A Complete Statistical Inverse Ray Tracing Approach to Multi-View Stereo

Shubao Liu
Brown University, Providence, RI
sbliu@lems.brown.edu

David B. Cooper
Brown University, Providence, RI
cooper@lems.brown.edu

Abstract

This paper presents a complete solution to estimating a scene's 3D geometry and appearance from multiple 2D images by using a statistical inverse ray tracing method. Instead of matching image features/pixels across images, the inverse ray tracing approach models the image generation process directly and searches for the best 3D geometry and surface reflectance model to explain all the observations. Here the image generation process is modeled through volumetric ray tracing, where the occlusion/visibility is exactly modeled. All the constraints (including ray constraints and prior knowledge about the geometry) are put into the Ray Markov Random Field (Ray MRF) formulation, developed in [10]. Differently from [10], where the voxel colors are estimated independently of the voxel occupancies, in this work, both voxel occupancies and colors (i.e., both geometry and appearance) are modeled and estimated jointly in the same inversey ray tracing framework (Ray MRF + deep belief propagation) and implemented in a common message passing scheme, which improves the accuracy significantly as verified by extensive experiments. The complete inverse ray tracing approach can better handle difficult problems in multi-view stereo than do traditional methods, including large camera baseline, occlusion, matching ambiguities, color constant or slowly changing regions, etc., without additional information and assumptions, such as initial surface estimate or simple background assumption. A prototype system is built and tested over several challenging datasets and compared with the state-of-the-art systems, which demonstrates its good performance and wide applicability.

1. Introduction

This paper addresses the problem of capturing 3D geometry and appearance from multiple 2D images taken from different views, which has wide applications in virtual reality, entertainment, human-computer interface, surveillance, navigation, and high-level vision tasks (e.g., visual tracking, classification, recognition, scene analysis for robotic task accomplishment), etc. As an extension of stereo reconstruction, the multi-view stereo problem is complicated by the following factors. **i) ambiguity of matching.** Most algorithms discard homogeneous regions and leave holes in the reconstruction. More complete reconstructions are to estimate the region based on the visibility constraints and regularity of the surface geometry. **ii) occlusion.** In complex environments, there are strong occlusions between objects and within objects (self-occlusion). While the effect of occlusion is not obvious in small-baseline stereo, it becomes important for the accuracy and completeness of the reconstruction in multi-view stereo, where the baseline between two views can be large. **iii) surface topology.** The topology of natural scenes can be very complex, e.g, flowers, trees, compared with planar buildings. **iv) background and transient clutter.** Many algorithms require a good initial surface estimate, usually gotten from object silhouettes. In the presence of the background clutter, it is not easy to extract silhouettes reliably. Most multi-view stereo algorithms also assume the scene is static. However in many applications, moving objects come and go, such as pedestrians and vehicles on a street, visitors in a museum, etc. They can completely confuse the reconstruction algorithm or lead to ghost effects.

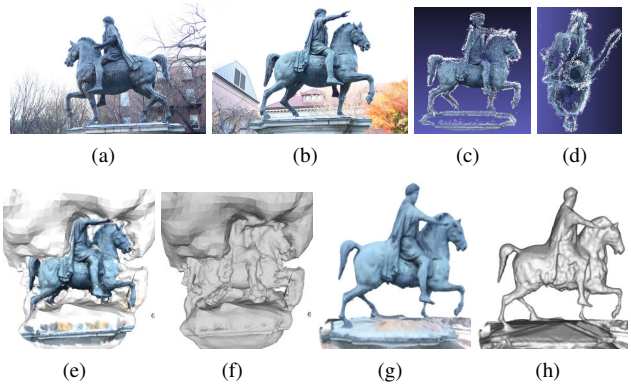


Figure 1. From left to right: (a, b) two out of 29 cluttered input images; (c, d) side and top views of the oriented point cloud from PMVS2 [5]; (e, f) textured and non-textured models from PMVS2+PoissonRecon [7]; (g, h) textured and non-textured models from our multi-view reconstruction algorithm

In the last decades, large numbers of research papers have been published on addressing these problems. These approaches can be broadly categorized into three classes based on the adopted 3D geometric representations: feature-based point cloud approach, surface evolution approach, and volumetric approach. 1) The feature-based point cloud approach implements the pipeline of feature extraction — feature matching — triangulation. There are

two popular variants of this approach, depth fusion and feature expansion. Depth fusion is a direct extension of stereo reconstruction. The depth maps from stereo pairs are merged to remove redundant and noisy points, resulting in a clean point cloud, from which a surface can be reconstructed [2, 9]. Another successful feature-based point cloud reconstruction method is the feature expansion approach, represented by the work of Furukawa and Ponce [5]. The feature expansion approach starts from a set of sparse feature matchings, and then the matchings are propagated to their neighbor pixels and refined to create a dense reconstruction. In the feature-based approach, the occlusion is handled sub-optimally, by either cross-checking as in stereo, or visibility filtering [5]. This approach works well for textured objects, for example buildings, but cannot handle textureless objects and objects of complex geometry, e.g., Fig. 9. The output of this approach is a set of unconnected points, which can be connected to form a mesh by some meshing algorithms, e.g., Poisson reconstruction [7]. But due to the irregularity of the point cloud and loss of information in the point cloud representation, the recovered shape could have incorrect shape and topology (See Fig. 1). In general, the feature-based approach can handle background and transient clutter easily. 2) The iterative surface approach adopts a surface representation of the scene, either explicitly with a mesh [3] or implicitly with a level set function [4, 6]. This approach starts from an initial surface, and iteratively refines it to increase its consistency with observed images. The convergence of the local refinement highly depends on the quality of the initial surface, which has become the main disadvantage of this approach. The background and transient clutter complicate the problem further by making the silhouettes hard to extract in these cases. 3) Volumetric methods work with occupancy volume directly, which can be easily converted to mesh surface representation if needed. Based on how to handle the visibility, these methods can be categorized into two groups: one relies on an initial surface to compute the visibility, and is represented by the graph cuts based methods [15, 17]; the other does not assume any prior visibility knowledge, and is represented by the work on space carving and its probabilistic variants [12, 8, 1]. Similarly to the surface evolution approach, the graph cuts based approaches also require an initial shape to estimate the visibility of each voxel, which strongly limits its general applicability. The space carving and its variants are general to handle large varieties of scenes. Due to the adopted greedy solution, it has not produced reconstruction with good geometry accuracy.

Different from most of the above approaches, Liu and Cooper [10] tackled the problem by first modeling the image generation process directly and then searching for the best model to explain these observations. That is, the multi-view stereo is treated as an inverse problem: instead of solving the reconstruction problem directly, all the constraints are first modeled, and 3D reconstruction is solved as finding the best model that satisfies all the constraints. In [10], the formulated optimization problem is solved with a hybrid approach: the geometry, modeled with voxel occupancies, is estimated by solving an MRF inference problem, given the voxel colors; and the voxel colors are estimated separately with a statistical robust estimation method. In this paper, we show that both voxel occupancies and colors (i.e., geometry and appearance) can be modeled and estimated jointly with a message passing algorithm, which significantly improves the reconstruction accuracy of both geometry and appearance. The voxel occupancy estimation algorithm in [10] is also reformulated to compute the intermediate terms that are shared with the color estimation algorithm developed later. Another improvement on accuracy is achieved through more accurate modeling of the ray-voxel intersection relationship. To achieve sub-voxel accuracy, the ray-voxel intersection relationship is improved from binary to fractional, and the ray is more accurately modeled with a cone instead of a line.

notation	meaning
Ω	the set of voxels indices
$\mathcal{N}$	voxel pair-wise neighborhood
x_i	vector variable for the ith voxel in the volume
x_i^o	occupancy component of voxel variable x_i
x_i^c	RGB color component of voxel variable x_i
x_{ri}	vector variable for ith voxel on the ray r
x_{ri}^o	occupancy component of voxel variable x_{ri}
x_{ri}^c	RGB color component of voxel variable x_{ri}
X_r	the set of variables for the voxels on the ray
$M_{i\to r}^o(x_{ri}^o)$	the message from ray r's ith voxel occupancy variable to the ray clique, i.e., $M_{x_{ri}^o\to r}(x_{ri}^o)$
$M_{i\to r}^o$	normalized voxel occupancy to ray message i.e., $M_{i\to r}^o(1) - M_{i\to r}^o(0)$
$M_{r\to i}^o(x_{ri}^o)$	the message from ray clique r to its ith voxel occupancy variable, i.e., $M_{r\to x_{ri}^o}(x_{ri}^o)$
$M_{r\to i}^o$	normalized ray to voxel occupancy message i.e., $M_{r\to i}^o(1) - M_{r\to i}^o(0)$
$M_{r\to i}^c$	the message sent from ray clique r to the its ith voxel's color variable

Table 1. Notations

2. Multi-View Stereo as an Inverse Ray Tracing Problem

2.1. Volumetric Ray Tracing

In statistical inverse ray tracing, a volumetric 3D model X is estimated from a set of image observations $\{I\}$ by modeling the prior $P(X)$ and the image generation process $P(\{I\}|X)$. Here the volume is divided into a set of voxels, so X represents the whole set of voxel variables $\{x_i\}$, with each voxel variable x_i having two components, binary occupancy x_i^o (value 0 for empty, 1 for solid) and RGB color x_i^c. To model $P(\{I\}|X)$, i.e., the process of generating image observations given a 3D model, we use the volumetric ray tracing. By volumetric ray tracing (here, the special case, volumetric ray casting), we know that the observation of each ray depends only on the voxels that the ray passes through. Here we use X_r to denote the set of voxels ray r passes through. The voxels are in the ray traversal order starting from the viewer. Similarly, X_r^o and X_r^c, respectively, denote the occupancies and colors of X_r. By volumetric ray tracing with a Lambertian model, the ray r's

rendered value can be computed as

$$\Upsilon_r(X_r) = \begin{cases} x^c_{r1}, & X^o_r = 1, \times, \times, \times, \times, \cdots \\ x^c_{r2}, & X^o_r = 0, 1, \times, \times, \times, \cdots \\ \cdots & \\ x^c_{ri}, & X^o_r = \underbrace{0, \cdots, 0}_{i-1}, 1, \times, \cdots \\ \cdots & \end{cases} \quad (1)$$

where $\times$ denotes either value 0 or 1, X^o_r is a vector of occupancy values for the voxels on the ray r sorted in the traversal order of the viewing ray. Eq. (1) says that the color of the view ray is equal to the color of the first solid voxel that is visible along the ray. Hence, $\Upsilon_r(X_r)$ is a vector of observable voxel colors in ray r conditioned on voxel occupancy states for ray r.

2.2. Ray Markov Random Field

Define the difference between the rendered value Υ_r and observed value I_r as the ray energy:

$$E_r(X_r) = ||I_r - \Upsilon_r(X_r)||^2, \quad (2)$$

which measures the 3D model's consistency with the measured pixel color.

In [10], a novel MRF model, called Ray MRF, is proposed to model not only the interaction between neighbor voxels, but also the interaction between voxels in the volume and pixels in the observed images. The ray energy defined in Eq. (2) determines the ray clique in Ray MRF, modeling the image generation process. Besides the ray clique, there are two other kinds of cliques in our model: unary cliques and pair-wise cliques, modeling the prior distribution of the whole set of voxel variables. Unary voxel occupancy cliques (E^o_{uv}) model the prior knowledge of voxels' occupancies (uv stands for "unary voxel"); pair-wise occupancy cliques (E^o_p) model the continuity of the 3D volume; pair-wise color cliques (E^c_p) model the spatial smoothness of voxel colors. Formally, the entire MRF can be expressed in the following energy form:

$$E(X) = \sum_{r \in R} E_r(X_r) + w^o_p \sum_{\langle i,j \rangle \in \mathcal{N}} E^o_p(x^o_i, x^o_j) + w^c_p \sum_{\langle i,j \rangle \in \mathcal{N}} E^c_p(x^c_i, x^c_j) + w^o_{uv} \sum_{k \in \Omega} E^o_{uv}(x^o_k) \quad (3)$$

where $E_r(X_r)$ is the clique energy for the ray r, and X_r is the set of voxels that ray r passes through; $E^o_p(x^o_i, x^o_j)$ is the pair-wise occupancy clique energy, and w^o_p is its weight; $E^c_p(x^c_i, x^c_j)$ is the pair-wise color clique energy, with w^c_p being its corresponding weight; and $E^o_{uv}(x^o_k)$ is the unary occupancy clique energy, and w^o_{uv} is its weight. Each energy function is defined as follows:

$$E^o_p(x^o_i, x^o_j) = \begin{cases} 0, & x^o_i = x^o_j \\ 1, & x^o_i \neq x^o_j \end{cases} \quad (4)$$

$$E^c_p(x^c_i, x^c_j) = ||x^c_i - x^c_j||^2 \quad (5)$$

$$E^o_{uv}(x^o_k) = (x^o_k - 1)^2 \quad (6)$$

Our goal is to find the best 3D model X^*, which minimizes the energy defined in Eq. (3), i.e.,

$$X^* = \arg\min_X E(X). \quad (7)$$

3. Voxel Occupancy Estimation

Formulating the image-based 3D modeling problem with Ray MRF is only the first step, the next and most critical step is to get an efficient algorithm to compute the maximum a posteriori (MAP) solution to the Ray MRF model, i.e., the best 3D model to explain all the image observations. There are many optimization methods for MAP estimation of MRFs, including local algorithms such as gradient descent and its variants, sampling methods such as Markov Chain Monte Carlo (MCMC) sampling, and approximate algorithms such as loopy belief propagation and graph cuts (GC), etc. Among them, LBP and GC are two of the most popular MRF inference algorithms in computer vision for their good convergence speed and quality. As in [10], LBPLBP is used here.

There are three kinds of cliques in Ray MRF: the unary clique, pair-wise clique, and ray clique. Since the message from random variable node to its adjoining factor node is simple and the message passing for the unary and pairwise clique is similar as traditional MRF, the following study will focus on the message sent from the ray clique to its participating voxels, i.e., $M^o_{r \to i}(x^o_{ri})$, where x^o_{ri} is the occupancy variable of the ith voxel on the ray r. Through belief propagation, the message $M^o_{r \to i}(x^o_{ri})$ can be computed as:

$$M^o_{r \to i}(0) = \min_{X_r : x^o_{ri} = 0} \left\{ E_r(X_r) + \sum_{j \neq i} M^o_{j \to r}(x^o_{rj}) \right\} \quad (8)$$

$$M^o_{r \to i}(1) = \min_{X_r : x^o_{ri} = 1} \left\{ E_r(X_r) + \sum_{j \neq i} M^o_{j \to r}(x^o_{rj}) \right\} \quad (9)$$

It is a combinatorial optimization problem with n random variables (the number of voxels a ray passes through is about $100 - 1000$ in normal case). The number of combinations is $n \times 2^n$ (i.e. $100 \times 2^{100} - 1000 \times 2^{1000}$), which is too large to enumerate in finite time.

Studies in [10] revealed that there is a "deep factorization" structure of the problem that can be explored by dynamic programming to solve the large scale combinatorial optimization. The final algorithm for computing $M_{r \to i}$ is summarized in Algorithm 1, which is a reformulation of the algorithm proposed in [10] to explicitly compute the intermediate variables useful for the color estimation later. In Algorithm 1, $M^o_r[i], E^o_r[i], E^c_r[i], E^{o\uparrow}_r[i], E^*_r[i], E^{*\uparrow}_r[i], E^{*\downarrow}_r[i]$ are all intermediate temporary variables, some of which will be reused later in color estimation.

For now, just pay attention to the computational complexity of the algorithm, by counting the number of "for loops": there are 5 sweeps of the voxels on each ray (3 forward sweeps and 2 backward sweeps). That is, the computational complexity is $5n$, or $O(n)$. For a clique with n

Algorithm 1 Message passing from ray cliques to voxel occupancy variables, i.e., computing $M^o_{r\to i}$

for each ray r **do**
  for $i = 1$ to n **do**
    $M^o_r[i] = M^o_{i\to r}$
    $E^o_r[i] = \min(0, M^o_r[i])$
    $E^c_r[i] = (I_r - x^c_{ri})^2$
  end for
  $E^{o\uparrow}_r[n] = 0$
  $E^*_r[n] = E^c_r[n] + M^o_r[n]$
  for $i = n-1$ to 1 **do**
    $E^{o\uparrow}_r[i] = E^{o\uparrow}_r[i+1] + E^o_r[i+1]$
    $E^*_r[i] = E^c_r[i] + M^o_r[i] + E^{o\uparrow}_r[i]$
  end for
  $E^{*\uparrow}_r[n] = \infty$
  for $i = n-1$ to 1 **do**
    $E^{*\uparrow}_r[i] = \min(E^{*\uparrow}_r[i+1], E^*_r[i+1])$
  end for
  $E^{*\downarrow}_r[1] = \infty$
  for $i = 2$ to n **do**
    $E^{*\downarrow}_r[i] = \min(E^{*\downarrow}_r[i-1], E^*_r[i-1])$
  end for
  for $i = 1$ to n **do**
    $M^o_{r\to i}(0) = \min(E^{*\downarrow}_r[i] - E^o_r[i], E^{*\uparrow}_r[i])$
    $M^o_{r\to i}(1) = \min(E^{*\downarrow}_r[i] - E^o_r[i], E^*_r[i] - M^o_r[i])$
    $M^o_{r\to i} = M^o_{r\to i}(1)$ - $M^o_{r\to x_i}(0)$
  end for
end for

binary random variables, the direct implementation of belief propagation takes $O(n\times 2^n)$ number of operations. Here for optimized message passing of ray-clique, the computation is reduced to $O(n)$. The dramatic reduction of the computational complexity changes the Ray MRF inference from intractable to efficient.

4. Voxel Color Estimation

In the last section, we have discussed the estimation of voxel occupancies, given voxel colors. Here the focus is turned to the voxel color estimation, given the voxel occupancies. Then the optimization problem can be solved by alternately estimating voxel occupancies and colors, as shown in Fig. 2. A natural question arises: can we estimate voxel colors with a similar approach as the occupancy estimation by the deep belief propagation? One obvious obstacle towards this direction is that the color is a 3-dimensional real-valued random variable, instead of binary variable. Treating it as a discrete variable is also problematic, because the number of states is too large ($256^3 = 16777216$). In [10], voxel colors are estimated through a robust statistical estimation method, independent of voxel occupancies. The problem with this approach is that the color estimation is sub-optimal and it requires another parameter related to the color variance across images. Another possible solution is to use non-parametric belief propagation by approximating the messages with mixture-of-Gaussians [16]). But is there a simpler solution?

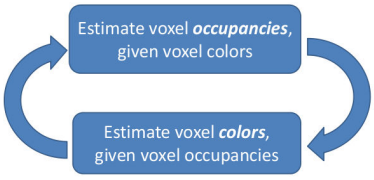


Figure 2. Alternating estimation of voxel occupancies and colors

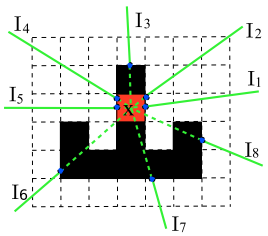


Figure 3. Voxel's visibilities in different views, given binary voxel occupancies. White squares denote empty voxels; dark squares denote solid voxels.

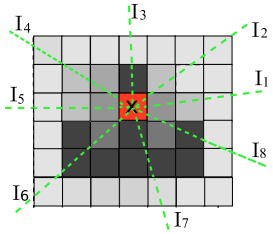


Figure 4. Voxel's visibilities in different views, given probabilistic voxel occupancies. Darker squares denote higher probability of the voxel being solid.

4.1. Joint Voxel Occupancy and Color Estimation

As shown in Fig. 3, the highlighted voxel is visible in 4 views. Then the marginal mean estimation of the color is the mean of the color of these 4 visible rays. What does this mean to our color estimation problem? It means that *given binary voxel occupancies, the voxel color can be estimated in a closed form*. In Ray MRF, the visibility of each voxel along a ray is measured in probability, as illustrated in 4. It turns out that given each voxel's visibility probability along each ray, the voxel's color can also be estimated in a closed form with Gaussian prior distribution. Formally the color of a voxel, x^c, is estimated as

$$\overline{x^c} = \frac{\sum_{r\in R_x} \overline{\mathsf{vis}}_r(x) I_r}{\sum_{r\in R_x} \overline{\mathsf{vis}}_r(x)} \tag{10}$$

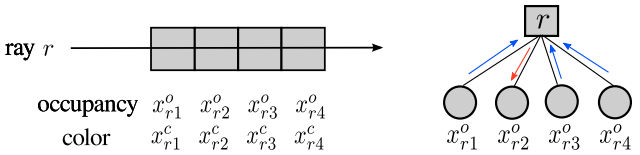
where

$$\begin{aligned}&\overline{\mathsf{vis}}_r(x)\\ =&\mathbb{E}_{x^o_{r(\ell^x_r+1)}}[\cdots \mathbb{E}_{x^o_{rn}}[P_r(\underbrace{0,0,\cdots,0}_{\ell^x_r-1},1,x^o_{r(\ell^x_r+1)},\cdots,x^o_{rn})]\cdots]\\ =&P_r(\underbrace{0,0,\cdots,0}_{\ell^x_r-1},1,\times,\cdots,\times).\end{aligned} \tag{11}$$

is the mean visibility of voxel x along ray r; R_x is the set of rays that pass through the voxel x; ℓ^x_r is the index of voxel x on the ray r; $\mathbb{E}_{x^o_{ri}}[\cdot]$ is the probabilistic expectation operator with respect to the occupancy variable x^o_{ri}, $P_r(x^o_{r1},\cdots,x^o_{rn})$ is the joint probability of the voxel occupancy variables on the ray r.

The maximal estimation of the color is computed as

$$x^{c*} = \frac{\sum_{r\in R_x} \mathsf{vis}^*_r(x) I_r}{\sum_{r\in R_x} \mathsf{vis}^*_r(x)} \tag{12}$$



(a) ray-voxel relationship (b) message passing from r to x^o_{r2}

Figure 5. Toy ray-clique example

where

$$\begin{aligned}&\mathsf{vis}^*_r(x)\\ =&\max_{x^o_{r(\ell^x_r+1)}}\{\cdots\max_{x^o_{rn}}\{P_r(\underbrace{0,0,\cdots,0}_{\ell^x_r-1},1,x^o_{r(\ell^x_r+1)},\cdots,x^o_{rn})\}\cdots\}\\ =&P^*_r(\underbrace{0,0,\cdots,0}_{\ell^x_r-1},1,\times,\cdots,\times).\end{aligned}\tag{13}$$

is the maximal visibility of voxel x along ray r. To be consistent with the occupancy estimation, we choose to stick with the maximal estimation, instead of the mean estimation, for the later derivations. How do we compute the maximal visibilities? Recall that in belief propagation, we can compute not only each variable's marginal/maximal distribution, but also a group of variables' joint marginal/maximal distribution, or more accurately speaking, a clique (or factor)'s marginal distribution.

Toy Example: A toy example $E_r(x_{r1},x_{r2},x_{r3},x_{r4})$ (Figs. 5) is used to show the essential ideas. The maximal visibility of the 2nd voxel can be computed as

$$P^*_r(0,1,\times,\times)=\frac{e^{-E^*_r(0,1,\times,\times)}}{e^{-E^*_r(\times,\times,\times,\times)}}\tag{14}$$

where $E^*_r(0,1,\times,\times)$ is computed as

$$\begin{aligned}&E^*_r(0,1,\times,\times)\\ =\;&\min(E^*_r(0,1,0,0),E^*_r(0,1,0,1),E^*_r(0,1,1,0),E^*_r(0,1,1,1))\\ =\;&\min\left\{\begin{array}{l}E_r(0,1,0,0)+M^o_{2\to r},\\ E_r(0,1,0,1)+M^o_{2\to r}+M^o_{4\to r},\\ E_r(0,1,1,0)+M^o_{2\to r}+M^o_{3\to r},\\ E_r(0,1,1,1)+M^o_{2\to r}+M^o_{3\to r}+M^o_{4\to r}\end{array}\right\}\\ =\;&E_r(0,1,\times,\times)+M^o_{2\to r}+\min(0,M^o_{3\to r})+\min(0,M^o_{4\to r})\end{aligned}\tag{15}$$

$E^*_r(\times,\times,\times,\times)$ is computed as

$$\begin{aligned}&E^*_r(\times,\times,\times,\times)\\ =\;&\min\left\{E^*_r(1,\times,\times,\times),E^*_r(0,1,\times,\times),E^*_r(0,0,1,\times),E^*_r(0,0,0,1)\right\}.\end{aligned}\tag{16}$$

General Case: For the case with n voxels on a ray, the maximal visibility of the ith voxel along a ray can be computed as following. Denote $\mathsf{vis}^{*i}_r = P^{*i}_r = P^*_r(\underbrace{0,0,\cdots,0}_{i-1},1,\times,\cdots,\times)$. The associated maximal energy distribution of the ith voxel along the ray is $E^{*i}_r = E^*_r(\underbrace{0,0,\cdots,0}_{i-1},1,\times,\cdots,\times)$. It can be computed as

$$E^{*i}_r=E^i_r+M_{i\to r}+\sum_{j=i+1}^{n}\min(0,M_{j\to r}).\tag{17}$$

Denote $E^{*0}_r=E^*_r(\times,\cdots,\times)$. It can be computed as

$$E^{*0}_r=E^*_r(\times,\cdots,\times)=\min_{i=1,\cdots,n}E^{*i}_r.\tag{18}$$

Then the maximal visibility of the ith voxel along ray r can be computed as

$$P^{*i}_r=\frac{e^{-E^{*i}_r}}{e^{-E^{*0}_r}}=e^{-(E^{*i}_r-E^{*0}_r)}.\tag{19}$$

A good news is that E^{*i}_r has been computed during the occupancy estimation in Algorithm 1 (the term $E^*_r[i]$ in line 11), which can be re-used here.

4.2. Message-Passing Algorithm for Color Estimation

The above operation can also be implemented with a message passing scheme. Denote $M^c_{r\to i}$ as the message sent from the ray r to voxel color variable x^c_i. Note that the message $M^c_{r\to i}$ is not a distribution as in the belief propagation. It has two values: visibility of the voxel x_{ri} on the ray r, and ray color I_r weighted by the visibility, i.e.,

$$M^c_{r\to i}:=(\mathsf{vis}^{*i}_r,\mathsf{vis}^{*i}_r*I_r).\tag{20}$$

The value of each voxel color can be computed by accumulating the messages received from all the rays that passes through the voxel:

$$x^{*c}=\frac{\sum_{r\in R_x}M^c_{r\to\ell^x_r}[2]}{\sum_{r\in R_x}M^c_{r\to\ell^x_r}[1]}\tag{21}$$

where $M^c_{r\to\ell^x_r}[1]$ and $M^c_{r\to\ell^x_r}[2]$ denote respectively the first and second value of the 2-d vector $M^c_{r\to\ell^x_r}$. With this message passing implementation of both occupancy and color estimation, the voxel updating is totally parallel with the ray, i.e., the message passing for rays can be computed in parallel. The above voxel color updating algorithm is summarized in Algorithm 2. The system diagram in Fig. 6 shows the whole information flow from the input images to the final surface mesh.

Algorithm 2 Computing the message from ray clique to voxel variables (including both occupancy and color), i.e., computing $M^o_{r\to i}$ and $M^c_{r\to i}$

1: **for** each ray r **do**
2: compute the occupancy messages as in Algorithm 1.
3: $E^{*0}_r=\infty$
4: **for** $i=1$ to n **do**
5: $E^{*0}_r=\min(E^{*0}_r,E^*_r[i])$
6: **end for**
7: **for** $i=1$ to n **do**
8: $\mathsf{vis}^*_r[i]=e^{-(E^*_r[i]-E^{*0}_r)}$;
9: $M^c_{r\to i}=(\mathsf{vis}^*_r[i],\mathsf{vis}^*_r[i]*I_r)$
10: **end for**
11: **end for**

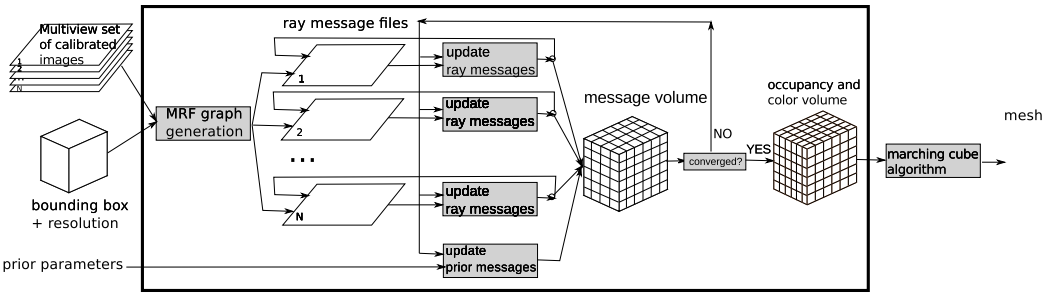


Figure 6. System diagram of the complete inverse ray tracing approach to multi-view stereo

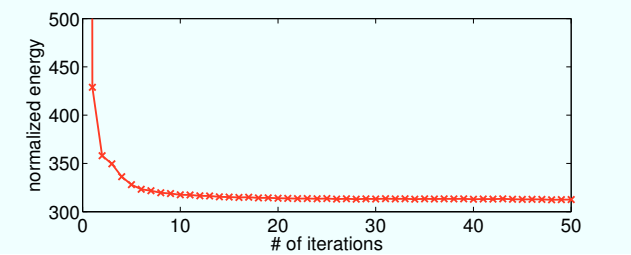


Figure 7. Convergence behavior of the proposed algorithm: the change of Ray MRF's normalized energy as the number of iterations increases.

4.3. Algorithm Convergence

The theoretical convergence property of the proposed algorithm is yet to be studied. Here its typical convergence behavior is reported empirically[1]. Fig. 7 shows the change of the normalized energy (defined as the whole Ray MRF energy divided by the number of rays) with respect to the number of iterations. It can be seen that the energy decreases exponentially and the algorithm converges in about 10-20 iterations.

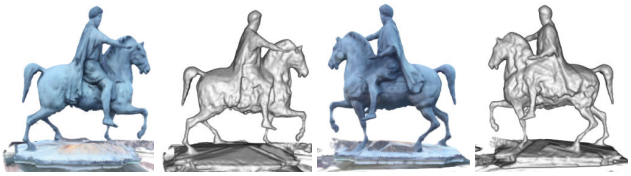
5. Experimental Results

The above developed algorithm for multi-view stereo is evaluated through a variety of datasets[2], reflecting different challenges in multi-view stereo and how our system responds to them. The performance of the proposed algorithm is compared with [10] and one of the state-of-the-art multi-view stereo algorithms, PMVS2 [5].

5.1. Horse Dataset (horse29)

Horse dataset is composed of 29 images, captured in a rough circle around a 6-meter tall horse statue. During capture, the camera is pointed upwards because of the height of the statue. Fig. 1 (a, b) show two of the 29 input images. It

(a) Results from [10]: two views of textured and non-textured model



(b) Our results: two views of textured and non-textured model

Figure 8. Comparison between [10] and our system on horse29 dataset

Figure 9. Two sample images of the elephants40 dataset

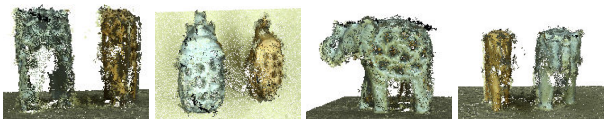


Figure 10. PMVS2's oriented point cloud output

can be seen that the horse scene is cluttered with trees and buildings in the background. Without silhouette inputs, the iterative surface approach will have difficulty in producing meaningful result for this dataset.

First our results are compared with results from [10]. Fig. 8 shows two different views of the generated models from [10] and our system. Although [10] has clearly recovered the object topology from the cluttered input images, it fails to reconstruct detail on the object. We can clearly see that the color is blurred, and the geometry is fuzzy. Our system, by optimizing geometry and appearance jointly, has reconstructed much more detail, for example, the horse-

[1]The experiment is conducted on the kermit dataset, which is provided by the Bundler software package [13], and has been used in [10] as a reconstruction example. It consists of 11 VGA-resolution images taken under normal indoor lighting.

[2]In these datasets, the images are captured with fixed normal lighting, auto-focused and fixed exposure settings (including aperture, shutter speed and ISO-value). The native resolution of the captured images is 4272x2848, which is used for the camera calibration with Bundler software [13, 14]. Then the images are resized to 641x427, for 3D reconstruction.

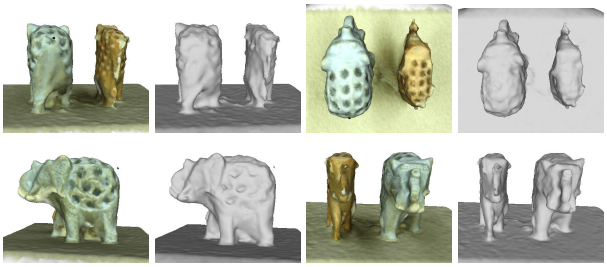


Figure 11. Four different views of the textured and non-textured models reconstructed by PMVS2 + PoissonRecon

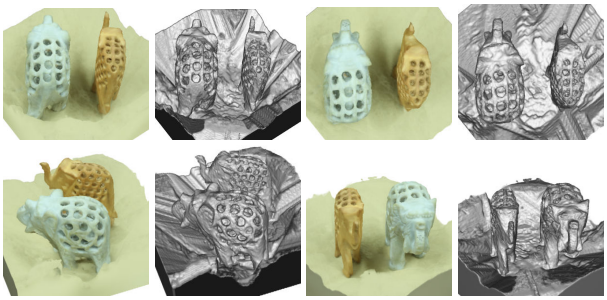


Figure 12. Four different views of the textured and non-textured models reconstructed by IRAY (our system); the holes on the body are well reconstructed.

Figure 13. Two sample images of building14 dataset

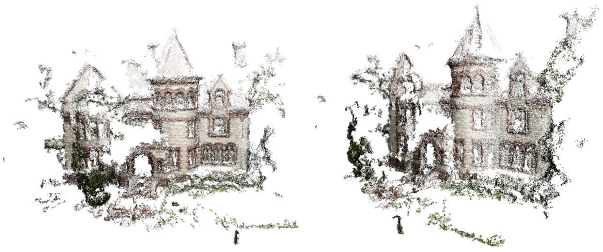


Figure 14. PMVS2's oriented point cloud output

rider's right hand and the horse's hair are more clearly visible. From this example, we can see that our improvement over [10] has significantly improved the inverse ray tracing approach's accuracy. In the next few experiments, we further demonstrate the advantages of the inverse ray tracing approach over the other state-of-the-art approaches represented by PMVS2 [5].

Fig. 1 (c, d) show the dense point cloud generated by PMVS2. It can be seen that the body part of the horse is reconstructed densely, while the tops of the horse, the horse-rider and the statue base are missing because these regions are not observed in any views from the bottom of

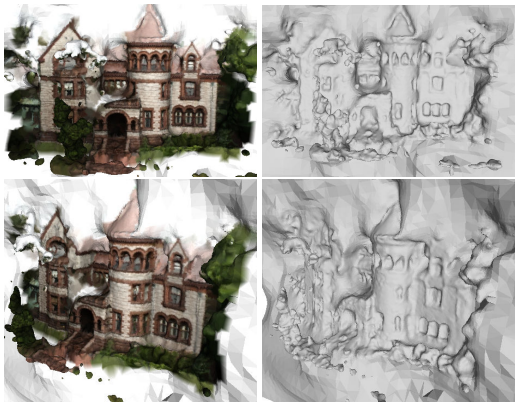


Figure 15. Two different views of the textured and non-textured models reconstructed by PMVS2 + PoisssonRecon

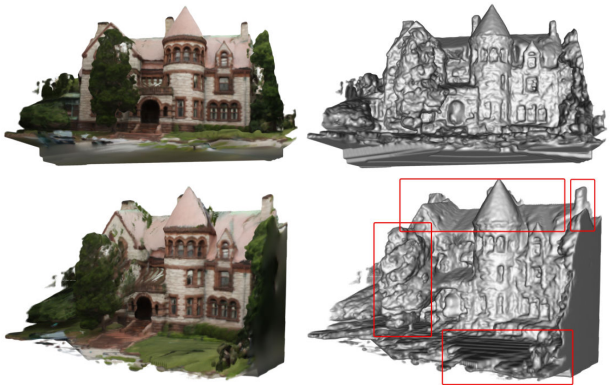


Figure 16. Two different views of the textured and non-textured models reconstructed by IRAY (our system); Interesting regions are highlighted in the last image.

the statue. Fig. 1 (e, f) show the reconstructed surface by PoissonRecon [7], taking PMVS2's point cloud as input. It can be seen that the generated surfaces totally depart from the true shapes. This is probably because the point cloud is not evenly sampled around the object, and missing regions are too large to be filled in reasonably by PoissonRecon.

Fig. 1 (g, h) and Fig. 8 (b) show the reconstruction results by our system[3]. Compared with PMVS2+PoissonRecon, we can see that our system does a much better job of recovering the geometry of the horse, horse-rider and statue base. It has successfully reconstructed the tops of these objects. Although these regions are not visible from the bottom, they are confined by the visibility constraints, and combined with the geometry smoothness prior our system outputs a reasonable reconstruction of these regions.

5.2. Elephants Dataset (elephants40)

The elephants dataset consists of 40 images taken around two carved elephants, one made of stone and the other made of wood. Interestingly, there are many holes in each elephant's body, and through these holes we can see that each elephant is "pregnant with her baby". These two elephants

[3]The volume resolution is 225x299x150 (about 10M voxels).

are put on a round plate, and a hand-held camera moves around the plate from the top. Fig. 9 shows two sample images.

Fig. 10 shows the reconstruction of a dense point cloud from four different view angles by PMVS2[4]. It can be seen that the side parts of the elephants are reconstructed densely, while the rear parts are almost completely missing, probably due to the small number of visible views.

Fig. 11 shows textured and non-textured 3D surface models from four different view angles reconstructed by PMVS2 + PoissonRecon[5]. PoissonRecon does a good job of filling the rear part of the elephants. However, it incorrectly fills the holes on the body of the elephants.

Fig. 12 shows the reconstruction results by our system[6]. Comparing Fig. 11 and 12, we can see that our approach can preserve the carved holes correctly, and reconstruct the baby elephants roughly. The rear parts of the elephants are reconstructed reasonably. There are because our algorithm better handles the occlusion relationship and is capable of reconstructing a region with small number of views. Also the volumetric representation, as used in our system, contains more geometric information than unconnected points, which make the surface reconstruction much easier with a simple iso-thresholding algorithm (e.g., marching cube algorithm [11]), compared with the point cloud based system.

5.3. Building Dataset (building14)

The building dataset consists of 14 images of a building cluttered by trees and moving vehicles. The images are captured with a hand-held camera with auto-focus and fixed exposure setting under normal outdoor lighting condition. Fig. 13 shows two example images.

Fig. 14 shows different views of the dense point cloud generated by PMVS2 software. It can be seen that most parts of the building are reconstructed densely, while the tree and grass regions of the scene are mostly missing, and the roof and chimney parts of the building are also missing. The absence of the tree and grass regions is very likely because the complex and repeated texture of the region makes the pixel-wise matching problematic. The missing of the roof and chimney is possibly due to the occlusion and small number of visible views. The point cloud generated by PMVS2 is further fed into the PoissonRecon software to generate a mesh surface representation. Fig. 15 shows two different views of the reconstructed surfaces.

Fig. 16 shows the reconstruction results by our system[7]. Compared with the PMVS2+PoissonRecon results, we can see that our system generates a more complete reconstruction of the scene, including the trees, grass, roof, chimney, etc. The textured rendering of the reconstructed 3D model is photo-realistic, while bearing in mind of the limited volume resolution (186x371x146) used.

[4]In PMVS2, the default parameters are set to work with high-resolution images. In the experiments, we modified two of the default parameters: 'level' is changed from 1 to 0, and 'csize' is changed from 2 to 1, to create denser point cloud than the default parameters.

[5]The PoissonRecon runs with parameters: octree depth = 10, solver divide = 8, samples per node = 1, surface offsetting = 1.

[6]The volume resolution in the experiment is 371x293x93 (about 10M voxels).

[7]The volume resolution is 186x371x146 (about 10M voxels).

6. Conclusion

This paper presents a complete solution to the optimum multi-view stereo problem based on the statistical inverse ray tracing method. In this approach, occlusion is modeled accurately and completely; voxel occupancies and colors are estimated jointly with an efficient message passing algorithm. Compared with point cloud based approaches, the proposed approach can reconstruct surfaces more completely and handle sparse cameras. Compared with surface evolution approaches and graph-cuts based volumetric approaches, it does not require an initial good surface estimate. Compared with space carving based volumetric approaches, it optimizes all the ray constraints together and has faster convergence rate and better geometric accuracy. The developed prototype system currently can only process tens of millions of voxels due to the cubic dependency of the computational and memory cost on the number of voxels in each of the three dimensions. In the future, sparse octree-based volumetric representation will be explored to reduce the computational cost in order to reconstruct large scale scenes.

Acknowledgement. This work was supported by the NSF IIS Program Grant 0808718.

References

[1] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *ICCV*, pages 388–393, 2001.

[2] N. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, pages 766–779, 2008.

[3] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *CVIU*, 96(3):367–392, 2004.

[4] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDEs, level set methods and the stereo problem. *IEEE Trans. Image Processing*, 7:336–344, 1998.

[5] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *TPAMI*, 32(8):1362–1376, 2010.

[6] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *IJCV*, 63(3):175–189, 2005.

[7] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symposium Geometry Processing (SGP)*, pages 61–70, 2006.

[8] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *IJCV*, 38:199–218, 2000.

[9] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang. Bundled depth-map merging for multi-view stereo. In *CVPR*, pages 2769–2776, 2010.

[10] S. Liu and D. B. Cooper. Ray Markov random fields for image-based 3D modeling: model and efficient inference. In *CVPR*, pages 1530–1537, San Francisco, CA, 2010.

[11] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987.

[12] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 35(2):151–173, 1999.

[13] N. Snavely. Bundler – structure from motion for unordered image collections, http://phototour.cs.washington.edu/bundler.

[14] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. In *SIGGRAPH*, 2008.

[15] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *CVPR*, pages 345–352, 2000.

[16] E. B. Sudderth, A. Ihler, W. T. Freeman, and A. Willsky. Nonparametric belief propagation. In *CVPR*, pages 95–103, 2003.

[17] G. Vogiatzis, C. H. Esteban, P. H. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *TPAMI*, 29(12):2241–2246, 2007.