Term Project Proposal
Normal Map Estimation using Model Geometry and Texture
Team: Avengers

Jang Wonjong, Lee Dahun, Jacob Morton
20140337, 20130221, 20172327
Computer Science and Engineering, POSTECH

1. Introduction

1.1. Motivation

It is an easy task to create a models geometry and texture. There are plenty of 3D modeling software available, like 3D Studio Max, Maya, and blender. It is not too difficult to program or load these into OpenGL as well. The problem of using only 3D geometry and Texture alone is that it can produce flat looking renderings, especially if the geometry is less detailed than the texture. This is typically the case, as texturing can hide simple geometry and flaws. There is more we can do with texture mapping than just wrapping the geometry with an RGB image. We can also add bumb maps, normal maps, displacement maps, and even subsurface scattering to make the rendering more realistic. But these additional mappings are difficult to produce and are often hand-made or produced in multi-step procedures available in some 3D modeling software. Adding tuned normals to the texture can improve the quality of 3D reconstructed scenes and objects while being illuminated by a light source. We wish for there to be a simple one-step procedure that can take an RGB picture and produce a high-quality normal map that takes into account lighting information.

1.2. Project Description

Inspired by Shape-From-Shading, we propose a tool for estimating a normal map from a given texture. We will use an RGB-D to capture both the RGB image and initial surface normals. We will use off-the-shelf Intrinsic Image Decomposition to estimate the albedo texture and removing shadows. We will first use Spherical Harmonics to estimate the light source direction in the given estimated texture and RGB-D normals. This lighting direction will be used with SH to compute a per-pixel normal map based on the diffuse surface illumination of the texture. We can then apply this normal map in addition to the model and texture during

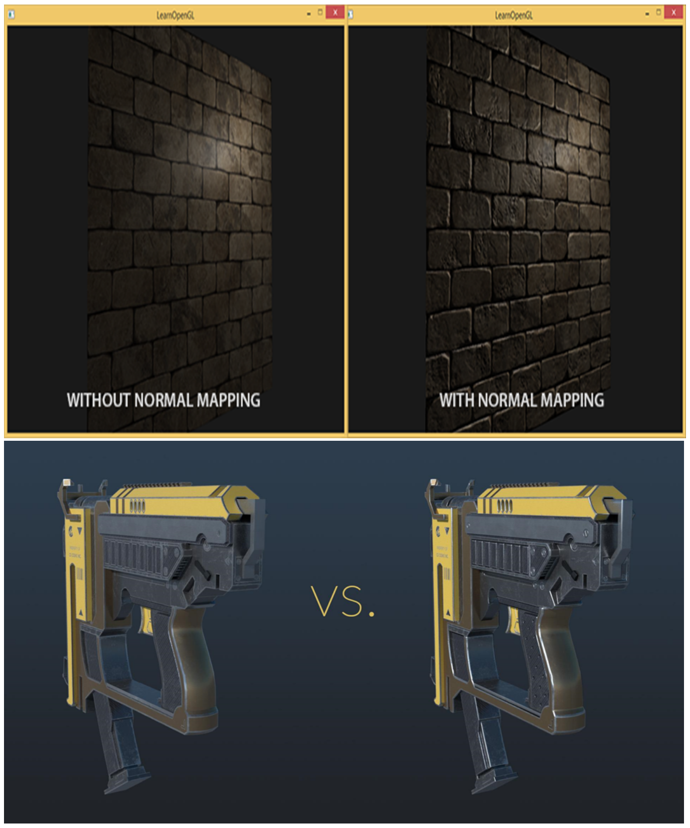

Figure 1. Comparison between **Left:** Geometry + Texture, **Right:** Geometry + Texture + Normal Map

the texturing process. We will demonstrate this by using a Fragment shader in OpenGL.

1.3. Background

This project will require knowledge in Spherical Harmonics, least-squares optimization, normal mapping in OpenGL using fragment shaders. Using lighting to understand surface topology is very common in 3D Reconstruction and is also something we humans do when viewing an object under lighting. The amount of energy a surface receives is proportional to the angle of a surface to a light source, which is related to the surface normal. We can use Spherical Harmonics (SH) to measure this energy at each pixel on an image and determine the pixel's normal vector. SH often assumes diffuse surface illumination and there may be some error in spectral illumination and self-shadowing. Solving for normals and lighting is very similar to SFS, intrinsic image decomposition, and environmental lighting.

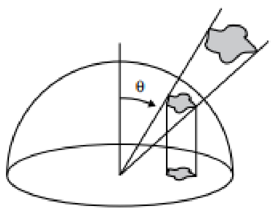

Figure 2. light energy illuminated on a surface is proportional to cosine between the surface normal N and the unit vector towards the light source L

2. Development Environment

Operating System:	Windows 10
IDE:	MS Visual Studios 2017
Libraries:	OpenGL, FreeGlut, GLEW, GLM, Ceres Solver, Eigen

3. Research and Development Plan

3.1. Research

Some time will be needed to understand and implement Spherical Harmonics (SH) for light source estimation. There are plenty of resources available since SH is widely used in the gaming industry for environment illumination and for 3D reconstruction. When estimating an environments illumination, SH uses a defined light source to compute the energy of a surface's illumination, which is the inverse of our problem. Also normal map estimation is usually an intermediary step in Shape-From-Shading (SFS), a popular 3D Reconstruction technique. Computing a generic normal map from an image is difficult. Assumptions on the geometry and diffuse surface reflectance (albedo texture) is needed. For face reconstruction, PCA models can provide both a normal map estimation from a generic face or fitted model, as well as for the albedo. With a RGB-D camera we can use the captured noisy normal map as starting point as well. Using and RGB-D camara for normals may be less accurate than aquiring normals from the geometry, but much easier. We plan on using an RGB-D because it is available and ease of use. The noisy normals will be refined during fitting. We refer to the initial normal map as N_{ref}. We also need a good approximation of the albedo texture ρ_{ref}, but it is difficult to aquire as we need an image without shading. We will pre-process the RGB image to remove shading using widely available open-source Intrinsic Image Decomposition. We will then Solve the least-squares problem to estimate lighting parameters using SH and given N_{ref} and ρ_{ref}. After lighting has been estimated, we will solve for the normal map by refining N_{ref} to produce N using SH, the found lighting parametes, and ρ_{ref}. We will render the resulting normal map N in OpenGL using a fragment shader.

3.2. Development Plan

Our plan is to separate the development tasks so we can work as a team, with equal contribution. The Project will be split up into 5 tasks: SH illumination estimation, Normal map generation from illumination, and rendering

3.3. Schedule

See Table 1.

Date	Task
11/03 - 11/21	Research, Understand, test examples
11/21 - 12/12	Implementation
12/12 - 12/21	Integration and Demo

Table 1. Schedule

3.4. Team Member Roles

See Table 2.

Name	Roles
WonJong	Intrinsic Image Decomposition + SH Light Source Estimation
Dahun	Render Model, Texture, and Normal Map
Jake	RGB-D Normal Map extraction + SH Normal Map Estimatio

Table 2. Team Member Roles

References

[1] R. Green. *Spherical Harmonic Lighting: The Gritty Details*. GDC 2003.

[2] Rich Forster. *Spherical Harmonics for Beginners*. https://dickyjim.wordpress.com/2013/09/04/spherical-harmonics-for-beginners/

[3] Igor Goldvekht. *Shape-From-Shading*. https://github.com/IgorGee/Shapes-From-Shading