Indian Institute Of Technology Delhi

Major Project Thesis

Rendering of Weathered Surfaces

Authors:
Devashish Tyagi

Supervisor:
Prof. Subodh Kumar

A thesis submitted in fulfillment of the requirements for the degree of B. Tech and 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 **Rendering of Weathered Surfaces** being submitted by Devashish Tyagi (2009CS50240) 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

INDIAN INSTITUTE OF TECHNOLGY DELHI

Abstract

Rendering of Weathered Surfaces

by Devashish Tyagi

Metal is one of the most frequently used substance for manufacturing of a wide variety of products. One of the most remarkable property of metal is ability to change appearance over time. Metal surfaces undergo various complex chemical changes that modify it's appearance drastically in some cases. Rendering such time varying effects on metal surfaces is a topic of considerable interest in the field of Computer Graphics. We focus on modeling and rendering of the effect of corrosion on a metal surface. To represent a metal object, we use a 3D array of voxels. A statistical process driven governed by chemical and physical interactions dictates the corrosion level of these voxels after certain time steps. Since corrosion of real world metal objects effect both colour and geometry, we segregate the rendering into the job rendering colour and rendering geometry. In our work we focus on getting an improved rendering of geometry of corroded objects. In order to achieve a realistic looking rendering of the corroded objects, we obtained normal variation data from real life corroded objects. We use the work by Wang et. al.[1] to obtain corrosion information about the object. Using corrosion level information and normal variation information of a real world object, we modify the geometry of the voxel model. We make effective use of shading languages to modify pixel level details and produce visually better rendering.

Acknowledgements

I would like to thank Prof. Subodh Kumar for advising us 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 us at all times. I would also like to thank Keshav Choudhary for being an immense help in the research effort. I am grateful to Sunanda Tyagi and Monalisa Gupta for encouraging me to finish the tedious job of writing the thesis. Finally, I owe my every achievement to my parents.

Devashish Tyagi (2009CS50240)
Indian Institute of Technology Delhi

Contents

Chapter 1

Introduction

A perennial challenge of computer image generation is the depiction of objects with weathered appearances. Much of the visual richness and ambience of household scenes is associated with weathered materials, such as rusting iron grails, corroded plate and copper pipes with patina formation. Unfortunately, the material models widely used in computer graphics assume that the materials are both pristine and immutable, even though real materials are neither. More sophisticated models would allow materials to change over time as they are exposed to the elements. Such models would then allow for the rendering of convincing images of weathered objects by considering a material's structure, its interaction with light, and the physical processes that affect its appearance[2].

These processes have been studied in biology, physics and mathematics[3]. In computer graphics, Dorsey and collaborators have developed a number of specific models for flows, patina formation and weathering [2, 4]. However, the full generality of pattern formation remains beyond the reach of any particular mathematical model or physical simulation.

In our work, we overcome the generalisation difficulties faced by mathematical models by using a date driven rendering technique. Our method is inspired by data driven approach adopted in [5, 6]. The previous works focused on obtaining TSV-BRDFs (Time and Spatially Varying BRDFs) or reflectance models from real life observations for the rendering purposes. We focus on obtaining normal and albedo maps from real life observations. We assume we are given a voxelized 3D data of an object containing the corrosion information. We use the inferred information from real world objects and modify the colour and shape information of the 3D data of the object. We present various methods by which we obtain normal maps from images of real world objects and various rendering algorithm that impart appearance information to meshes of the object. The rendering pipeline presented in this work is much more versatile and produce visually appealing results.

1.1 Prior Work

Several different approaches to modelling/or rendering surface weathering phenomenon have been introduced. These works can be categorised into three main categories: visual and physical simulation techniques, procedural texture generation and data driven approaches. Dorsey *et. al.*[2, 4] have introduced various physical simulation techniques for varied weathering phenomenons. They modelled and rendered both surface and subsurface effects using a stack of layers[4]. On a similar note, works on flow simulation over 2D surfaces[7, 8] demonstrated the ability to affect the appearance of a surface via flow simulation. Methods that implement the subsurface scattering of light within materials are also relevant[9]. Even though results obtained by these methods are realistic, these methods are highly sensitive to environment parameters and it is really difficult to obtain good renderings if random environment settings. They demonstrate the rigidness of the simulation techniques. These techniques are specific in the environments they simulate or render. Another way surface weathering has modelled is through procedural texture generation methods. Procedural texturing[10, 11] is a traditional approach for creating irregular patterns and shapes. Perlin noise[12] is a widely-used for modeling stochastic textures. Turk[13] and Witkin and Kass[14] introduced reaction diffusion textures inspired by biochemical processes. Badler and Becket[15] modeled blemishes based on fractal subdivision techniques and relatively simple distribution models. Chen[16] used $\gamma - ton$ tracing to simulate weathering effects that introduce blemishes such as dirt, rust, cracks and scratches. It works by introducing age inducing particles called $\gamma - ton$ particles to the visual scenery. These methods fail to take mimic real life weathering processes and require significant manual intervention. An emerging technique for rendering weathering process is data driven methods[5, 6]. These techniques derive some kind of visual information from real world data in this case TSV-BRDFs and Reflectance models. Though these methods demonstrate the versatility needed to model various weathering processes, they are unable to produce good renderings mainly due to the absence of geometric details. We overcome this problem by explicitly adding geometric details using normal map. In Chapter 2 we describe our rendering pipeline in detail and also illustrate the results obtained using it in Chapter 3.

Chapter 2

Rendering Pipeline

As indicated in the preceding chapter most of the past work relied on expensive approximate simulations of the chemical and physical processes that lead to altering of appearance of metal objects[2, 4, 17, 18]. Such style of rendering imposes many types of restrictions on the resulting rendering technique. The first and most important of them is limitation of the kind of corrosion process that can rendered. Different metals corrode differently under different conditions resulting in varied appearances[19]. This intricacy makes entirely simulation based techniques limited in their scope. Another problem associated with such techniques is their inability of capture geometric features. Most of the past techniques rely on modifying the reflectance characteristics of the surface and hence modifying their appearance[18]. Other techniques rely on their mathematical model (physical simulation) to generate the varied appearances[4, 16, 17]. These problems in essence arise due to the close coupling of modelling and rendering pipelines. This leads to over-dependency of rendering on modelling.

We overcome the above mentioned problems by using a data driven approach to rendering. Our rendering pipeline is illustrated in Figure 2.1. In our rendering pipeline we decouple the job of modelling and rendering. The rendering solely relies on the real world data obtained from corroded objects. The input consists of a photograph of a corroded object with all the necessary information and an approximate height map. The photograph of the corroded object can be obtained from any camera with decent resolution. The height map of the object can be obtained from any low or medium quality 3D scanner. The mid processing unit consists of 3 separate image processing algorithms that extract the following information from the inputs

- **Albedo Map** This map contains the diffuse reflecting power of the surface. Such information is helpful in obtaining and estimating the diffuse reflecting power of a surface with particular corrosion value.

- **Weathering Map** This map contains the weathering degree for each pixel corresponding to the input image.
- **Normal Map** This map contains the normal variation of the surface of the corroded object.

The obtained inputs serve as precursors to the mid processing unit which produces all the information needed by the rendering algorithm.

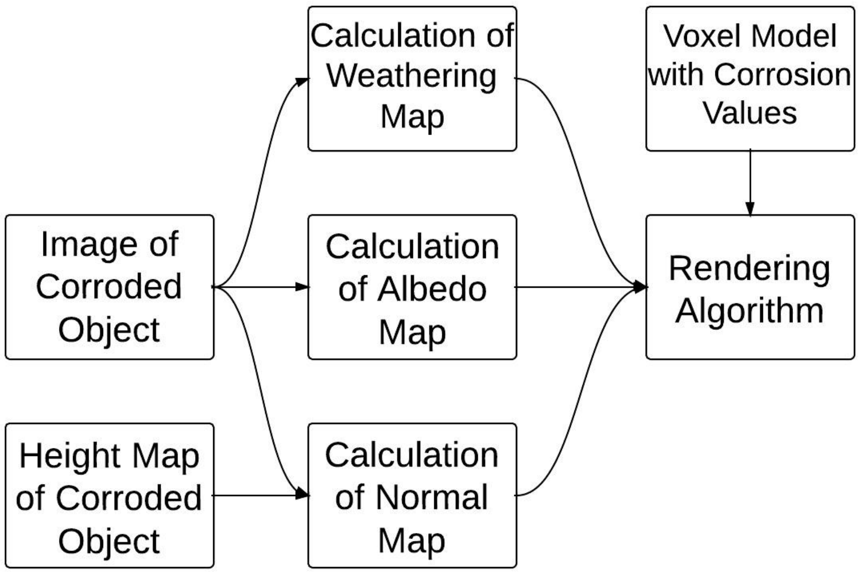

Calculation of Weathering Map
Voxel Model with Corrosion Values
Image of Corroded Object
Calculation of Albedo Map
Rendering Algorithm
Height Map of Corroded Object
Calculation of Normal Map

FIGURE 2.1: Figure shows the data driven pipeline implemented in our work. The data used to drive our rendering in this case includes an image of the corroded object obtained under natural illumination

In the following section we will describe the pipeline stages in detail and then we will move onto the kind of geometric details we need to capture and how we go about calculating the normal map.

2.1 Pipeline Stages

The rendering pipeline consists of 3 main stages

1. Input Collection
2. Processed Information Generation
3. Rendering Algorithm

Since our goal is to achieve a rendering that is very similar to the real world objects, we harness real world data for the purpose of rendering. We use an intrinsic image of a corroded object to obtain appearance information needed for rendering. An image taken in natural illumination is a result of many underlying properties such as diffuse coefficient, specular coefficient, normal map and in some cases occlusion map. Each of them define an intrinsic property of the surface in question. We derive and use some of this information to define the surface properties and hence the appearance of our model. More specifically we use diffuse coefficient and normal map obtained from real world objects to decide how our model would look. Obtaining the desired information from just an intrinsic image without any prior information is a long standing information in the field of computer vision. Even after dearth of literature on the subject, this problem is still very hard to solve. The difficulty of the problem stems from the fact that it is an under constrained problem. In order to aid calculation of intrinsic surface properties from an image, we make use of height map obtained from a low/medium quality 3D scanner (0.5 - 1mm 3D resolution). After we have acquired our input in form of an image of a corroded object and it's corresponding height map we move onto processed information generation phase. In the Processed Information Generation phase we generate three main information regarding the surface of the real world object

1. Diffuse Map
2. Weathering Degree Map
3. Normal Map

The process and intricacies of generating a diffuse coefficient map from a single image is discussed by Choudhary[20]. Weathering degree map is generated from a semi automatic process inspired by work of Wang *et al* [1]. The proposed technique called appearance manifold is used for modeling the time variant surface appearance of a material from a single image. This method takes advantage of the key observation that concurrent variations in the appearance over a a surface represent different degrees of weathering. It reorganizes these various appearances in a manner that reveals their relative order with respect to weathering degree. This method requires the user to mark the areas with least corrosion values and areas with most corrosion values. The method assumes that course of weathering produces gradual monotonic transformations in appearance that exist at spatially varying degrees over the surface of the material sample, such that various states of per-pixel appearance form a manifold in appearance space. Since this manifold exhibits a non-linear structure, the authors approximate the underlying manifold by plotting the captured BRDF data of each surface point in appearance space defined by reflectance features and construct a neighborhood graph among these sample points, which is referred to as appearance manifold. Given the input from the user, the method infers from relative positions of points on the manifold their relative degrees of weathering. A sample result of the technique is illustrated in Figure 2.2. Black areas indicate areas of high corrosion values and white areas indicate areas of low corrosion values. Normal map generation is much more intricate and is discussed in the following sections. Given the level of detail present in normal maps and the under constrained nature of the problem extra input is required. In this scenario we used the captured height map of a real world object to guide the normal generation algorithms for revealing more details in the generated normal maps.

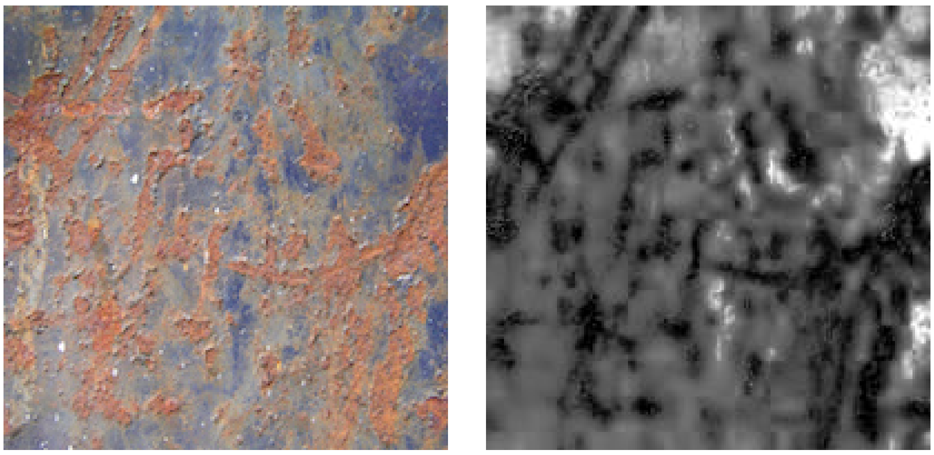

(A) Input Image (B) Output Weathering Map

FIGURE 2.2: Black areas in Figure 2.2b represent areas of high corrosion in Figure 2.2a

After we have produced all the necessary surface information from our Processed Information Generation phase, we move onto the rendering algorithm phase. An important step here is finding correspondence between the points on the model and pixels in the various maps obtained. We use heuristics to obtain this correspondence and illustrate resulting rendering in next chapter.

2.2 Computing Normal Map

Recovering normal map from a single image comes under the classical problem of recovering shape from a single image. Shape recovery is a classical problem in computer vision. The goal is to derive a 3-D scene description from one or more 2-D images. The recovered shape can be expressed in several ways: depth $Z(x, y)$, surface normals (n_x, n_y, n_z), surface gradient (p, q), and the surface slant, ϕ, and tilt, θ. The depth can be considered either as the relative distance from camera to surface points, or the relative surface height above the x-y plane. The surface normal is the orientation of a vector perpendicular to the tangent plane on the object surface. The surface gradient, $(p, q) = (\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y})$, is the rate of change of depth in the x and y directions. The surface slant, ϕ, and tilt, θ, are related to the surface normal as $(n_x, n_y, n_z) = (l \sin\phi \cos\theta, l \sin\phi \sin\theta, l \cos\phi)$, where l is the magnitude of the surface normal. We concentrate on representing shape with normals. Our choice is justified by ease of use in lighting and ability to capture more details as compared to height map. Before we discuss the methods considered for achieving the goal, we have to discuss the phenomenon of image formation since all these techniques rely on this information. A simple model of image formation is the Lambertian model, in which the color at a pixel in the image depends on the light source direction, the surface normal and the diffuse coefficient. In shape from image problem , given a image, the aim is to recover the light source, the surface shape and diffuse coefficient at each pixel in the image. However, real images do not always follow the Lambertian model. Even if we assume Lambertian reflectance and known light source direction, and if the brightness can be described as a function of surface shape and light source direction, the problem is still not simple. This is because if the surface shape is described in terms of the surface normal, we have a linear equation with three unknowns, and if the surface shape is described in terms of the surface gradient, we have a non-linear equation with two unknowns. Therefore, finding a unique solution to shape from image is difficult and it requires additional constraints.

2.2.1 Photometric Stereo

This method intends to solve the problem of determining shape of an object from intensity information under different illumination, but from the same viewing position. This method, known as Photometric Stereo, was first formulated by Woodhan[21]. The photometric stereo is based on the following fact: if a pair of images of the same image are obtained by varying the direction of incident illumination but from the same viewing direction, we can draw a pair of different reflectance maps, because reflectance maps depends on the direction of light source. Surface orientation is determined locally by the intensity pairs recorded at each image point as an intersection of constant brightness lines on the reflectance maps. This photometric stereo method is very rapid and free from noise due to the local nature of calculations. Earlier works in this field were focused on diffusely reflecting surfaces and employed point source of illumination. These methods suffered from many drawbacks. The two most important drawbacks included working only for diffused surfaces when most industrial applications require or use specularly reflecting surfaces and using point source of illumination which is quite difficult to achieve in real world settings.

Ikeuchi[22] made definitive progress in this field by extending the method to specularly lit surfaces and using distributed light source instead of point source of light. Their method requires two kind of tasks : one is off-line (precomputing) job and other is on-line (real time) job which is rather simple as compared to off-line job. The off-line job consists of calculating the reflectance map and constructing the lookup table. The lookup table is a 2D table that contains the shape information of a point corresponding to a pair of brightness information. The on-line job consists of reading the image brightness and determining orientations of a surface patch based on the lookup table. Each dimension of the table corresponds to brightness measurements on the surface patches under one of the two sources. Each entry of the table contains a surface orientation corresponding to a pair of brightness measurements. In the image acquisition settings the object is illuminated by three line sources placed symmetrically at 120°. Each lamp source is turned on one at a time giving rise to three different reflectance map. By using these three reflectance values, the surface orientation is determined. Theoretically, it is possible to determine surface orientation using two reflectance maps only. Details of calculation of reflectance map and the lookup table is detailed by Ikeuchi[22]. For the on-line job, the brightness is obtained from TV camera. To reduce the noise typical of these devices, more than one picture per light source are taken, and arrays corresponding to the same light source are averaged. The resulting three brightness arrays, one for each light source, are input to the photometric stereo system.

Though this method works appreciably well for most objects, it is not used by our rendering method. The primary reason for this is the fact that it requires controlled image acquisition settings. Our method aims to achieve the goal of rendering various weathering phenomenon by obtaining data from varied sources which are likely to be images under uncontrolled settings. We don't want to restrict our rendering pipeline by data acquisition techniques. That's why we choose to pursue techniques that work with less information and are not dependent on acquisition conditions or parameters. Such a restriction also rules out promising techniques such as global illumination technique[23].

2.2.2 Shape from Shading

Work by Choudhary[20] illustrate that fairly good approximation of albedo map from image can be obtained for corroded objects which implies a good approximation for shading from image. An example output of the results obtained by their work can be seen in Figure 2.3. This warrants the use of shape from shading techniques to obtain the normal map for single images.

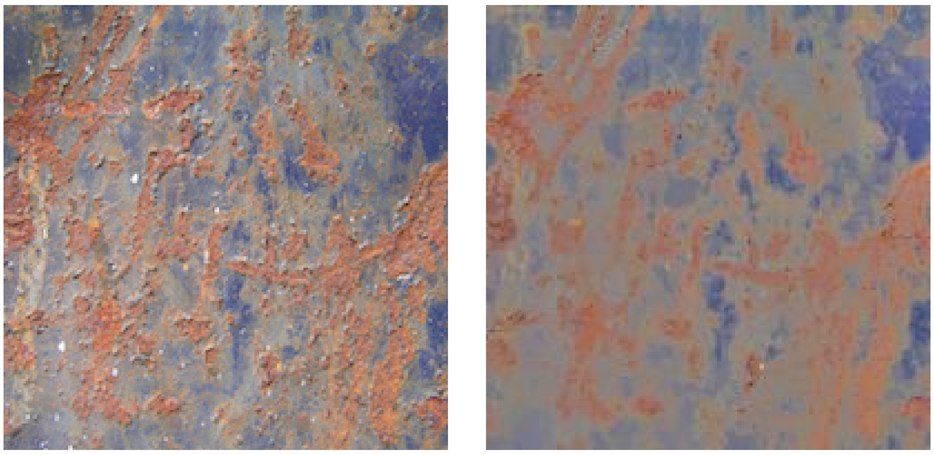

(A) Input Image (B) Output Albedo Map

FIGURE 2.3: Technique proposed by Choudhary[20] result in the albedo map as shown in Figure 2.3b corresponding to image in Figure 2.3a

Shape-from-shading (SFS) deals with the recovery of shape from a gradual variation of shading in the image. Shading has been employed by many famous artists to convey vivid illusions of depth in paintings. Shading is one of the cues through which humans are able to accurately perceive the shape of an object from an image. This fact has inspired many researchers in human vision to understand and simulate the process by which our eyes and brains actually use the shading information to recover the 3-D shapes. It's not just the shading, but also the outlines, elementary features, and the visual system's knowledge of objects that help the brain to recover shape information. This was illustrated by Ramachandran[24]. Stereoscopic processing heavily affects the extraction of SFS by visual system. Barrow and Tenenbaum discovered that it is the line drawing of the shading pattern that seems to play a central role in the interpretation of shaded patterns[25]. Mingolla and Todd's study of human visual system based on the perception of solid shape[26] indicated that the traditional assumptions in SFS-Lambertian reflectance, known light source direction, and local shape recovery, are not valid from psychology point of view. One can observe from the above discussion that human visual system uses SFS differently than computer vision normally does.

SFS techniques can be classified into four groups

1. **Minimization approaches** These approaches obtain the solution by minimizing an energy function.
2. **Propagation approaches** These approaches propagate the shape information from a a set of surface points (e.g., singular points) to the whole image.
3. **Local approaches** These approaches derive shape based on the assumption of surface type.

4. **Linear approaches** These approaches compute shape based on linearization of reflectance map.

A detailed analysis of all the state of art SFS algorithms in all the above categories has been conducted by Zhang *et al*[27]. They implement and compare performance and result of six well known SFS algorithms. All the algorithms were implemented in C and were evaluated on three synthetic images and two natural images. Mean and standard deviation of depth(Z) error, mean of surface gradient (p, q) error and CPU timing were used as evaluation metric. None of the algorithms worked well for all the images. In general minimization SFS algorithm are more robust while other algorithms are faster. Also these algorithms rely on the assumption of homogeneity and smoothness which if often violated in case of corroded surfaces. These surfaces have sharp changes in shape information which is not captured well by these algorithms. Also the lack of coherence between shading recovery algorithms and shape from shading algorithms leads to under performance of the technique. A better alternative is to look at a superset of these techniques which recovers shape and illumination at the same time. Thus, we move on to the next technique SIRFS[28].

2.2.3 SIRFS

Traditional methods of recovering scene properties such as shape, reflectance or illumination, like photometric stereo, rely on multiple observations of the same scene to over-constrain the problem. Recovering these properties from a single image is almost impossible in general setting as there are infinite number of shapes, paints and lights that all produce the same image. However, certain explanations are much more likely than others: surfaces tend to be smooth, paints tend to be uniform, and illumination tends to be natural. Barron and Malik therefore pose this problem as one of statistical inference, and define an optimization problem that searches for the most likely explanation of a single image[28]. The technique for recovering intrinsic scene properties from a single image of a object is labeled as “Shape, Illumination, and Reflectance from Shading”or “SIRFS”.

SIRFS is a combination of classical Shape from Shading and “intrinsic image”techniques to recover shading and reflectance. In case of SIRFS not only shape but also illumination and reflectance are unknown. The SIRFS problem formulation is:

$$\begin{aligned} \underset{R,Z,L}{\text{maximize}} \quad & P(R)P(Z)P(L) \\ \text{subject to} \quad & I = R + S(Z, L) \end{aligned} \tag{2.1}$$

where R is a log-reflectance image, Z is depth map, L is a spherical harmonic model of illumination[29]. $S(Z, L)$ is a "rendering engine"which linearizes Z into a set of surface normals, and produces a log-shading image from those surface normals and L. $P(R)$, $P(Z)$, and $P(L)$ are priors on reflectance, shape, and illumination, respectively, whose likelihoods we wish to maximize subject to the constraint that the log-image I is equal to a rendering of our model $R + S(Z, L)$. We can simplify this problem formulation by reformulating the maximum-likelihood aspect as minimizing a sum of cost functions (by taking the negative log of $P(R)P(Z)P(L)$) and by absorbing the constraint and removing R as a free parameter. This gives us the following unconstrained optimization problem:

$$\underset{Z,L}{\text{minimize}} \quad g(I - S(Z, L)) + f(Z) + h(L) \tag{2.2}$$

where $g(R)$, $f(Z)$ and $h(L)$ are cost functions for reflectance, shape, and illumination respectively which are often referred to as "priors"on scene properties. Since we are concerned with recovery of shape from image, we will examine only the "priors"on shape in detail in the following sub-section.

2.2.3.1 Priors on Shape

The prior on shape consist of three main components:

1. An assumption of smoothness (that shapes tend to bend rarely), which is modeled by minimizing the variation of mean curvature.
2. An assumption of isotropy of the orientation of surface normals (that shapes are just as likely to face in one direction as they are another) which reduces to a well-motivated "fronto-parallel"priors on shape.
3. A prior on the orientation of surface normals near the boundary of the objects, as shapes tend to face outward at the occluding contours.

Formally, the shape prior is a weighted combination of three costs:

$$f(Z) = \lambda_k f_k(Z) + \lambda_i f_i(Z) + \lambda_c f_c(Z) \tag{2.3}$$

where $f_k(Z)$ is our smoothness prior, $f_i(Z)$ is our isotropy prior, and $f_c(Z)$ is our bounding contour prior. The λ multipliers are learned through cross validation on the training set.

One overarching theme in the field of natural image statistics is regularizing some function of the second derivative of a surface is effective. However, this method is susceptible to severe issues because of lack of invariance of these priors to out of plane rotation and scaling. Instead of using traditional priors from natural image statistics, the authors use mean curvature of the surface. Mean curvature is simply the divergence of the normal field. Mean curvature is defined as mean of principle curvatures $H = \frac{1}{2}(\kappa_1 + \kappa_2)$. It can be approximated on a surface using filter convolutions that approximate first and second partial derivatives, as show in [30].

$$H(Z) = \frac{(1 + Z_x^2)Z_{yy} - 2Z_xZ_yZ_xy(1 + Z_y^2)Z_{xx}}{2(1 + Z_x^2 + Z_y^2)^{\frac{3}{2}}} \tag{2.4}$$

The smoothness prior for shapes in Gaussian scale mixture on the local variation of the mean curvature of Z:

$$f_k(Z) = \sum_i \sum_{j \in N(i)} c(H(Z)_i - H(Z)_j; \alpha_k, \sigma_k) \tag{2.5}$$

$N(i)$ is the 5x5 neighborhood of pixel i, $H(Z)$ is mean curvature of shape Z, and $H(Z)_i - H(Z)_j$ is the difference between mean curvature at pixel i and pixel j. $c(;\alpha_k, \sigma_k)$ is defined as follows:

$$c(x; \alpha, \sigma) = -log(\sum_{j=1}^{M} \alpha_j \mathcal{N}(x; 0, \sigma_j^2) \tag{2.6}$$

It can be described as the negative log likelihood (cost) of discrete univariate Gaussian scale mixture (GSM), parametrized by α and σ, the mixing coefficients and standard deviations, respectively, of the Gaussians in the mixture. We set M = 40 (the GSM has 40 discrete Gaussians), and α_k and σ_k are learned from our training set using expectation-maximization.

The prior on surface isotropy is inspired by the fact that surface tend to be oriented isotropically. This implies that it is equally likely for a surface to face in any direction. Imposing this prior seems like any easy task as there is equal cost associated with each direction of the surface normal. But we have ignored the fact that we have observed the surface in space, which implies that is more likely for the surface to face the observer than to be perpendicular to the observer. Hence, a prior is imposed in order to undo the bias. The prior for imposing surface isotropy is given by:

$$f_i(Z) = -\sum_{x,y} log(N_{x,y}^z(Z)) \tag{2.7}$$

where $N^z_{x,y}(Z)$ is the z-component of the surface normal of Z surface at (x,y) position (x,y). The shapes that are made most likely by this prior are “fronto parallel”surfaces which help in combating the bas-relief ambiguity.

Since the images in our scenario won’t be accompanied by a contour image stating the presence and position of contour pixels, the prior on occluding contours doesn’t have much effect on our results. For the purpose of completion we will state the prior used for occluding contours:

$$f_c(Z) = \sum_{i \in C} (1 - (N^x_i(Z) n^x_i + N^y_i(Z) n^y_i)^{\gamma_c} \tag{2.8}$$

where n^x_i is the estimated local normal to the occluding contour in the image place, $N(Z)$ is the surface normal of Z. $\gamma_c = 0.75$ as it fits the training set best.

2.2.3.2 Optimization

To estimate shape, illumination, and reflectance, we must solve the optimization problem in Equation 2.2. This is a challenging optimization problem, and naive gradient-based optimization with respect to Z and L fails badly. Therefore the authors use an effective multiscale optimization technique, which is similar in spirit to multigrid methods [31], but extremely general and simple to implement. We will describe our technique in terms of optimizing $a(X)$, where $a()$ is some loss function and X is some signal.

Let us define $\mathcal{G}$, which constructs a Gaussian pyramid from a signal. Because Gaussian pyramid construction is a linear operation, we will treat $\mathcal{G}$ as a matrix. Instead of minimizing $a(X)$ directly, we minimize $b(Y)$, where $X = \mathcal{G}^T Y$:

$$\begin{aligned} &[l, \nabla_Y l] = b(Y) : \\ &X \leftarrow \mathcal{G}^T Y // \text{ reconstruct signal} \\ &[l, \nabla_X l] \leftarrow a(X) // \text{ reconstruct signal} \\ &\nabla_X l \leftarrow \mathcal{G} \nabla_X // \text{ back propagate gradient} \end{aligned} \tag{2.9}$$

Y is initialized to a vector of all 0’s and then solved for $\widehat{X} = \mathcal{G}^T(argmin_Y b(Y))$ using L-BFGS.

2.2.3.3 Experiments

In this section we demonstrate how the technique performed on various inputs provided to it. Figure 2.4 shows how SIRFS perform on an image of corroded object without any changes. It is evident that all the major details are not captured by the algorithm. The reason for this is primarily based in the choice of smoothness prior. On one hand it prevents the algorithm from capturing the fine details that are present on the surface of a corroded object, on the other hand it maintains surface integrity and gives rise to smooth surface shapes.

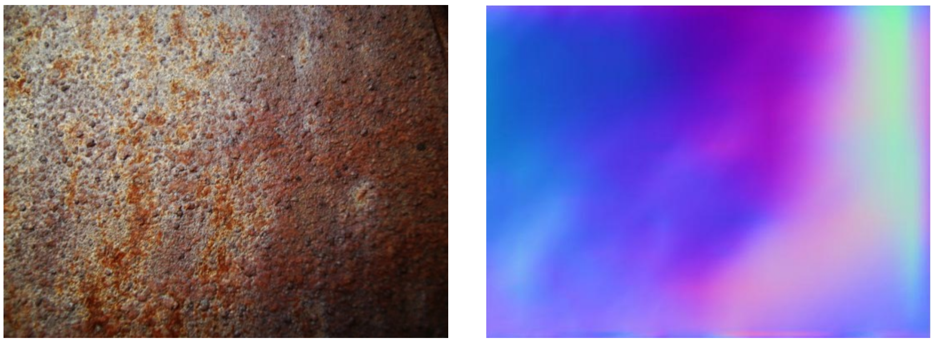

(A) Input Image (B) Output Normal Map

FIGURE 2.4: Normal SIRFS performs badly on surfaces with fine surfaces variations as in this case. SIRFS even fails to capture the broader variations in Figure 2.4a

In order to overcome the bias created by the smoothness prior, the weight of this prior was adjusted to achieve a better result. Figure 2.5b shows the result when the weights were adjusted to bring the best possible result.

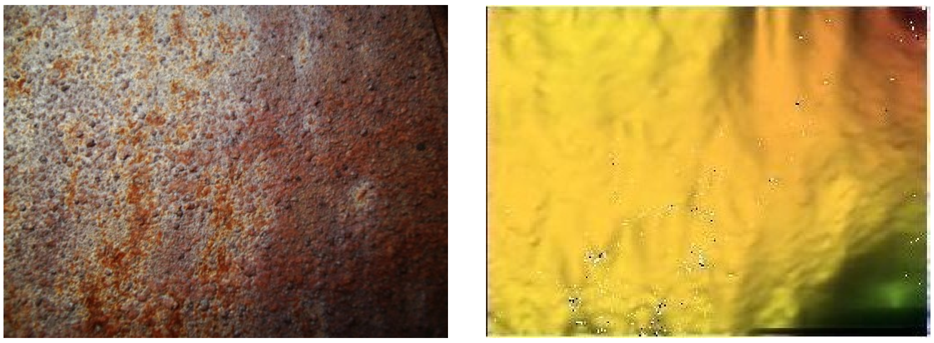
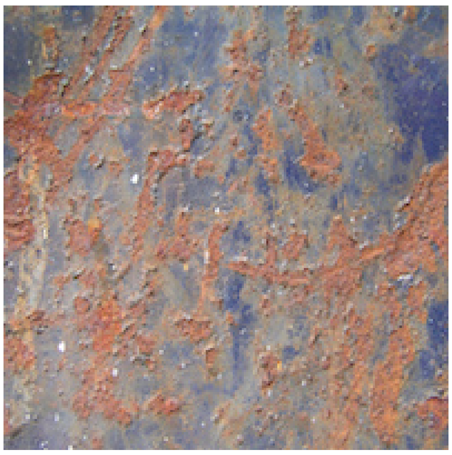
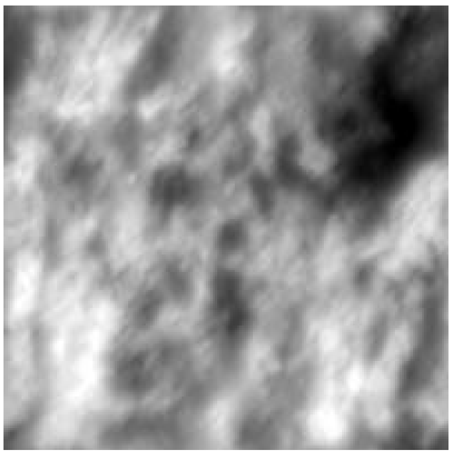

(A) Input Image (B) Output Normal Map

FIGURE 2.5: Since the smoothness prior prevents the SIRFS algorithm from capturing fine details on the surface of the objects, we change the weights of smoothness priors. As a result we obtain a result which shows much more variation on the surface.

SIRFS fail because of one more reason and that is failure to capture the spatially varying illumination. As we can see in Figure 2.4a the illumination varies from left to right which right being on the darker side. This shading effect is observed even though the object is mostly horizontal. The output from SIRFS in Figure 2.5b follows the shading cues and varies the normal from left to right even though there should not be any such change. The best way to overcome is problem is to provide some kind of hint to the SIRFS algorithm. In our case we would be using a height map. SIRFS optimization algorithm can initialized with a provided height map and used for further inference. Figure 2.6b shows the height map provided to modified SIRFS as input and Figure 2.6c shows the output obtained. The height map in the rendering pipeline is collected by using a low quality scanner.


(A) Input Image


(B) Input Height Map

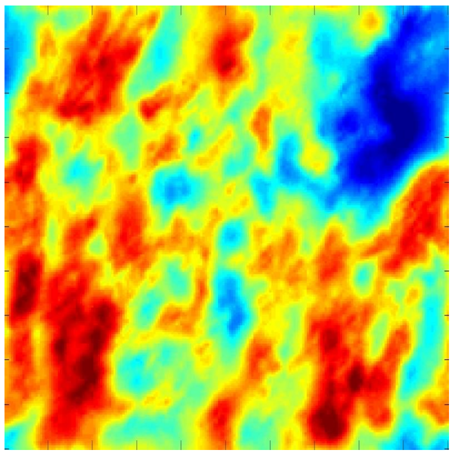
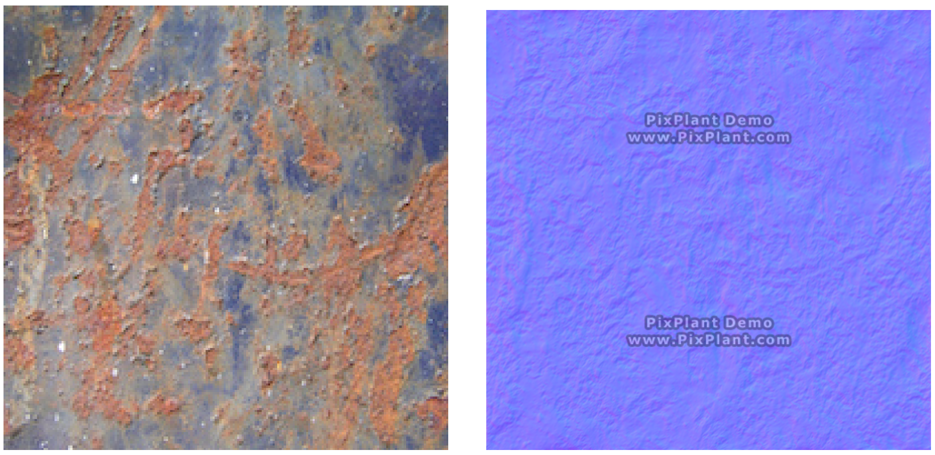

(C) Output Normal Map

FIGURE 2.6: The height map in this case was obtained by commercial software CrazyBump using a low resolution image of the original object. This the output obtained when SIRFS is supplied with a height map that enables the initial optimization steps to be taken in the desired direction. This improvement also overcomes the inability of original SIRFS to remove the effects of spatially varying illumination

2.2.4 Commercial Software

The quality of normal maps generated from real objects haven't been satisfactory this far. We decided to look for options outside the realm of academic research. A lot of open source and commercial organizations have invested tremendous effort in producing software that generate normal maps, height maps and other rendering information just from an image. Some of the most popular software in this category include CrazyBump[32], NeoTextureEdit[33], MaPZone[34], Knald[35] etc. The list of offerings is varied and includes various normal and height map manipulation tools too. We evaluated some of the most popular tools used for the above stated purpose and decided to use PixPlant[36] for generating normal maps for our rendering purposes. The example obtained from the software can be seen in Figure 2.7.



(A) Input Image (B) Output Normal Map

Figure 2.7: The normal map in this case is obtained from commercial software PixPlant. The quality of normal map obtained is much better than any of the purely methods based purely on academic research.

Even though using commercial software for generating normal maps is fairly easy and the quality of the results is good, we intend to improve upon the traditional academic methods and come up with a fully automated method of our own. Since the software that works best are not open source, a solution to this problem requires much more effort and a detailed survey of techniques available.

Chapter 3

Rendering Geometry

3.1 Input Data

As indicated by the rendering pipeline described in 2, our rendering algorithm requires several inputs. albedo map, normal map, weathering map, object description with corrosion values and parameterization of the object. Methods and issues arising in parameterization of general closed body objects are discussed in [20]. The input image 3.1, albedo map (Fig 3.2), normal map (Fig 3.3), weathering map (Fig 3.4) and the object (Fig 3.5) with corrosion values which we would be using to evaluate our rendering algorithms are illustrated below.

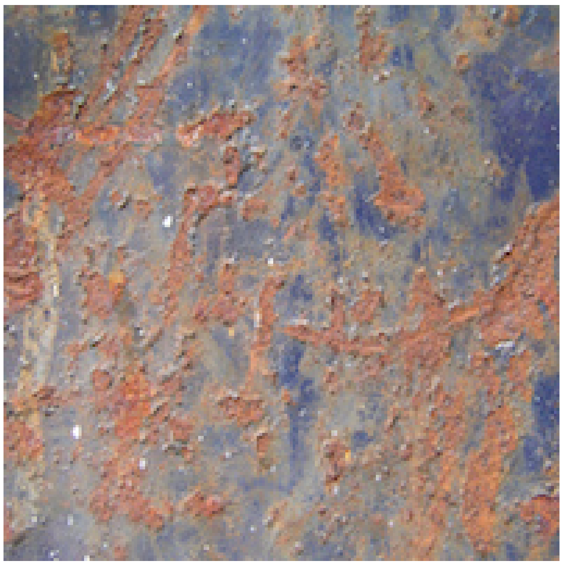


Figure 3.1: Input image of the object. Information from this image is used to infer colour and shape for various weathering degree.

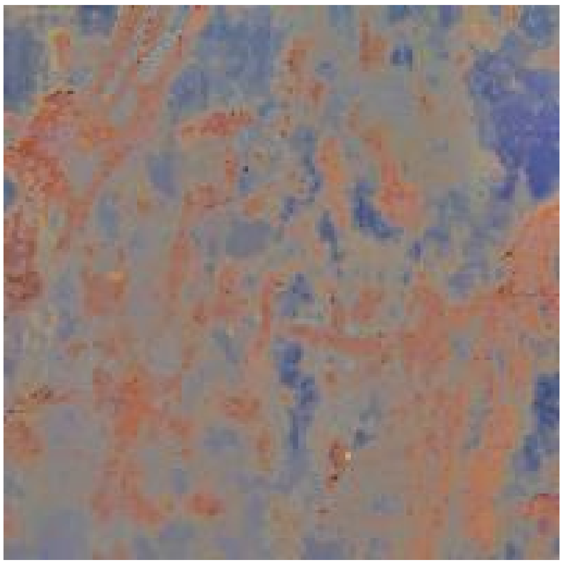


FIGURE 3.2: Albedo map corresponding to the input image. This image is obtained by the methods discussed in [20].

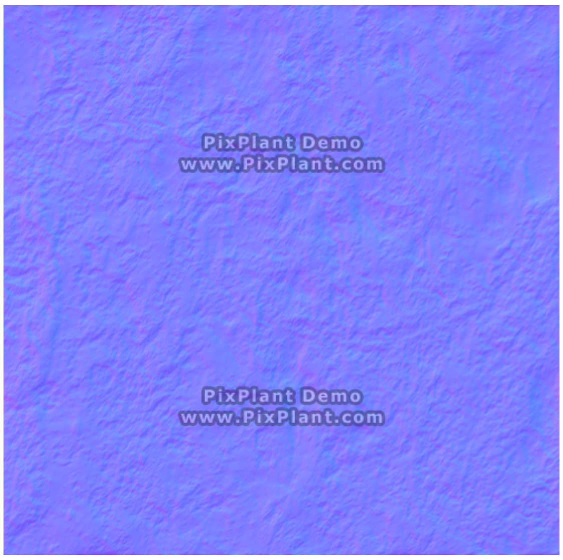


FIGURE 3.3: Normal map corresponding to the input image. This map contains all the shape information that would be used to alter the geometry of the object with corrosion information.

3.2 Rendering Algorithm

The main challenge of the rendering algorithm is to assign colour and normal values to the pixels in the final rendering so as to maintain local coherence. There are two primary ways in which colour and normals can be assigned. First, assign colour and normal per vertex and interpolate for the pixels. Second, assign colour and normal per pixel. The first method has the advantage of being computationally cheaper while the second method is able to achieve much more details. We present the algorithms we

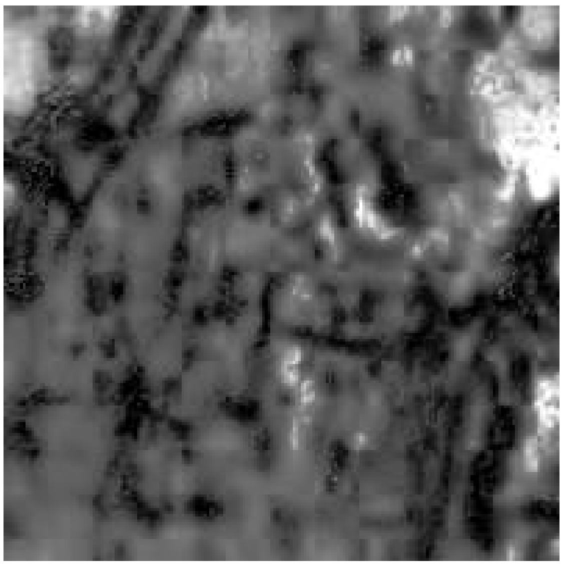


Figure 3.4: The central piece of information that connects information extracted from image to the actual object model with corrosion values. Dark areas indicate areas with higher corrosion values and vice-versa.

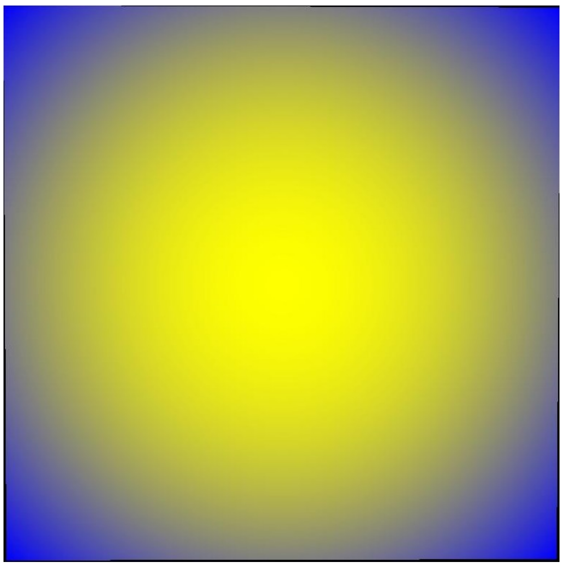


Figure 3.5: The object we will use for evaluation purposes consists of a simple plate with corrosion values increasing radially outwards. The corrosion values decrease as a linear function of distance from the centre of the plate. The plate has corrosion value variation from 0.0-1.0.

developed for our rendering engine and illustrate how we improve rendering at each step without sacrificing the computational efficiency by a significant amount.

3.2.1 Nearest Neighbour Match per Vertex

We receive object description as a mesh with corrosion value associated with each vertex. Let us denote the corrosion value associated with in a vertex i as c_{v_i} and let the corrosion level at a pixel (p, q) in the weathering map be $W(p, q)$. Average weathering degree at a vertex is defined as:

$$\gamma_{v_i} = \frac{\sum\limits_{j \in N(i)} c_{v_j}}{|N(v_i)|} \tag{3.1}$$

where $N(i)$ denotes 16 closest neighbours (in breadth first manner) of vertex v_i in the mesh. Average weathering degree at a pixel in the weathering map is defined as:

$$\beta_{p,q} = \frac{\sum\limits_{r,s \in N(p,q)} W(r, s)}{|N(p, q)|} \tag{3.2}$$

where $N(p, q)$ is a 4x4 neighbourhood of pixel (p, q). In our first iteration of our algorithm we compute the average weathering degree for all the vertices in the mesh and pixels in the weathering map. Then for each vertex we find the pixel in the weathering map for which $\gamma_{v_i} - \beta_{p,q}$ is minimised. The colour and normal corresponding to that pixel are picked up form the normal map and the albedo map. These colour and normal values are assigned to the vertex in question. In the next iteration we render the mesh where each vertex has been assigned a colour and normal value. The result obtained by this method can be seen in Figure 3.6.

3.2.2 Nearest Neighbour Match per Pixel

The earlier rendering failed to capture the fine rendering details. This can be primarily attributed to the fact that colour and normals were assigned at vertex level and were interpolated to pixels. There is smoothing happening here which prevents fine details from emerging. A naive solution to this problem would be interpolate corrosion values to pixels and apply the above algorithm at the pixel level. But this operation is computationally very expensive. A more efficient way to achieve pixel level rendering is to find the texel (p, q) at which $\gamma_{v_i} - \beta_{p,q}$ is minimised and assigned this as texture coordinates to the vertices. These texels are then interpolated to pixel level and used to lookup colour and normal values. This method has the advantage of being computationally efficient and achieving the goal of detail assignment at pixel level.The result obtained by this method can be seen in Figure 3.7. An important aspect of this rendering algorithm

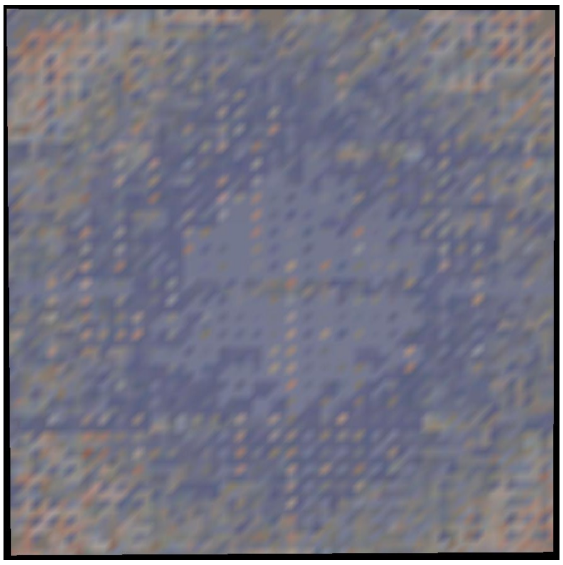


FIGURE 3.6: The rendering obtained when we assign colour and normal values per vertex. The mesh contains 4,000 vertices. Since most of the details are interpolated, the rendering obtained is hazy. A solution to this problem is assigning colour and normal values at pixel level.

is how the texel values assigned to vertices are distributed over the object surface. A visualisation of this distribution can been seen in Figure 3.8. In the visualisation we have assigned each pixel (r, g, b) value as $(u, v, 0)$ where (u, v) is their corresponding texel co-ordinates. Figure 3.1 the least corroded areas are concentrated in the top right corner. In the (u, v) space this region corresponds to higher v value as compared to u value. Therefore we observe a dominant green colour in the centre of the plate. Similarily, the rusted region is concentrated along top left and middle right. This region in (u, v) space corresponds to either dominant u or dominant v. Therefore, we observe darker green as we move towards the periphery of the plate which then gives way to shades of red. It is evident from the visualisation that the texel are picked from all over the weathering map and kind of allow neighbour coherence.

This rendering still suffers from some issues. There are concentric rings of colour that alternate. Since there is a smooth variation of corrosion values on the object surface, the colour pattern should largely mimic the behaviour.

3.2.3 Discrete Optimization

On a theoretical level, our goal is to achieve texel values for vertices in a manner such that their values don't deviate much from their neighbours. In the previous algorithm we picked a texel value for a vertex and assigned that as the only possible value for that vertex. Instead of having such a rigid assignment algorithm which just seeks to minimise $\gamma_{v_i} - \beta_{p,q}$ solely, we devise an algorithm that chooses K texel values corresponding to

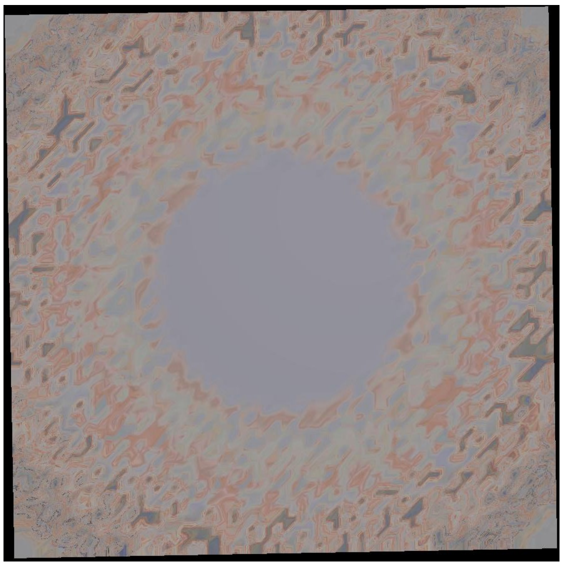


FIGURE 3.7: The rendering obtained when we assign colour and normal values per pixel.

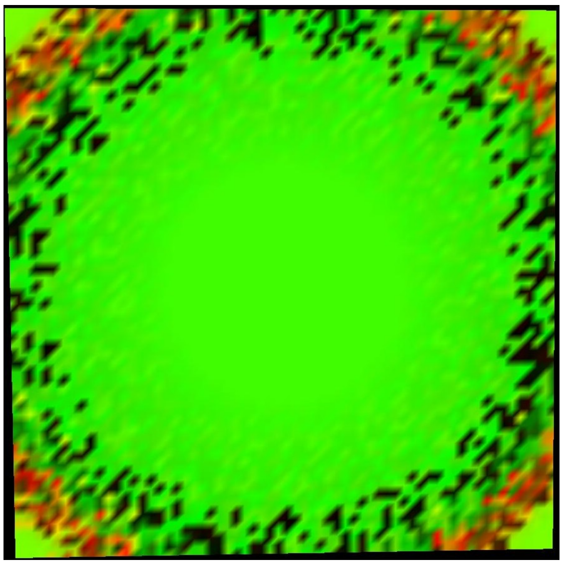


FIGURE 3.8: The distribution of texel values on the surface of the input object.

smallest K instances of $\gamma_{v_i} - \beta_{p,q}$ for a vertex. Now a vertex has K values of texels to choose from. The final choice will depend upon minimising the distance from adjacent vertices in texel space. In order to achieve this goal we use a modified voting algorithm. The algorithm starts from a random vertex in the mesh and executes the voting algorithm to assign it a texel value. It then proceeds to it's neighbours in a breadth first manner and keeps assigning them texel values using the same voting algorithm. The voting algorithm for a vertex v_i is detailed below:

1. Initialise $VotingCount[K] \leftarrow \{0\}$
2. For each neighbour v_j of v_i do the following

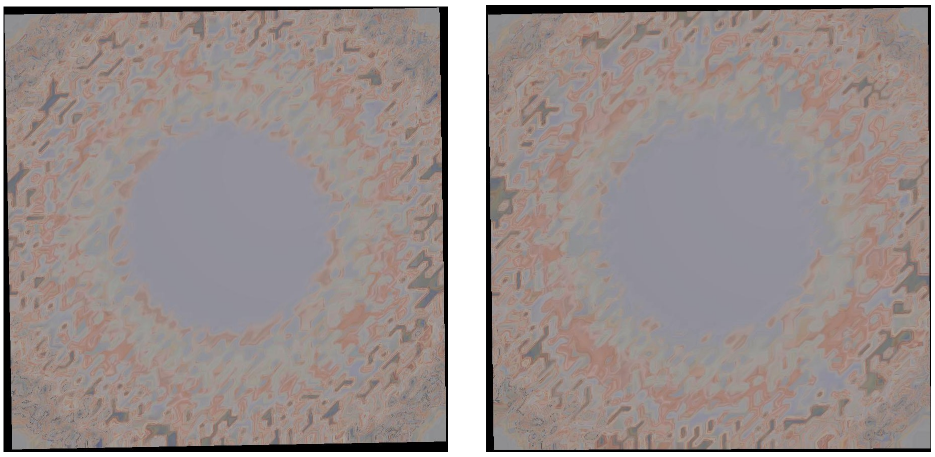
   (a) Find the minimum distance between all possible texel values for v_j and v_i.

   (b) Assume the minima is achieved for r^{th} texel value out of K for v_j and s^{th} texel value out of K for v_i.

   (c) $VotingCount[s] \leftarrow VotingCount[s] + 1$

3. Find for which value of $x \in [1, K]$ is $VotingCount[x]$ the highest. Assign the x^{th} texel value out of K to vertex v_i.

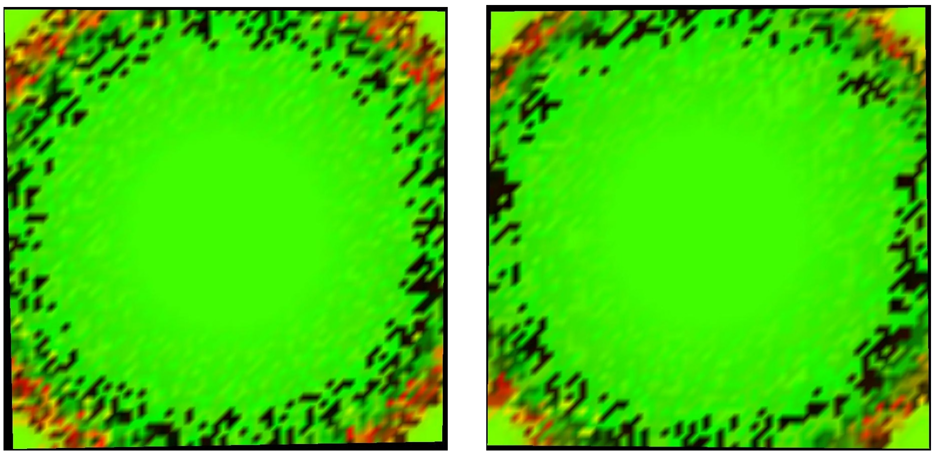
This heuristic gives more leeway to the choice of texel values for the vertices and thereby ensuring there is much more neighbourhood coherence. The results of this rendering are compared with our previous algorithm in Figure 3.9.



(A) Rendering Algorithm 2 (B) Rendering Algorithm 3

FIGURE 3.9: The result from Rendering Algorithm 3 has much more neighbourhood coherence as observed in middle section and edges of the surface. Also due to increased coherence the details are much more prominent.

We also compare the distribution of texel values over the surface of the object. By use of discrete optimization technique we hope to eliminate the problem of dark and abrupt patches observed in Figure 3.8. As Figure 3.10 illustrates we are successful in achieving our goal.



(A) Texture Coordinate Distribution 2

(B) Texture Coordinate Distribution 3

FIGURE 3.10: The result from Rendering Algorithm 3 has much more neighbourhood coherence as observed in middle section and edges of the surface. Also due to increased coherence the details are much more prominent.

Chapter 4

Conclusion

In our work we illustrate a methodology to render weathering effects on general surfaces. We presented a novel rendering pipeline which provides much needed freedom to the modelling and rendering modules. We implement a fully functional pipeline and experiment with various rendering algorithm. Even though we have a fully functional rendering pipeline, there are multiple points of failure in it. One issue that we haven't addressed well is the parameterization of general surfaces. This functionality is very essential to rendering of general surfaces. Computational difficulty of parameterization of general surfaces means we can't render each and every object. The other issue is computation of normal maps. We have discussed this issue in great detail in our work and illustrated why this problem doesn't have a satisfactory solution till date. One way to overcome this problem is through the use of high resolution 3D scanner. We can use the 3D scan to generate the height map which in turn can be used to generate the normal map. Even though this is a much more accurate technique then the methods presented here, it's a cumbersome technique and can't incorporated into a fully automatic rendering pipeline. The issue of generating good texel values for the vertices comes under the realm of optimization techniques. Even though we have discussed a discrete optimization technique for assignment of texel values to vertices, better optimization formulations are possible. One of the most promising formulations in this scenario includes mass-spring optimization formulation. Due the absence of pre-determined spring strength or mass weights, solving this formulation becomes hard. Despite the several pitfalls of our method, we have made an important step in the right direction.

Bibliography

[1] 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. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 754–761, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1179352.1141951. URL http://doi.acm.org/10.1145/1179352.1141951.

[2] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis, and Hans Køhling Pedersen. Modeling and rendering of weathered stone. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 225–234, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: 10.1145/311535.311560. URL http://dx.doi.org/10.1145/311535.311560.

[3] Mark C Cross and Pierre C Hohenberg. Pattern formation outside of equilibrium. *Reviews of modern physics*, 65(3):851, 1993.

[4] Julie Dorsey and Pat Hanrahany. Modeling and rendering of metallic patinas. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1185657.1185722. URL http://doi.acm.org/10.1145/1185657.1185722.

[5] Jinwei Gu, Chien-I Tu, Ravi Ramamoorthi, Peter Belhumeur, Wojciech Matusik, and Shree Nayar. Time-varying surface appearance: acquisition, modeling and rendering. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 762–771. ACM, 2006.

[6] Wojciech Matusik. *A data-driven reflectance model*. PhD thesis, Massachusetts Institute of Technology, 2003.

[7] Cassidy J Curtis, Sean E Anderson, Joshua E Seims, Kurt W Fleischer, and David H Salesin. Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 421–430. ACM Press/Addison-Wesley Publishing Co., 1997.

[8] Julie Dorsey, Hans Køhling Pedersen, and Pat Hanrahan. Flow and changes in appearance. In *ACM SIGGRAPH 2005 Courses*, page 3. ACM, 2005.

[9] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 165–174. ACM, 1993.

[10] Robert L Cook. Shade trees. *ACM Siggraph Computer Graphics*, 18(3):223–231, 1984.

[11] David S Ebert. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[12] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[13] Greg Turk. *Generating textures on arbitrary surfaces using reaction-diffusion*, volume 25. ACM, 1991.

[14] Andrew Witkin and Michael Kass. Reaction-diffusion textures. In *ACM Siggraph Computer Graphics*, volume 25, pages 299–308. ACM, 1991.

[15] Welton Becket and Norman I Badler. Imperfection for realistic image synthesis. *The Journal of Visualization and Computer Animation*, 1(1):26–32, 1990.

[16] Yanyun Chen, Lin Xia, Tien-Tsin Wong, Xin Tong, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Visual simulation of weathering by γ-ton tracing. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1127–1133. ACM, 2005.

[17] Stephane Merillou, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Corrosion: Simulating and rendering. In *Proceedings of Graphics Interface 2001*, GI '01, pages 167–174, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society. ISBN 0-9688808-0-0. URL `http://dl.acm.org/citation.cfm?id=780986.781007`.

[18] Jinwei Gu, Chien-I Tu, Ravi Ramamoorthi, Peter Belhumeur, Wojciech Matusik, and Shree Nayar. Time-varying surface appearance: Acquisition, modeling and rendering. *ACM Trans. Graph.*, 25(3):762–771, July 2006. ISSN 0730-0301. doi: 10.1145/1141911.1141952. URL `http://doi.acm.org/10.1145/1141911.1141952`.

[19] I Gurappa. Characterization of different materials for corrosion resistance under simulated body fluid conditions. *Materials Characterization*, 49(1):73–79, 2002.

[20] Keshav Choudhary. Rendering of weathered surface. Master's thesis, Indian Institute of Technology Delhi, July 2014.

[21] Robert J Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *22nd Annual Technical Symposium*, pages 136–143. International Society for Optics and Photonics, 1979.

[22] Katsushi Ikeuchi. Determining surface orientations of specular surfaces by using the photometric stereo method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):661–669, 1981.

[23] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224. ACM Press/Addison-Wesley Publishing Co., 1999.

[24] Vilayanur S Ramachandran. Perceiving shape from shading. *Scientific American*, 259(2):76–83, 1988.

[25] Harry G Barrow and Jay M. Tenenbaum. Retrospective on interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 59(1):71–80, 1993.

[26] Ennio Mingolla and James T Todd. Perception of solid shape from shading. *Biological cybernetics*, 53(3):137–151, 1986.

[27] Ruo Zhang, P-S Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8): 690–706, 1999.

[28] Jonathan Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. Technical Report UCB/EECS-2013-117, EECS Department, University of California, Berkeley, May 2013. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-117.html`.

[29] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500. ACM, 2001.

[30] Paul J. Besl and Ramesh C Jain. Segmentation through variable-order surface fitting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(2): 167–192, 1988.

[31] Demetri Terzopoulos. Image analysis using multigrid relaxation methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):129–139, 1986.

[32] CrazyBump software. `http://www.crazybump.com`. Accessed: 2014-06-12.

[33] NeoTextureEdit procedural texture editor. http://neotextureedit.sourceforge.net. Accessed: 2014-06-12.

[34] MaPZone 2.6 editor. http://www.mapzoneeditor.com/. Accessed: 2014-06-12.

[35] Knald. https://www.knaldtech.com/. Accessed: 2014-06-12.

[36] PixPlant software. https://www.pixplant.com/. Accessed: 2014-06-12.