Indian Institute Of Technology Delhi

Major Project Thesis

Towards Rendering of Weathered Surfaces

Author:
Keshav Choudhary

Supervisor:
Prof. Subodh Kumar

A thesis submitted in partial fulfillment of the requirements for the degree of M. Tech in Computer Science and Engineering

in the

Department of Computer Science

July 2014

Certificate

This is to certify that the project report titled **Towards Rendering of Weathered Surfaces** being submitted by **Keshav Choudhary (2009CS50244)** to the Indian Institute of Technology, Delhi, is a result of bonafide work carried out by them under my supervision. The matter and results obtained in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Prof. Subodh Kumar
Dept. of Computer Science and Engineering
Indian Institute of Technology
New Delhi 110016

Abstract

Towards Rendering of Weathered Surfaces

One important sub-area of computer graphics is the generation of realistic images. A challenge in computer graphics' realism is creating the digital models of shape and materials that are required for such rendering. We attempt to achieve a photorealistic rendering of the weathering phenomena. Weathering can be approximated as the application of a certain stochastic process on any object. We propose a pipeline for rendering the results of the application of such a stochastic process on a mesh in the best possible manner. There are five major modules in the pipelines which include, Generation of diffuse map; Generation of normal map; Generation of a weathering degree map; Parameterization of the input mesh model and Rendering of the given input mesh model with the help of the maps previously generated.

Acknowledgements

I would like to thank Prof. Subodh Kumar for advising me enthusiastically throughout the course of the major project. I would also like to thank him for giving us this opportunity to work on this exciting project, and for encouraging me at all times. I would also like to thank Nisha Jain for providing us with all the input meshes and helping us with key insights throughout the course of the project. Finally, I would like to thank my partner Devashish Tyagi for helping me throughout with the project.

Keshav Choudhary (2009CS50244)
Indian Institute of Technology Delhi

Contents

List of Figures

Chapter 1

Introduction

One important area of research is the rendering of realistic images. Photo-realistic rendering techniques are capable of rendering images that predict the appearance of yet to be manufactured objects. A challenge in achieving realism is creating the digital models of shape and materials that are required for such rendering.

We attempt to achieve a photo-realistic rendering of the weathering phenomena. Weathering is the chemical decomposition and physical disintegration caused by weather exposition. It is often linked to geology but is also related to the kind of material. Presently, designers either compose multiple textures manually, or directly modify object geometries when realism is needed. Instead, we present an automated technique for the rendering of weathered surfaces. The reality in rendering is achieved with the help of color and shading variations learnt from some sample diffuse and normal maps of weathered surfaces. We use these learned variations to color and shade our input models according to its weathering character. Diffuse maps are obtained after the removal of lighting effects from images and contains the raw color channel of the object that the image captures. Normal maps helps in modifying shading. This modification helps to add a roughness to the rendering, rather than being completely flat.

A diffuse map of a real weathered surface would help in providing realistic colors to the surfaces we are trying to render. Similarly, a normal map of a weathered surface would help us in modifying the shading appropriately. We discuss various techniques to generate relevant diffuse and normals maps along with ways to use them in our rendering. In fact, Figure 1.1 presents a complete pipeline for rendering weathering surfaces. It consists of five modules, (a) module for parameterizing the input mesh to be rendered – normal mapping requires a mesh which has texture co-ordinates which makes parameterization a necessity, (b) module for generating diffuse maps – generating diffuse map given an input image of a corroded surfaces, (c) module for generating normal

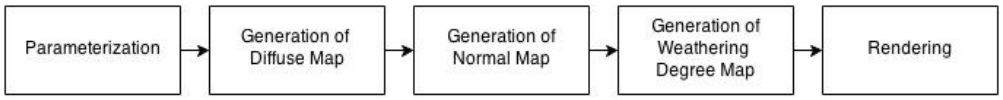


FIGURE 1.1: Rendering Pipeline

maps – generating normal map given an image of a corroded surface, (d) module for generating weathering degree maps – this modules aims at associating useful meaning to the previously generated diffuse map for the purpose of rendering,i.e., we have an input mesh model with weathering degree values associated with every vertex and a diffuse map of a real sample of a weathered surface. We need to have some way of assigning weathering degrees to each of the diffuse coefficients present in the diffuse map. Such an association is called a weathering degree map, (e) Rendering – uses all the maps generated in a meaningful way to produce a realistic rendering of a weathered surface.

The organization of the rest of the thesis is as follows, chapter (2) explains the various techniques tried for parameterizing the input mesh, chapters (3) and (4) explain how to obtain the best diffuse and weathering maps from images of corroded objects, chapter (5) explains the technique used to obtain the weathering degree map from diffuse map and chapter (6) presents the results obtained.

Chapter 2

Choosing a Parameterization Technique

2.1 Introduction

A parameterization of a surface can be viewed as a one-to-one mapping from a suitable domain to the surface. In general, the parameter domain itself will be a surface and so constructing a parameterization means mapping one surface into another. Typically, surfaces that are homeomorphic to a disk are mapped into the plane. Usually the surfaces are either represented by or approximated by triangular meshes and the mappings are piecewise linear.

Parameterizations have many applications in various fields of science and engineering, including scattered data fitting, reparameterization of spline surfaces, and repair of CAD models. But the main driving force in the development of the first parameterization methods was the application to texture mapping which is used in computer graphics to enhance the visual quality of polygonal models. Later, due to the quickly developing 3D scanning technology and the resulting demand for efficient compression methods of increasingly complex triangulations, other applications such as surface approximation and remeshing have influenced further developments.

Parameterization plays an important role in achieving a photorealistic rendering. Normals mapping is essential to simulate the impression of a detailed 3D surface, by modifying the shading as if the surface had lots of small angles, rather than being completely flat and normal mapping requires that tangent plane basis vectors be calculated for each vertex in a mesh, which in turn requires texture coordinates. Moreover, in the future if we think of synthesizing a 3D texture, texture coordinates would play a role in it too.

So, it was important for us to obtain a good parameterization for all kinds of possible input meshes but parameterizations almost always introduce distortions in either angles or areas and a good mapping in applications is one which minimizes these distortions in some sense. Many different ways of achieving this have been proposed in the literature, so we tried out quite a few of them to figure out which one of those best served our purpose. Following are the different methods that we tried.

2.2 Techniques for Parameterizing a Mesh

2.2.1 IsoParameterization

MeshLab provides a parameterization algorithm based on abstract parametrization which was proposed by Pietroni et al. [1]. The main idea proposed by them is rather simple. Usually textures are defined in a dominion that is just the $(0,0) - (1,1)$ square on the plane but in their approach as a domain of the parametrization they use a different 2-dimensional domain, the surface of a very coarse simplicial complex that has the same topology of the original mesh and it is composed by just a few hundred triangles. Such an approach is interesting because this abstract parametrization can be used for a number of things, like for example remeshing, texturing, tangent space smoothing etc.

Meshlab provides us with a filter "Isoparamterization", which can be used to isoparameterize a simplicial complex that has the same topology of the original mesh. The created isoparametrization then can be used to build a standard parametrization over any mesh that is reasonably close to the original one.

The technique did not work out for us as it requires water tight and single component meshes. But the input meshes which we wanted to render were not water tight. So we moved on to other techniques available for paramterization.

2.2.2 Sintef's Go Tools

Sintef's GoTools is the name of a collection of C++ geometry library which include a library for parameterization. The purpose of the GoTools Parametrization library is to compute a reasonable parametrization for a discrete geometrical object. This geometrical object can be a planar graph embedded in R^2 orR^3 (triangulation, rectangular grid, etc.) or a point cloud without any explicit connectivity, but with an identified border. There are some standard algorithms which the GoTools' library implements :-

- Floater's Shape Preserving Parameterization [2]

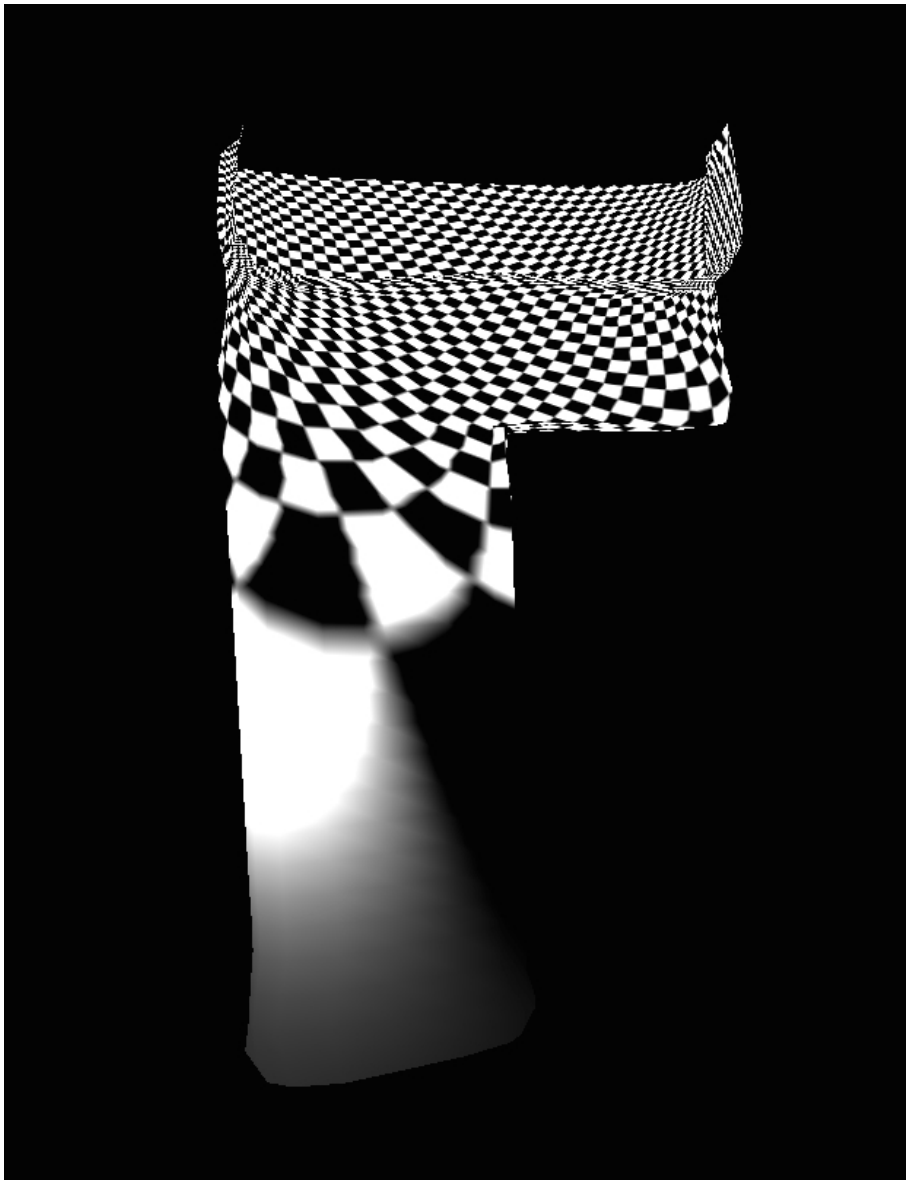


FIGURE 2.1: Mean Value Paramterization - GoTools

- Mean Value Parameterization [3]

We tried parameterizing a sample mesh using both the above mentioned techniques but the results were not satisfactory. Result on a sample input mesh can be seen in figure 2.1. As is eveident from the result there is a lot of stretching in parameterization, thus we looked into techniques which minimizes this stretching.

2.2.3 CGAL

CGAL (Computational Geometry Algorithms Library) is another C++ library to provide easy access to efficient and reliable geometric algorithms. It has a module "Planar Parameterization of Triangulated Surface Meshes", which implements some of the state-of-the-art surface parameterization methods, such as discrete conformal map [4], discrete authalic parameterization [5] and Tutte barycentric mapping [6]. These methods are mainly distinguished by the distortion they minimize (angles vs. areas), by the constrained border onto the planar domain (convex polygon vs. free border) and by the guarantees provided in terms of bijective mapping. Result of using discrete authalic parameterization on a sample input mesh in shown in figure 2.2. A similar problem of stretching was observed in all other techniques tried using this library.

2.2.4 Stretch Minimizing Mesh Parameterization

CGAL implements all the standard state of the art mesh parameterization techniques but there was the problem of stretching in all of them. Yoshizawa et al. [7]propose a way of stretch-minimizing mesh parameterization. Given a triangle mesh, they start from the Floater shape preserving parameterization and then improve the parameterization gradually. At each improvement step, they optimize the parameterization generated at the previous step by minimizing a weighted quadratic energy where the weights are chosen in order to minimize the parameterization stretch.

If we take a surface $S \in \mathbb{R}^3$ which is parametrically given by $r(s,t) = [x(s,t)y(s,t)z(s,t)]$, the Jacobian Matrix corresponding to the mapping r given by $J = [\frac{\partial r}{\partial s}, \frac{\partial r}{\partial t}]$ gives us all the properties of the parameterization such as angle and length distortions. Further Sander et al. [8] derive a texture stretch metric based on the maximum (Γ) and minimum (γ) singular values of the Jacobian matrix. They define the two stretch norms over a triangle T as $L^2(T) = \sqrt{((\Gamma^2 + \gamma^2)/2)}$. Yoshizawa et al. [7] use this metric to define the stretch for each vertex u_i in the parameter domain as

$$\sigma_i = \sqrt{\sum A(T_j)\sigma(U_j)^2 / \sum A(T_j)}$$

where $A(T_j)$ denotes the area of the triangle T the sums are taken over all triangles T_j surrounding mesh vertex p_i corresponding to u_i. The boundary vertices of mesh M are mapped into the boundary vertices of U which form a polygon in the parameter plane $\mathbb{R}^2_{s,t}$ and for each inner vertex p_i of M its corresponding vertex u_i inside the polygon is

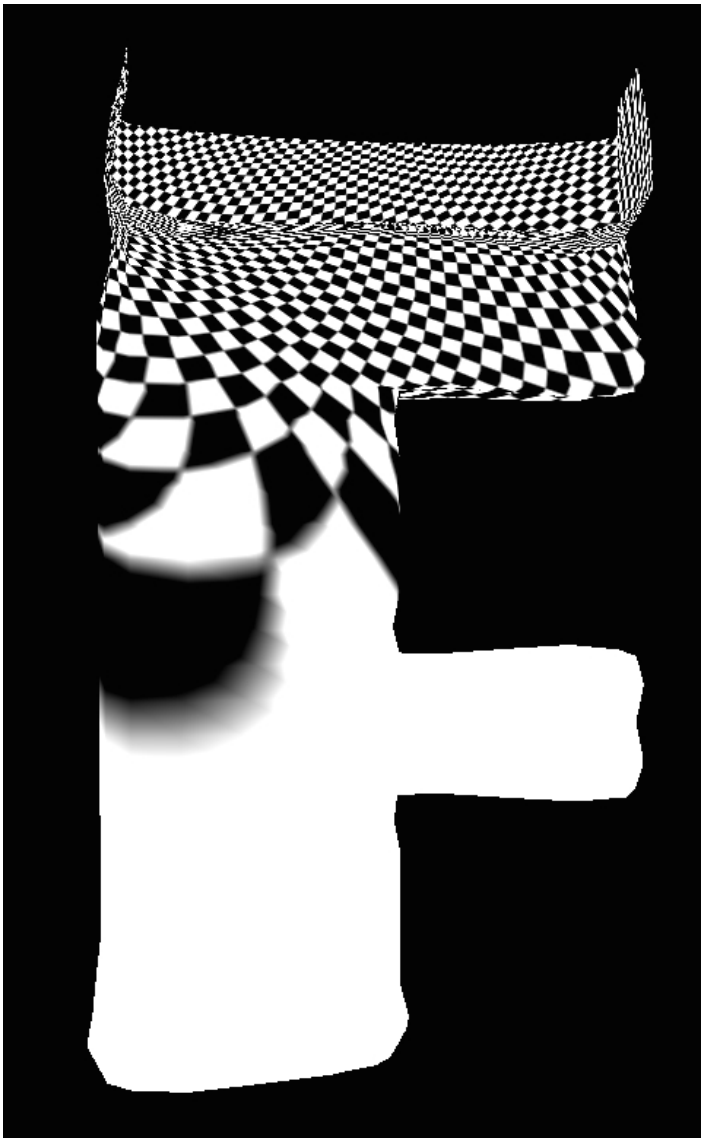


FIGURE 2.2: Paramterization using CGAL library

selected such that the following local quadratic energy

$$E(u_i) = \sum_j w_{ij}(u_j - u_i)^2$$

achieves its minimal value. Here $\{u_j\}$ are vertices corresponding to the mesh one-link neighbours of $p_i \in M$ and $\{w_ij\}$ are positive weights. The local stretches are then redistributed using

$$w_{ij}^{new} = w_{ij}^{old}/\sigma_j$$

Initial weights and texture coordinates are obtained using Floater's shape preserving

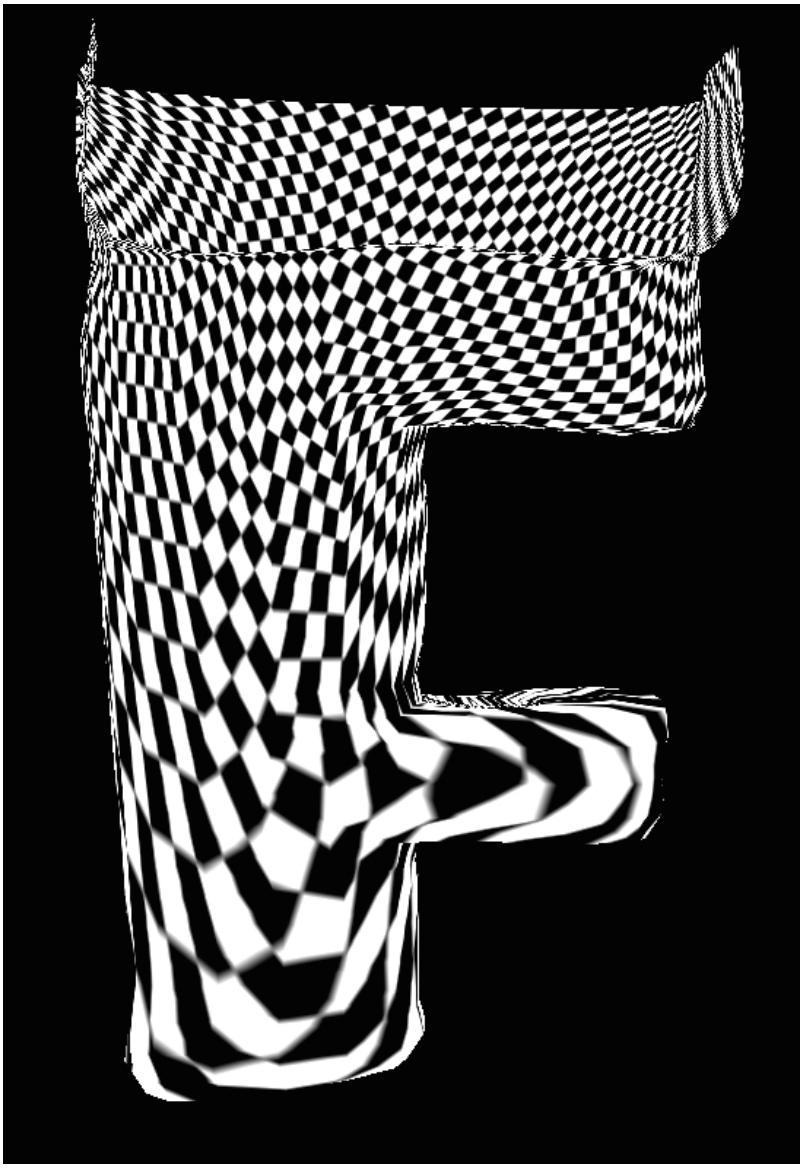


FIGURE 2.3: Paramterization after Application of Stretch Minimization

parameterization [2]. We present the results of the technique tested on a sample input mesh in Figure 2.3. As we can see the results are much better then all the approaches previously tried. The stretch has been minimized to a great extent but the mesh parameterization procedure generates regions consisting of slim triangles, which could be due to uneven distribution of the stretch. Also such thin and long triangles are more evident near the mesh boundary, where large area and angle distortions play a role. One can hope that an appropriate relaxation of boundary conditions will reduce those area and angle distortions while maintaining low stretch.

2.2.5 Parameterization using Texture Atlas Generation

Levy et al. [9] propose a new optimization based parameterization technique called Least Square Conformal Maps which used Texture Atlas. The model to be textured is partitioned into a set of parts homeomorphic to discs, referred to as charts, and each of them is provided with a parameterization. The generation of a texture atlas can be decomposed into the following steps:

- Segmentation: The model is partitioned into a set of charts. The authors introduce a way in which charts boundaries are positioned in such a way that most of the discontinuities between the charts are located in zones where they do not cause texture artifacts and charts are homeomorphic to discs, such that it is possible to parameterize them without introducing much deformation.
- Parameterization: Each chart is 'unfolded', i.e. put in correspondence with a subset of R^2. The criteria introduced minimizes angle deformations and non-uniform scalings.
- Packing: The charts are gathered in texture space. The authors propose an algorithm that packs the charts directly rather than their bounding rectangles which previously was the generally followed practise.

This texture atlas generation technique could be used to parameterize the input model, the issue being the chart boundaries. So, at the seams across chart boundaries, we could mark thin strips on either sides. Within these strips, we could alter the texture co-ordinates such that they fit some continuous function. In this way, we would not have evident chart boundaries.

This parameterization technique has also been implemented in CGAL. We tested it on a sample input mesh. The results can be seen in figure 2.4. We have not tested the idea discussed above to do away with the chart boundaries.

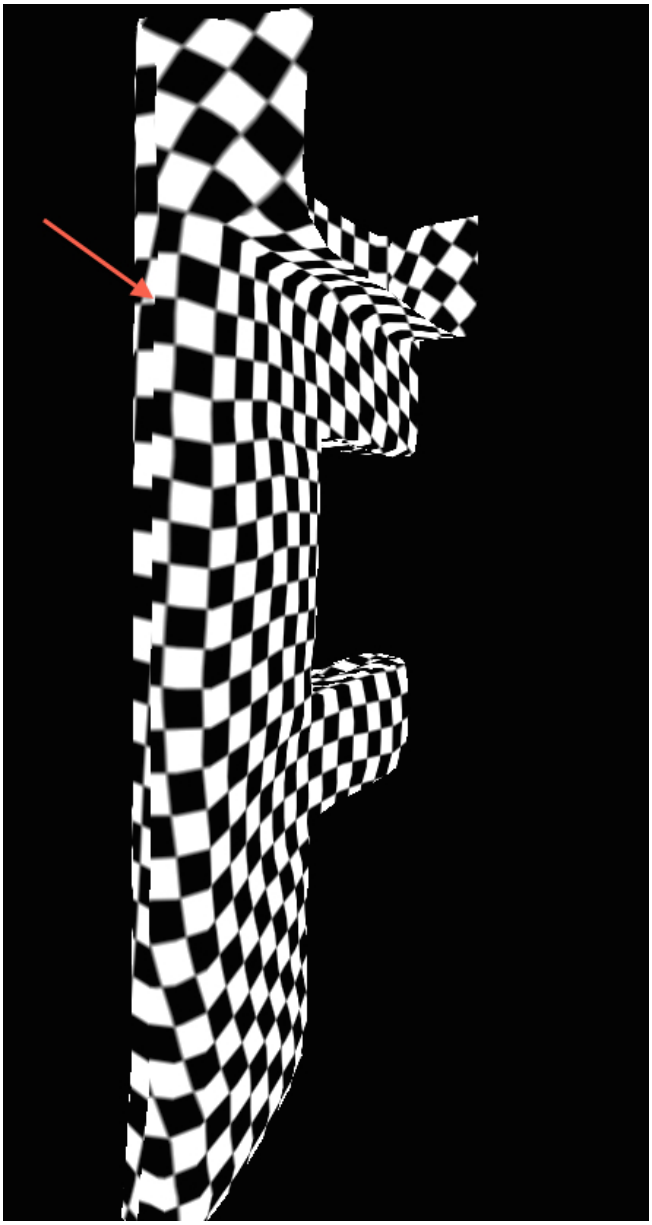


FIGURE 2.4: Paramterization using Texture Atlas. A chart boundary is marked.

Chapter 3

Diffuse Map

3.1 Introduction

We present a data driven pipeline , which given an image of a corroded surface produces Diffuse and Normal Maps to help us enhance the quality of rendering. This chapter discusses the various approaches that we tried to obtain good quality diffuse maps from images. Diffuse maps are obtained after the removal of lighting effects from images and contains the raw color channel of the object that the image captures. A diffuse map of a real weathered surface would help in providing realistic colors to the surfaces we are trying to render.

3.2 Approaches tested to obtain Diffuse Maps

3.2.1 Shape, Illumination, and Reflectance from Shading

Barron et al. [10] address the problem of recovering shape, albedo, and illumination from a single image of an object. Recovering these properties from a single image is a fundamentally under-constrained problem as there are an infinite number of shapes, paint, and lights that exactly reproduce a single image. The authors therefore pose this as an optimization problem that searches for the most likely explanation of the given image. They construct priors similar to those used in natural image statistics and solve the optimization under these constructed priors.

The implementation was publicly available. It was tested on a variety of images. The results were promising in the case of some sample images but it did not perform as well

(A) Original Image

(B) Diffuse Image

FIGURE 3.1: Results obtained using SRIFS

as expected on images of corroded surfaces. Results on a sample input image can be seen in figure 3.1.

3.2.2 Crazy Bump

Crazybump is a publicly available tool for the creation of normal, diffuse and height maps from a given texture. The results for normal maps were somewhat satisfactory but for diffuse maps only the highlights were removed as can be observed in figure 3.2. Moreover,

(A) Original Image

(B) Diffuse Image

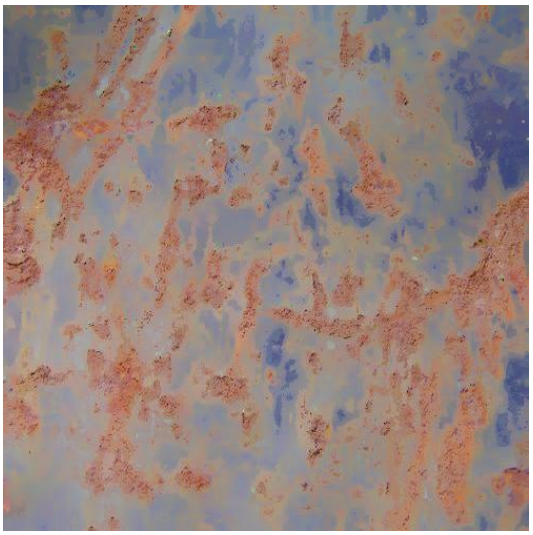
FIGURE 3.2: Results obtained using Crazy Bump

no documentation of the technique used for creation of these maps was available, so we decided not to use it for our purpose.

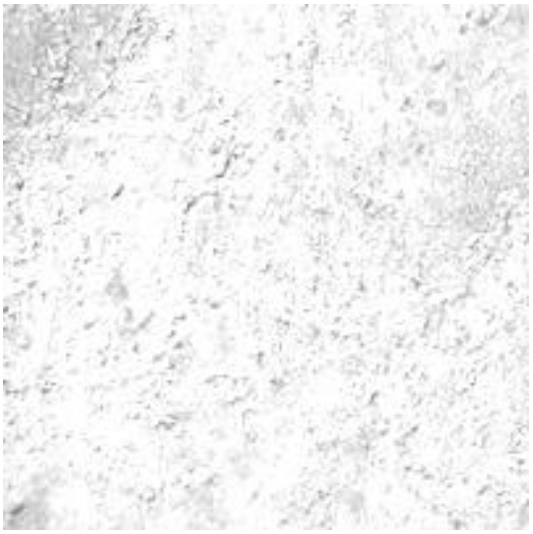
3.2.3 AppGen: Interactive Material Modeling from a Single Image

Dong et al. [11] present a method to model the detailed spatially-varying reflectance properties (diffuse, specular and roughness) and surface normal variations with minimal user interaction, given a texture image of a nearly planar surface lit with directional lighting. They propose a two step system, (1) Removing highlight and shadow pixels

(A) Original Image



(B) Diffuse Image



(C) Shading Image

FIGURE 3.3: Results obtained using AppGen

in the input image and fill them by image inpainting. This leaves an image only of diffuse contribution, (2) An algorithm to separate the the diffuse component I_d into a shading map S_d and a dissue albedo map ρ_d. For this purpose, the image value $I_d = (I_d^r, I_d^g, I_d^b)$ is represented by its intensity $I_d^i = (I_d^r + I_d^g + I_d^b)/3$ and chroma value $I_d^c = (I_d^r/I_d^i, I_d^g/I_d^i, 3 - I_d^r/I_d^i - I_d^g/I_d^i)$. The light is assumed to be white $I_l = (1.0, 1.0, 1.0)$, so that the image chroma comes from the chroma of the diffuse albedo $\rho_d^c(x) = I_d^c(x)$, while the image intensity is the product of shading and albedo brightness $I_d^i(x) = \rho_d^i(x) S_d^i(x)$. Now the goal remains to seperate the diffuse intensity I_d^i into an albedo intensity map ρ_d^i and a shading intensity map S_d^i. To solve this, the authors formulate the separation as an optimization problem and solve the initial albedo map and shading map by an EM (Expectation- Maximization) like algorithm.

They assume that in each local region Ω, pixels with the same chroma value $\rho_d^c(x) = I_d^c(x) = c$, belong to the same material and have the same albedo intensity i_c. Based on this local albedo assumption, we have

$$\rho_d^i(x) = i_c \forall x \in \Omega_c,$$

where Ω_c refers to the set of pixels that are in Ω and have the same chroma value c. The authors also assume that the in Ω groups of pixels with different chroma values share the same global geometry and thus have the same average shading. Based on this local shading assumption, we have

$$E(S_d^i(x)|x \in \Omega_c) = E(S_d^i(x')|x' \in \Omega).$$

The intensity estimation of pixels in Σ_c can be computed as

$$\begin{aligned} E(I_d^i(x)|x \in \Omega_c) &= \frac{i_c \sum_{x \in \Omega_c} S_d^i(x)}{N_{\Omega_c}} \\ &= i_c E(S_d^i(x')|x' \in \Omega), \end{aligned}$$

where N_{Ω_c} is the number of pixels in Ω_c. And so the albedo intensity $\rho_d^i(x)$ of a pixel x in region Ω_c should satisfy

$$E_0(\Omega, c, I_d^i, \rho_d^i) = \frac{E(I_d^i(x')|x' \in \Omega_c)}{E(S_d^i(x')|x' \in \Omega)} = \frac{\frac{1}{N_{\Omega_c}} \sum_{x' \in \Omega_c} I_d^i(x')}{\frac{1}{N_\Omega} \sum_{x' \in \Omega} \frac{I_d^i(x')}{\rho_d^i(x')}}$$

Based on this local constraint, they formulate the separation as an optimization problem by minimizing the following energy function of $\rho_d^i(x)$:

$$F_0(\rho_d^i(x)) = \sum_{\Omega \in \Sigma} ||\rho_i^d(x) - E_0(\Omega, c, I_d^i, \rho_d^i)||^2,$$

where Σ is the collection of all fixed local regions that contain x. In our implementation we defined Ω to be a $W * W$ window and set the region size to $W = 20$. The optimization is then solved using an iterative algorithm similar to Expectation Minimization. The authors also propose an extension in which users can interactively mark regions that violate the assumptions using a few rough strokes and then they augment the optimization procedure with some constraints to further refine the separation results.

The proposed technique was implemented in Matlab (except the part with user interaction) and the results looked very promising. The algorithm was able to deliver a flat diffuse map for a variety of samples tested, so we decided to use these technique for our purposes. Results could be seen in figure 3.3.

3.2.4 Estimating the diffuse map using normal map

We also gave thought to the idea of using normals maps for the purpose of deriving diffuse maps. We concluded that a high resolution scan with registered image of a corroded object could proof to be useful. The steps involved would be

- Take a high resolution scan of a corroded object and estimate the height map from the scan.
- Use the height map to obtain the normal map.
- Take a high resolution image of the corroded object under directional lighting, preferably under LED lights.
- Estimate the light direction.
- Use the phong lighting equation

$$I_d(x) = \rho_d(x) S_d(x) = \rho_d(x)(N(x).L) I_l$$

  to estimate ρ_d, the diffuse component, assuming $I_l = (1.0, 1.0, 1.0)$.

This technique could not be tested as we could not obtain a scan of the object but this could be tried out in the future.

3.2.5 Time Varying Surface Appearance

Jinwei et al. [12] mention that they have acquired the time-varying database of 26 samples, encompassing a variety of natural processes including burning, drying, decay

and corrosion. They propose a Space-Time Appearance Factorization (STAF) model, which factors space and time-varying effects and includes an overall temporal appearance variation of diffuse characteristics. We could use their database to learn the variation of diffuse coefficients over time, align the time axis with the corrosion degree axis and learn the diffuse coefficients for different corrosion values.

Chapter 4

Normal Map

Normal maps help to simulate the impression of a detailed 3D surface, by modifying the shading as if the surface had lots of small angles, rather than being completely flat. Hence, a good quality normal map helps to achieve a more realistic rendering. It also helps us to see the rendered mesh model in different lighting conditions and provide feedback to the process generating the model. There are quite a few techniques for obtaining normal maps and they are explained in detail in [13]. We focus on particular way known as **Shape from Shading**.

Shape from Shading deals with the basic problem of recovering 3D information of the captured object in a given input gray-scale image. In Literature, shape from shading algorithms can be categorized according to the method used to solve the image irradiance equation $I = \rho \mathbf{N}.\mathbf{L}$. There are approaches that try to recover the surface by minimizing the error between the two sides of the equation, i.e. minimization approaches. A second class of approaches, called propagation approaches, starts from a set of points where the solution is known and propagates the shape information to the entire image. The last class includes approaches that reduce the complexity of the problem by either converting the nonlinear image irradiance equation into a linear one (linear approaches) or by approximating the surface locally by simple shapes (local shading analysis approaches). Shah et al. [14] propose a way to reduce the non-linear problem into a linear through the linearization of the image irradiance equation. The idea is based on the assumption that the lower order components in the reflectance map dominate.

The reflectance function for Lambertain surfaces is modelled as follows:

$$\begin{aligned} E(x,y) &= R(p,q) \\ &= \frac{1+pp_s+qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+p_s^2+q_s^2}} \\ &= \frac{cos\sigma + pcos\tau sin\sigma + qsin\tau sin\sigma}{\sqrt{1+p^2+q^2}} \end{aligned} \tag{4.1}$$

where $E(x,y)$ is gray level at pixel (x,y), $p = \frac{\partial Z}{\partial x}$,$q = \frac{\partial Z}{\partial y}$,$p_s = \frac{cos\tau sin\sigma}{cos\sigma}$,$q_s = \frac{sin\tau sin\sigma}{cos\sigma}$, τ is the tilt of the illuminant and σ is the slant of illumination. Using the following discrete approximation for p and q

$$p = \frac{\partial Z}{\partial x} = Z(x,y) - Z(x-1,y) q = \frac{\partial Z}{\partial y} = Z(x,y) - Z(x,y-1),$$

the above reflectance equation can be rewritten as:

$$\begin{aligned} 0 &= f(E(x,y), Z(x,y), Z(x-1,y), Z(x,y-1)) \\ &= E(x,y) - R(Z(x,y) - Z(x-1,y), Z(x,y) - Z(x,y-1)) \end{aligned} \tag{4.2}$$

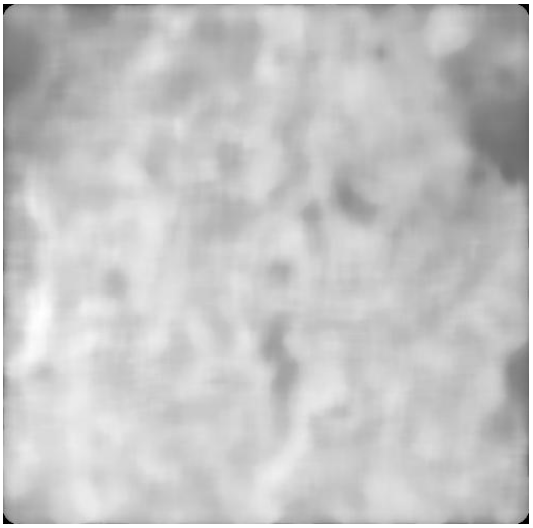
Using linear approximation (Taylor expansion up through the first order terms), we can form N^2 such equations for a $N * N$ image which will form a linear system, which can be solved using the Jacobi iterative method. We implemented the idea in Matlab. The height map obtained on a sample input image can be seen in figure 4.1. It is evident that the global details are being captured by the approach but the finer details are missing and so a normal map obtained using this approach would not be useful.

Wu et al [15] present a new interactive Shape from Shading algorithm that recovers high-frequency details from a single image under the assumptions that the image is of an object with nearly homogeneous albedo. They use the lambertian lighting model

$$I = \rho N.L$$

where I is the input image, ρ is the surface albedo, N is a unit vector representing the surface normal and L is a unit vector representing the direction of a distant light source. They first try to compute the light source direction and then form an optimization problem to best fit the albedo and normal vectors according to the lambertian lighting model. The approach looks promising and could be tried out to obtain some sample normal maps.

(A) Original Image



(B) Height Image

FIGURE 4.1: Height Map obtained using Shah's Linear Approximation

Chapter 5

Weathering Degree Map

The rendering pipeline introduced shows a module for generation of diffuse & normal maps and another module for rendering the given mesh input using these maps. There is another module in between them which implements a way of associating useful meaning to these maps for the purpose of rendering i.e. we have an input mesh model with weathering degree values associated with every vertex and a diffuse map of a real sample of a weathered surface. We need to have some way of assigning weathering degrees to each of the diffuse coefficients present in the diffuse map. Such an association is called a weathering degree map.

A possible way to achieve the above said is presented by Wang et al. [16]. From spatially variant BRDF (diffuse coefficient in our case) data captured from a weathered material sample at a single instant in time, the appearance variations on the sample are analysed for the synthesis of weathering degrees.

The authors describe the construction of an **Appearance Manifold**, which is the plot of the captured BRDF data of each surface point in an appearance space defined by reflectance features. They construct a neighbourhood graph among these sample points. Further, they describe a method to infer,from user annotations and the relative positions of points on the manifold, their relative degrees of weathering.

In our implementation, we construct the Appearance Manifold of a given sample diffuse image $I(x, y)$ by organizing the 3-dimensional diffuse coefficients of each pixel (x,y) into a vector and plotting as a point in a corresponding 3-dimensional appearance space. For constructing edges between these points, the k-rule is used, whereby each point is connected to its k nearest neighbours. Then certain out-liers are pruned using the ϵ-rule, which allows connections only between points separated by a distance less than a threshold ϵ. The threshold is set as $\epsilon = 1.5d_p$ where d_p is the average pairwise distance

between connected points after applying the k-rule. Next, the user is asked to mark the initial set of most and least corroded points namely X_1' and X_0'. The set are then expanded to include more points using the rule

$$X_1 = \{x_i | \phi(x_i, x_0') > \lambda \Phi(X_0', X_1')\}$$

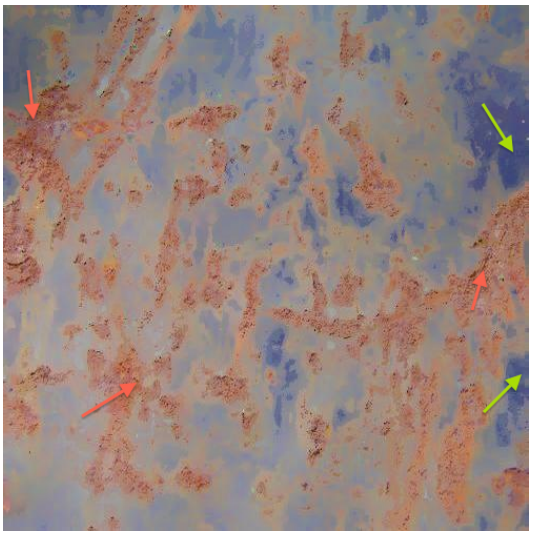
, where $\phi(x_i, x_j)$ between points x_i and x_j is defined as the shortest path along the connected nodes between x_i and x_j, $\Phi(X_0', X_1')$ is the minimum distance between sets X_0' and X_1', x_0' denotes the closest point in X_0' to x_i, and λ is a parameter set to 0.95 in our implementation. Set X_0 is defined similarly.

The weathering degree of a point x is computed using

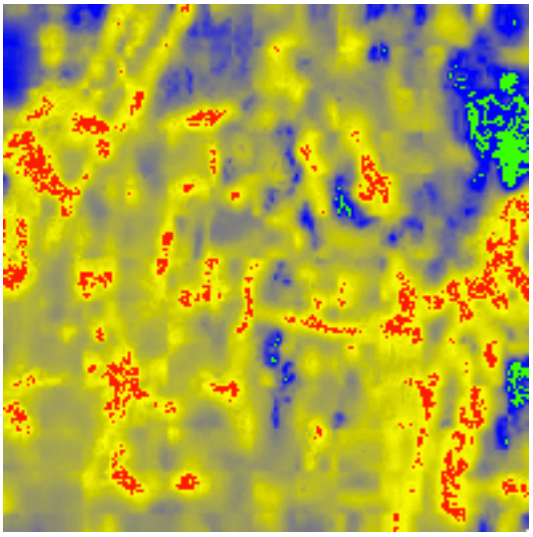
$$\varphi(x) = \frac{\phi(x, x_0')}{\phi(x, x_0') + \phi(x, x_1')}$$

where x_0' and x_1' are respectively the closest points in X_0 and X_1 to x. These weathering degree map is then used in rendering to provide diffuse color to the vertices using certain heuristics described later.

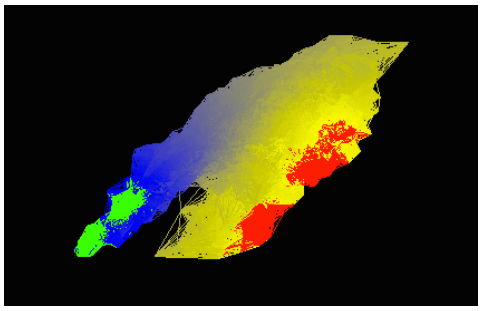
In figure 5.1, we can see that the resulting weathering degree map. The most corroded vertices have been represented using red colour, the least corroded ones using green colour, and the rest of them have been represented using the colour scheme $(r, g, b) =$ (weathering degree,weathering degree, 1-weathering degree).



(A) Diffuse Image. Red arrows indicate marked as most corroded & green arrows indicate marked as most corroded

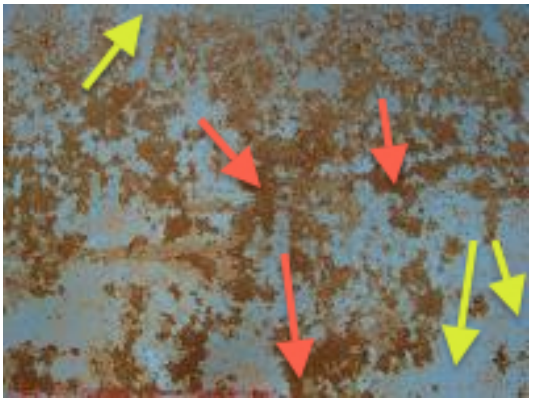


(B) Weathering Degree Map

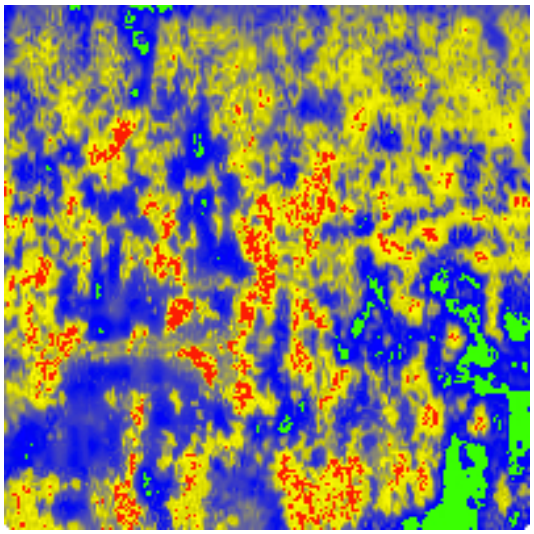


(C) Appearance Manifold

FIGURE 5.1: Weathering Degree Map and Appearance Manifold.



(A) Diffuse Image. Red arrows indicate marked as most corroded & green arrows indicate marked as most corroded



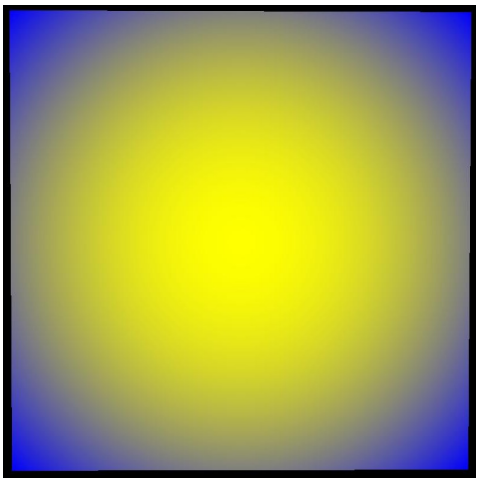
(B) Weathering Degree Map

FIGURE 5.2: Weathering Degree Map for a sample diffuse map

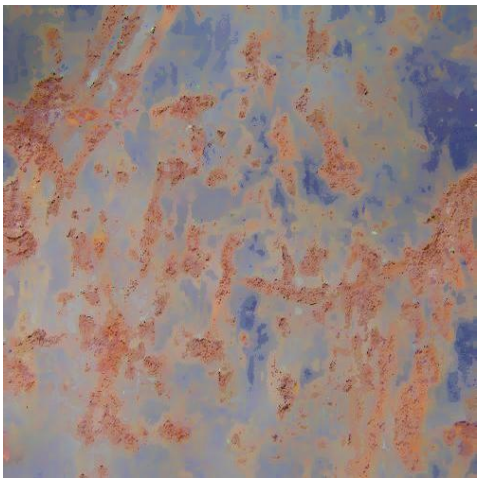
Chapter 6

Rendered Results

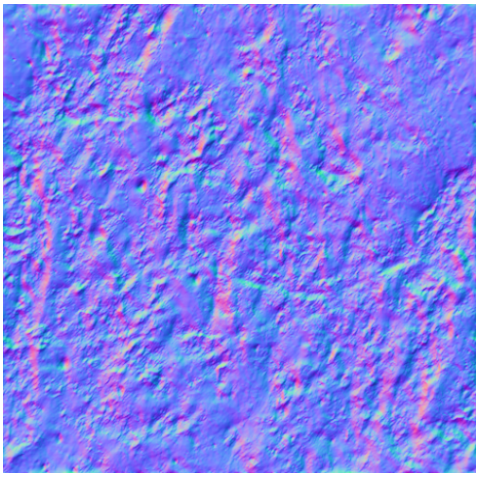
The final module in the rendering pipeline is the renderer. It takes in a mesh model and all the maps created in the previous modules to create a rendering of the mesh with color and normal mapping. We have used Tessellation shaders to help us improve our rendering. The details of the various rendering algorithms tested can be found in detail in [13]. I present the results 6.1 on a sample input mesh. It is a simple plate with weathering degrees gradually increasing as we move radially outward.



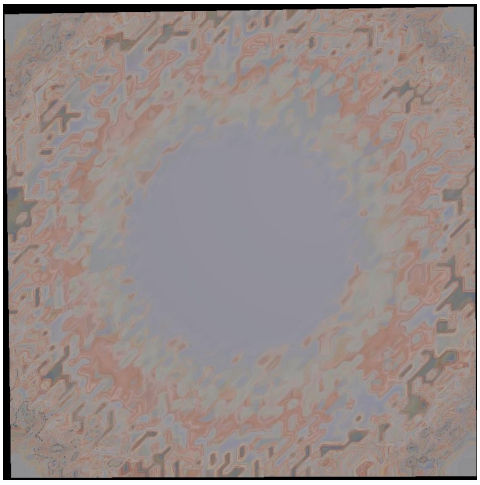
(A) Variation of Corrosion Values provided as input



(B) Diffuse Map used



(C) Normal Map used



(D) Result

FIGURE 6.1: Results obtained on a sample

Chapter 7

Conclusion

We introduce a data driven pipeline for rendering of weathered surfaces. Given an input mesh with weathering degrees associated with each vertex , the pipeline uses images of corroded objects to determine what diffuse coefficients and normals should be used for rendering the input mesh.

The results obtained indicate that there is scope for further improvement in generation of normal maps. There are few other techniques which could be tried to improve the quality of normal maps. High resolutions scans of corroded surfaces can provide us with a good height map. The height map can be used to generate detailed normal maps. Moreover, the approach suggested by Wu et al [15] also looks promising and could be tried out to obtain some sample normal maps.

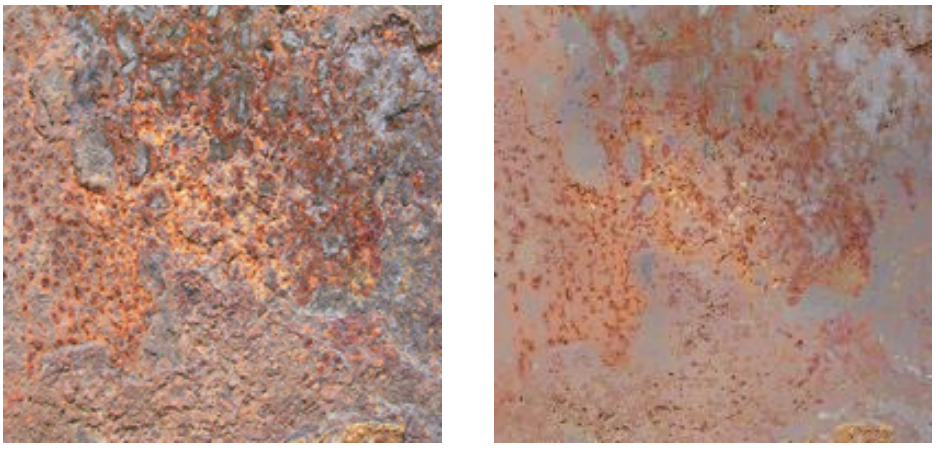
Jinwei et al.[12] work, as discussed earlier could also be used as an alternative technique for the generation of diffuse maps. The results could then be compared with Appgen and the better of the two could be used for the final rendering purposes.

Further, some alternative parameterization techniques that could solve the problem of texture stretching could be looked at. A better paramterization would help to improve the quality of normal mapping.

Appendix A

Additional Results – Diffuse Maps

We present some additional diffuse maps that were generated using AppGen [11] technique.



(A) Original Image (B) Diffuse Image

FIGURE A.1: Diffuse Maps generated for a sample input image

(A) Original Image

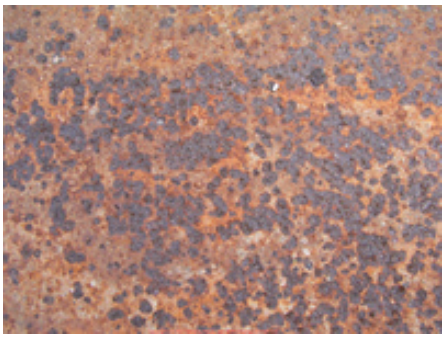
(B) Diffuse Image

FIGURE A.2: Diffuse Maps generated for a sample input image

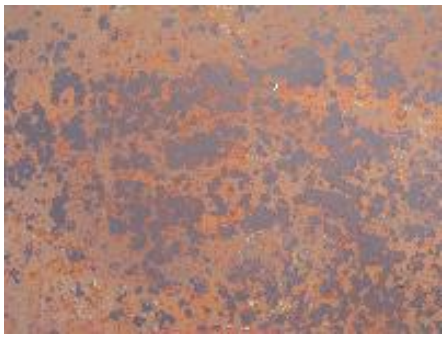
(A) Original Image

(B) Diffuse Image

FIGURE A.3: Diffuse Maps generated for a sample input image



(A) Original Image



(B) Diffuse Image

FIGURE A.4: Diffuse Maps generated for a sample input image

(A) Original Image

(B) Diffuse Image

FIGURE A.5: Diffuse Maps generated for a sample input image

(A) Original Image

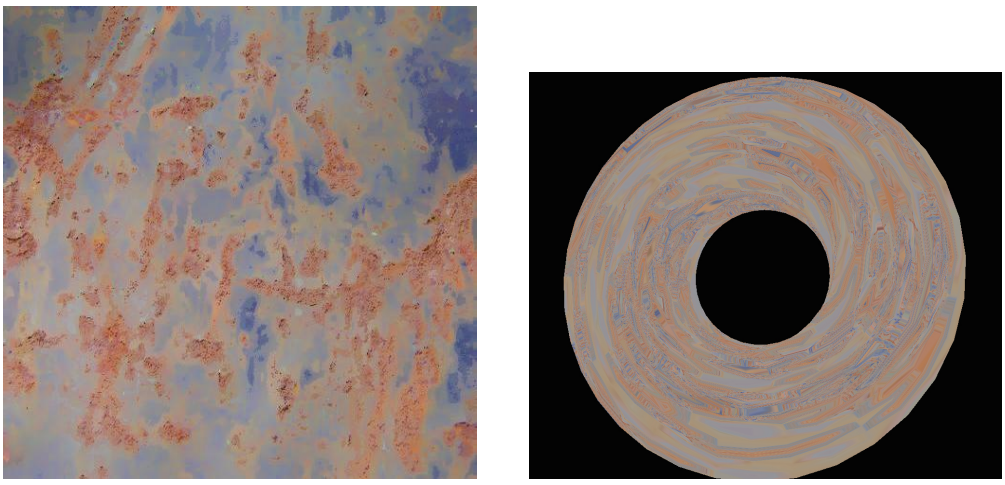
(B) Diffuse Image

FIGURE A.6: Diffuse Maps generated for a sample input image

Appendix B

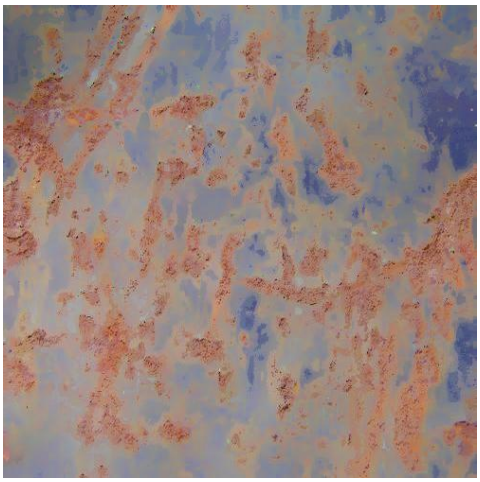
Additional Results – Final Rendering

We present some additional results for final rendering obtained.

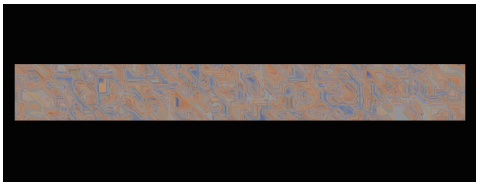


(A) Diffuse Map used (B) Result

FIGURE B.1: Results obtained on a sample cylinder input



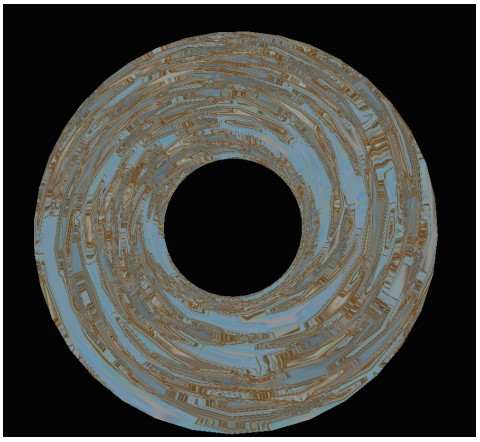
(A) Diffuse Map used



(B) Result

FIGURE B.2: Results obtained on a sample rod input

(A) Diffuse Map used



(B) Result

FIGURE B.3: Results obtained on a sample cylinder input

(A) Diffuse Map used

(B) Result

FIGURE B.4: Results obtained on a sample rod input

Bibliography

[1] Nico Pietroni, Marco Tarini, and Paolo Cignoni. Almost isometric mesh parameterization through abstract domains. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):621–635, 2010. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.96.

[2] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[3] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[4] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH*, pages 173–182, 1995. URL http://dblp.uni-trier.de/db/conf/siggraph/siggraph1995.html#EckDDHLS95.

[5] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. *Comput. Graph. Forum*, 21(3):209–218, 2002. URL http://dblp.uni-trier.de/db/journals/cgf/cgf21.html#DesbrunMA02.

[6] W. T. Tutte. How to draw a graph. *Proc Lond Math Soc*, 13:743–767, 1963. URL http://www.ams.org/mathscinet-getitem?mr=28:1610.

[7] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. A fast and simple stretch-minimizing mesh parameterization. In *SMI*, page 390, 2004.

[8] Pedro V. Sander, John Snyder, and Steven J. Gortler. Texture mapping progressive meshes. pages 409–416. ACM Press, 2001.

[9] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3): 362–371, 2002. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/566654.566590. URL http://portal.acm.org/citation.cfm?id=566590.

[10] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. Technical Report UCB/EECS-2013-117, EECS, UC Berkeley, May 2013.

[11] Yue Dong, Xin Tong, Fabio Pellacini, and Baining Guo. Appgen: interactive material modeling from a single image. *ACM Trans. Graph.*, 30(6):146:1–146:10, December 2011. ISSN 0730-0301. doi: 10.1145/2070781.2024180. URL http://doi.acm.org/10.1145/2070781.2024180.

[12] Jinwei Gu, Chien I. Tu, Ravi Ramamoorthi, Peter Belhumeur, Wojciech Matusik, and Shree Nayar. Time-varying surface appearance: acquisition, modeling and rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 762–771, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1179352.1141952. URL http://dx.doi.org/10.1145/1179352.1141952.

[13] Devashish Tyagi. Rendering of weathered surfaces. Master's thesis, IIT Delhi, 2014.

[14] Ping-Sing Tsai and Mubarak Shah. Shape from shading using linear approximation. *Image Vision Comput.*, 12(8):487–498, 1994. URL http://dblp.uni-trier.de/db/journals/ivc/ivc12.html#TsaiS94.

[15] Tai-Pang Wu, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Interactive normal reconstruction from a single image. *ACM Trans. Graph.*, 27(5):1–9, 2008. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1409060.1409072.

[16] Jiaping Wang, Xin Tong, Stephen Lin, Minghao Pan, Chao Wang, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. Graph.*, 25(3):754–761, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141951. URL http://dx.doi.org/10.1145/1141911.1141951.