Satish Kumar Rajagopal

3D Texture Reconstruction from Multi-View Images

Master’s Thesis
Degree Programme in Computer Science and Engineering
May 2017
ABSTRACT

Given an uncontrolled image dataset, there are several approaches to reconstruct the geometry of the scene and very few for reconstructing the texture. We analyze two different state of the art fully integrated texture reconstruction frameworks to generate a textured scene. Shan et al.’s approach [1] uses a shading model inspired by Computer Graphics rendering to formulate the scene and compute the texture. Texture is stored as albedo reflectance parameter per vertex for each color channel. Waechter et al. [2] uses a two-stage approach, first stage where a view is selected for each face and second stage where global and local adjustments are performed to smooth out seam visibility between patches. Both approaches have their own occlusion removal stage.

We analyze these two drastically different approaches under different conditions. We compare the input images and rendered scenes from the same angle. We discuss about occlusion removal in an unconstrained image dataset. We modify the shading model proposed by Shan et al. to solve for a controlled indoor scene. The analysis shows the advantages of either approaches on specific conditions. The patch based texture reconstruction provides a visually appealing scene reconstructed in a considerable time. The vertex based texture reconstruction has a complex model providing us the framework to solve for lighting and environment conditions under which the images are captured. We believe that these two approaches provide fully integrated frameworks that reconstruct the scene for both geometry and texture from an uncontrolled image data set despite all the inherent challenges in a reasonable time.

Keywords: texture modeling, shading model, structure from motion, multi-view stereo
# TABLE OF CONTENTS

ABSTRACT
TIIVISTELMÄ
TABLE OF CONTENTS
FOREWORD
ABBREVIATIONS

1. INTRODUCTION ........................................................................................................... 6
2. 3D SCENE RECONSTRUCTION .................................................................................. 8
   2.1. Feature Extraction ................................................................................................. 9
       2.1.1. SIFT .............................................................................................................. 9
       2.1.2. SURF ........................................................................................................ 10
   2.2. Feature Matching .................................................................................................. 11
   2.3. Structure from Motion ........................................................................................ 12
       2.3.1. SfM Model ................................................................................................. 12
       2.3.2. Incremental SfM ...................................................................................... 14
   2.4. Multi-View Stereo .............................................................................................. 15
   2.5. Surface Reconstruction ....................................................................................... 16
3. TEXTURE RECONSTRUCTION .................................................................................... 17
   3.1. Vertex based Texture Reconstruction .................................................................. 19
       3.1.1. Preprocessing ............................................................................................. 19
       3.1.2. Shading Model ........................................................................................... 19
       3.1.3. Detection of Cloudy Images ....................................................................... 20
       3.1.4. Algorithm ................................................................................................... 21
   3.2. Patch based Texture Reconstruction .................................................................. 24
       3.2.1. Baseline Method ......................................................................................... 24
       3.2.2. Preprocessing ............................................................................................. 25
       3.2.3. View Selection ............................................................................................ 26
       3.2.4. Color Adjustment ....................................................................................... 27
4. EXPERIMENTS AND RESULTS .............................................................................. 29
   4.1. Vertex based Texture Reconstruction .................................................................. 29
   4.2. Patch based Texture Reconstruction .................................................................. 30
   4.3. Impact of mesh resolution ................................................................................. 31
   4.4. Visual Turing Test ................................................................................................ 32
   4.5. Occluder removal ............................................................................................... 32
   4.6. Indoor scene ........................................................................................................ 34
5. DISCUSSION .............................................................................................................. 35
6. CONCLUSION ............................................................................................................. 36
7. REFERENCES .............................................................................................................. 37
8. APPENDICES ............................................................................................................. 42
FOREWORD

This work was carried out at Center for Machine Vision and Signal Analysis in University of Oulu, Finland under the supervision of Prof. Janne Heikkilä. First, I would like to express my gratitude to Prof. Janne Heikkilä for his valuable insights and guidance in accomplishing this work. He has helped me take a step towards completing my studies and provided his motivation and advice about my career. I would like to thank Dr. Markus Ylimäki for his help with code review and feedback at the initial stages of study. I am grateful to Prof. Vili Kellokumpu who has been helpful with every aspect of life inside university.

I would like to thank Martina Majerčáková for her continuous support and belief in me. Last but not the least; I am grateful to my parents who have stood by my side and supported me at every stage of my life so far.

Oulu, 21.05.2017

Satish Kumar Rajagopal
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization And Mapping</td>
</tr>
<tr>
<td>SfM</td>
<td>Structure from Motion</td>
</tr>
<tr>
<td>SIFT</td>
<td>Scale Invariant Feature Transform</td>
</tr>
<tr>
<td>SURF</td>
<td>Speeded Up Robust Features</td>
</tr>
<tr>
<td>DoG</td>
<td>Difference of Gaussian</td>
</tr>
<tr>
<td>LoG</td>
<td>Laplacian of Gaussian</td>
</tr>
<tr>
<td>RANSAC</td>
<td>Random Sample Consensus</td>
</tr>
<tr>
<td>DLT</td>
<td>Direct Linear Transformation</td>
</tr>
<tr>
<td>MVS</td>
<td>Multi-View Stereo</td>
</tr>
<tr>
<td>CMVS</td>
<td>Clustering Views for Multi-View Stereo</td>
</tr>
<tr>
<td>PMVS</td>
<td>Patch-based Multi-View Stereo</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>EXIF</td>
<td>Exchangeable Image File Format</td>
</tr>
<tr>
<td>FSSR</td>
<td>Floating Scale Surface Reconstruction</td>
</tr>
<tr>
<td>BRDF</td>
<td>Bidirectional Reflectance Distribution Function</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

3D reconstruction is one of the fundamental research problems in Computer Vision and Computer Graphics. In its basic form it produces 3D model of a scene based on multiple 2D images. It is widely used in many fields such as archaeology, quality inspection, robotics, virtual reality, gaming and film industry. Real time reconstruction is primarily approached using Simultaneous Localization and Mapping (SLAM) [4] [5]. Advancements in embedded hardware have enabled to use this approach effectively in full-fledged applications. Offline reconstruction based on large-scale data uses a Structure from Motion (SfM) approach [6]. SfM produce reliable large-scale geometry models irrespective of the dissimilarities between the images in terms of lighting conditions and camera parameters.

Despite making significant progress in 3D reconstruction techniques, we are still far away from ability to generate 3D models that look similar to real world objects. Texture reconstruction has not attracted as much attention as the geometric reconstruction. 3D digitization is a relevant field for many industries such as photo tourism, entertainment, preservation of cultural heritage, gaming and augmented reality. It would be easier to generate a 3D textured model of a statue from images rather than from special active scanning 3D data acquisition hardware. Many of the 3D reconstruction pipelines do not produce textured models that can be plugged in many computer graphics applications without much editing. Texture is an important component in the visualization of model and it requires significant manual editing. Thousands of images are used in the reconstruction and they vary significantly. The changes include illumination, camera parameters, field of view, uneven distribution of image data set, occlusions, etc. Majority of the reconstruction methods fall in short to handle these complexities in their approach as they worked on controlled data sets or imposed certain environment conditions such as viewpoints and lighting.

The last decade has seen some progress in the field of 3D reconstruction from 2D images. We have come to a stage where we are able to reconstruct 3D geometric textured model of objects from the vast internet photo collection [3]. It can be difficult to differentiate low-resolution rendered images using state of art technology from the original images scaled at similar resolution. Most of the existing methods are computationally expensive and demand high processing power to achieve large-scale texture reconstruction. We are still far from achieving detailed texture reconstruction methods that do not demand high processing power and whose output can be directly used in graphics applications without or little manual post processing. Inaccurate camera parameters, illumination, exposure and scale differences between images, out of focus blur and computation complexity are some of the challenges faced in the texture reconstruction process.

When analyzing the state of the art technologies, we observe two prominent approaches. The first one takes an approach based on computer graphics, modeling each image as a shading model. The lighting parameters for each image and reflectance parameters for each vertex are estimated by minimizing an objective function. In this approach, texture resolution is linked directly with mesh resolution. In other words, the texture color is encoded as per-vertex color. The second approach is based on matching surface faces to corresponding image area, either single view or multiple views. The matching function can vary depending on objective function. Local or global color adjustments are made to smooth the model at the seams.
The objective of thesis is to give an overview of the texture reconstruction pipeline and analyzing the two different approaches in detail. In our experiments, we use images obtained from online databases. For indoor scene, we use images captured using a standard mobile phone. A comparison in terms of perceived visual quality, computation parameters and performance under different environment conditions is done as a part of the analysis.

The thesis is structured in an order similar to the reconstruction pipeline. In Chapter 2, we introduce pre-processing pipeline of 3D scene reconstruction. We discuss steps involved in SfM. In chapter 3, we discuss both the texture reconstruction approaches in detail. We run the approaches on images obtained from standard databases and indoor images captured using standard mobile phone. We explain the experiment setup and discuss the results in chapter 4.
2. 3D SCENE RECONSTRUCTION

In a 3D scene reconstruction process, information from several images captured by different cameras at multiple viewpoints with some common overlapping scenes is used to form a 3D model. Typical stages involved in the 3D reconstruction process are shown in Figure 1. The input images do not have to be restricted to a particular single camera or restricted viewpoints. They could be data acquired in a careful systematic manner or they can be just a collection of photos of a location captured by different camera setups at different times. Internet photo collections of famous touristic locations would fit the latter scenario. There are full-fledged applications [17] [21], which take a collection of images as input and produce scene geometry as output.

![Large scale 3D reconstruction pipeline.](image)

Figure 1. Large scale 3D reconstruction pipeline.
All stages of the reconstruction process are discussed in detail in the following subsections. In Section 2.1, we will discuss about some popular techniques to extract features from an image. The features of the image are unique points in image with which we can identify a real world point. The way to establish that two images share some common image patches is by finding correspondences between the features of the images. This is explained in Section 2.2. We will also discuss here optical flow calculation, which assists in feature pairing especially for subsequent time instance frames such as in videos. Structure from Motion (SfM) is one of the highlights of Multiple View Geometry. Using feature correspondences from unstructured image collection, camera poses can be extracted. The camera parameters are used in the reconstruction of features as 3D points in the world coordinates as described in Section 2.3. The result from SfM comprises only of a sparse set of points, which have high correspondence match and pass the numerical stability involved in the reconstruction calculations. In Section 2.4, we discuss the technique of Multi View Stereo through which we can calculate a denser model by projecting individual images.

2.1. Feature Extraction

The goal of extracting 2D features in the image is to identify sets of matching points between the images. Each feature with high correspondence confidence is reconstructed into a 3D point in the latter stage. Image features extracted and / or matched using human vision can be different from features identified by the machine vision system. Harris corner [12] is one of the most common feature extraction methods. They output features in an image. However, we should be able to identify the same feature in a different image. Feature descriptors add information to the extracted features and this information is used in identifying one feature from other. Several feature descriptors have been developed in the past. However, we need feature descriptors that would help us match image feature from an unconstrained image database that have scale and rotational changes. Therefore, the descriptors should produce the same description irrespective of aforementioned changes. With these constraints in consideration, Scale-Invariance Feature Transform (SIFT) [9] and Speeded Up Robust Features (SURF) [10] feature descriptors are commonly used in 3D reconstruction process. They have their distinct advantages, which have been discussed in relevant literature [15] [16]. Table 1 highlights the performance of these descriptors under different conditions.

Table 1. Comparison between SIFT and SURF [15]

<table>
<thead>
<tr>
<th>Method</th>
<th>Execution Time</th>
<th>Scale Invariance</th>
<th>Rotation Invariance</th>
<th>Blur Invariance</th>
<th>Illumination Invariance</th>
<th>Affine Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>Common</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
<td>Common</td>
<td>Good</td>
</tr>
<tr>
<td>SURF</td>
<td>Best</td>
<td>Good</td>
<td>Common</td>
<td>Good</td>
<td>Best</td>
<td>Good</td>
</tr>
</tbody>
</table>

2.1.1. SIFT

SIFT detector is based on scale-space extrema. It involves four stages: (1) Finding Scale-Space extrema, (2) Keypoint localization and filtering (3) Orientation
assignment and (4) Descriptor. The first two stages detect keypoints along both scale and space using a continuous function of scale. Gaussian kernel is a good candidate for continuous function of scale. Although Laplacian of Gaussian performs best, Difference of Gaussian is preferred due to its cost effectiveness. The third stage accounts for changes in orientation and scale. In the final stage, orientation data of pixels around keypoints are used to form an orientation histogram. This orientation histogram data of the feature and its neighboring pixels describe the feature. Hence, the size of the descriptor is a product of number of arrays in orientation histograms and number of bins used to describe the orientation histogram.

Figure 2. Scale Invariant Feature Transform (SIFT) Keypoints: Image showcasing SIFT keypoints calculated using OpenCV library. Each keypoint is represented by a circle. The radius of the circle describes the size of the keypoint. The orientation of the keypoint is described by the line inside the circle.

2.1.2. SURF

SURF is similar to SIFT as it uses space scale analysis as well. In fact, it can be considered as a speeded up version of SIFT. Instead of approximating LoG to DoG, SURF approximates it further to a box filter. By doing this the scale space extrema can be calculated using integral images. Haar wavelet responses are used to calculate the descriptor vectors. The Haar response of neighborhood pixels along the orientation direction is the descriptor vector. The vector is normalized to be invariant to scale and rotation.
2.2. Feature Matching

Once all features from all images are extracted, feature correspondences are calculated. For feature correspondences, all features of an image are compared with all features of the second image. This is done for all image pairs, which makes this stage computationally heavy. So usually, a high-level image similarity is done to select image pairs for calculating feature matches. Some tricks could be used to perform this faster [3]. When comparing one image to another a many to one approach is used. Any feature on second image with more than one possible match from the first image is discarded. A simple confidence measure of the image is calculated based on number of matched features. Images with insufficient matched features are unreliable and discarded.

Further reduction in image pairs can be performed by applying geometrical constraints. A feature pair from two images should point to the same 3D point in the real world. Assuming that all images are captured by pinhole camera, an epipolar geometry constraint can be applied successfully to valid image pairs. As shown in Figure 3, optical rays through corresponding features from two images must intersect as they belong to same real world point. The homogenous image points, x and x', must satisfy the fundamental matrix equation. For any pair of corresponding image points x and x’ in two images, there exists an initial unknown fundamental matrix F, which satisfies equation (1). Geometrically F projects image point x from a two dimensional projective plane on to one-dimensional projective line l’, a line through epipole e’.

\[ x'^T F x = 0 \]  

(1)

Figure 3. Real world point X, image points x and x’ form the epipolar plane.

Due to noise, invalid matches, etc. not all feature matches will satisfy the condition. Generally, Random Sample Consensus (RANSAC) algorithm [14] to maximize the number of satisfying feature pairs (inliers) is applied to find F. An
eight-point algorithm [11] [44] is used to calculate fundamental matrix. If there are not sufficient inliers, the image pair is discarded. Among valid image pairs, outlier features are discarded. The inliers across images can be grouped together to form tracks to support incremental triangulation of the 3D point in latter stage. In case of sequential photo collection such as videos, optical flow [16] can be used to speed up the whole process of feature matching. This is predominantly the case in real time systems.

2.3. Structure from Motion

Structure from Motion (SfM) refers to the technique of reconstructing three-dimensional structure from motion information obtained from multiple images or video frames captured at different viewpoints. In the simplest of cases, it is equivalent of a stereo vision or binocular vision. Using the images from each eye (or camera) and available pose information between the eyes (equivalent of relative pose between stereo cameras), we can roughly triangulate the depth of visible object in real world. Instead of matching features explicitly, our brain does that internally to match features between the two eyes. Conventionally given two poses of the camera and their feature correspondences between images, it is possible to triangulate the 3D point. Whereas in SfM, no information of the camera poses is known at the beginning and we could still estimate both camera poses and 3D point simultaneously using feature correspondence between images.

Broadly, there are two different approaches to do this. The first, batchwise SfM considers all cameras parameters together, and tries to solve for all in a single batch. This is numerically inefficient due to the large number of parameters that need to be solved and is affected by outliers. In addition, this method is not viable for scaling to large-scale internet photos collection. The later, incremental SfM is the recent and more commonly used pipeline. It starts with a smaller reconstruction and builds on it by adding newer images to the reconstruction every cycle. Hence it is dependent on images used for initial reconstruction and can result in local minima. To mitigate any errors being propagated into next iteration, bundle adjustment and outlier removal operations are performed between cycles. Bundle adjustment minimizes geometric cost function by collectively optimizing both camera parameters and 3D point locations. Bundle adjustment [19][43] is a non-linear minimization technique and can significantly affect the running time. Incremental SfM is discussed below, as it is the approach used in our experiments.

2.3.1. SfM Model

To estimate a 3D point, the camera poses of its corresponding images must be known. The camera pose includes position and orientation, collectively called as extrinsic parameters of the camera. For every camera, the extrinsic parameters comprise of rotation and translation represented by 3x3 rotation matrix $R_i$ and 3x1 translation vector $t_i$. Sometimes the translation is represented using camera center $c_i$, where $t_i = -R_i c_i$. In addition to extrinsic parameters, camera intrinsics modeled by internal parameters of the camera should also be known. Camera intrinsics are represented by 3x3 upper diagonal intrinsic matrix $K_i$, mapping 3D points to homogenous 2D image positions. Camera intrinsic matrix is modeled by focal length
of lens, offset between camera and lens arrangement, aspect ratio of pixel, skew and scaling factor for unit conversion to pixel. In many cases, $K_i$ is assumed to be a diagonal matrix, $K_i = diag(f_i, f_i, 1)$ where $f_i$ is the focal length of camera. In a general $K$ matrix, elements off the diagonal are determined by skew (deviation of angle between image axes from 90°) and principal point (intersection point of optical axis on image sensor plane). For most digital cameras, skew is zero and principal point is at the center of image sensor. Also in most modern cameras, the pixel is square shaped leading to an aspect ratio of one and hence the first two diagonal elements are equal. Although the $K$ matrix model is linear, the lens distortions are non-linear. Brown’s radial distortion model [18], can be used to correct for distortion. The distorted $X$ position coordinate $x_d$ is a function of $x_u$, undistorted $X$ position coordinate and $r$, distance of point from image center. This is extended to $Y$ dimension as well. In general, first two coefficients ($k_1$ and $k_2$) are sufficient to model the distortion as long as the lenses do not have large field of view (FOV) like spherical lenses.

$$x_d = x_u + k_1 r^2 + k_2 r^4$$  \ (2)

A single camera can thus be represented as $C_i = \{c_i, R_i, f_i, k_{1i}, k_{2i}\}$. There are a total of nine parameters, three representing intrinsic and six representing extrinsic parameters. The camera parameters represent how a 3D point $X_j = (X_{jx}, X_{jy}, X_{jz})$ in real world is projected on to the image $i$ as a 2D image point $P(C_i, X_j)$ through projection equation [13]. Point tracks are formed during the feature matching stage. Each track $j$ contains observed image points $q_{ij}$ of a single 3D point in different camera viewpoints $i$. The difference between observed image point $q_{ij}$ and projected image point $P(C_i, X_j)$ is called as reprojection error, as shown in Figure 4.

![Reprojection error](image)

Figure 4. Reprojection error is the distance between the projected image point $P(C_i, X_j)$ and observed image point $q_{ij}$ of 3D world point $X_j$ in camera $C_i$.

The SfM problem is formulated as to minimize sum of squared reprojection errors of all 3D points on its visible image frames. However, not all points are visible in all images. Hence, a visibility indicator parameter $w_{ij}$ is used in the problem. If the
visibility indicator $w_{ij} = 1$, it signifies that point track $j$ is visible in camera viewpoint, and $w_{ij} = 0$ otherwise. The objective function is given as below, where $n$ is the number of views and $m$ is the number of 3D points or equivalently tracks.

$$g(C, X) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \left\| q_{ij} - P(C_i, X_j) \right\|^2$$  \hspace{1cm} (3)

The goal of SfM is to find optimum camera and scene parameters that minimize the objective function. This is carried out using an incremental approach with new camera viewpoint being added at every stage and objective function minimized using a non-linear least squares optimization algorithm such as Levenberg–Marquardt algorithm.

### 2.3.2. Incremental SfM

We have to recover camera parameters and 3D location for all the feature tracks. The recovered model should minimize the reprojection error, which is calculated as sum of squared distances between the projections of each feature track and its corresponding image feature. The minimization problem is a non-linear least squares problem and solved using Levenberg-Marquardt algorithm [42]. These types of algorithms find local minima and especially with large-scale photo collection, this could be a potential issue. It is important to initialize with a good starting parameter estimates to avoid solution parameters stuck in wrong local minima.

An incremental SfM pipeline starts with a reliable two-view reconstruction and hence the choice of images used is critical. The initial image pair used for two-view reconstruction should have large number of matches. They should also have a large baseline between them to ensure that the estimated 3D points are well conditioned. Homography could be used to check if the image pair forms a large baseline. Homography represents a plane to plane transformation. If a homography could be reliably fit to correspondences, it would mean that the two images are of a single plane or the images are captured at same location, but with different camera orientations. RANSAC can be applied for homography estimation between images using feature correspondences. If homography cannot be fit to correspondences between two images, it indicates that the two cameras are apart with sufficient baseline. The image pair with least inliers, but with a large number of correspondences should be chosen. The chosen pair is solved for relative camera pose estimation using the five-point algorithm [7] and triangulation of the feature correspondences is performed to find 3D locations of the points.

Subsequently newer images are added one by one. The image with largest number of feature tracks whose 3D points are already estimated is chosen. Direct Linear Transformation (DLT) on RANSAC is used on already estimated 3D points to find the intrinsic and extrinsic parameters of the new camera [6]. This would be used as initial estimates for bundle adjustment to refine the same. Once the new camera parameters are known, triangulation of feature tracks from new camera is performed assuming they are seen from at least two cameras with sufficient separation between them. This process is repeated until there are no more images to add. For the purpose
of speed and robustness, any outliers with larger reprojection error are removed between iterations.

2.4. Multi-View Stereo

Structure from Motion (SfM) outputs camera poses and a point cloud. However, the point cloud is sparse as only image features that are matched well enough across photographs are used in the reconstruction. A denser point cloud can be reconstructed by performing Multi-View Stereo (MVS) reconstruction [3] using the sparse 3D points and registered individual images. Recovering depth information using stereo vision is similar to how human vision system perceives depth. A basic approach to calculating depth parameter using stereo is shown in Figure 5. Although depth perception using stereo requires only two cameras (or viewpoints), there might be more than two images that capture the same 3D point. In this case, the optimal point can be calculated by weighting the histogram of depth calculated from several pairs of images.

![Figure 5. Window based MVS algorithm: Given a pixel and an image window around it, we hypothesize a finite number of depths along its viewing ray. At each depth, the window is projected on to other images and consistency of texture at these image projections is evaluated. The score is maximum at its true depth.](image)

In an unconstrained large dataset, the number of photos is beyond what a standard MVS algorithm could handle. To address this issue, clusters of images viewing particular part of the scene are formed and each cluster is passed through MVS separately. This also facilitates parallelization and each cluster can be run in separate processors [22].
2.5. Surface Reconstruction

The dense point cloud is reconstructed into a surface using surface reconstruction methods. Poisson surface reconstruction [23] is one of the most common methods used for this purpose. It takes 3D points and their normal as inputs and outputs a surface model.

The objective is to triangulate a surface mesh based on the point data (i.e.) in other words, to find the indicator function. The indicator function is defined as 1 at points inside the model, and 0 at points outside. Note that the indicator gradient is 0 except at points near the surface. Thus, the problem of computing indicator function reduces to inverting the gradient operator. In other words, finding the indicator function $\chi$ whose gradient best approximates a vector field $\vec{V}$ defined by the samples (i.e.) $min_{\chi} \|\nabla \chi - \vec{V}\|$. On applying the divergence operator, the variation problem transforms into Poisson problem: compute the scalar function $\chi$, whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$.

$$\Delta \chi = \nabla \cdot \nabla \chi = \nabla \cdot \vec{V}$$

![Figure 6. Illustration of Poisson reconstruction in 2D.](image)

As one can notice, Poisson surface reconstruction is dependent on the point samples or mesh resolution. The denser the point cloud, the smaller the triangles and hence adding more realism in surface. The resolution is analogous to scale parameter in sampling. There are other methods, which use patch size or depth maps from Multi-View Stereo for scale [41] or treat scale as a continuous to account for sample resolution differences [40].
3. TEXTURE RECONSTRUCTION

While 3D reconstruction has seen tremendous improvements over the last decade or two, they still do not produce good enough models that could be used in end applications without extensive manual editing. Most of the texture reconstruction techniques have been developed for a controlled laboratory environment or with additional constraints. In addition, some of them [20] have complex models that it is not scalable to a larger dataset. Our study primarily deals with large-scale internet photo collection where there is very little prior information and control on the viewpoint and lighting consistency. There are only a few approaches, which scale up to large-scale 3D reconstruction pipelines for internet photo collections [3] [26]. In addition to the size of inputs, internet photo collections pose additional challenges. Image information such as illumination and exposure is unknown. The input images vary in illumination, exposure and image scale factor (image area occupied by a real world object vary between images) that may vary by several magnitudes between a close up view and a far distant view. As we are discussing about outdoor scene, there are occluders (pedestrians or other dynamic objects) which are in the image, but do not belong to the scene geometry. Many occlusions are removed in the scene reconstruction stage.

We will be discussing two types of texture reconstruction framework. The first one uses reflectance model representing per vertex colors or albedo parameters. The second one uses patch based texture modeling representing a per view texture. Computer graphics algorithms have used reflectance and illumination model for few decades and there is a large literature regarding the same. Estimating illumination and reflectance properties of scene using image based techniques is well studied. Most of them work only on controlled datasets, such as the lighting conditions of all input images are identical. With this constraint, it is possible to compute illumination and surface reflectance [48] [49].

One of the early reconstruction pipelines to deal with diverse illumination is Haber et al.’s wavelet-based framework [20] that allows simultaneous estimation of illumination per image and surface reflectance. The approach uses an all frequency relighting framework based on Haar wavelets. This allows for incorporating various reflection models. The simulation of light propagation in the scene is based on reflectance equation for every rendered pixel given by equation (4). In equation (4), $\rho$ is the Bidirectional Reflectance Distribution Function (BRDF) at position $\hat{x}$ for incident illumination from direction $\omega_i$ and outgoing direction $\omega_o$. The incident illumination and occlusion at $\hat{x}$ is represented by $\hat{L}$ and $V$ respectively. $\Omega$ denotes the hemisphere of incident illumination. Once the three components: illumination, occlusion and BRDF is represented in a common Haar wavelet basis, equation (4) can be represented as triple-wavelet product integral [50]. It is shown in equation (5), where $C_{kim}$ is the tripling coefficient. For a given point, $V$ is constant and hence the equation can be reduced to bilinear system in $\hat{L}$ and $\rho$. The lighting and reflectance model is complex involving bilinear system of equations and without any bootstrapping mechanism to reduce the computational time.

$$L(\hat{x}, \omega_o) = \int_{\Omega} \rho(\hat{x}, \omega_o, \omega_i)V(\hat{x}, \omega_i)\hat{L}(\hat{x}, \omega_i) d\omega_i$$  \hspace{1cm} (4)
In image-based texture generation, texture information comes from images. In many approaches, the granularity of source of information is kept at texel level. In other words, the source of texture information for a particular texel can differ from the source of texture of its neighboring pixel. In most approaches, blending of information from multiple source images is performed to compute the texture of single texel [53] [54]. Sinha et al. [33] uses a single image per texel.

In a multi-view environment, surface patches are shared across images. Goldluecke and Cremers [51] use this to apply a superresolution model and hence obtaining a texture map in a resolution higher than the individual input image resolution. This demonstrates texel interdependencies between neighboring texels instead of a per texel lighting model. It is developed based on 2D superresolution and the idea is to model the downsampling by the cameras in image formation process. The image formation model is reduced to an energy function, which is solved for unknown texture optimally fitting all input images simultaneously.

Unlike the above approaches, some methods circumvent texel-wise blending by selecting large regions, which are assigned texture data from a single view. Usually this is done per mesh face. Some approaches use a single image per face [52] [36] [27] and others blend multiple images per face [34] [55]. Blending of images can cause problems due to inaccurate camera parameters or geometry. Also if the chosen images are out of focus or of different scales, that could add to the problems. This is solved in some approaches [34] by weighting the images. The weights depend on angle and proximity to model. The resulting texture patches may still suffer from discontinuities between patches due to exposure or luminance differences. Hence adjacent texture patches need to be photometrically adjusted so that their seam becomes invisible. These differences are adjusted either locally or globally to reduce seam visibility. Velho and Sossai [36] employ a local adjustment. They set the color at seam to be the mean of the color from its left and right patches. A diffusion step is done to smoothen the color transition towards this mean value at the seam. This has some effect on the area near the seam (Figure 8: center). Lempitsky and Ivanov [27] take a global approach to reduce seam visibility. They calculate globally optimal luminance correction values that are added to vertex luminance values. The constraints used for calculating these correction values are as below.

(i) After adjustment, the luminance differences at the seams should be as small, which is the primary motive behind color adjustment.

(ii) Derivative of adjustments within a texture patch should be small. This is to ensure that the correction values do not affect the integrity of a single patch and its features.

Texture reconstruction is important for projecting realism in the reconstructed models. In an ideal case, it should be a fully automatic, scalable and fully integrated with a surface reconstruction framework. We discuss two texture reconstruction pipelines from each category that provide a full-fledged framework that can handle large uncontrolled and hence realistic datasets. They are discussed in detail in the following sections.
3.1. Vertex based Texture Reconstruction

Shan et al. [1] proposed a large-scale framework for finding reflectance parameters of the reconstructed model and simultaneously estimating the viewpoint and illuminating conditions. The framework demonstrated the ability to render the reconstructed model in a simulated illumination conditions as the reflectance parameter model is a simple color model per vertex. Each vertex in the mesh will have a reflectance or albedo vector for each color channel. This is similar to basic computer graphics rendering. The object has a reflectance value and based on the illumination, it reflects different proportions of light across the three channels. Although it was originally showcased with outdoor images, where the illumination model is simple, the model and the equations can be modified to fit the necessary environment.

3.1.1. Preprocessing

SfM along with Multi-View Stereo (MVS) form the major processing block in the pre-processing stage. There are many readily available software to perform these operations. Camera poses are recovered with SfM using VisualSFM [21]. Shan et al. uses a two-step approach for initial point cloud reconstruction. This approach speeds up the reconstruction for large data sets by selecting limited images with high coverage area. After filtering unwanted images, image set I is supplied to VisualSFM for reconstructing initial point cloud. A dense, oriented point cloud is obtained using Patch-Based Multi-View Stereo (PMVS) [8] or Clustering Views for Multi-View Stereo (CMVS) [22]. A triangle mesh is generated using Poisson Surface Reconstruction [23].

In addition to these, per vertex visibility is obtained for the set of images. The objective is to identify a set of images (from I) for each vertex in which it is visible. This could be a long process. However, because PMVS already outputs a visibility set, we can use that as initial set. For each vertex \( v \) in the Poisson mesh, we group the visibility sets of 30 nearest points. In this superset, nine images that appear most frequently is selected. An average color for all vertices in the 7-ring neighborhood of \( v \) is computed by projecting the 7-ring neighborhood vertices onto this nine-image set and averaging it. For each image \( I \) in I, images facing away from \( v \) and images whose normal hits other faces of the mesh before hitting \( v \) are rejected. For the other images, 7-ring neighborhood is projected on to \( I \) and resampled color values from \( I \) are compared against the average color computed. If the normalized cross correlation is higher than threshold (0.8), image \( I \) is added to the visibility set of vertex \( v \). Fast Library for Approximate Nearest Neighbors (FLANN) [39] can be used to accelerate the process. Pre-computed z buffers and ray casting can be used for occlusion testing.

3.1.2. Shading Model

The inputs for lighting and reflectance estimation comprises of outdoor images (I) with reconstructed mesh and per vertex visibility sets. The objective is to estimate lighting parameters for each image and reflectance parameters for each vertex. We also identify cloudy images as this simplifies the problem and speeds up the optimization. As this is an outdoor scene, we adopt a simple model for illumination
and materials. We assume that in addition to the ambient uniform hemispherical light from the sky, there exists a directional sunlight. In addition, we consider all the materials to be diffuse.

Let us consider a vertex $P_i$, an image $I_j$ for which we are modeling the rendered pixel intensity. The parameters related to the vertex are surface normal $N_i$ and surface albedo parameters $a_i$. The albedo parameters vary between the three color channels. Hence this would be a three dimensional vector. As you will see, we initially use this as a scalar value when we process on grayscale images and then expand into the three color channels. The skylight (ambient) intensity, sunlight (diffuse) intensity and lighting direction in spherical coordinates of the single directional light source (sun) during capturing of image $I_j$ are represented by $k_j^{sky}$, $k_j^{sun}$ and $L_j$. Note that the camera exposure is modeled into the light intensities automatically. $\delta_i,j$ models the sunlight visibility (i.e.) it has a value of one if $P_i$ is in sunlight in image $I_j$, else it has a value of zero. The rendered pixel intensity $R_{i,j}$ of vertex $P_i$ in image $I_j$ is modeled with an ambient + diffuse model as follows:

$$R_{i,j}(\theta) = a_i \left\{ f(N_i) k_j^{sky} + \max[0, L_j, N_i] k_j^{sun} \delta_{i,j} \right\}$$  \hspace{1cm} (6)

$$\theta = \{ N_i, a_i, L_j, k_j^{sky}, k_j^{sun}, \delta_{i,j} \}$$

The function $f(N_i)$ models the skylight visibility of the point. The ambient light is modeled as a uniform hemispherical light from sky. Although the surrounding surface mesh can be included in the modeling, a simpler version dependent only on the vertex under consideration is used [24]. Given that the up direction is $U$, it is modeled as:

$$f(N_i) = \frac{1 - N_i \cdot U}{2}$$  \hspace{1cm} (7)

If the observed pixel intensity of vertex $P_i$ in image $I_j$ is $\bar{R}_{i,j}$ and $V_i$ is the set of image indices in which the vertex $P_i$ is visible, the lighting and reflectance parameter estimation can be formulated as follows:

$$\arg\min_{\theta} \sum_i \sum_{j \in V_i} \sqrt{\bar{R}_{i,j}} \left\| R_{i,j}(\theta) - \bar{R}_{i,j} \right\|^2$$  \hspace{1cm} (8)

Note that the objective function is a weighted sum of squared differences between the observed pixel intensity and the rendered pixel intensity according to the shading model. The weight is square root of the observed pixel intensity and this is to weigh down the effect of points in shadow. This is particularly effective in the initial stages when the sunlight indices $\{\delta_{i,j}\}$ are initialized to one.

### 3.1.3. Detection of Cloudy Images

The objective function (8) can be computationally expensive when processing thousands of images gathered from several sources. The non-linearity of the function
increases the risk of settling down on one of the numerous local minima. Cloudy images have negligible sunlight intensity and hence can be approximated to an ambient illumination model. Using this approximation, we can calculate skylight intensities and albedo parameters for the points visible in those images. An image can be considered cloudy if it passes one of the three tests described below.

(i) Usage of image metadata information stored in the EXIF tag. The exposure value of the image can be calculated using exposure time, ISO setting and focal number as \( \frac{\text{exposure time} \times \text{ISO value}}{F-\text{number}^2} \). Lower exposure value represents sunnier and brighter illumination. The image is categorized as cloudy if it has a value in the range of \([0.5, 5.0]\).

(ii) The amount of ‘skyness’ can be calculated by analyzing the top 3% of the image. An image is classified as cloudy if in the top section of the image, the condition \( 2B_{avg} - R_{avg} - G_{avg} < 100 \) is true.

(iii) The image is classified as cloudy if skylight intensity is higher than sunlight intensity by several folds. In specific terms, the image is classified cloudy if \( k_j^{sky}/k_j^{sun} > 10 \).

Unlike the first two tests, the last test could only be applied after lighting estimation. Hence, the first two tests are used at the initial stages of the algorithm and the third test is added in later to update the cloudy image dataset.

### 3.1.4. Algorithm

The algorithm comprises of three steps. Firstly, we use the special case of cloudy images, to estimate albedos for a subset of the points. In the second step, we estimate the lighting parameters for each image in the dataset. Once lighting parameters are available for all the images, each individual vertex can be solved for their albedo and normal parameters.
Usage of Cloudy images

All cloudy images can be considered to have only ambient light (skylight) and hence we could remove the directional light component from the shading equation (6). The shading equation can be simplified to

\[
R_{i,j}^c(N_i, a_i, k_j^{sky}) = a_i f(N_i) k_j^{sky}
\]  

(9)

Let \( I^c \) be the set of cloudy images. We use set of points \( P^c \) that are present in more than three cloudy images; hence, our estimations are more reliable. It is difficult to estimate normal of a vertex without directional light. We have normal estimated as part of the surface reconstruction and can be used as an input. The objective function can be simplified to

\[
\arg\min_{\{a_i, k_j^{sky}\}_i} \sum_{i \in I^c} \sum_{j \in V_{i \cap I^c}} \sqrt{R_{i,j}} \left\| R_{i,j}^c(a_i, k_j^{sky}) - R_{i,j} \right\|^2_2
\]  

(10)

Lighting estimation

Not all points are available in cloudy images. It is computationally expensive to solve for all the points and image lighting conditions at the same time. Since not all points are required to determine lighting conditions, the focus is set to solving for lighting parameters for each image using an effective subset of the points. An ideal set of points for lighting estimation would contain points visible in many images. However, it should also achieve coverage by ensuring that there are enough points
from each image. The point set $\mathbf{P}^l$ is initialized with 2000 points that are visible in most images. To ensure coverage, randomly sampled points are added from each image until there are at least $m (=1000)$ points from each image. To give preference to points visible in more images, the sampling probability is proportional to the number of images in which the point is visible.

The information obtained from solving cloudy images can be used to steer the solving of objective function. A damping term to bias the solution towards albedo estimates from (10) for points belonging to $\mathbf{P}^c$ and normal estimates from input mesh is added. A damping factor $\lambda_1 (=1)$ is used in the modified objective function (11). As discussed in Section 3.1.3, we can use the estimated lighting parameters to update the cloudy image set $\mathbf{I}^C$.

$$
\arg\min_\theta \sum_{i \in \mathbf{P}^l} \sum_{j \in V_i} \sqrt{\hat{R}_{i,j}} \left\| R_{i,j}(\theta) - \hat{R}_{i,j} \right\|^2_2 \\
+ \lambda_1 \sum_{i \in \mathbf{P}^c} \left\| a_i f(N_i) - \hat{a}_i f(\hat{N}_i) \right\|^2_2
$$

(11)

**Per-vertex Albedo and Normal estimation**

Once the lighting parameters are estimated for each image, we could estimate the point parameters $\{N_i, a_i, \delta_{i,j}\}$ independently. Some of the points might not be visible in many images. Estimation of both albedo parameters and normal for these points can be noisy. Since the normal from surface estimation is reasonably accurate, we add a dampening term towards it. The weight parameter increases for points with fewer images. This is calculated as a sub-product of lighting estimation parameters in the previous step.

$$
\arg\min_{\{N_i, a_i, \delta_{i,j}\}} \sum_{j \in V_i} \sqrt{\hat{R}_{i,j}} \left\| R_{i,j}(\theta) - \hat{R}_{i,j} \right\|^2_2 \\
+ \lambda_1 \sum_{i \in \mathbf{P}^c} \left\| a_i f(N_i) - \hat{a}_i f(\hat{N}_i) \right\|^2_2 \\
+ \omega_i \left\| N_i - \hat{N}_i \right\|^2_2
$$

(12)

One of the parameters, sunlight visibility $\delta_{i,j}$ is a binary variable. This is initialized to one at the beginning. To solve for the sunlight visibility binary variable, it is solved in three steps:

(i) Set $\delta_{i,j}$ and solve other parameters.

(ii) Solve for $\delta_{i,j}$ while other parameters are set for each point. This is done independently for each point, a simple binary decision.

(iii) Fix $\delta_{i,j}$ and solve for other parameters again.

While describing the shading model in Section 3.1.2, we described that all reflectance parameter and lighting intensities are color values. In practice, the color image is converted into monochrome to solve for $L_j$, sunlight direction and monochrome reflectance parameter. With $L_j$ known, we solve the objective function in each color channel independently.
3.2. Patch based Texture Reconstruction

Texture reconstruction is vitally important if we want to achieve realism in the rendering. Ideally, this should be achieved as without increasing the geometric complexity of the reconstruction. As discussed in earlier sections, it should still be integrated with scene construction pipeline. Waechter et al. [2] proposed a fully unified texturing pipeline that handles all the inherent challenges while efficiently producing textured real world models from hundreds of images on reasonable time duration.

The approach is based on a single view per texture. In the first step, a single view is selected for representing the patch’s preliminary texture. In the second step, adjustments are made to ensure smooth texture along the seam or border between the neighboring patches. We first discuss the baseline method in Section 3.2.1 on which this approach is based on before discussing the modified proposed solution in the latter subsections.

3.2.1. Baseline Method

The input images are registered using SfM and the scene geometry is reconstructed using multi-view stereo techniques [22] [26] [45]. With very little post processing, this yields a good reconstruction. The geometry reconstruction setup yields image registration against the 3D reconstruction. Lempitsky and Ivanov [27] proposed a two-step approach for per face texturing. The proposed method is based on it and we will be discussing relevant steps involved in their method so that we could point out the differences between the baseline method and proposed method in future discussion.

Lempitsky and Ivanov compute a labeling pattern \( l \) based on pairwise Markov random field energy formulation. A view \( l_i \) is used as texture for each mesh face \( F_i \) so that the Markov energy value (equation (13)) of the labeling is minimum. The data term chooses good views for the face. The smoothness terms tends to force the solution towards reducing seam visibility; in other words increasing smoothness between patches.

\[
E(l) = \sum_{F_i \in \text{Faces}} E_{\text{data}}(F_i, l_i) + \sum_{(F_i,F_j) \in \text{Edges}} E_{\text{smooth}}(F_i, F_j, l_i, l_j) \quad (13)
\]

The baseline method uses a measure of the angle between face normal and viewing direction in data term. This measure is insufficient for real data set due to the following reasons: (a) it does not consider proximity of the images to the object in the measure (b) it does not consider resolution of face on the image area (c) it does not remove out of focus or blurry images. Allène et al. [29] projects the face onto the view and includes that in data term to account for proximity and image resolution. Similarly, Lumigraph’s [30] view blending weights account for the same effects as well. The expression as it stands can favor faces closer to the camera as they have large projection area. However, it still does not account for out of focus images. Gal et al. [28] uses gradient as a measure of focus. Their proposed data term uses gradient magnitude of the image integrated over the face’s projection on the image.
This term is large if the image is on focus (gradient is large) or images are close, parallel to the face and has high resolution (large projection area).

Lempitsky and Ivanov [27] use smoothness term as difference between left and right sides integrated over the seam. This prefers seams in regions where the images are accurately registered or where the texture is smooth. The smoothness terms are a computational hindrance and are not precomputed due to the number of combinations possible. This could also favor low resolution, distant or blurry images, unlikely in a controlled experiment image database.

Before adjusting patch colors, we must ensure that each mesh vertex belongs to one texture patch. Therefore, each vertex is duplicated into left vertex and right vertices, one for each texture patch on the left and right side of the seam. Note that this is the simplest case. If the seam vertices belong to more than two patches, then many numbers of duplicates are created. We will consider the simpler case in our formulations to describe the approach. Each vertex $v$ is split into $v_{\text{left}}$ and $v_{\text{right}}$ belonging to left and right patches respectively. An additive correction $g_v$ is computed for each vertex that minimizes the following function.

$$
\argmin_g \sum_v \left( f_{v_{\text{left}}} + g_{v_{\text{left}}} - (f_{v_{\text{right}}} + g_{v_{\text{right}}}) \right)^2 + \frac{1}{\lambda} \sum_{v_i, v_j \text{ are adjacent and in same patch}} (g_{v_i} - g_{v_j})^2
$$

The first term minimizes the difference between left and right patches, making them as similar as possible. The second term minimizes the differences between adjacent vertices belonging to the same texture patch, making changes inside a single texture patch as gradual as possible. With corrections $g_v$ known for all vertices, corrections for each texel are calculated by interpolating from corrections of surrounding vertices in barycentric coordinates.

### 3.2.2. Preprocessing

Face visibility for all combinations of views and faces can be calculated by using back face and view frustum culling. View frustum culling is the process of removing objects that lie completely outside the viewing frustum from the rendering process. It is usually done using bounding volumes surrounding the objects than the objects...
themselves. Occlusion detections can be done using available libraries [32] to compute intersections of viewing rays of camera to the triangle of interest. In equation (13), the data term can be precomputed for all face-view combinations, as they remain constant and are used several times during the algorithm.

Figure 9. Left: Each color represents a different label in the mesh patch. Right: The textured result with visible luminance differences between patches.

3.2.3. View Selection

The view selection is similar to the baseline method, based on minimizing energy computed by equation (13) using graph cuts and alpha expansion [31]. However, there are differences in data and smoothness terms along with an additional photo consistency check.

Data term

Gal et al.’s data term $E_{data} = -\int_{\phi(F_i,l_i)}\|\nabla (l_i(p))\|_2 dp$, integration of gradient of the face’s projection over image is used. The gradient of the image $\|\nabla (l_i(p))\|_2$ is calculated using a Sobel operator and summed over area $\phi(F_i,l_i)$ in which face $F_i$ is projected on the image. In case of the projection area being less than a pixel, the gradient at the pixel centroid is multiplied with the projection area.

The data term’s preference for large gradients is prone to risks in uncontrolled data sets and has not been accounted for in Gal et al.’s method. Occluders such as pedestrians have high gradients, especially when the background is smooth, such as a
wall. If these views pass the visibility check, this should not be chosen as preliminary texture. For this purpose, a photo consistency check of the texture in the view is performed.

**Photo Consistency Check**

The assumption is that the view with occluder is an outlier that sees the face. With that assumption, the majority of views should see the correct color. A minority would see wrong colors; and are less correlated. Sinha et al. [33] and Grammatikopoulos et al. [34] made use of this assumption to remove inconsistent views using mean and median colors. The median color approach does not work well on smaller view sets as the median can be towards the outlier. Fixing the mean can be dangerous as well, since the initial mean can be quite far away from the inlier mean. Sinha et al. [33] uses user interventions and this could be avoided in our proposed approach. The proposed approach uses a mean shift algorithm to remove the inconsistent views.

1. Calculate face projection’s mean color $c_i$ for each view $i$ in which the face is visible.
2. Initially declare all views as inliers.
3. Compute mean $\mu$ and covariance matrix $\Sigma$ of all inliers.
4. Evaluate multivariate Gaussian expression $\left(-\frac{1}{2}(c_i - \mu)^T\Sigma^{-1}(c_i - \mu)\right)$ for each view in which the face is visible.
5. Clear all inliers and add the faces with the above function value greater than a threshold (0.006).
6. Repeat steps 3 to 5, for 10 iterations or all entries of $\Sigma$ drop below $10^{-5}$, the inversion of $\Sigma$ becomes unstable or there are less than 4 inliers.
7. For each face’s outliers we add a penalty for those views to prevent the selection of those data terms for that face.

**Smoothness term**

As discussed in Section 3.2.1, Lempitsky and Ivanov’s smoothness term is computationally expensive. In addition, it can favor blurry or far off views, counteracting the data term’s preference for close up views. The proposed smoothness term is based on Potts model: $E_{\text{smooth}} = [l_i \neq l_j]$ where $[\cdot]$ is Iverson bracket. The proposed approach favors close up views and is fast to compute.

### 3.2.4. Color Adjustment

Color adjustment is made to adjust discontinuities between patches and minimize seam visibility. The proposed approach is to use modified version of baseline method’s global adjustment, followed by local adjustment with Poisson editing [35].

**Global Adjustment using Color support region**

The baseline method’s adjustment in equation (14) is only evaluated at a single location: vertex’s projection on two images both side of seam. There is usually registration error and both projections do not match to the same spot on real object. Also in an uncontrolled data set, the two images maybe of different scales and hence the looked up pixels scan different footprints in 3D. This can also lead to incorrect global adjustments and artifacts.
To account for this problem, instead of using color value at vertex’s projection we use the color value obtained from along the vertex’s seam as illustrated in Figure 10. Vertex $v_1$ is on the seam between red and blue patches. The color in the red patch $f_{v_1, \text{red}}$ is calculated as an average color of samples from seams $\overline{v_0v_1}$ and $\overline{v_1v_2}$. We draw twice as many samples as the length of edges. The weights are inversely proportional to the distance from vertex under consideration (i.e.) the weight is 1 at vertex $v_1$ and decreases linearly to a weight of 0 at vertex $v_0$ and $v_2$. The reason behind the weights is that the compensation is applied to texels using barycentric coordinates. The average colors of seams $\overline{v_0v_1}$ and $\overline{v_1v_2}$ is average weighted with the edge lengths to obtain $f_{v_1, \text{red}}$. In a similar manner $f_{v_1, \text{blue}}$ is calculated and used in equation (14).

![Figure 10](image.png)

Figure 10. Vertex $v_1$ is shared between blue and red patches. Its color is looked up as weighted average of samples from seams $\overline{v_0v_1}$ and $\overline{v_1v_2}$. The weight is 1 at vertex $v_1$ and decreases as the distance from $v_1$ increases.

**Local Adjustment using Poisson editing**

Global adjustment might not be sufficient to eliminate all seam visibility. Gale et al. [28] performs a local Poisson image editing merge texture patches seamlessly. They do it at a computational cost as the whole texture patch is edited, resulting in huge linear systems. The proposed approach by Waechter et al. [2] is to restrict the Poisson editing to a 20 pixel wide border strip. The strip’s outer and inner rim is used as boundary conditions. The outer rim pixel’s color value is set as the average color of pixel’s color in the image assigned to the patch and neighboring patches. The inner rim pixel’s color value is the pixel’s color in the image assigned to the patch. In case of smaller patches, inner rim is omitted. The guidance field for the Poison equations is the Laplacian of the strip. The linear system can be solved using SparseLU factorization. The system matrix for linear system of equations is the same for all color channels. The factorization is computed only once and reused for all channels. Adjusting strips is more time and memory efficient when compared to adjusting the whole texture patch.
4. EXPERIMENTS AND RESULTS

As seen from Section 3, texture reconstruction is a complex process. It involves several nonlinear optimization problems at different stages of the process. Shan et al.’s approach [1] is dependent on mesh resolution. The time consumption is proportional to the number of vertices in the mesh. For this reason, a Linux server is used when memory or processing requirements is higher. In other cases, a laptop (Intel® Core™ i5-3200M CPU, 16GB RAM) with Graphics Processing Unit (GPU) is used. Although many scene images are obtained from standard databases, some indoor scenes are captured using mobile phone cameras.

4.1. Vertex based Texture Reconstruction

For the vertex based texture reconstruction, a Matlab implementation was done for albedo and lighting estimation process of the texture reconstruction. For the remaining stages, VisualSFM with PMVS was used. For the optimization problems, *lsqnonlin* function was used. Visibility estimation is a highly time consuming process for large datasets in the fully integrated large scale texturing process [1]. It can easily run for several hours on a server. For our implementation, we used the visibility set available from VisualSFM with PMVS. As the reflectance parameter is different for each color channel, theoretically this section of the program could be parallelized. However, note that the requirements for local memory would be high as you are solving for thrice the number of mesh vertices. Table 2 shows the runtime involved for the datasets on a laptop, where the local memory is limited.

![Figure 11. Colosseum reconstructed using vertex based texture approach [1].](image)

Table 2. Time profile for vertex based texture reconstruction

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Images</th>
<th># Mesh vertices</th>
<th>Running time (hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colosseum</td>
<td>100</td>
<td>680k</td>
<td>28</td>
</tr>
<tr>
<td>Eiffel tower model (indoor scene)</td>
<td>20</td>
<td>55k</td>
<td>8</td>
</tr>
</tbody>
</table>
4.2. Patch based Texture Reconstruction

Waechter et al. [2] have implemented an open source C++ implementation [37] and the same was used for our experiments. We use several datasets (der Hass, castle, city wall, etc.) for analyzing this method. The data term is the main computational bottleneck as it involves integration of gradient of the face’s projection over the image. However this approach is still more time efficient than some of the other methods [38] which produce theoretically optimal results at a computational cost. The data term is linear against number of views and number of faces. It can be precomputed and stored in advance. We use the Multi View Environment platform [46] for reconstruction of geometry. It uses Floating Scale Surface Reconstruction (FSSR) [47], which treats sampled points as a representation of a finite area instead of a point on a real world object.
Table 3 displays the runtime statistics for our experiments and this does not include time taken for surface reconstruction. Number of faces is the key parameter determining time consumed unlike number of vertices in vertex based approach. The reflectance parameters have to be computed for each color channels in addition to bootstrapping stage where the lighting parameters of each image are computed.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Images</th>
<th># Mesh faces</th>
<th>Running time (hr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>City Wall</td>
<td>565</td>
<td>1.3m</td>
<td>4</td>
</tr>
<tr>
<td>Der Hass</td>
<td>80</td>
<td>0.25m</td>
<td>1.5</td>
</tr>
</tbody>
</table>

### 4.3. Impact of mesh resolution

In Shan et al.’s approach, texture is stored as albedo parameter per vertex. Based on the material model, the reflectance is different for different light channels and hence the albedo parameter is a three-channel vector, one for each channel. Some triangle meshes can be larger. The larger the triangles, the less texture information is available at the middle of the triangle. This could cause blending artifacts as can be seen in Figure 14. The mesh does not have enough number of vertices to cover the patch representing the changes. Poisson Surface Reconstruction [23] converts a point cloud into a triangle mesh. The mesh resolution is controlled by a depth parameter. Increasing the depth parameter increases the mesh resolution. However, it leads to surface artifacts on the mesh. Instead, the depth parameter is kept to a reasonable number and triangle subdivision is used to increase the resolution. Triangle subdivision is the process of splitting the triangle into smaller triangles. This could be done at both in a global and local scale. Unlike Shan et al.’s approach, patch based texture reconstruction is not dependent on mesh resolution.

Figure 14. Mesh resolution is not sufficient to accurately represent the chessboard texture. Triangle subdivision can be done to increase the mesh resolution.
4.4. Visual Turing Test

Due to the nature of the problem, there are not many techniques to quantify the realism of the texture. One definition of realism is that people should be unable to tell apart a rendered image from original photo. For any photo, I can produce a rendering (for the same viewpoint and lighting conditions) that appears so realistic that you cannot tell which one is real. This is the grand challenge problem, called the Visual Turing Test for scene reconstruction. As part of the Visual Turing Test, Shan et al. compared rendered images against their original reference images. This is possible because the shading model accounts for the texture and environment factors (lighting) separately. The lighting parameters of each image are computed as part of texture reconstruction process and this information is supplied to the rendering (MeshLab). Patch based texture reconstruction performs a color adjustment in global and local regions. Hence, the luminance values of images are merged into the texture. Figure 15 shows a rendered image on the left and the reference photo on the right. Since the light information of image is available, it is possible to render invisible parts in the scene under the same lighting conditions.

![Figure 15. Visual Turing Test: Rendered image on left and reference photo on right.](image)

4.5. Occluder removal

In an unconstrained data set, the images are not consistent and there are occlusions in the images. Both the approaches described in the thesis, have their visibility estimation to remove the occluders in the image. Shan et al. [1] uses correlation with average color from the most frequent images in the neighborhood to determine if it is an occluder or any other foreground objects. Waechter et al. [2] uses a similar approach. An iterative modified mean-shift algorithm is used to determine inliers and outliers (occluders). Figure 16 and 17 show examples from both the approaches. In some cases such as in Figure 18, a foreground object, which in reality belongs to the scene, is categorized as occlusion. This is common with thin structures as they either occupy insignificant area in some of the long shot images. The proposed frameworks are better than some other approaches [33], where user interactions are required to mark occluding objects to keep them away from being used in texture.
Figure 16. Correlation with average color is the approach used in Shan et al.’s approach [1] to remove any occluders in foreground. Left: Input reference photo with occlusion; Right: Rendered image.

Figure 17. A modified mean-shift algorithm is used by Waechter et al. [2] to remove occlusions. Left: Pedestrian used in the rendering when photo consistency check is not performed. Right: Mesh textured with photo consistency check.

Figure 18. An example where part of the scene (lamp post) is categorised as an occluder and removed in the reconstruction.
4.6. Indoor scene

Vertex based texture reconstruction has the complexity in its model to segment the environment conditions of the images from the actual texture. The shading model can be modified for different lighting conditions such as an indoor scene or night shots. Images of a plastic model of Eiffel Tower (shown in Figure 19: left) was used in a controlled setup to test indoor scene reconstruction. The images were captured in an indoor scene with one light source. The same shading model described in Section 3.1.2 was used. However, a modification was made to the directional lighting model. As it was known that there is only one directional light source at the same location, the directional light parameters were kept constant between the images. The reduced parameter set is $\Theta = \{ N_i, a_i, L, k^{sky}, k^{sun}, \delta_i \}$. One of the sample images used and reconstructed scene with texture is shown in Figure 19.

![Figure 19. Plastic model of Eiffel tower captured in indoor scene and reconstructed with a modified shading model. Left: Example image; Right: textured reconstruction](image-url)
5. DISCUSSION

Most texture reconstruction approaches deal with only indoor constrained datasets. While these might be viable for small-scale reconstructions, they do not satisfy the requirements for a large-scale reconstruction. The models employed for some of these approaches are quite complex and take several hours of execution even for less than ten images. Large-scale texture reconstructions discussed in this thesis take similar time for hundreds of images. Other than having a simpler model to solve, they benefit from integrating with 3D scene reconstruction framework. The vertex based texture reconstruction uses the visibility estimation from MVS to speed up the visibility estimation stage. In our experiments, we used the visibility mask from MVS directly and managed to get visibly appealing results.

Texture reconstruction is the key step towards achieving realism in reconstructed scene. There is not much research about parameters to calculate this quality. Existing benchmarks in the field of scene reconstruction, such as Middlebury [56] and KITTI [57] concentrate on geometry and not on the appearance of the scene. These benchmarks typically compare the location accuracy of the point cloud or surface with ground truth data (usually obtained through other sensors such as laser scanner). To measure the quality of reconstructed texture, we need ground truth data for comparison. There is not a clear way of measuring ground truth texture. In addition, texture is represented differently either as reflectance parameter per vertex or as a texture map. In vertex based texture reconstruction, albedo and light information are coupled. The quality of the texture can be analyzed on the rendered images. Visual Turing Test does the same by using judgments from the test group. Instead, a quantitative comparison of the rendered (and relighted) image with the source image from the same viewpoint would give a measurable result. This is not a simple pixel-by-pixel comparison, as we will have to account for null space and occluder differences between reference and rendered image. Null space refers to the empty holes in the 3D scene that are rendered on the image. They are dynamic and cannot be rejected by a simple rectangular region of interest. A binary mask can be created by applying an appropriate geometry culling technique on the rendering and only pixels inside the binary mask can be used for quantitative error computation.

Most available large-scale datasets are for outdoor environments and images under consideration are captured in daylight. It would be interesting to see the results from nighttime and lowlight shots of the same scene. In fact, it would be a better way to test the decoupling of lighting and albedo parameters. The shading model would need to be modified accordingly. Similar experiments for indoor scenes and other scenarios will need to be performed as well.
6. CONCLUSION

In this thesis, we have studied and discussed the state of the art texture reconstruction approaches for unconstrained large collection of photos. The texture reconstruction approaches are integrated with geometry reconstruction framework. It uses sub products from the latter such as visibility set. A closer look and analysis of Shan et al.’s [1] vertex based and Waechter et al.’s [2] patch based approaches was done.

The vertex based texture reconstruction system computes reflectance parameters of vertices and lighting conditions of the images. A Matlab implementation of vertex based texture reconstruction was done and used with outputs from VisualSFM. The key highlight of the method is the framework to perform Visual Turing Test at a large scale. The computed lighting conditions were used to relight and render the scene at the same viewpoint as the original image. The quality of the texture is linked with mesh resolution. Unlike the former, patch based texture reconstruction selects a view per face and adds a color correction to reduce seam visibility between faces. Typical effects in uncontrolled datasets such as out of focus, image scale variation and unreconstructed occluders are automatically handled automatically without any user interactions. Additionally, it is much faster than vertex based approach as the number of parameters is less and still produces visibly appealing results.

As a step towards solving the Visual Turing Test, there are still some limitations in the discussed approaches. There exist coupling between albedo and lighting parameters. Only outdoor environments in daylight are modeled and tested. Although, some initial tests have been done in indoor scenes, more analysis is required. Other lighting conditions such as nighttime shots need to be modeled in the framework.
7. REFERENCES


[37] Algorithm to texture 3D reconstructions from multi-view stereo images. www.gris.informatik.tu-darmstadt.de/projects/mvs-texturing


8. APPENDICES

Appendix 1.  Projection equation

Appendix 2.  Focal length initialization

Appendix 3.  Random Sample Consensus
Appendix 1. Projection equation

A perspective camera can be represented by eleven parameters. The camera orientation (rotation) $\omega$ and camera center (translation) $c$ is represented by three parameters each. These are the camera extrinsic parameters. There are seven camera intrinsic parameters representing focal length (in x and y direction), skewness and offset between projection center and image center. Making the common assumptions of modern day cameras that pixels are square, no skewness and that the projection center is coincident with image center, it is reduced to one parameter, focal length $f$. The camera intrinsic matrix will be $K = diag[f, f, 1]$.

Each point is parameterized by 3D position $X$ and each camera is parameterized by camera parameter $C = [\omega, c, f]$. Let $P(C,X)$ be the equation mapping a 3D point $X$ to its 2D projection in a camera with parameters $C$. $P$ transforms the $X$ to homogenous image coordinates and performs the perspective decision. Structure from Motion tries to reduce the sum of reprojection errors for $m$ points captured by $n$ cameras as described in equation (3).

$$P'(C,X) = KR(P - c)$$

$$P(C,X) = [-P'_x/P'_z - P'_y/P'_z]^T$$

$$g(C,X) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \left\| q_{ij} - P(C_i, X_j) \right\|^2$$
Appendix 2. Focal length initialization

When initializing a new camera, an estimate of focal length is usually used. Most cameras nowadays embed useful metadata about photo into the image. The information is encoded in Exchangeable Image File Format (EXIF) tag of image. The information often includes exposure time, focus, aperture, image dimensions and focal length. Note that the focal length need to be in pixel units that can be directly used in SfM. However, the focal length in metadata is usually in millimeter $f_{mm}$. Given the size of image sensor in millimeter $sensor_{mm}$ and dimensions of image in pixels $(w_{pix}, h_{pix})$, the focal length in pixel units $f_{pix}$ can be calculated as in equation (15). The required inputs are available in EXIF tag.

$$f_{pix} = \frac{f_{mm}}{sensor_{mm}} \left(\max(w_{pix}, h_{pix})\right)$$ (15)
Appendix 3. Random Sample Consensus

Random Sample Consensus (RANSAC) is an iterative method to fit mathematical model to a set of observed data. The method is used to reject outliers from the observed data and fit a model to the remaining data. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability. A basic assumption is that majority of the data is inliers which fits more closely to the model. And outliers are data that do not fit the model by a large error. The outliers can be extreme values of noise or erroneous measurements. The procedure of the algorithm is below:

1. Select a random subset of the original data. The size of the subset is the minimum number of inputs required to estimate the model.
2. Fit a model to the subset.
3. Check if the remaining data fits the model. If the error value is larger than a particular threshold, it is categorized as an outlier. The set of data that fits the model is called the consensus set.
4. The estimated model is reasonably good if sufficiently many observations belong to the consensus set.
5. Repeat steps 1 to 4 with a valid exit condition. This could be maximum number of iterations or once the consensus set is large enough.
6. Additionally, reestimate the model using all members of the consensus set to improve the model.