國立交通大學

多媒體工程研究所

碩 士 論 文

由單張影像之最佳化紋理合成

Bidirectional texture synthesis using optimization
from a single image

研 究 生：林展宇
指導教授：林奕成　教授

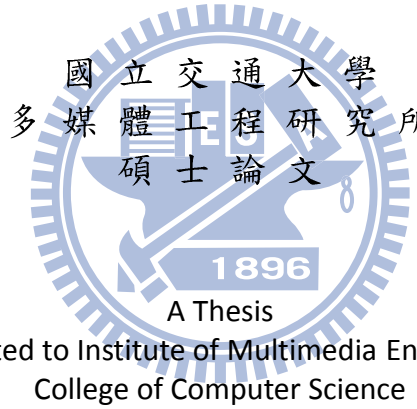
中 華 民 國　九 十 八　年 七 月

由單張影像之最佳化紋理合成

Bidirectional texture synthesis using optimization from a single image

研 究 生：林展宇　　Student：Chan-Yu Lin

指導教授：林奕成　　Advisor：I-Chen Lin



國立交通大學

多媒體工程研究所

碩士論文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

由單張影像之最佳化紋理合成

研究生：林展宇　指導教授：林奕成　助理教授

國立交通大學

多媒體工程研究所

摘要

本論文提出由單張影像利用最佳化的方式，來合成出物體在不同光源位置時，表面的光影變化之情形。我們利用影像之顏色以及反射夾角，得到最佳化之目標函式，進而使用類似最大期望演算法(EM algorithm)，來解此目標函式，得到每個點之顏色。此外，使用者可以指定目標表面的法向量，讓紋理合成在使用者所指定的表面。而我們可利用陰影造形法(shape from shading)來估計影像之法向量。藉由上述方式，我們就能經由單一影像，去合成出目標表面在不同光源下的光影變化。

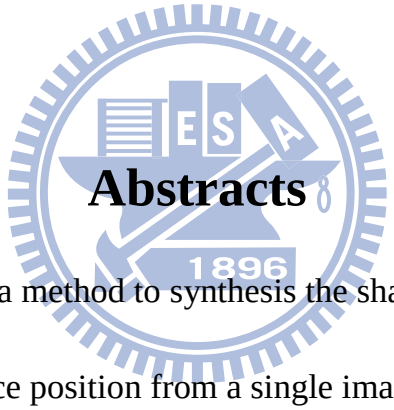
關鍵字：最大期望演算法，陰影造形法，紋理合成

Bidirectional texture synthesis using optimization from a single image

Student: Chan-Yu Lin Advisor: Dr. I-Chen Lin

Institute of Multimedia Engineering

National Chiao Tung University



Abstracts

This thesis proposes a method to synthesis the shading of object surface under different light source position from a single image. We obtain the objective function from image colors and the reflection angles, and then apply an EM-like algorithm to solve this function. Besides, user could specify the normal of target surface which we synthesize. And we use shape from shading to recover the object surface from the image. Finally, we can synthesize a sequence of images in different lighting configuration from a single image.

Keyword: EM algorithm, shape from shading, texture synthesis

致謝

在二年的碩士學習生涯中，受到了許多師長及朋友們的協助，能夠順利的完成論文，我由衷地感謝他們：

1. 首先得感謝我的父母，在學習這條漫長的路上，他們始終給予最大的支持與鼓勵，給予我最大的學習自由，是我最強力的精神後盾及心靈的避風港。
2. 指導教授 林奕成博士，是碩士生涯中的良師兼益友，總是能提出精湛的見解，指引我研究的方向。在兩年的碩士中，不僅教導我研究的專業知識，也讓我學習許多待人處事的道理。
3. 實驗室的各位同學，在課程上的相互討論、課餘的休閒活動，都是令我難忘的回憶。也因為有他們的陪伴，讓我的二年生活過的多采多姿。

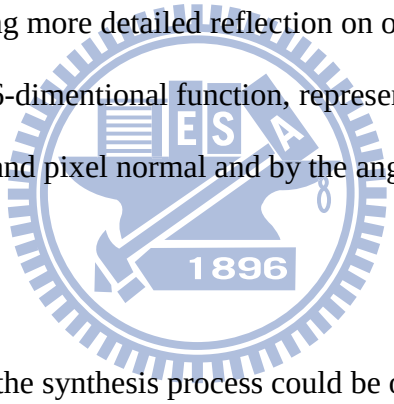
Contents

摘要 I
Abstracts II
致謝 III
Contents IV
1. Introduction 1
1.1 Motivation 1
1.2 Overview 2
2. Related Work 5
2.1 Texture Synthesis 5
2.2 Bidirectional Texture function 6
2.3 Shape From Shading 8
3. Normal Reconstruction 11
4. Optimization Process 18
4.1 Texture Synthesis 18
4.2 Multi-level Synthesis 22
4.3 Synthesis with Reflection Angle 23
4.4 Appearance Consistency 25
5. Results 28
6. Summary & Future Work 31
7. Reference 32

1. Introduction

1.1 Motivation

In order to reproduce the visual realism of the real world, many different texture synthesis methods have been researched and applied in computer graphics, vision and image processing. For instance, pixel-based and patch-based texture synthesis, extracting the colors from input images by comparing the neighbors for visual consistency. For synthesizing more detailed reflection on object surface, Bidirectional Texture Function (BTF), a 6-dimentional function, represent each pixel color by the angle between light vector and pixel normal and by the angle between view vector and pixel normal.



However, the input of the synthesis process could be obtained from a variety of sources such as hand-drawn images or photographs. In order to synthesize realistic reflection on the surface, BTF requires a large database containing thousands of captured photographs from different viewing and lighting configurations as the input. Nevertheless, it is a very difficult and tedious work to obtain such large amount of images even with professional facilities.

We present a simple way to synthesize view-dependent surfaces from lighting directions. The input to our system is only a single image of a nearly convex object, where the surface materials are nearly identical. We assuming the surface is mainly dominated by Lambertian reflection but also with slight self-shadow or subsurface

scattering and we considerate the surface shading based on the normal and the light vector. Furthermore, we solve the view-dependent detail reflection by texture optimization. In order to keep consistency of the images during light source moves, we treat it as texture flow and use a consequent initial guess to keep the pattern consistency.

1.2 Overview

The requirement of the input to our system is very simple; we just use an off-the-shelf camera and take a photo of a nearly single-material object under directional lighting conditions, e.g. under the sun light. We assuming that the surface is a Lambertian-dominant, so that the color of pixels is strong connected to the surface normal vector and the normalized light vector. Then we apply the shape from shading to reconstruct the normal of object surface. Because the input surface is single material, the normal we reconstructed will less affected by noise.

Once we obtain the rough surface normal according to shady information, we could compute the angle between normal and light vector for each pixel in the input image. Also, the angle between surface normal and target light vector could be calculated by $N*L$. According to Lambertian reflectance model, we could use the angles to re-compute the surface color under the target illumination condition to get a new photo. Nevertheless, while simply scaling according to the incident angles, view-dependent properties, such as self-shadowing, subsurface scattering can not be synthesized. Therefore we use the Lambertian-based re-shading image as the soft-constraint in our optimization process. Besides, in order to keep the appearance

consistency of the texture pattern, the similarity of neighboring pixels is another strong constraint in the energy function.

The other issue in this thesis is how to keep the consistency of textures synthesized from different lighting settings. General texture synthesis would synthesize quit different results with different initial guesses. To keep the view-consistency texture, we take the synthesis result of the first illumination configuration as an initial guess in the optimization for other light settings during entire synthesis process.

We apply an Expectation Maximization (EM)-like algorithm to solve the minimization of the energy function [McLachlan and Krishnan 1997]. Through successive iterations with different resolutions, the texture energy will decrease gradually and the result could be refined. For different resolution, different neighborhood sizes will be applied to catch the feature in the input. The large neighborhoods allows large scale pattern to be settle in the output, the smaller neighborhoods is used to refine the texture.

In this thesis, we first introduce researches related to our work in chapter 2, such as texture synthesis and shape from shading. In chapter 3, we combine shape from shading and image segmentation technique to reconstruct a smooth surface normal. The recovering normal will be the constraint during the synthesis process. After normal recovering, we present the texture synthesis by optimization for detail at chapter 4. Experiment results and further discussion will be presented at last two chapters.

Our system can use a successive of optimization process to get a sequence of results under different light position. The following figure is the flow chart of our system:

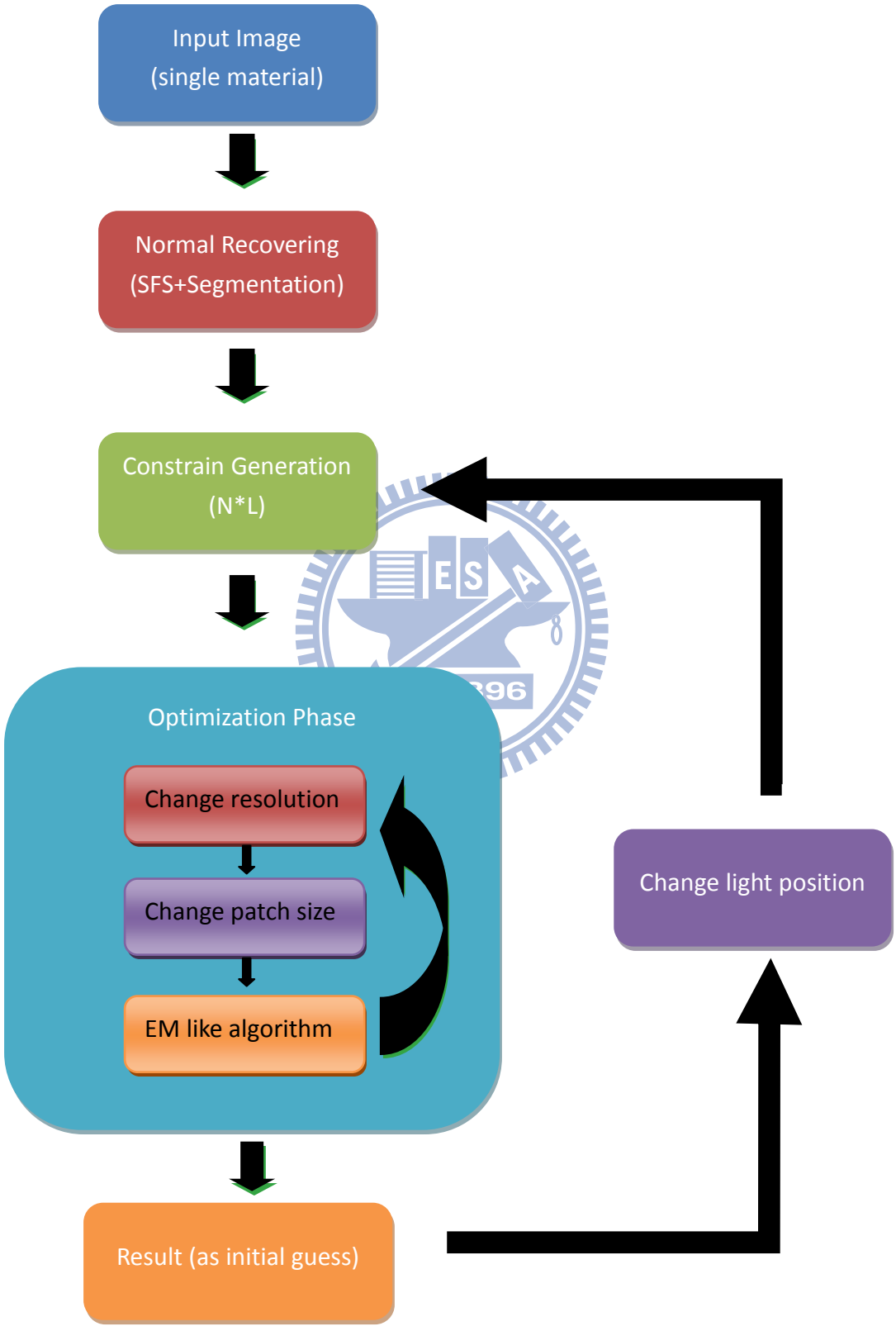


Figure (1): The flow chart of our system

2. Related Work

This goal of this thesis is to avoid the tedious acquisition of the enormous BTF database but keep view-dependent properties as well. We survey researches about BTF. Besides, we apply shape from shading to reconstruct the surface normal as soft constraints and apply texture synthesis to obtain the result.

2.1 Texture Synthesis

Texture synthesis has been widely researched for years; there are a variety of methods toward synthesizing textures. Pixel-based methods are easier to assign constraint on the texture because they synthesize the texture color one pixel at a time by comparing partially synthesized neighborhood with exemplar neighborhood [Efros and Freeman 2001; Kwatra et al. 2003]. In contrast, patch-based methods are amenable to keep the global structure well by synthesizing one patch at a time [Efros and Leung 1999].

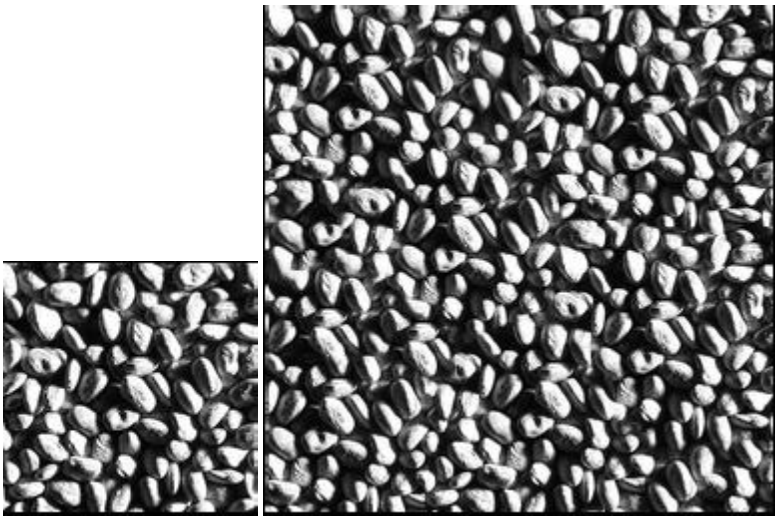


Figure (2): The input is on the left and the output on the right.

Optimization also has been applied to texture synthesis with Markov Random Field (MRF). Paget and Longstaff [1998] use a multi-scale synthesis algorithm incorporating local annealing to obtain larger realizations of texture visually indistinguishable from the training texture.

Besides, there are some researches about texture relighting. Using photometric stereo [Hertzmann and Seitz 2003] to reconstruct the rough normal map, Y-C Shen et al.[2006] relight the object by interpolation similar lighting conditions. They need to take a few photographs for a fixed viewpoint under controlled light. In contrast, our system uses only a single image as input.

2.2 Bidirectional Texture function

The Bidirectional Texture Function (BTF) is a 6D function that can describe textures arising from both spatially-variant surface reflectance and surface details. The BTF captures not only the surface appearance, but also the fine-scale shadows, occlusions, and specularities caused by surface mesostructures. Therefore, it can greatly increase the surface relism in the rendering.

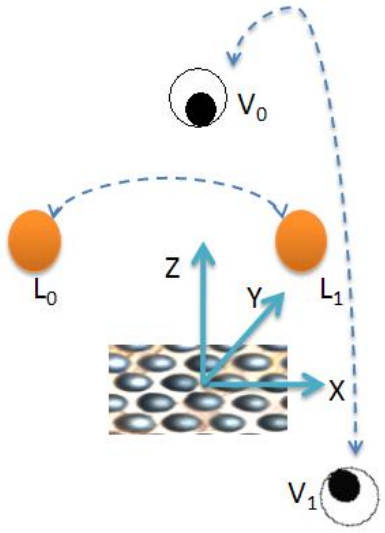


Figure (3): Capturing photos from different viewing and light position.

Surface appearance modeling using the BTF can be roughly subdivided into the following phases[Leung et al. 2007]:

1. BTF acquisition (real or synthetic data),

2. BTF compression,

3. BTF synthesis,

4. BTF rendering.

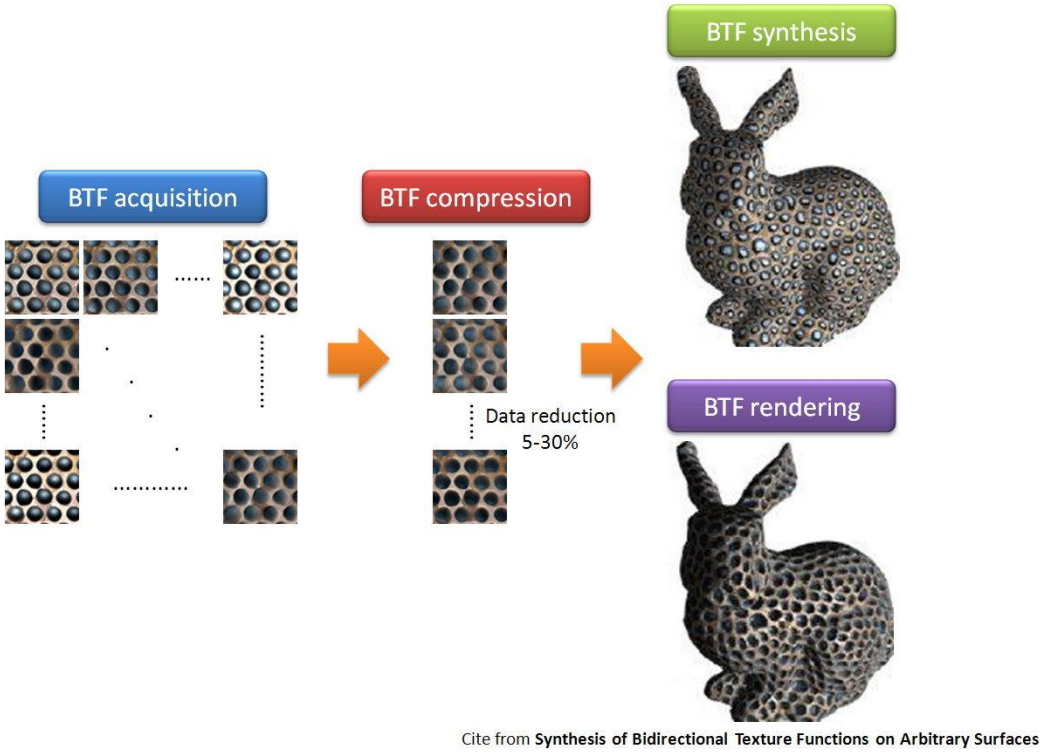


Figure (4): The simple flow chart of BTF process.

BTF acquisition is the capturing and modeling the BTF data from real-world material. It may take hundreds of range cameras to construct a camera array in order to get photograph from different viewing and lighting settings. However, the high dimensionality and enormous storage of the BTF is difficult and computationally expensive for synthesis directly. The PCA method and some multi-linear methods were applied to compactly represent the whole database to reduce the dimension, and this is call BTF compression.

The BTF is directly synthesized with reference to the target geometry before

rendering the data on target surface. If we want to change the appearance with another BTF, we need to re-synthesize the BTF data because the data is tied to the geometry when we first synthesize. The BTF captures not only the surface appearance, but also the mesostructure, it can greatly increase the surface relism in the rendering.

2.3 Shape From Shading

In computer vision, shape from shading is a technique to recover the shape from a gradual variation of shading in the image. A widely used model that relates shape with image information is the Lambertian model, in which the image intensity depends on the light source direction and the surface normal. The formulation for recovering the surface normals was given by Horn[1990] for a wide range of reflectance functions.

In SFS, given a image intensity, the aim is to recover the light source and the surface shape 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 SFS is difficult; it requires additional constraints. Ruo et al[1999] implement and compare six well-known SFS algorithm, such as propagation approach and linear approach.

Propagation approaches start from a single reference surface point, or a set of surface points where the shape either is known or can be uniquely determined (such as

singular points), and propagate the shape information across the whole image.

Given initial values at the singular points (brightest points), the algorithm looks in eight discrete directions in the image and propagates the depth information away from the light source to ensure the proper termination of the process.

Linear approaches reduce the non-linear problem into a linear through the linearization of the reflectance map. The idea is based on the assumption that the lower order components dominate in the reflectance map.

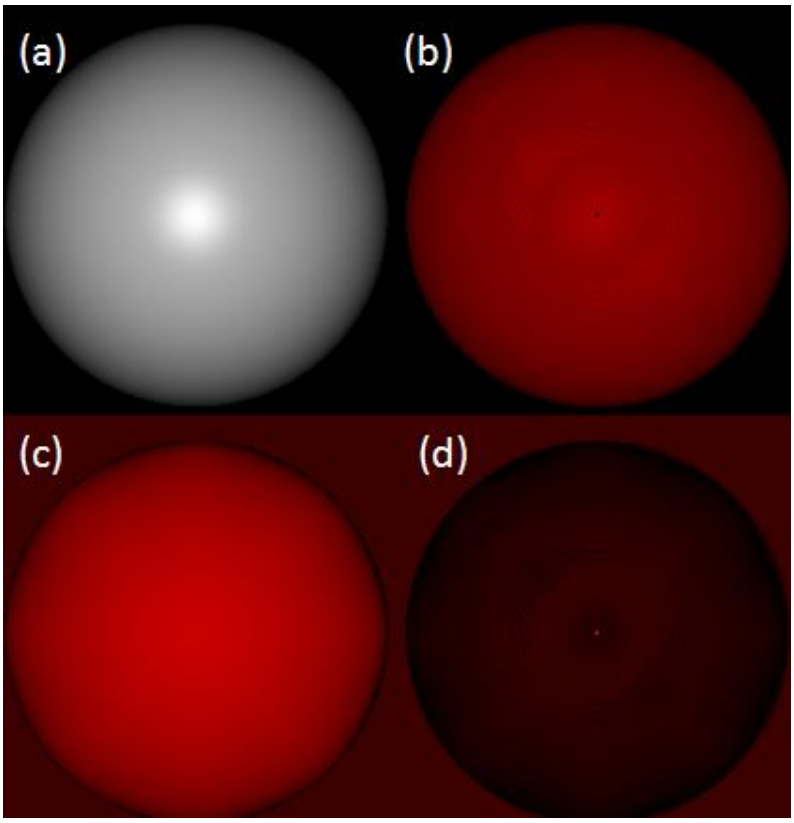


Figure (5): (a)The input image (b)The depth map from linear approach (c)The depth map from propagation approach (d) The difference map of these two

In this thesis, we reconstruct the normal of the surface by the image gradient, the light vectors and the intensity of pixels [Fang, H., and Hart, John C 2004].

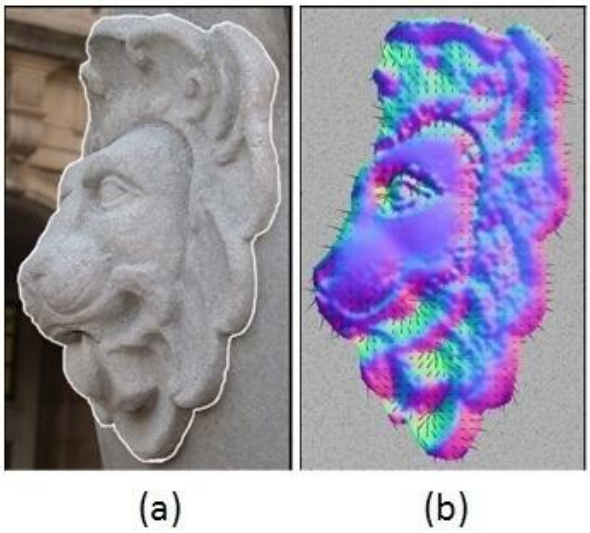


Figure (6): (a) An ordinary photo is scanned in.
(b) Normal map reconstructed from shape from shading.

3. Normal Reconstruction

The goal of this chapter is to reconstruct the normals of the input image for obtaining the relationship between pixel colors and normals. The angle between the estimated normal and the light vector will be the constraint during the optimization process. In order to obtain a smooth surface normal, we first apply image segmentation to the input. There are two important properties in doing segmentation：

1. *Capture perceptually important clusters, which often represent the global feature in the image*. Capturing precise characterizations that is perceptually important, and being able to specify what a given segmentation technique does is two central issues.
2. *Be highly efficient, run time is nearly linear in pixel number.* Segmentation technique could be widely used if the method is highly efficient. For example, it runs several frames per second can be applied to video processing field.

We applied an efficient image segmentation method based on graph-cut proposed by Pedro et al[2004]. It constructs a graph that each pixel is a node in the graph and certain neighboring pixel are connected by undirected edges. Let G= (V, E) be an undirected graph with vertices $v_i \in V$, the elements to be segmented, and edges (v_i, v_i) $\in E$ corresponding to each pair of neighboring vertices. Each edge (v_i, v_i) $\in E$ has a corresponding weight w((v_i, v_i)), which describes the dissimilarity between neighboring vertices v_i and v_i. In our case of image segmentation, the elements in V refer to the pixels in the image and the weight of corresponding edges is the measure of the dissimilarity of two pixels connected by the edge. The edges between two pixels in the same segment should have relatively low weights, while edges between

pixels in different segment should have higher weights.

The segmentation algorithm is based on the dissimilarity among neighboring pixels within each of the two segments. The *internal difference* of a region $R \subseteq V$ to be the largest weight in the minimum spanning tree of the region, *MST*(*R*,*E*). That is

$$Int(R) = \max_{e \in MST(R,E)} w(e)$$

The other measurement describe the *difference between* two regions $R_1, R_2 \subseteq V$ to be the minimum weight edge connecting the two regions. That is

$$Dif(R_1, R_2) = \min_{v_i \in R_1, v_j \in R_2, (v_i, v_j) \in E} w((v_i, v_j))$$

Then we could use these two kinds of difference to decide whether the regions could be merged or not by the following pair wise comparison,

$$D(R_1, R_2) = \begin{cases} true & if \quad Dif(R_1, R_2) > MInt(R_1, R_2) \\ false & otherwise \end{cases}$$

where the minimum internal difference , *Mint*, is defined as,

$$MInt(R_1, R_2) = \min(Int(R_1) + \tau(R_1), Int(R_2) + \tau(R_2))$$

τ is a threshold function based on the size of the region,

$$\tau(R) = k / |R|$$

where $|R|$ denotes the size of R, and k is a constant parameter. In practice k sets a scale of observation, a larger k will cause larger region segmentation. The threshold function controls the degree to which the difference between two regions must be greater than their internal difference in order to be evidence of a combination them or not. Figure (7) simply shows how the segmentation works.

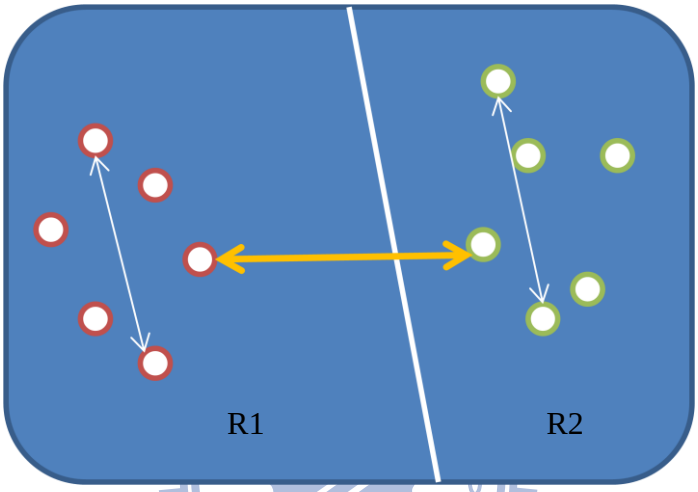


Figure (7): The white arrows in the figure represent $Int(R_1)$ and $Int(R_2)$. While the orange arrow is the $Dif(R_1, R_2)$. We can tell that the orange arrow is longer than the two white arrows, that is to say R_1 and R_2 can not be merged.

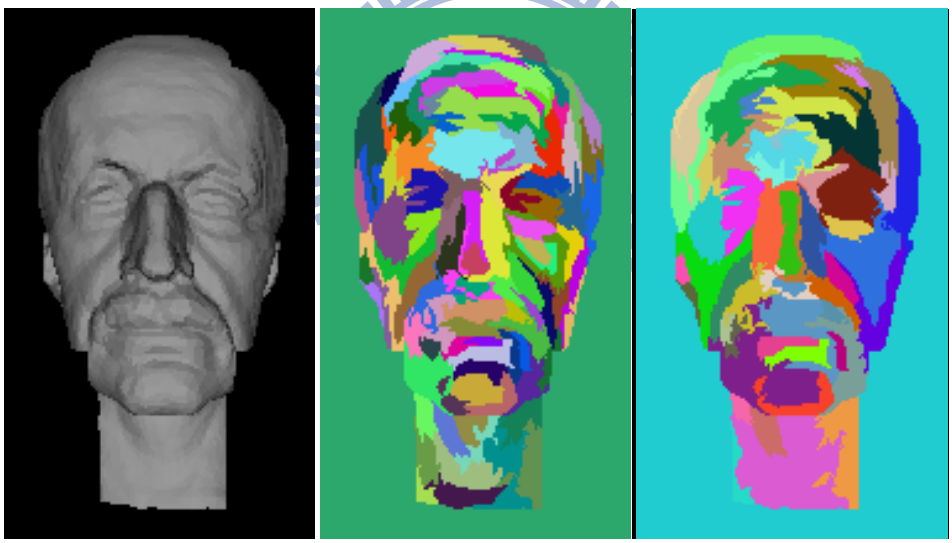
The input is a image, we take it as a graph $G=(V,E)$, each pixel represent a vertex in V and any two pixels form a edge in E. Assuming there are n vertices and m edges in G. The output is a segmentation of V into regions $S = (R_1,\ldots,R_r)$. The following is the algorithm,

0. Sort E into into $\pi=(o_1,\ldots,o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Construct S^0 given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering , i.e., $o_q = (v_i, v_i)$. If v_i and v_i are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of

both those components, then merge the two components otherwise do nothing. More formally , let R_i^{q-1} be the component of S^{q-1} containing v_i and R_i^{q-1} the component containing v_j. if $R_i^{q-1} \neq R_j^{q-1}$ and $w(o_q) \leq$ Mint(R_i^{q-1}, R_j^{q-1}) then S^q is obtained from S^{q-1} by merging $R_i^{q-1} and\ R_j^{q-1}$. Otherwise . $S^q = S^{q-1}$.

3. Repeat step 2 for q=1,…,m.
4. Return $S = S^m$

We could get the segmentation result after the algorithm. Figure (7) and Figure (8) show the segmentation results from synthetic model and real objects.



(a) (b) (c)

Figure (8): [Synthesis data]
(a)The input image, (b)The segmentation result with k=150
(c) The result with k=500.

Figure (9): [Real data]
Left top is the input image.
Right top is the segmentation result with k=150.
Left bottom is the result with k=500.

After the segmentation, we could recover a more stable normal of the input image. Horn [1990] presents the formulae for recovering the shape of a surface from a shaded image which can deal with complex, wrinkled surfaces. And the following simple Lambertian reflectance model works well for our purposes. Let *S* be the unit vector from the center of the object toward a sufficiently distant point light source. We further assume the point on the surface with largest intensity *Imax* towards the light source position. The darkest point is shadowed and its intensity *Imin* indicates the ambient light in the scene. The function $c(x,y) = (I(x,y)-Imin)/(Imax-Imin)$ estimates the cosine of the angle between incidence and normal. We could reconstruct the surface normal by the cosine angle and the image gradient.

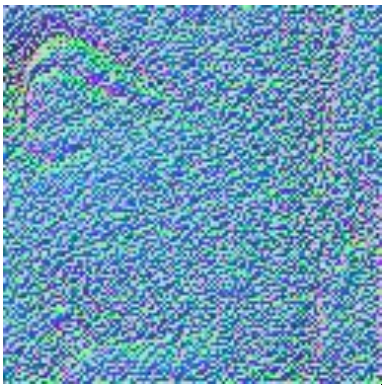
$$G(x,y) = \nabla I(x,y) - (\nabla I(x,y) \cdot S)S$$

$$N(x,y) = c(x,y)S + s(x,y)G(x,y)/\|G(x,y)\|$$

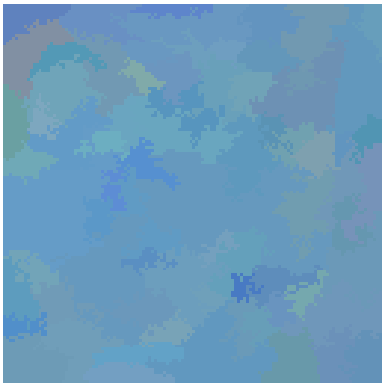
where $\nabla I(x,y) = (\partial I/\partial x, \partial I/\partial y, 0)$ is the image gradient.

Because the reconstructed normal is rough and easily affected by noise, we use the segmentation result to get a smooth surface normal. We compute the average normal of each region and assign to the center of the region. Then we could interpolate the normal of each pixel by the distance between specified pixel and each region center. That is, the closer the region is the higher weight it has. The following figure shows the difference of interpolation or not. Figure. 9 shows the normal map of different methods.

(a)

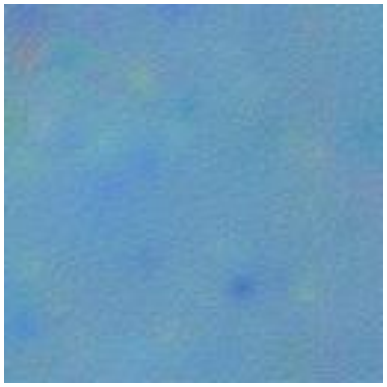


(b)



(c)

(d)



(e)

Figure (10): (a)The input image
(b)Rough normal from SFS
(c)Grouping normal by the segmentation result
(d)Interpolating pixel normal
(e)The interpolated normal map from (b) and (d)

4. Optimization Process

In this chapter, we will describe how our synthesis process works and how to keep the surface appearance consistency. The first section is about basic texture synthesis using optimization. Then we present how we consider normal as a soft-constraint during the optimization. The last section is about how to keep the appearance consistency.

4.1 Texture Synthesis

The input to our system is just a single image; we use this image to perform basic texture synthesis using optimization based on Kwatra et al [2004]. According to the Markov Random Field (MRF) property of texture, *locality* and *stationary* is two major issues. The color of pixel is only related with the color of its neighboring pixel that is *locality*; *stationary* implies that the locality property is independent with the actual position of the pixel, it means no matter what location the pixel is, its color is only depend on its neighboring pixels.

Now we would like to describe the similarity of the input image and the synthesized output based on the MRF theory mentioned above. Base on the assumption that the synthesized output will perceptually similar to the input image if all the patches in the output be similar to some patches in the input image. The objective function could be defined by the similarity between input patches and output patches. Let Z be the input image which we want to capture the color information and X be the output we want to synthesize. We could concatenate the value of all the pixels in X to form a vectorized version of X, x. Then let N_p be the patch which is centered at p in the output, and its corresponding vector is the sub-vector of x, x_p.

Under the same definition, z would be a vectorized version of input Z, and z_p is a sub-vector of z. z_p is the most similar patch corresponding to x_p under the Euclidean norm.

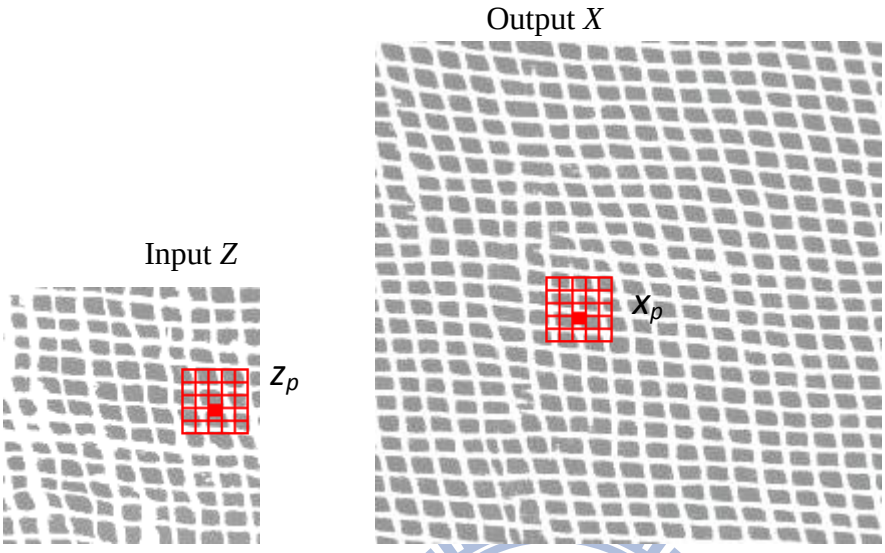


Figure (11): x_p is the patch centered at pixel p, and z_p is the closest corresponding patch in the input.

The objective function could define as:

$$O(x;\{z_p\}) = \sum_{p\in X^+} \|x_p - z_p\|^2 \qquad (1)$$

If we consider the patch formed by all the pixels in the image, it would be very computationally expensive and inefficient in practice. So selecting a subset of the patches whose center is located at $X^+ \subset X$ is more efficient. The elements in X^+ are choosing by pixels in the output which is $w/4$ apart and w is the patch size. The patches we choosing would be sufficiently overlapped to avoid sampling not enough. Besides, choosing patches in the sparse grid manner could prevent too many patches affect one pixel simultaneously getting the synthesized output too blurry.

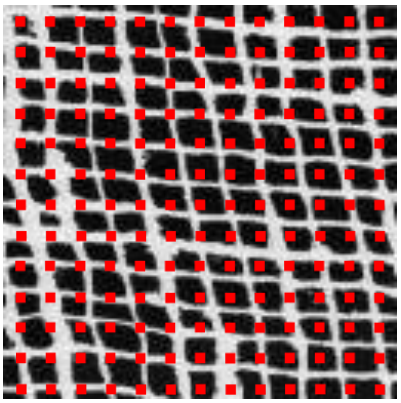


Figure (12): The figure shows how we spread the sample point. Each center is $w/4$ apart and w is the patch size.

Using the formula mentioned above, we could refine the synthesized output progressively. If the initial status of x is unknown, there are two ways to do the initial guess. First, we could full the output resolution with random color; the other is assigning random patches to each x_p. At each iteration, we treat x and z_p as variable at different phase. Given an initial guess of x, we could find the corresponding z_p that is closest to each x_p. Then we could update the value of x according to the corresponding patches. Once we update the value of x, the patch set that contain the corresponding z_p of x may change. So we have to iteratively repeat these two steps until it coverage. When the patch set of z_p at current iteration is equal to the patch set of z_p at previous iteration, the optimization process is complete.

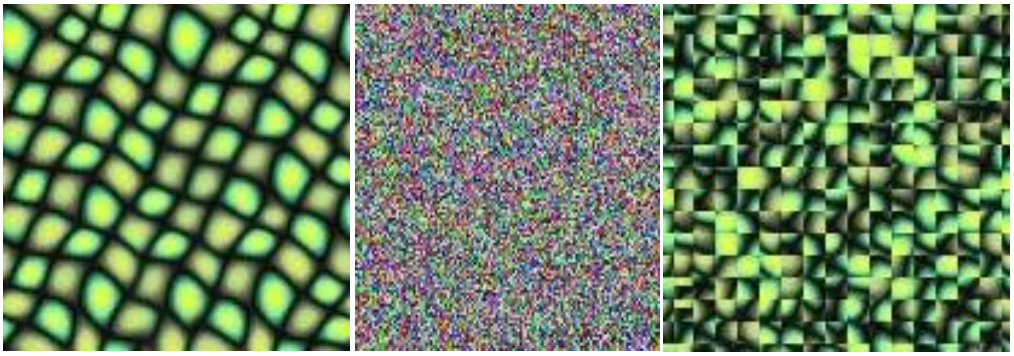


Figure (13): Left: input image, Middle: random initial guess, Right: patch initial guess.

An expectation-maximization (EM) algorithm is used in statistics for finding

maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. EM is an iterative method which alternates between performing an expectation (E) step, which computes an expectation of the log likelihood with respect to the current estimate of the distribution for the latent variables, and a maximization (M) step, which computes the parameters which maximize the expected log likelihood found on the E step. These parameters are then used to determine the distribution of the latent variables in the next E step.

In our synthesis case, we applied an EM like algorithm to solve the objective function in (1) [McLachlan and Krishnan 1997]. At E step, we want to get an estimation of each pixel in the synthesized output by minimize the error function in (1). This could be archive by setting the derivative of (1) with respect to x to zero. The outcome is a linear system that can be solve for x. As mentioned above, each term in (1) is a patch similarity error between the input and the output and the summation can be treated as the global error. In the other words, the goal of the minimization is pulling the pixels in x_p towards the pixels in z_p. Because pixels in X may exist in more than one patch in x_p, the corresponding z_p would simultaneously affect the value of specified pixel. Through E steps, we could get the color of each pixel in X by averaging its corresponding pixel in z_p from the input image.

The M step keeps the x value updated from E step as constants, treat z_p as valuables to minimize (1). After E step, a new set of x_p could be obtained from the estimation result, x. The minimization needs to find out the closest z_p for each x_p. We could compute the difference patch by patch to get the closest patch we want. But obviously, using brute-force to do this search process is very inefficient and time consuming. To accelerate the searching problem, we simply perform the *k-means* clustering, one of the widely used and intuitive algorithm. After defining k cluster

centers (k=4), we have to take center of patch in z_p and associated it to the nearest centers. Base on the new clusters, we could re-compute each cluster center and re-associate each pixel. A loop has been generated. The clustering process will stop when no more changes are done. Once the clustering is complete, we could perform the search process according to the clustering result to accelerate the optimization performance.

4.2 Multi-level Synthesis

In order to refine the synthesized result, we applied the multi-level scheme to the optimization process. The synthesis process is starting from a lower resolution. At coarse resolution, the feature patterns on the input are spatially close to each other, it is easier to keep the appearance consistency through the desired output. The result at lower resolution then serves as the initial guess at higher resolution via interpolation. During each resolution, a specific patch size is applied to the optimization process in order from largest to smallest. Larger patch could capture image structure first, then the smaller patch removes the fine scale errors to refine the output. In our case, we use three resolutions and three patch size of 33*33, 17*17, 9*9 at each resolution.

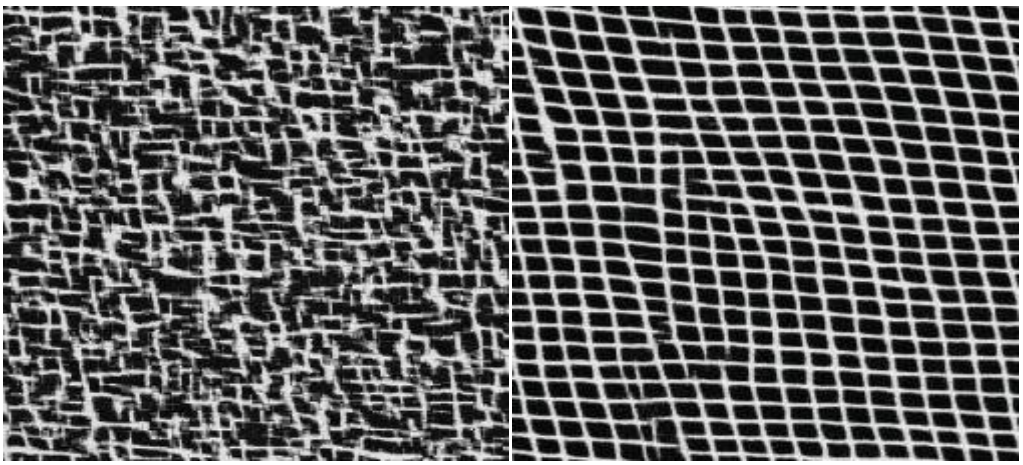


Figure (14): Left: the result without multi-level. Right: the result with multi-level.

4.3 Synthesis with Reflection Angle

Controllable synthesis technique could be applied in our case as the soft-constraint. Kwatra et al [2004] provide a modulation of objective function with appropriate weights allows for additional control over individual pixel values. We propose adding reflection parameters in the objective function as the constraints. For example, if certain color is desirable at specified location, we can express the control term as:

$$O_c(x) = \sum_{m\in\delta}\left(x(m) - x^c(m)\right)^2 \qquad (2)$$

where δ is a set of specified pixels and x^c is a constrain vector that contains desirable color and reflection informations. If we combine equation (2) into equation (1), it would be:

$$O(x) = \sum_{p\in X^+}\left\|x_p - z_p\right\|^2 + \lambda\sum_{m\in\delta}\left(x(m) - x^c(m)\right)^2 \qquad (3)$$

where λ is a relative weighting coefficient. The former term in (3), tries to keep the image pattern through the similarity comparison of neighboring patches. While the control attempts to meet certain control criteria in the synthesized output.

In our case, we try to synthesize the input image on a given reflection angle. Not only keep the structure from input, but also the shading variation through the predefined normal. We assuming the surface is dominated by Lambertian reflection so that we could considerate the surface shading based on the normal (*N*) and the light vector (*L*).

$$I_D = N \cdot LCI_L$$

where I_D is the surface brightness, C is the surface color and I_L is the intensity of incoming light. Because

$$N \cdot L = |N||L|cos\alpha$$

where α is the reflection angle between normal and light vector. The intensity will be the highest if the normal vector in the same direction as the light vector (cos(0) = 1, the surface will be parallel to the direction of the light), and the lowest if the normal vector is perpendicular to the light vector (cos(π/2) = 0, the surface runs perpendicular with the direction of the light).

With a given output surface normal and target light source vector, we can compute the reflection angle between normal and light for each pixel in the synthesized output, x_θ. We can also reconstruct the surface normal of the input image and compute the reflection angle between the normal of image and input light source vector, z_θ. Then we can use x_θ.and z_θ to get a constrain map.

For each pixel p in X, we can find a corresponding pixel q in Z that the reflection angle p_θ is closest to the reflection angle q_θ. Then the constraint map can be simply obtained by pasting the color of q with the resolution equal to the synthesized output. Base on the assumption that the surface is dominated by Lambertian reflection, we could obtain a simple soft-constraint. Then the detail view dependent properties , such as self-shadowing, subsurface scattering can be synthesized through the texture optimization.

In the E step, we solve the differential of (3) with respect to x. For the normal constrain mentioned above, solving the differential problem is corresponding to compute the weighted average of the constraint map and those corresponding patches.

While in the M step, we search the closest patch z_p depending on not only the Euclidean norm but also the similarity of the normal. Considering the Euclidean norm between patches can capture the surface patter well. In order to keep the shading variation on the surface is smooth, we have to consider with the criteria about normal.

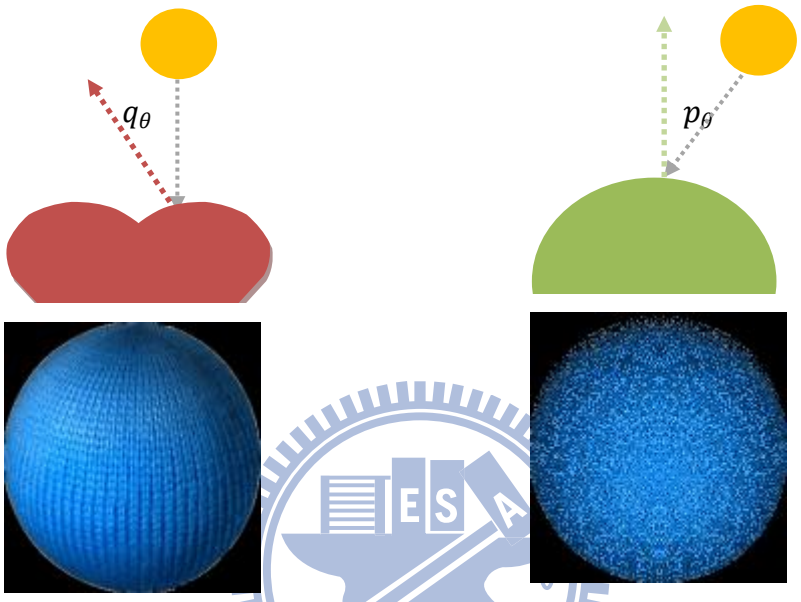


Figure (15): Left is the input image with q_θ. With the given sphere normal, we can compute the p_θ. Right is the constrain map.

4.4 Appearance Consistency

Texture synthesis could keep the feature pattern of repeating element on the sample and generate the output that is larger in size than the input sample but perceptually similar to it. Although the outcome would be perceptually similar to the input, but the results would be different through different synthesis phases. That is to say, even the same program, the synthesized result from first time experiment and the result from second time experiment wouldn't be perceptually the same. If we want to simulate the shading variation on the given surface normal through different light position, we have to keep the appearance consistent. Then we can tell whether the

change in brightness over the surface is correct or not perceptually.

We want to maintain spatial and temporal continuity to keep the appearance consistency. So we can treat this problem as texture flowing control. Kwatra et al.[2003] achieve texture flowing by warping the result of the previous frame with given vector field. The warped image denotes the soft-constraint during the optimization in current frame.

Let L denotes the light position which we want to simulate the shading variation. L is composed by a sequence of 2D synthesized output, (L_1,L_2,…,L_{p-1}). Where p is the number of light source position. The shading variation is treated as shading flow and L_i is the desirable shading flow between light position i and light position i+1. Given the color of pixel p at synthesized output under light position i, we can get the color of p under light position i+1 after shading through L_i (p).

Figure (16): This figure shows how we take synthesized result as soft-constraint at next optimization process.

Let X_i and X_{i+1} denote two synthesized results from different lighting configurations. Each pixel p in X_i could be related to each pixel q in X_{i+1} through L_i. Hence, the control term in the objective function can be defined as:

$$O_c(x;L) = \sum_{i\in 1:L-1} \sum_{p\in X_i} (x(p,i) - x(q,i+1))^2 \qquad (3)$$

We apply basic texture synthesis to get a first synthesized result of the sequence, L_1. Then the later optimization process will base on the constraint texture synthesis that using previous results as soft-constraint.

5. Results

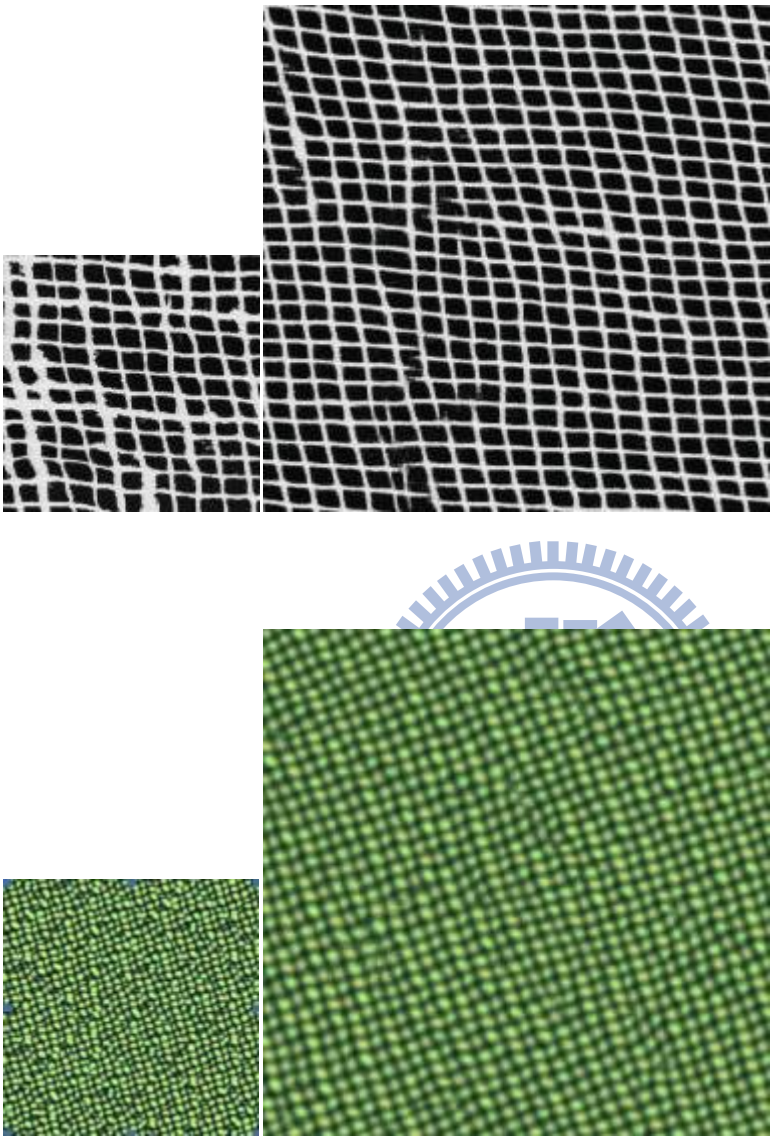


Figure (17): Result from basic texture synthesis by optimization

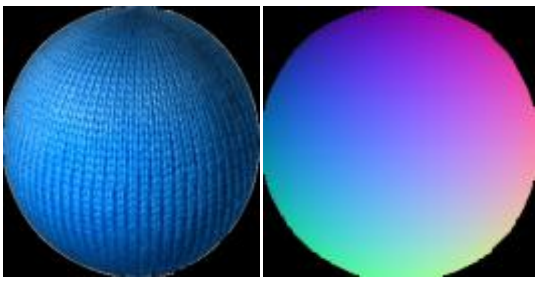


Figure (18): Left is the input image. Right is the given sphere normal.

Figure (19) and Figure (20) show the results of different movement of light source with appearance consistency.

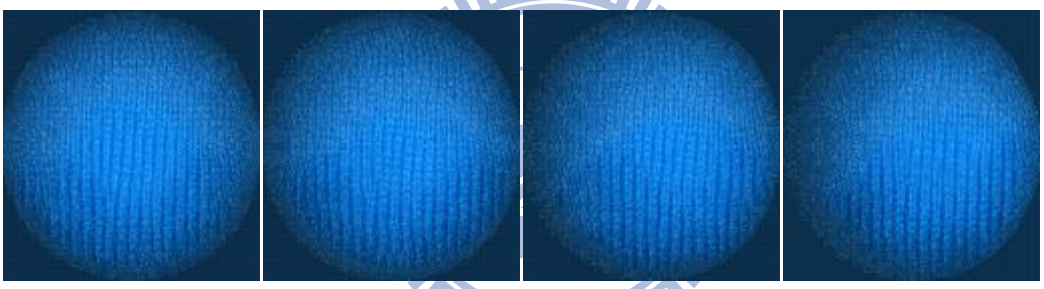


Figure (19): Light position moves from left to right.

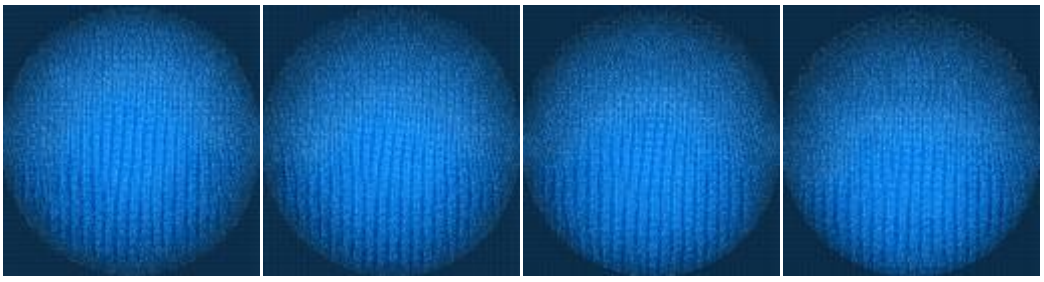


Figure (20): Light position moves from right to left

Figure (21) show the results of smaller patch size during optimization.(5*5)

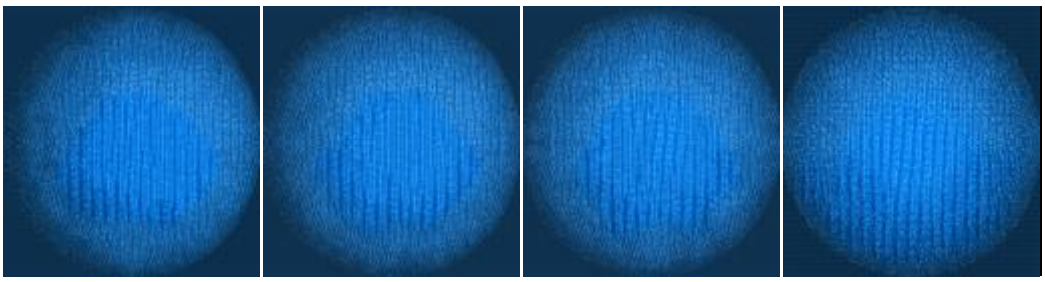


Figure (21): Light position moves from left to right with smaller patch size

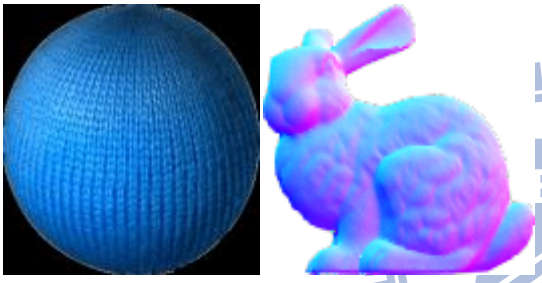
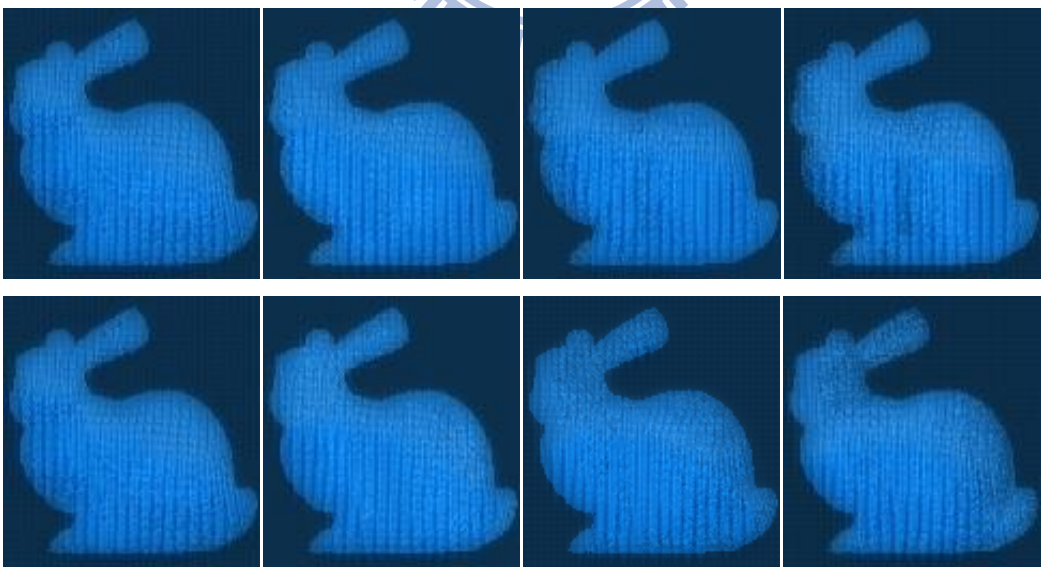


Figure (22): Left is the input image. Right is the given sphere normal.



6. Summary & Future Work

We present a simple way to simulate a given surface normal with texture from the input image under different lighting positions. Base on the assumption that the surface is mainly dominated by Lambertian reflection with slight self-shadow or subsurface scattering, we generate the constraint map from the angle between the normal and the light vector. Then the optimization is applied with the constraint map to synthesis the output. We also show how to keep the appearance consistency by the previous synthesized result. A previous result serves as the constraint map that keeps the appearance of current result consistent.

There are several issues that might improve our results. We generate the constraint map base on the normal and light vector. If the recovering normal is more specific, the synthesized output will preserve more details and shading variations. A more robust normal reconstruct method may apply to our thesis. Besides, when applying multi-resolution synthesis, we set the patch sizes at constant. Sometimes it can not capture the feature of the image well. Techniques about feature detection can help us with more precise patch size that obtains the surface feature well. For the future work, we want to generate a database like BTF database that containing different viewing and lighting configuration from a single image. More issues besides image color and reflection angle should be considerate to reach that goal.

7. Reference

Debonet, J. S. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. *Proceedings of ACM SIGGRAPH 97* (August), 361–368.

Debevec, P. Image-based lighting, 2001.

Efros, A., amd Leung, T. 1999. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, 1033–1038.

Efros, A. A., AND Freeman, W. T. 2001. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, 341–346.

Fang, H., AND Hart, John C. 2004. Textureshop: Texture Synthesis as a Photograph Editing Tool, Proc. *SIGGRAPH 2004*.

Horn, B. K. 1990. Height and gradient from shading. 1990. International journal of computer vision, 5:1,, 37-75, 1990.

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., AND Salesin, D. H. 2001. Image analogies.*Proceedings of SIGGRAPH 2001* (August), 327–340. ISBN 1-58113-292-1.

Hertzmann , A. AND Seitz, S. M.. Shape and materials by example: A photometric stereo approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, 533–540, 2003.

Kwatra, V., Sch¨odl, A., Essa, I., Turk, G., AND Bobick, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July), 277–286.

Kwatra, V., Essa, I., Bobick, A., AND Kwatra, N. 2005. Texture Optimization for Example-based Synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*

Lefebvre, S., AND Hoppe, H. Appearance-space texture synthesis. 2006. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3), 541-548.

Leung MK, Pang WM, Fu CW, Wong TT, AND Heng PA. 2007. Tileable BTF. *IEEE transactions on visualization and computer graphics* 13(5):953-65, 2007

Mclachlan, G., AND Krishnan, T. 1997. *The EM algorithm and extensions*. Wiley series in probability and statistics. JohnWiley & Sons.

Paget, R., AND Longstaff, I. D. 1998. Texture synthesis via a noncausal nonparametric multiscale markov random field. *IEEE Transactions on Image Processing 7*, 6 (June), 925–931.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2004. Efficient Graph-Based Image Segmentation. International Journal of Computer Vision, Volume 59, Number 2, September 2004

Ruo Z. P-S Tsai. James EC And Mubarak S. 1999. Shape-from-shading: a survey.

IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume 21, 690-706

Y-C Shen, W-C Ma, Y-Y Chuang, B-Y Chen, AND M Ouhyoung. 2006.Coherent Texture Synthesis for Photograph Relighting and Texture Transfer. *Proceedings of 2006 Workshop on Computer Vision & Graphic Image Processing*, Taoyuan, Taiwan, 2006.