higher end systems and GPU optimization to be practically used. Even the simpler algorithms at this stage will require the designer's input to ensure a correct appearance, though the amount of this required intervention seems like it will continue to decrease over time.

In terms of the 2D image version of the problem, the results have been quite satisfactory, especially given the initial challenge of interpreting depth. Post-processing

techniques like smoothing on the final relief can improve the shape of the mesh, but must be carefully balanced to avoid smoothing out finer details. Most of the 2D image results still tend to be flatter than their 3D model counterparts, but the high emphasis on the edges and notable details in the scene help the viewer perceive the finer differences. Since the process is highly dependent on the gradient field, additional processing to handle the natural downsides of the discrete gradient (floating-point accuracy, higher instability) could improve current results to a slight extent. In terms of implementation, computational matrix operations remain a slow process. To improve algorithms that utilize the Poisson Systems specifically, iterative matrix solving functions could be considered in more detail. Sparse systems already improve performance by a significant magnitude, but the speed is still highly dependent on the matrix structure and the type of solver used.

3D model reconstruction results are still of higher quality, due to the combination of more research on the topic and less undetermined factors. Many of the more advanced algorithms still require more computation time, which isn't necessarily bad, but for applications (particularly in manufacturing) some more optimization in the methods may be desired.

In both problem types, finding a good balance in compression between purveying depth and maintaining details remains a core component of generalizing the procedures. Most algorithms still maintain limitations on the details of the input type they accept. Particularly, many struggle based on the degree of details within the input. Another avenue that may be useful to consider in the field is to approach problems that contain more shading and salient details within them. A technique that specifically filters out different levels of details and then recombines them for the result might be of interest, if not already being researched.

One other area I have interest in is more research into separation between foreground and background layers. Most algorithms at this stage mainly use the gradient to make this decision. The 2D region based algorithm discussed in detail proposes the graph structure in determining relations, but also notes that the user is still required to handle some of the connections. In my implementation, I was considering more research into procedures of background subtraction that could provide another set of information to distinguish the image layers.

Relief-generation methods have made great progress within the last decade overall, and many of the current results are sufficient for modern applications, including the original purpose of art and storytelling. However, a fully automated process that is robust with most instances of a given input is still far from completion. Improvements can still be made in terms of required user influence, research into different materials and even combinations of materials, and, like most computational operations, speed. There remain many aspects in which continuing research in this topic can be beneficial.

5.2 Additional Results

Figure 5.1: More results from the 2D algorithm implemented for this thesis.

(a)-(b) Backpack (c)-(d) Grandfather Clock

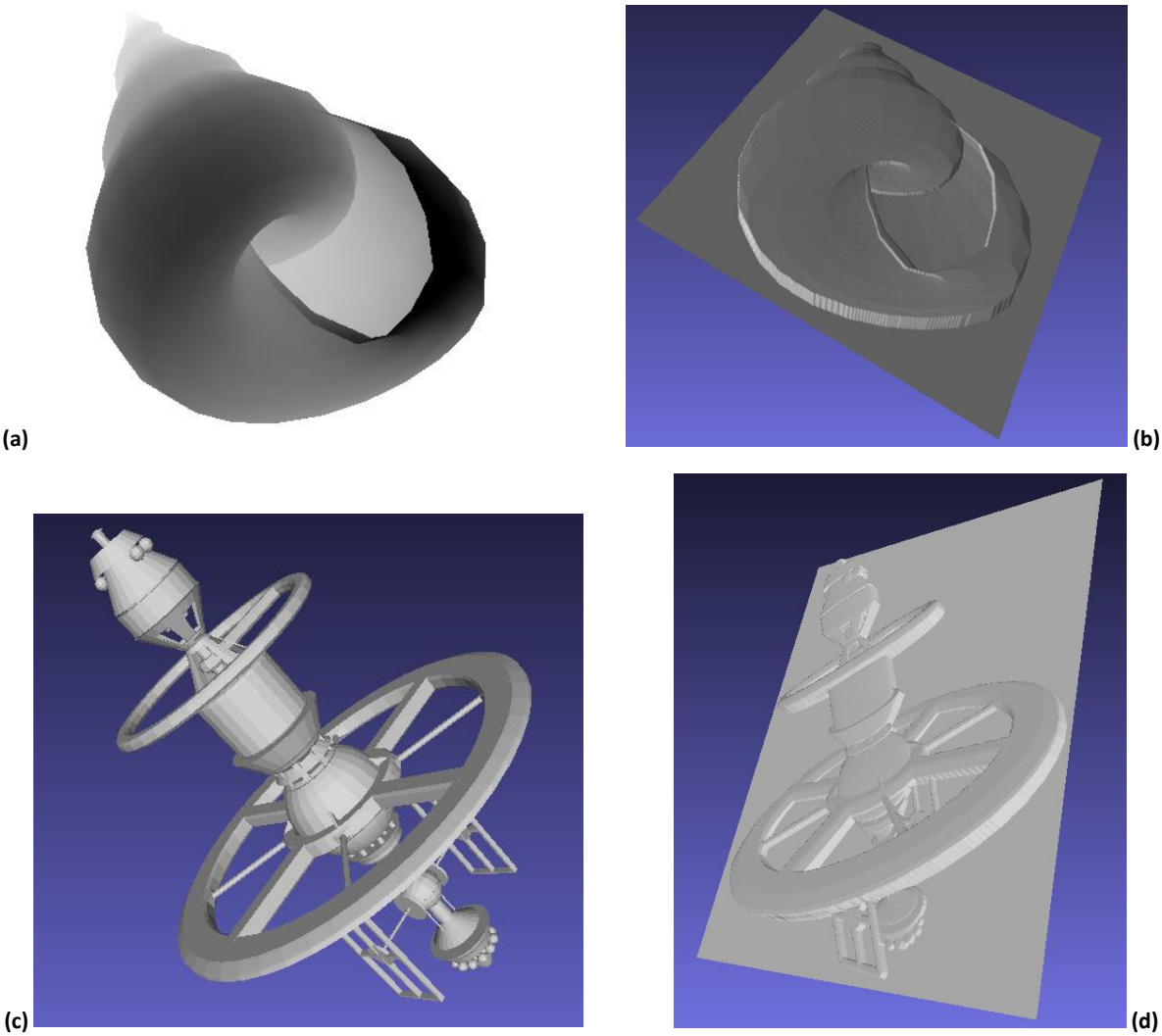


Figure 5.2: More results from the 3D algorithm described in the thesis.

(a)-(b) Shell (c)-(d) Space Station

Works Cited

Arpa, S. (2014). *Representations of 3D Scenes in a Limited Depth Range.*

Henry Kang, S. L. (2007, August 5). Coherent Line Drawing. St. Louis, Missouri, United States.

J. Kerber, M. W.-P. (2011). Computer Assisted Relief Generation - a Survey. *COMPUTER GRAPHICS*, 1-11.

J. Wu, R. M.-F.-K.-H. (2013, January). Making Bas-Reliefs from Photographs of Human Faces. Cardiff, UK. Retrieved from Elsevier.

Jens Kerber, A. T.-P. (2009). Feature Sensitive Bas Relief Generation. Saarbrücken, Germany.

Jens Kerber, A. T.-P. (2010). Real-time Generation of Digital Bas-Reliefs. Saarbrücken, Germany.

Q. Zeng et al. (2013, October 1). Region-based bas-relief generation from a single image. *Graph*, pp. 1-12.

Tim Weyrich, J. D. (2007). Digital Bas-Relief from 3D Scenes. *ACM SIGGRAPH 2007*, (pp. 1-7).

Wang, M. (2011). *3D Digital Relief Generation.*

Xianfang Sun, P. L. (2008). Bas-Relief Generation Using Adaptive Histogram Equalization. *IEEE TRANS. VISUALIZATION AND COMPUTER GRAPHICS*, pp. 1-12.

Zhongping Ji, W. M. (2007). Bas-Relief Modeling from Normal Images with Intuitive Styles. *Journal of LaTeX Class Files*, 1-12.