Photo-Realistic Scene Modeling and Visualization using Online Photo Collections

Qi Shan

A dissertationsubmitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Brian Curless, Chair

Steven M. Seitz, Chair

Richard Szeliski

Paul D. Sampson (GSR)

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Photo-Realistic Scene Modeling and Visualization using Online Photo Collections

Qi Shan

Co-Chairs of the Supervisory Committee:
Professor Brian Curless
Computer Science and Engineering

Professor Steven M. Seitz
Computer Science and Engineering

Reconstructing 3D scenes from online photo collections has attracted a tremendous amount of interest from both academia and industry. The progress in the past decade has been exceptional, in terms of scale and reconstruction quality. Yet, we are still far from creating 3D models that support consumer level graphics applications. The challenge is twofold. First, modern geometry reconstruction and illumination/reflectance estimation techniques are generating low quality 3D models that are severely contaminated by visual artifacts, i.e., geometry holes, over-inflated boundaries, noisy surface details, low-resolution texture, etc. These artifacts are often extremely noticeable, thus largely limit the applicability of these 3D reconstruction approaches. Second, the real-world is dynamic, but very little research has been devoted to modeling and visualizing transient objects in photos. Therefore, typically we see ghost town 3D models in even the best-of-the-breed work.

In this thesis, I first introduce the Visual Turing Test with two of the first relight-able city-scale MVS models. The results show that poor geometry reconstruction and the lack of transient scene elements significantly reduce the photorealism of the rendered images. Our grand vision is that eventually the 3D reconstruction research will be able to pass the Visual Turing Test. While we are still far from that, this dissertation proposes new approaches to photo-realistic scene modeling and visualization. This line of research addresses both of the two aspects of the challenge, i.e., reducing

severe artifacts and incorporating some transient objects (people) in renderings, by improving various key components of modern 3D reconstruction pipelines. To be more specific, our work pushes the limit of (1) Structure-from-Motion research by solving the ground-to-aerial geo-registration with pixel level accuracy; (2) Multi-View Stereo by incorporating occluding contour information, and show dramatically improved geometry; (3) lighting/texture estimation by explicitly modeling outdoor illumination, and optimizing for lighting parameters and scene albedo, (4) image-based rendering to improve visualization of a scene with erroneous geometry, and (5) modeling transient objects. This dissertation describes work that can be considered as early effort towards the goal of making 3D reconstruction technologies widely applicable in real-world graphics applications.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

One thing I noticed in the past five and half years of pursuing a Ph.D. degree is that I don't always know what I am actually doing. For that, I wish to thank my advisors Brian Curless and Steve Seitz for their guidance, inspiration, support, and patience. I have learnt a lot from them about the taste to research problems, the attitude to work and to life. The graduate student life was not all about being productive or making progress. It is also about all the obstacles and struggles along the way. I am deeply grateful to have spent the memorable past few years working with Brian and Steve. I have benefited a great deal from knowing and closely working with Yasutaka Furukawa and Carlos Hernandez, and appreciate their great contribution to the work in this dissertation. I am also honored to have worked with Changchang Wu, Sameer Argawal, and Tadayoshi Kohno.

I would also like to thank Richard Szeliski and Paul Sampson for their service as my thesis committees. Their contribution is invaluable to the dissertation. I am also thankful that I have worked with Tony DeRose, Mark Meyer, and John Anderson when interning at Pixar. I learned a lot during the eight-month experience. Very special thanks to Lindsay Michimoto, as well to my friends and labmates at the University of Washington, Kathleen Tuite, Yun-En Liu, Supasorn Suwajanakorn, Aditya Sankar, Ricardo Martin Brualla, Mark Yastkar, and Paraschos Koutris, who have made my time at UW and in the lab such a pleasure.

Finally, I wish to thank my parents for their love and support, and my wife You Ren for always being there, bringing warmth and goodness to my life.

Chapter 1
INTRODUCTION

"For me, it's absolutely inevitable that entertainment will be 3D, it'll all be 3D, eventually, because that's how we see the world." – James Cameron [54]

It is foreseeable that in near future a significant portion of productivity and entertainment activities would be built upon 3D technologies. Video gaming, film production, virtual conferencing, and CAD are in the transition stages where exciting new technologies (for example, holographic displays, virtual reality headsets, 3D printers, etc.) are booming. However, high quality 3D content creation for consumer grade applications is still an expensive, mostly manual, and labor intensive process.

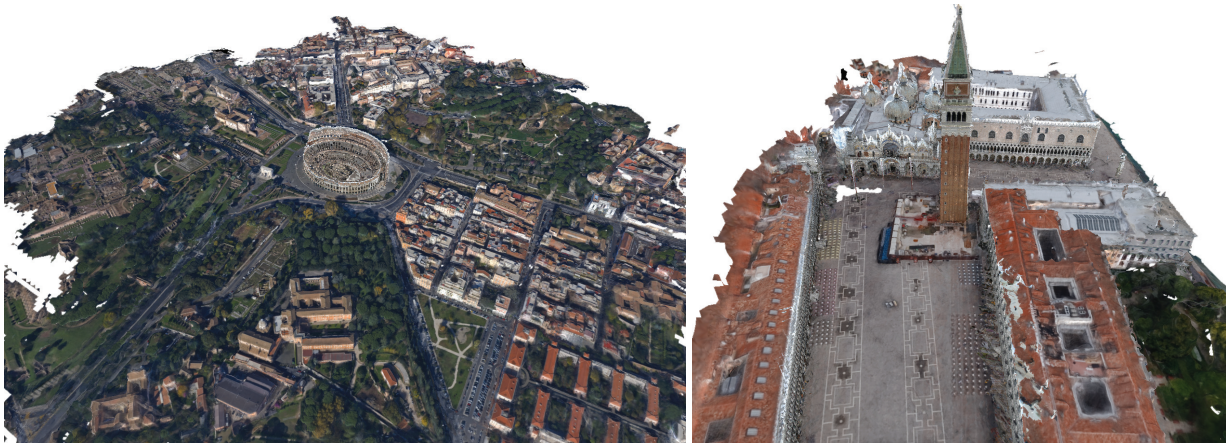
Academic research on automatically reconstructing 3D scenes is an area that has received a great deal of attention. Arguably the most exciting progress is from the line of work which uses Internet photo collections as input. Researchers have succeeded in developing very large-scale 3D reconstruction pipelines for Internet photo collections [4, 22, 67]. These systems are able to scale to millions of photos and produce increasingly better quality 3D models. Commercial applications, e.g., Apple Maps [7], Google Earth [29], and Bing 3D Maps [51], that are inspired by these reconstruction pipelines, have reached a point where most of the world is reconstructed at least to a moderate level of quality.

Although the research progress is encouraging, the gap between the academic community and the stricter demands of industry is still wide. Indeed, the current stage of 3D reconstruction work is not ready for most commercial graphics applications. Although we are able to render real world scenes using sparse point clouds or partial mesh reconstructions on a computer monitor, the visualization is often severely contaminated with artifacts including surface noise, holes, undesired

Figure 1.1: The Visual Turing Test: One of the two images is rendered from our reconstructed 3D model, and the other is a real photograph. We ask the question to a test subject: "which of the two images do you think looks more photo-realistic?"

geometry, and low-resolution texture. These artifacts largely limit modern 3D reconstruction techniques to support graphics applications that have been promised to the community, for example, virtual tours that grant the visual experience of being physically present at a real-world landmark, a.k.a, "being there" experience. In fact, most graphics applications (e.g., video games and virtual reality) require high quality 3D reconstructions that are basically artifact-free, and thus are beyond the reach of current state-of-the-art approaches.

In order to address the problem of bridging the gap between academic research and industry, we need to better understand the challenges. The first question is: "how do we know if we have achieved the goal?" In fact, this question is about how to evaluate 3D reconstruction work, and is far from trivial. State-of-the-art work typically conducts evaluations with Middlebury datasets, and report distance to a ground truth. This evaluation doesn't capture the photo realism [61]. Some work simply visually inspects the reconstructed 3D models, the conclusion could vary depends on the perspectives. In this thesis, we propose to evaluate by randomly sampling the images rendered from 3D models and measure how photo-realistic they are comparing to actual photos taken from same viewpoints, a process we call the "Visual Turing Test" (Figure 1.1). We argue this is more meaningful than the Middlebury evaluation, but passing it is a grand challenge, and requires the



Colosseum San Marco Square

Figure 1.2: Two datasets used in our experiments, where the reconstructions are rendered from aerial viewpoints with albedo colors without additional lighting effects. Top: Colosseum. Bottom: San Marco Square.

3D models to be *complete* and *re-lightable*. We create two city scale 3D models from an unusually large number of photos (100K+), and successfully estimate scene albedo and lighting variations (Figure 1.2). The quantitative results of the visual Turing test show that we can fool human perception when making comparisons at low image resolution, although further progress is necessary to achieve photo realism at higher resolution levels.

Figure 1.3: Typical bad results in our visual Turing test. (left real, right rendered). More than 90% of test subjects pick the reference photos as more realistic in every resolution level.

The visual Turing test is important for identifying key challenges. Some of the most interesting results are the images with low passing rates (Figure 1.3). These cases help to draw a road map for the work described in the thesis by answering the next question "what are the main challenges for

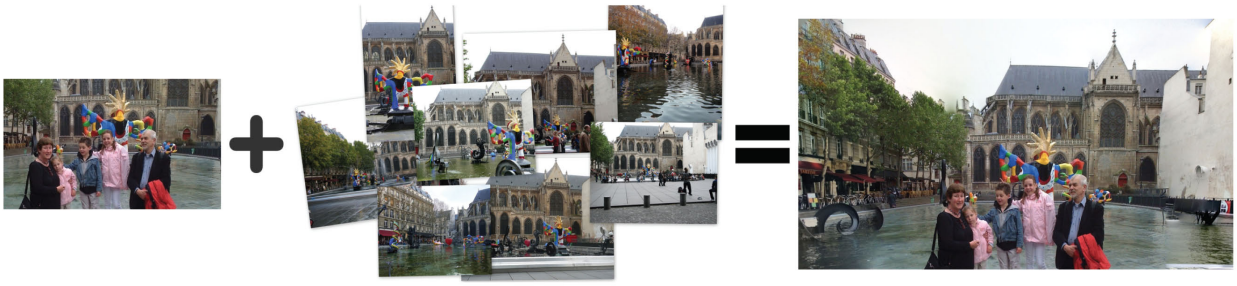


Figure 1.4: Photo Uncrop. Travel photos have limited field of view (FOV). We address the problem of extending the field of view of a photo – an operation we call uncrop, by utilizing Internet photos from near-by viewpoints.

3D reconstructions to pass the Visual Turing Test?" The answer is twofold:

- reducing geometry artifacts to a level where they are almost unnoticeable by human perception;
- visualizing transient but salient scene elements that are hard to model.

To address the above two challenges, in this dissertation, I propose making progress on photo-realistic scene modeling and visualization. My work focuses on a fully automatic, end-to-end pipeline of 3D reconstruction and visualization. The input of my system is a collection of photos collected from popular image hosting websites (Google [28] and Flickr [19]) by typing in the names of real-world landmarks (e.g., "Colosseum in Rome" or "Place de la Concorde in Paris"). The output is a 3D model of the particular landmark [63, 64, 66], or a visualization created through the 3D model using image-based rendering [65]. Our work significantly advances the state-of-the-art in 3D reconstruction and visualization, offers innovations in multiple critical aspects, and is able to achieve *photo-realistic* rendering quality for applications like "Photo Uncrop" [65] (Figure 1.4).

Geometry artifacts often appear as holes, overly expanded boundary fill, and noisy surface details. These artifacts are very noticeable, but extremely difficult to fix for automatic scene reconstructions. Geometric holes are common for texture-less or un-captured regions. We address this problem by using multiple sources of images (tourist photos, street-view imagery, and aerial photos) and interpolating depth information based on occluding contours. Boundary artifacts and noisy

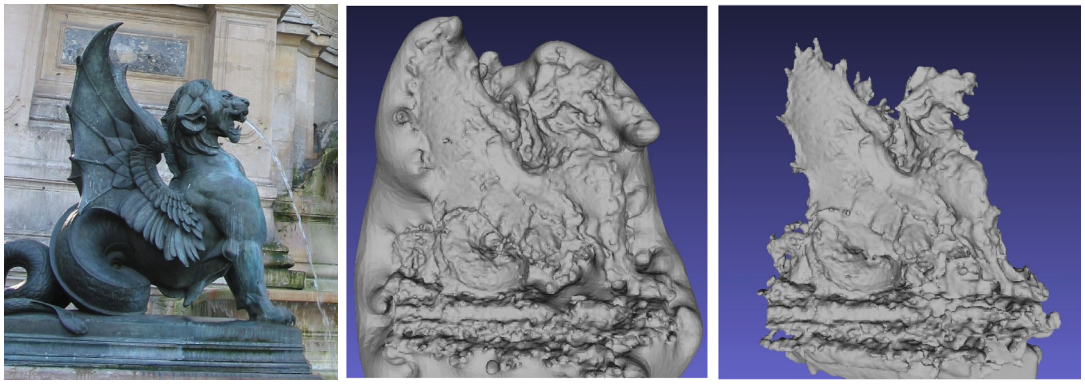


Figure 1.5: Improved geometry from occluding contours. Left: one of the input images. Middle: the mesh reconstructed using a state-of-the-art reconstruction pipeline. Right: the mesh reconstructed from our improved pipeline that estimates dense visibility from occluding contours.

surface details are two other major causes of geometric errors. The cause lies within the standard 3D reconstruction pipelines which start with images, then estimate camera poses, compute dense 3D oriented points, and in the end reconstruct a mesh from these dense points. In most methods the last step fits a mesh that best explains the 3D oriented points, without using the color information from input images. We improved the pipeline by putting the image information back in the mesh reconstruction stage. Our approach estimates dense visibility from occluding contours and infers free space information to better regularize surface reconstruction (Poisson-based approach in [41]), so that the reconstructed surface would be constrained in a tight envelop. This constraint helps to suppress overly expanded boundary artifacts (Figure 1.5) and improve surface details .

Transient but salient scene elements (e.g., people in the scene) are also crucial. Modeling them exposes new challenges. Existing Multi-View Stereo (MVS) methods rely on the *photometric consistency metric* for inferring scene depths. This metric assumes a static scene across multiple images. A person does not usually appear at the same location with exactly the same posture in multiple images, especially in Internet photo collections. Our approach takes advantage of human pose detection/estimation [82] and image compositing [65] techniques, and explicitly models people in images (Figure 1.6).

Most of the techniques described in this dissertation use publicly available online photo col-

Input image | Photo Popup [Hoiem et al. 2005] | Our depth map rendering

Figure 1.6: Comparing with single image modeling.

lections. These photos are taken by a large number of people using different cameras at different times of day with different illumination conditions. However, these techniques are not limited to online photo collections. And unsurprisingly, they are able to achieve even better results while working with carefully taken photos. We choose Internet photos as our datasets for the following three reasons. First, it is easy to obtain a huge amount of images to enable a large scale evaluation. Second, these photos are more "general purpose" as they are typically taken by ordinary non-tech-savvy photographers. We believe they better represent the population of the user groups of our system than us (academic researchers). Datasets carefully captured by researchers are typically biased towards a few particular applications and less suitable for large scale scene reconstruction work. Finally, Internet photos have a much longer time span (taken over years). These photos capture a wide range of scene appearance variation, and are interesting to applications including scene change detection [50] and lighting/texture estimation [63].

This dissertation is organized as follows. It first presents the Visual Turing Test and our solution to reconstruct complete and re-lightable city-scale models in Chapter 2. We examine the quantitative results and delve into the images with low passing rates. These images reveal a clear road map for next steps of our work, including reliable ground-to-aerial geo-registration (Chapter 3) and higher quality geometry reconstruction (Chapter 4). We further note that visualization is another crucial component of the pipeline, especially when our 3D models are reconstructed from

online photo collections and the underlying geometry is not perfect. We demonstrate the effectiveness of a novel Image-Based Rendering (IBR) visualization approach and show success in creating "photo-realistic" renderings for a novel application called "Photo Uncrop" (Chapter 5). Chapter 6 introduces an extension of "Photo Uncrop" which explicitly models people in images. It creates immersive 3D visualization from a single input image by adding parallax and defocus effects. Each of the following chapters is organized in a self-complete way where I first introduce the problem, then describe related work, our solution, and present experimental results and discussions. Note that the mathematical notations of the equations are only consistent within each chapter.

Chapter 2

THE VISUAL TURING TEST FOR SCENE RECONSTRUCTION

The last few years have seen dramatic progress in the area of 3D reconstruction from photographs, to the point that much of the world has been reconstructed and can be browsed in tools like Microsoft's Photosynth, Google's Photo Tours, and Apple's Flyover 3D Maps.

Yet, we are still far from being able to generate 3D geometric models that look just like the real thing. Far from it; even the best-of-breed vision-based 3D reconstruction techniques are not good enough to support most computer graphics applications (games, films, virtual tourism, etc), and instead require extensive manual editing (in the case of 3D maps) or image-based rendering (e.g., Photosynth) to compensate for deficiencies in the reconstructed geometry.

But what exactly does it mean to look just like the real thing? One definition is that people should be unable to tell apart photos from renderings. For any photo, we can produce a scene rendering (for the same viewpoint and illumination conditions) that appears so realistic that you can't tell which one is real. This is a grand challenge problem; we call it the *Visual Turing Test* for Scene Reconstruction.

While we are still far from being able to pass the Visual Turing Test, it defines a useful benchmark, and a goal to strive for in 3D reconstruction research. Achieving this goal also necessitates two new capabilities that have not previously been demonstrated. First, we have to match *any* photo. This requires building a model that leverages *all available imagery*, from the ground, the air, the walking paths, and the streets. To this end, we are the first to demonstrate models derived from Flickr photos, Streetview, and aerial imagery, merged into one reconstruction. We employ an unusually large number of photos (100K+) to create our models, to ensure that they are as complete as possible. Second, we must be able to render the scene to match the viewpoint and illumination in any photo. The latter requires estimating not only geometry, but surface reflectance, as well

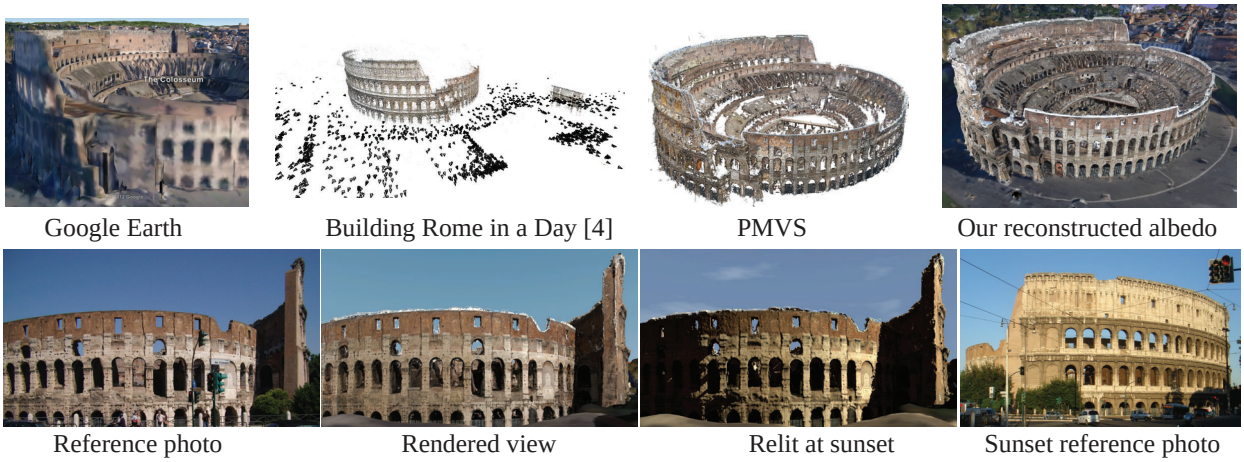


Figure 2.1: Comparison with state of the art (top row) and results for rendering and relighting of one viewpoint (bottom row).

as the lighting in that particular photo. In this chapter, we present the first large scale results on reflectance estimation and lighting matching for Internet Photo Collections. To the best of our knowledge, we are also the first to conduct a large scale Visual Turing Test which consists of 100 randomly selected renderings, each of which is shown in 4 resolution scales, and 142 human test subjects.

The remainder of the chapter is organized as follows. I first give a brief overview on related work in scene reconstruction in Section 2.1. Our approach is described in Section 2.2 (Preprocessing) and Section 2.3 (Lighting and Reflectance Estimation). I show our datasets, both qualitative and quantitative results of our visual Turing test in Section 2.4. In Section 2.5, I present how the visual Turing test draws the road map of my research and inspires the work in following chapters.

The work in this chapter was published in the 2013 International Conference on 3D Vision [63].

2.1 Related Work

In the past few years, researchers succeeded in developing very large-scale 3D reconstruction pipeline for Internet photo collections [4, 22] that can scale to millions of photos. These models are detailed but incomplete, containing holes in areas of sparse coverage. In contrast, aerial-based

reconstructions in products like Google Earth and Apple's 3D Maps provide more uniform scene coverage, but lack the detail and resolution of ground-based models. We seek to achieve the best of both worlds, also leveraging Google Streetview imagery to fill gaps in coverage.

There is a large literature on the topic of reflectance and illumination modeling in the graphics and vision communities, going back multiple decades. The vast majority of prior work, however, assumed a controlled laboratory environment or imposed restrictions such as fixed lighting, or materials that do not vary over the surface. Most closely related to our work are methods that operate outside, "in the wild," with widely varying, unknown viewpoints, illumination, and surface material variation.

A few researchers have reconstructed depth maps from time-lapse webcam videos of outdoor scenes [1, 2], where the fixed viewpoint and strong directional sun-light allow application of photometric stereo techniques.

Internet photo collections pose additional challenges, as both the viewpoint and illumination are unknown and vary arbitrarily in each photo. Indeed, the only previous work to attempt relighting for Internet photo collections is [31] and they provided results for only one dataset (Statue of Liberty) consisting of six photos. While their lighting and reflectance model is more sophisticated than ours, it is not scalable–it took three hours to process the six image dataset. By using a more streamlined illumination and reflectance model, we are able to process tens of thousands of images, while achieving high quality visual results.

Also related is work on multi-view intrinsic image decomposition [45]. They recover a PMVS point cloud, which is then used to estimate per-point, per-view illumination (a single color value to represent the combined illumination and shading) which can be spread smoothly across each view; they leverage multi-view constraints on the (Lambertian) reflectance during estimation. This approach enables transferring illumination from one image to another that has many PMVS points in common, again after smoothly spreading the illumination across the second image. However, their datasets are fairly small, typically tens of images. Further, the method does not support general relighting, instead copying sparse illumination (a per-point color) from one image to another image, and both images must cover roughly the same portion of the scene.

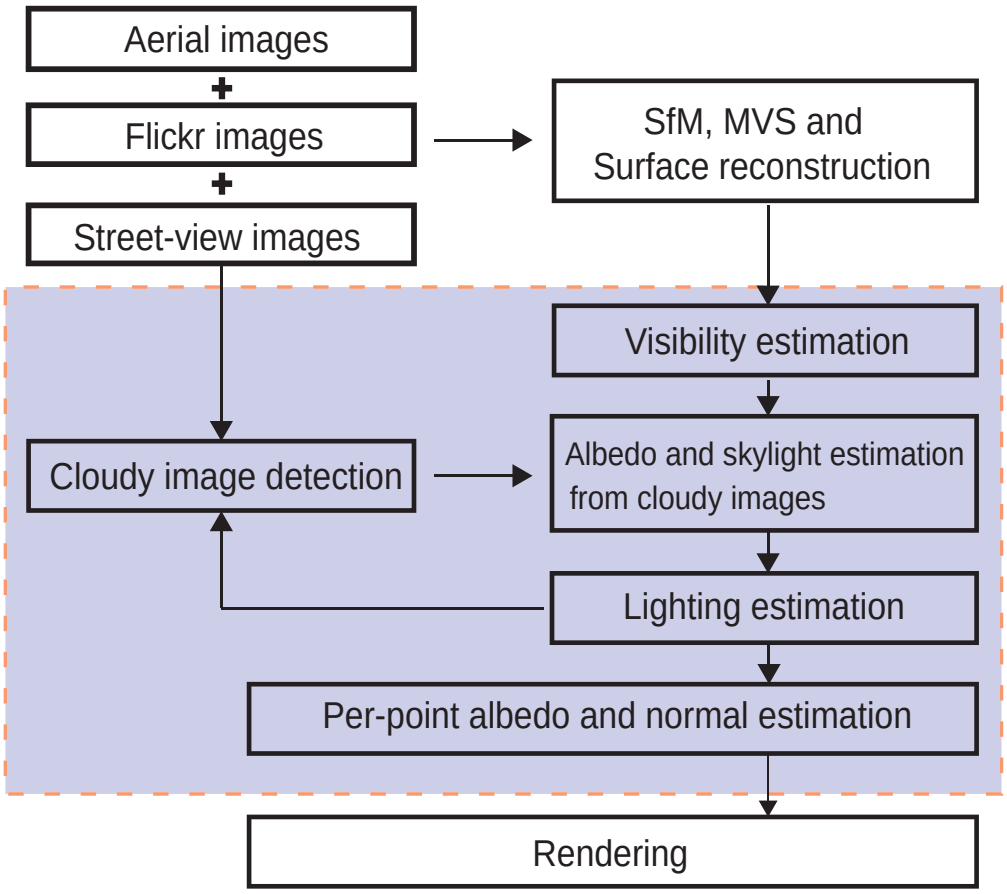
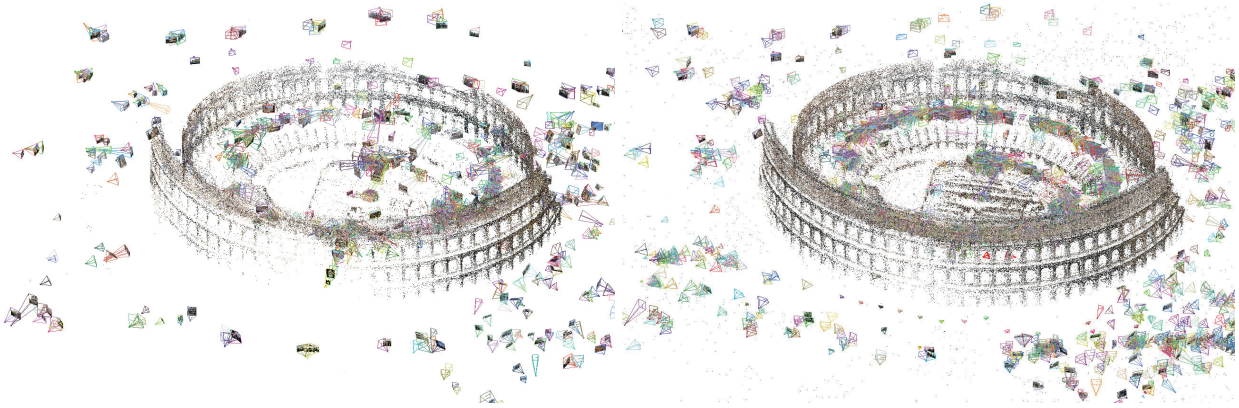


Figure 2.2: Workflow overview.



(a) (b)

Figure 2.3: Ensuring a uniform SfM reconstruction. (a) SfM model from a randomly subsampled image set. (b) The final SfM model after augmenting the image set.

2.2 Preprocessing

Our pre-process consists of collecting images, recovering camera poses with structure-from-motion (SfM), recovering a point cloud with multi-view stereo (MVS), and recovering a mesh with per-vertex visibility to sets of images.

Given a landmark (e.g., the Colosseum), we download ground-level images from Flickr [19] and obtain aerial images from Google. We augment this set with Google Streetview images in regions that are poorly covered by Flickr photos; these images are not directly available to us in raw form, so we simply capture them from the in-browser Streetview rendering. We also approximately invert the sRGB function typically applied to photographic imagery to put the pixel values in a linear space.

We then recover a triangle mesh for the landmark using freely available software. In particular, we employ VisualSFM [79] to estimate camera poses, PMVS/CMVS [26, 27] to recover a dense, oriented point cloud, and Poisson Surface Reconstruction [40] to reconstruct a triangle mesh. We remove large (typically inaccurate) "hole-fill" triangles from the Poisson reconstruction; specifically, we filter out all triangles with average edge length greater than 20 times the average edge length of the entire mesh. Finally, we estimate a set of visible images per vertex, i.e., a set of

images in which the vertex is visible. Below, we describe the SfM and visible image estimation steps in more detail.

2.2.1 SfM Reconstruction

For some datasets, the number of available images is quite large (e.g., $140K$ Flickr images of the Colosseum) with a considerable amount of overlap. Matching all the images to each other would be quite slow. Further, photos are typically concentrated around a small number of popular viewpoints [4], heavily oversampling those particular regions and needlessly slowing the entire pipeline. We take a two-step SfM approach that limits the number of images used and encourages good coverage around the landmark.

In the first step, we randomly subsample K images (we use $K = 1000$) to give an initial subset $\mathbf{I^K}$. The remaining images form a set $\mathbf{I^R}$. We apply VisualSFM to $\mathbf{I^K}$ to recover camera poses and image matches; images match if they have common SfM features. The resulting set $\mathbf{I^1}$ contains images that were matched successfully by VisualSFM.

In the second step, we augment $\mathbf{I^1}$ with another subset $\mathbf{I^A}$ taken from $\mathbf{I^R}$ and then re-run VisualSFM. To add images to $\mathbf{I^A}$ (which is initially empty), we match each of $\mathbf{I^1}$'s images against the images in $\mathbf{I^R}$, where a match must have a large number of features in common (≥ 300) and be geometrically consistent with the initial SfM reconstruction. To encourage good coverage, we process the images in $\mathbf{I^1}$ in order, starting with the images that have the fewest number of matches, thus giving priority to sparsely covered areas. Further, we do not consider images in $\mathbf{I^1}$ that are already well-matched (having ≥ 30 matches). To promote high quality reconstruction and to control the total number of images, we sort the images in $\mathbf{I^R}$ according to image resolution and iterate through them (highest resolution first) when matching to an image in $\mathbf{I^1}$, stopping after finding a fixed number of matches (we set the number to 10). After an image from $\mathbf{I^R}$ is matched, it is removed from $\mathbf{I^R}$ and added to $\mathbf{I^A}$. When finished building $\mathbf{I^A}$, we perform a second pass of VisualSFM on $\mathbf{I^1} \cup \mathbf{I^A}$ yielding a set $\mathbf{I^2}$ containing images that matched successfully during reconstruction. Figure 2.3 illustrates the results on the Colosseum after the two steps.

2.2.2 Visibility Estimation

After reconstructing a Poisson mesh and automatically trimming out large hole-fill triangles, we estimate a set of images in $\mathbf{I^2}$ that can see each vertex, i.e., one visibility set per vertex. The original PMVS points already have a conservative visibility set per oriented point, a set comprised of images that matched well at that point; we use the PMVS points and visibility sets to bootstrap the process of estimating per-vertex visibilty.

Specifically, for each vertex v in the trimmed Poisson mesh, we collect the 30 nearest PMVS points and their visibility sets. We then select the 9 images that appear most frequently in those visiblity sets. Next, we project all vertices in the 7-ring neighborhood of v (i.e., vertices within 7 edge hops from v) into the selected images and compute an average color at each of those vertices. We then consider each image I in $\mathbf{I^2}$. If v is facing away from I or if a ray cast from I to v hits another part of the Poisson model first, then I is eliminated from consideration. Otherwise, the 7-ring neighborhood is projected into I, and the resampled colors from I are compared against the average colors of the vertices, using Normalized Cross Correlation (NCC) as the metric. If the NCC score is higher than a threshold (0.8), then I is added to v's visibility set. To accelerate the process, we find nearest PMVS neighbors using FLANN [53] (set for exact neighbor-finding), and we use pre-computed z-buffers instead of ray casting for occlusion testing.

2.3 Lighting and Reflectance Estimation

Given the images with recovered poses and the reconstructed mesh with per-vertex visibility sets, we estimate lighting parameters for each image and reflectance parameters for each vertex. In the remainder of the section, we explain our shading model and objective function, how to detect cloudy images (useful for bootstrapping the optimization), how we optimize for the shading model parameters, and finally some implementation notes.

2.3.1 Shading model and objective function

For the outdoor scenes we are reconstructing, we adopt a simple, but fairly effective representation for illumination and materials. In particular, we assume the lighting is comprised of uniform, hemispherical sky illumination plus directional sunlight, and we assume all materials are diffuse.

Given a point P_i (a vertex in the mesh), an image I_j, and their associated shading parameters, the rendered pixel intensity $R_{i,j}$ of P_i in I_j is calculated with an ambient+diffuse shading model as follows:

$$R_{i,j}(\Theta) = a_i \left\{ f(N_i) k_j^{sky} + \max[0, L_j \cdot N_i] k_j^{sun} \delta_{i,j} \right\}. \tag{2.1}$$

$$\Theta = \{N_i, a_i, L_j, k_j^{sky}, k_j^{sun}, \delta_{i,j}\}. \tag{2.2}$$

a_i and N_i are the surface albedo and normal at P_i, respectively. k_j^{sky} and k_j^{sun} are the skylight (ambient) and sunlight (diffuse) intensities, respectively. L_j is the lighting direction, parameterized in spherical coordinates. Note that we have not explicitly modeled camera exposure; instead, this is a scale factor that is implicitly pre-multiplied into the light intensities. $\delta_{i,j}$ models sunlight visibility and is 1 if P_i is in sunlight in image I_j, else it is 0. $f(N_i)$ models the ambient (skylight) occlusion, that is, how much of the hemisphere is visible from, and hence, illuminates the point. In principle, we can use the input mesh model to take into account occlusions caused by the surrounding structure as in [31]. For efficiency, we just use the normal N_i to determine hemispherical sky visibility, ignoring occluders. $f(N_i) = (1 - N_i \cdot U)/2$ (derived in [68]) where U is the unit-length "up" direction in the scene. Though we expect that adding sky occlusion due to surrounding geometry could improve results, we found that our simplified model works surprisingly well in practice.

We assume, for the moment, that our images, albedos, and lighting are grayscale; we discuss color in Section 2.3.5.

Based on our shading equation (2.1), the lighting and reflectace estimation problem can be formulated as follows:

$$\underset{\Theta}{\operatorname{argmin}} \sum_i \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \| R_{i,j}(\Theta) - \tilde{R}_{i,j} \|_2^2. \tag{2.3}$$

$\tilde{R}_{i,j}$ is the observed pixel intesity of point P_i in image I_j, and V_i is the list of image indexes in which P_i is visible, where i and j are indexes to points and images. Note that the objective is simply the sum of squared differences of image intensities between the observation and what is predicted by our shading model, weighted by the squared root of the observed intensity. The weight is intended to give less weight to points that may be in shadow. This weighing scheme has proven effective, particularly in the early stage of the optimization, where $\{\delta_{i,j}\}$ are all initialized to 1, i.e., all the points are assumed to be sunlit.

2.3.2 Identifying Cloudy Images

Solving (2.3) on a large mesh with thousands of images is a very challenging problem because optimizing it has a high computational cost, exacerbated by the non-linearity of the functional which gives rise to numerous local minima. To improve both the computational efficiency and to avoid local minima, we will make use of cloudy images, which have negligible sunlight intensity and can directly lead to estimates of skylight intensities and surface albedos for points visible in those images. (In our experiments, around 15% of photos are taken under cloudy weather and 40% of all the 3D points are visible in at least one of the cloudy images. This section describes how we identify cloudy images.

An image is identified as cloudy, if it passes at least one of the following three tests.

• The first test is on the camera shot setting stored inside the EXIF tag. We compute the exposure value as $\frac{\{\text{exposure-time}\}\{\text{ISO-value}\}}{\{\text{F-number}\}^2}$, and identify the image as cloudy, if the value is modest, that is, within the range $[0.05, 5.0]$. A small value typically indicates a sunny day with strong illumination, while a large value indicates a night-time shot.

• The second test is on the *skyness* at the top portion of an image, as inspired by Ackermann et al. [2]. Given an image, we compute the average intensities in the RGB channels over the top 3% of the image region, and identify the image as cloudy, if $(2B_{avg} - R_{avg} - G_{avg} < 100)$ holds.

• The last test is on the ratio between the skylight and sunlight intensities (k_j^{sky}/k_j^{sun}) after lighting estimation (i.e., during optimzation). An image is identified as cloudy if the ratio is more than 10.

As described in Sec. 2.3.3, the first two tests are initially used to identify cloudy images. After

the first lighting estimation, we include the third test to update cloudy images.

2.3.3 Algorithm

The core estimation algorithm consists of three steps: 1) Partial albedo estimation from cloudy images; 2) Lighting estimation; and 3) Per-point albedo estimation (See Fig. 2.2 for an entire algorithm flow).

Skylight and partial albedo from cloudy images

For cloudy images, the shading equation (2.1) does not have a sunlight component and is simplified to

$$R^C_{i,j}(N_i, a_i, k^{sky}_j) = a_i f(N_i) k^{sky}_j. \tag{2.4}$$

Let $\mathbf{I^C}$ denote a set of cloudy images. We collect a set of points $\mathbf{P^C}$ that are visible in at least three cloudy images, where estimation becomes reliable. The optimization problem (2.3) can similarly be reduced as follows:

$$\underset{\{a_i, k^{sky}_j\}}{\operatorname{argmin}} = \sum_{i \in \mathbf{P^C}} \sum_{j \in V_i \cap \mathbf{I^C}} \sqrt{\tilde{R}_{i,j}} \| R^C_{i,j}(a_i, k^{sky}_j) - \tilde{R}_{i,j} \|^2_2. \tag{2.5}$$

Note that surface normal N_i is technically a variable in (2.5). However, we instead use the Poisson normal, because normal estimation is unreliable without the directional lighting component, and the input mesh model has already fairly accurate surface normal estimates.

Lighting Estimation

Even with partial surface albedo estimates, it is very expensive to solve (2.3) by using all the points, which could number into tens of millions. On the other hand, we observe that not all the points are necessary to estimate lighting parameters; thus, we first focus on solving lighting parameters for each image, while operating on a small but effective set of points.

For lighting estimation, we would like to select a subset of points (vertices) that are visible in many images, but also achieve coverage by ensuring each image contains at least $m(= 1000)$

such points. After initializing the set $\mathbf{P^L}$ with 2000 points that have the most number of visible images, we pick an image that has less than m visible points, and add 100 points from that image to $\mathbf{P^L}$. The 100 points are randomly sampled, where the sampling probability is proportional to the number of visible images for each point, so that points with more visible images are more likely to be added. The process repeats until all the images have more than m points or no more points can be added.

Now, we finally solve (2.3), but with two modifications. First, we use the subsampled point set. Second, we add a damping term to bias our solution to the surface albedo estimate $\tilde{a}_i$ from (2.5) and the normal $\tilde{N}_i$ in the input mesh, giving a new objective:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{i \in \mathbf{P^L}} \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \| R_{i,j}(\Theta) - \tilde{R}_{i,j} \|_2^2 + \lambda_1 \sum_{i \in \mathbf{P^C}} \| a_i f(N_i) - \tilde{a}_i f(\tilde{N}_i) \|_2^2. \tag{2.6}$$

$\lambda_1 = 1$ is used in our experiments. Note that the damping term is added for points $\mathbf{P^C}$ that are visible in some cloudy images and has estimates from (2.5).

After solving (2.6), we update the cloudy image set $\mathbf{P^C}$ by using the estimated lighting parameters as in Section 2.3.2. Then, we solve (2.5) and (2.6) in exactly the same way.

Per-point Albedo Estimation

The final step is to fix the lighting parameters $\{L_j, k_j^{sky}, k_j^{sun}\}$, then solve for the remaining parameters $\{N_i, a_i, \delta_{i,j}\}$, which can be optimized for each point independently,

$$\underset{\{N_i, a_i, \delta_{i,j}\}}{\operatorname{argmin}} \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \| R_{i,j}(\Theta) - \tilde{R}_{i,j} \|_2^2 + \lambda_1 \sum_{i \in \mathbf{P^C}} \| a_i f(N_i) - \tilde{a}_i f(\tilde{N}_i) \|_2^2 + w_i \| N_i - \tilde{N}_i \|_2^2. \tag{2.7}$$

The third term arises from the observation that when a point is visible only in a few images, normal and albedo estimation become very noisy. In this case, since the surface normal estimation from the input mesh model is fairly accurate, we add a damping term on the surface normal itself, while adaptively weighing the term based on the amount of available image information for that point. Note that $\tilde{N}_i$ is the surface normal in the input mesh. More detailed definition of the per-point weight w_i is given in Sec. 2.3.4.

2.3.4 Regularization Weight on Per-point Albedo Estimation

For per-point albedo estimation, here we re-iterate Equation 2.7

$$\underset{\{N_i,a_i,\delta_{i,j}\}}{\operatorname{argmin}} \sum_{j\in V_i} \sqrt{\tilde{R}_{i,j}} \|R_{i,j}(\Theta) - \tilde{R}_{i,j}\|_2^2 + \lambda_1 \sum_{i\in \mathbf{P^C}} \|a_i f(N_i) - \tilde{a}_i f(\tilde{N}_i)\|_2^2 + w_i \|N_i - \tilde{N}_i\|_2^2.$$

The first term is the data term measuring the image discrepancy between observed pixel intensities and what is predicted by our model. The second term is a regularizer based on cloudy images; it keeps the estimated albedos for points seen in cloudy images close to the estimates recovered by optimizing Equation 5. Reference normals come from the Poisson reconstruction.

In this section, we focus on the third term, a regularizer that encourages adherence to the Poisson normals when the weight w_i is high. Ideally, we would make w_i depend on the first, image discrepancy term in the objective; i.e., when the discrepancy is high, the normal estimation is unreliable, and the weight should be high. Of course, we do not know the magnitude of the image discrepancy term before actually solving Equation (7). Nonetheless, for a subset of points $\mathbf{P^L}$, we have already solved Equation 6 to estimate lighting and thus have computed image discrepancies for those points. Our approach is to use that information to construct per-point weights based on just these image discrepancies.

More precisely, we first compute an average image discrepancy measure d_j^{image} for each image I_j by taking the average of the discrepancy term over points $\mathbf{P_j^L}$ that are in $\mathbf{P^L}$ and visible in I_j:

$$d_j^{image} = \frac{1}{|\mathbf{P_j^L}|} \sum_{P_i \in \mathbf{P_j^L}} \|R_{i,j}(\Theta) - \tilde{R}_{i,j}\|_2^2.$$

Then, we define the reliability r_i^{point} of imagery information at each point P_i by aggregating d_j^{image} over P_i's visible images V_i:

$$r_i^{point} = \sum_{I_j \in V_i} \sqrt{|\mathbf{P_j^L}|} \exp(-\kappa\, d_j^{image}).$$

The exponentiated discrepancy measure from each image is weighted by the square root of the number of contributing points $\mathbf{P_j^L}$. Note that the reliability of a point should increase when it is

visible in more images, which is also modeled by the formula above. We used $\kappa = 20$ in our experiments.

Finally, we normalize r_i^{point} so that its mean is 1.0 over the entire point set to give a normalized reliability measure $\hat{r}_i^{point}$, and we then define the regularization weight w_i to be inversely proportional to $\hat{r}_i^{point}$:

$$w_i = \frac{\lambda_2}{\hat{r}_i^{point}},$$

where $\lambda_2 = 0.01$ in our experiments.

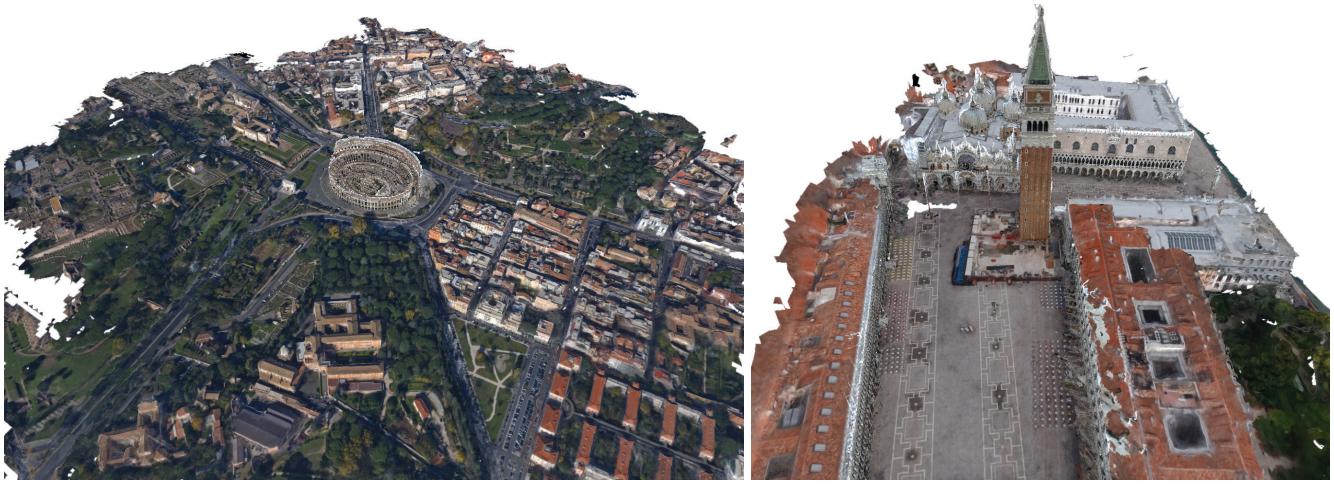
2.3.5 Implementation details

Here we describe several implementation details. First, we employ the Matlab function *lsqnonlin* to solve all of the the optimization problems.

Second, $\delta_{i,j}$ is a binary variable and cannot be optimized easily. $\delta_{i,j}$ is initialized to be 1 at the beginning. When $\delta_{i,j}$ is a free variable in an optimization problem, we solve it in three steps: 1) Fix $\delta_{i,j}$ and solve the other parameters with *lsqnonlin*; 2) Solve $\delta_{i,j}$ while fixing the others for each point independently (a simple binary decision); and 3) Fix $\delta_{i,j}$ again and solve the other parameters by *lsqnonlin*.

Finally, the albedos a_i and lighting intensities, k_j^{sky} and k_j^{sun}, are all color values. In practice, when sunlight direction L_j is not a free variable, we simply solve the optimization problem in each color channel independently. When L_j is a free variable in an optimization, we first map the colors to grayscale (luminance) to solve the problem, then solve the same problem again in each color channel independently while fixing the lighting direction.

Poisson Surface Reconstruction [40] converts a PMVS point cloud into a triangle mesh, where the output mesh resolution can be controlled by a parameter (*depth*). We noticed that increasing this parameter (and hence resolution) too much introduces surface artifacts, however, we still want to match the texture resolution of the mesh to that of the input images for optimal rendering. Therefore, we use a modest parameter for *depth*, in particular, 12 for the San Marcos Square and 13 for the Colosseum. Then, we simply apply the triangle subdivision – split a triangle into four



Colosseum San Marco Square

Figure 2.4: Two datasets used in our experiments, where the reconstructions are rendered from aerial viewpoints with albedo colors without additional lighting effects. Top: Colosseum. Bottom: San Marco Square.

smaller ones – to increase the mesh resolution, where the surface subdivision is adopted in two ways in our system. First, we subdivide the entire mesh once uniformly to increase its resolution. Second, regions of interest, which are specified by drawing rectangles on images, are subdivided by three times to increase resolution locally.

Lastly, inaccurate geometry at the top of the structure often projects to sky pixels in the input images. Since sky pixels are usually much brighter, the estimated albedos become very high and cause visible artifacts in the rendering. To address this, we adopt a simple thresholding method that removes mesh vertices that have near upward normals and have high albedo values. More concretely, we drop a mesh vertex, if the associated surface normal is within 9 degrees from the up-direction, and the estimated albedo value is more than 255×2 in any of the three color channels.

2.4 Experimental Results

This section presents the first large scale 3D reconstructions with lighting and reflectance models from community photo collections mixed with aerial and streetview imagery. Figure 2.4 shows

Dataset	# Input Images			# Images after SfM	# MVS points	# Mesh vertices	Running time [hour]		
	Flickr	Aerial	Street-view				Visibility estimation	Lighting estimation	Per-point albedo estimation
Colosseum	140k	77	14	3,267	10m	27m	20	~ 1	3
San Marco	14k	33	0	2,687	23m	34m	23	~ 1	4

Table 2.1: Statistics of our datasets.

the two datasets used in our experiments, where some statistics are given in Table 2.1. The computational time is collected by running the system on a cluster of 12 nodes. The visibility and the per-point albedo estimation processes distribute workload to all the 12 node while the lighting estimation process runs on a single core. Figure 2.4 shows overlook albedo renderings. Note that in the Colosseum model rendering, the points on the Colosseum are mostly reconstructed from ground level images, thus are much denser than the points on the rest of the city. The rest of the city is mostly created from aerial images that are taken within a short period of time. Since there is not much lighting variation in the aerial images, shadows on the ground are baked into the albedo.

2.4.1 Visual Turing Test

We conducted a series of Visual Turing Tests to evaluate the realism of our renderings using Amazon Mechanical Turk. We present a pair of images, one real and one rendered, from the same viewpoint and illumination condition, then ask the subject to specify which is "more realistic".

The results of the Visual Turing Tests depend on the image resolution. Simply put, the higher the image resolution, the easier it is to detect small imperfections in the reconstructed model. Therefore, we conducted tests for four different image resolutions, in particular, when the longer side of an image is 100, 200, 400, and 600 pixels in length.

We chose hundred randomly selected Flickr photos as reference views. Each view is presented in four resolution levels, and there are in total four hundreds image pairs in the study. Each image pair is shown to twenty test subjects. Low resolution images are sent to workers prior to high resolution images for the same viewpoint, to avoid having the high-res results (which are easier for

Figure 2.5: Visual Turing test. In each image pair, the ground truth image is on the left and our result is on the right.

subjects to get right) pollute the low-res tests. Some of the image pairs are shown in Figure 2.5, where the real photos are on the left and our rendered images are on the right. Note that we don't feed the estimated shadow map into the rendering process as it contains shadows from foreground occluders. The sky is rendered with a simple sky dome texture mapping.

Visual Turing Test results are provided in Figure 2.6a, where the x-axis corresponds to the hundred examples, and the y-axis is the probability, in which our renderings succeeded on the test, that is, fooled the subjects. Examples are sorted along the x axis in the ascending order of the success rate in the 100 pixel resolution. Clearly, the test is easier at lower resolution. Indeed, a handful

of low-res rendered images actually passed the Visual Turing Test, meaning that the majority of the subjects believed our renderings are more realistic than the photos. The average success rate at the four resolutions are 0.3455, 0.16, 0.0735, and 0.034, respectively. Moreover, 30% of the subjects were fooled on almost half of the low-res tests, which suggests that passing the low-res Visual Turing Test is perhaps a goal within reach for 3D reconstruction research. Interestingly, there are a couple viewpoints in which subjects had trouble identifying real photos even for the highest resolution.

Figure 2.6b summarizes the statistics of the success rates (y-axis) over the hundred examples for the four resolution levels (x-axis). For each resolution, the five horizontal markings correspond to the minimum, lower quartile, median, upper quartile, and the maxium of the success rates.

Figure 2.6c illustrates how many tests (out of four hundreds) are completed by each of the 142 subjects. Note that one worker participates in multiple tests, but cannot do the same test more than once (same photo, same resolution).

Finally, there are a number of other factors (apart from resolution) that appear to be correlated with success or failure on the Visual Turing Test. One is the presence of people (Figure 2.7). Subjects are much more likely to pick a photo as more realistic if the photo contains people, which suggests that adding people to the renderings could improve realism. Visual artifacts can also give our results away. For the right-most result in Figure 2.7, the viewpoint is located right behind some geometry fragments which occlude the object of interest and cause severe artifacts.

2.4.2 More Evaluations

To further validate the accuracy of our lighting estimation, we render images with and without shadow effects (Figure 2.8). The shadows (highlighted with green ellipses) rendered with our estimated lighting configurations match those in the input images. The rightmost column shows the renderings from an aerial viewpoint, illustrating the presence of large shadows cast by the tower. Note that since we compute reflectance and lighting parameters, it is easy to relit any photos from any viewpoint using any lighting configurations, as the example in Figure 2.1.

One may suggest an alternative solution for reproducing lighting effects without estimating

lighting and albedo, for example, by computing average/median colors over the mesh from visible images and applying a histogram matching to ground truth images. However, as illustrated in Figure 2.9, such an approach does not produce any directional lighting effects, which is crucial to visual fidelity. It also suffers from inconsistent colorization, because it does not properly handle widely varying viewpoints and illumination conditions that are present in Internet photo collections.

Figure 2.10 illustrates the importance of the visibility test in our system, which removes the influences of foreground occluders that are often present in community photo collections. Our system can reveal structure behind occluders as if they are lit under the same illumination conditions. It rejects fore-ground occluders by examining the color consistency over the entire image collection, and can reveal the structure behind, while removing the foreground objects. Figure 2.10c shows a close-up view of structure where the subdivision scheme was used to increase the mesh resolution, which illustrates the fidelity of our reconstruction even at an inch-scale in a city-scale 3D model, thanks to two components. First, combining aerial and ground level images enables dynamic scale range so that our mesh has an adaptive resolution as more points are devoted to more exposed regions. Second, the subdivision scheme produces extremely fine grain local texture.

We provide additional details and results on the Visual Turing Test. Figure 2.11 shows a screen shot of the test user interface. We randomized the order in which the reference photo and rendered result were shown to avoid position bias (e.g., 50% of the time, the reference photo appears above the rendered image). Figure 2.12 illustrates the four resolution levels.

We show additional typical good and bad image results in Figures 2.13 and 2.14. Even with good geometry rendering, subjects are still more likely to choose the reference photo if people are present (e.g., the first example in Figure 2.14). Figure 2.15(b) shows the increase in performance if we omit photos containing people; observe that the scores are significantly increased, compared to 2.15(a). In some cases, the photos themselves may appear unrealistic; for example, 2 of the 100 randomly selected photos are black-and-white. Subjects rate them as less realistic than the (colored) renderings (Figure 2.16).

2.5 Conclusions and Limitations

In this chapter, I have presented a system to capture and render relightable scene reconstructions from massive unstructured photo collections consisting of Flickr photos, streetview and aerial images. Our system captures a wide range of lighting variations and scene reflectance, and recovers fine grain texture details. The evaluation on a large scale Visual Turing Test demonstrates the effectiveness of our system.

As a step towards solving the grand challenge of *Visual Turing Test*, our system has notable limitations and thus a number of areas for future work. We have not modeled ambient occlusion which is an important lighting effect. There is one coupled scale ambiguity between lighting colors and albedo values. Simple geometry might not provide enough information for light estimation, which further introduced ambiguity. Our system models outdoor environments under the sun and sky illuminations. It would be interesting to extend the framework to night-time shots. Lastly, our system optimizes surface normals, but it is not clear so far if this improves the geometric fidelity.

In the previous section, I discussed the images with low pass score in our Visual Turing Test. These "failed" results suggest next steps (Chapter 3-6) of our work on pushing the research of *photo-realistic scene modeling*. Specifically, to be able to eventually passing the test, we need reliable ground-to-aerial registration (Chapter 3), much better geometry with fewer artifacts (Chapter 4), better visualization techniques that take geometry error into account (Chapter 5), and being able to model transient objects in the scene (Chapter 6).

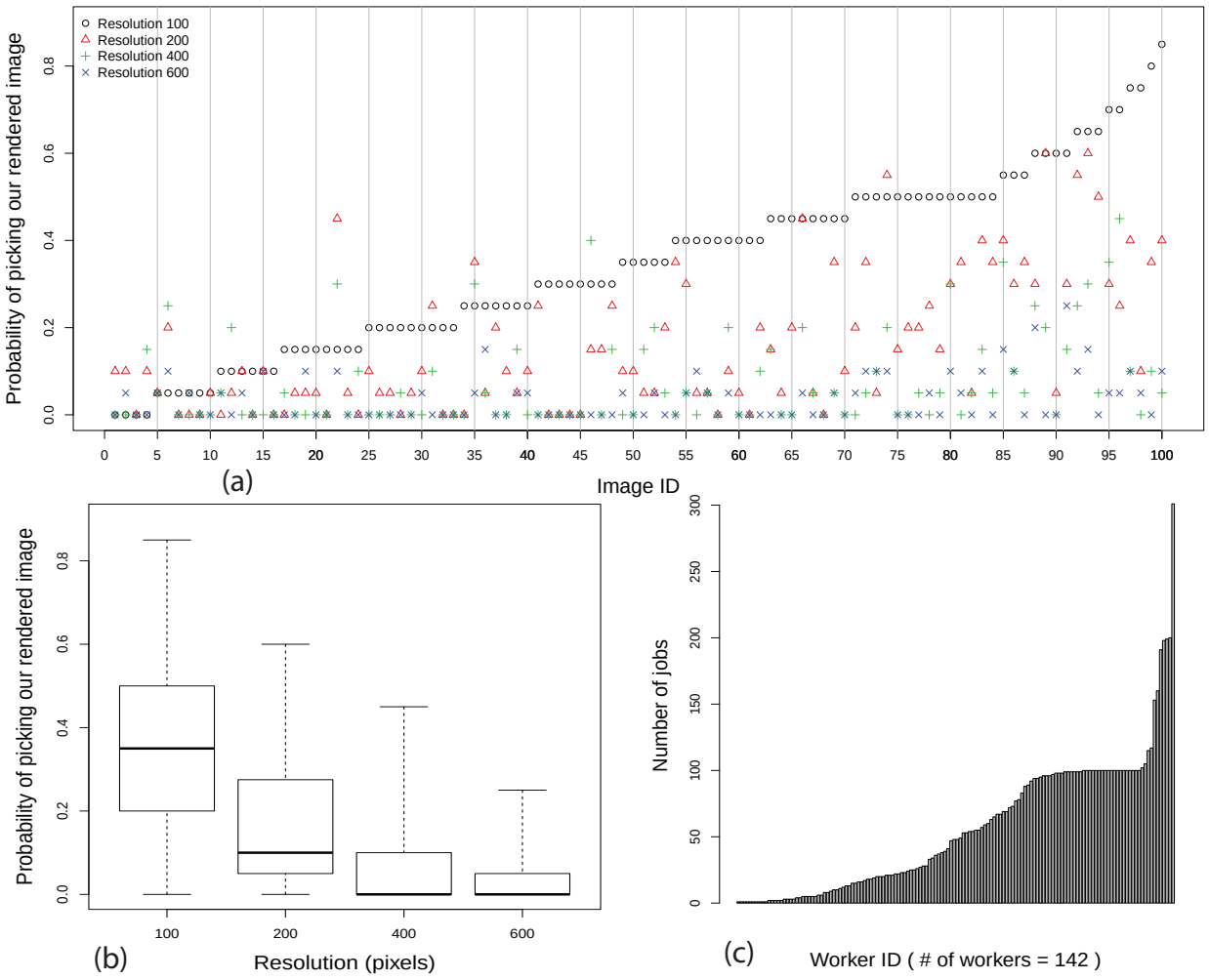


Figure 2.6: Statistics of the Visual Turing Test. Please zoom in for a better visualization of the plot. (a) Per-image average. (b) Per resolution level statistics. (c) Worker plots.

Figure 2.7: Typical failure cases. Bad geometry and people are two major causes for our method to fail the Visual Turing Test. More than 90% of test subjects pick the reference photos (left) as more realistic in every resolution level for these examples.

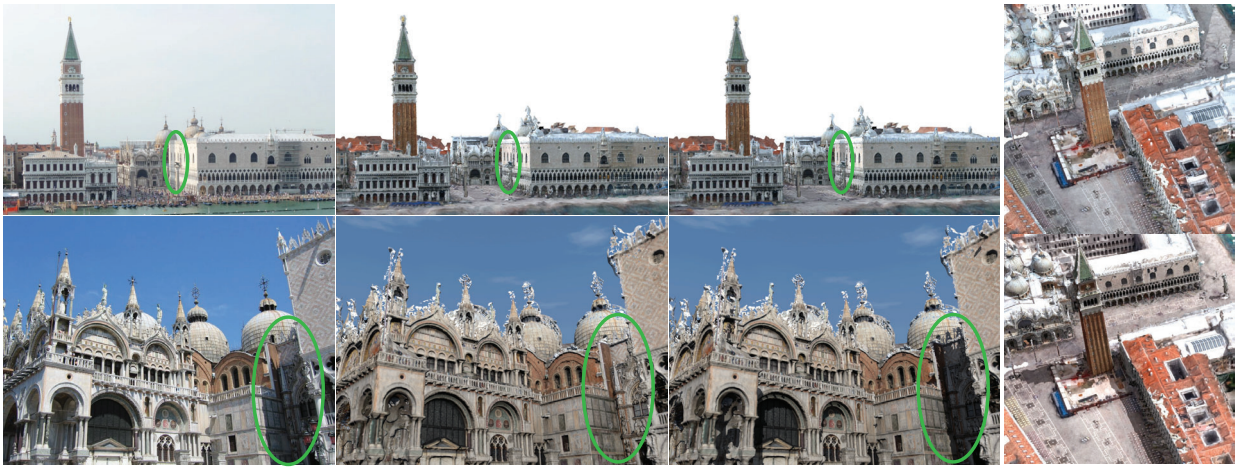


Figure 2.8: Validating the accuracy of our lighting estimation.

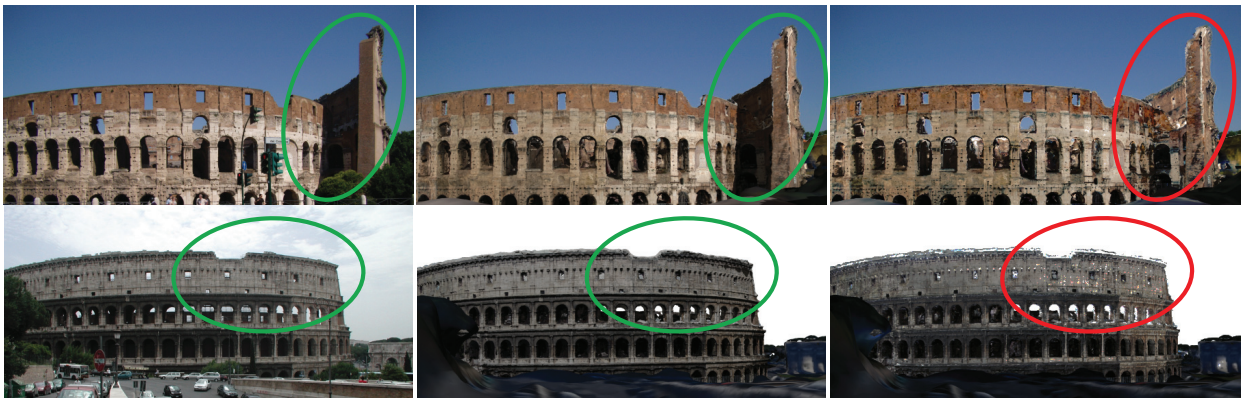


Figure 2.9: Rendering with median color and histogram matching. Left: input ground truth images; middle: our results; right: images rendered from average pixel color and applying histogram matching to the ground truth image. Note the lack of directional lighting effects and color noise in red ellipses.

(a) (b) (c)

Figure 2.10: Our rendered image removes fore-ground objects. (a) A reference photo. (b) Rendered result. (c) A close-up view of (b).

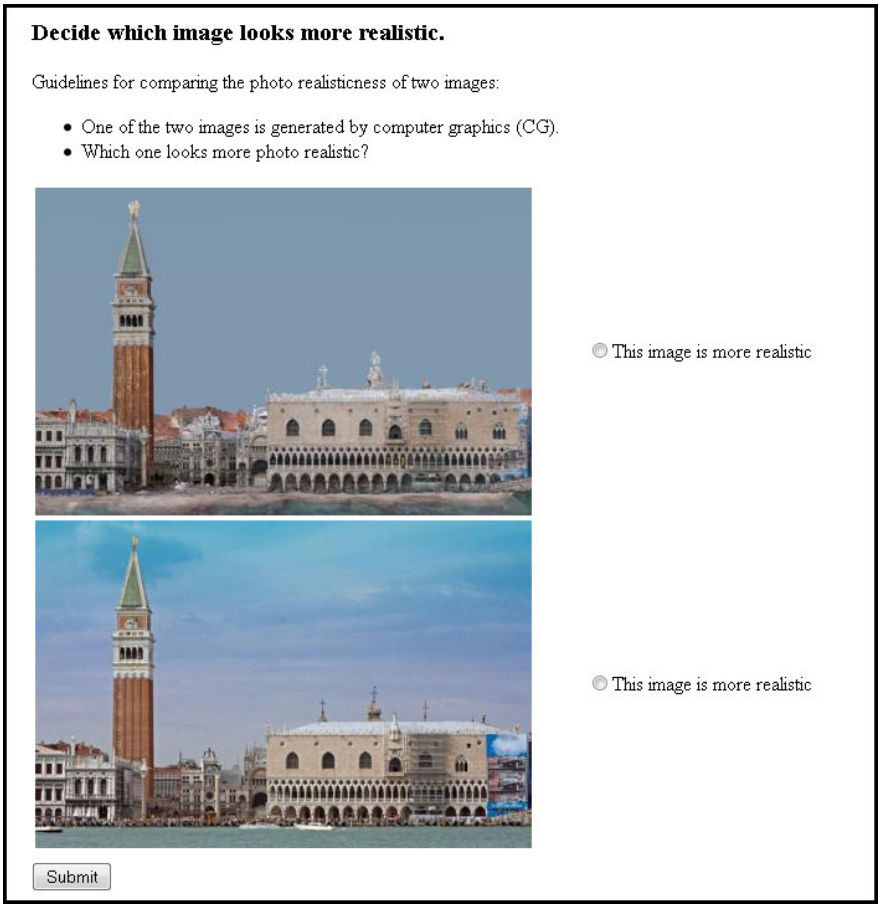


Figure 2.11: The UI for the Visual Turing Test shown to workers on Amazon Mechanical Turk. In this case, the top image is our rendering, while the bottom one is a real photo.

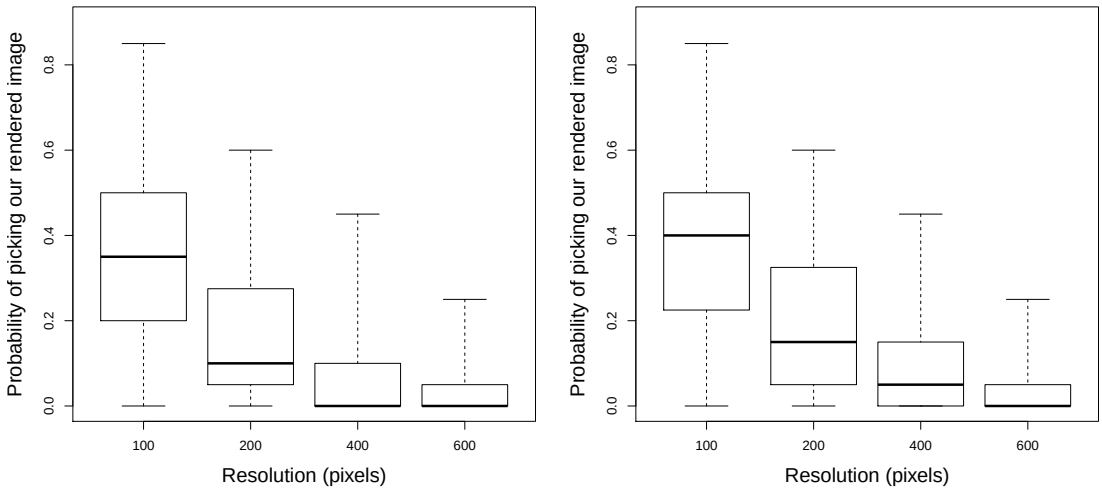
100px 200px 400px 600px

Figure 2.12: Test images at different resolutions. For each pair at a given resolution, the reference photo is on the left, and the rendered image is on the right.

70%, 5% 60%, 25% 65%, 15% 85%, 10%

Figure 2.13: Additional typical good results (top real, bottom rendered). The numbers are the probabilities that our rendered image fools test subjects at resolution levels 100 and 600, respectively.

Figure 2.14: Additional typical bad results (left real, right rendered). More than 90% of test subjects pick the reference photos as more realistic in every resolution level.



(a) On all 100 images. (b) On 75 images with no people.

Figure 2.15: Performance increase from omitting photos with people.

Figure 2.16: Black and white photos are anomalous (left real, right rendered). 10% and 20% test subjects choose the rendered image as more photo-realistic at the 600 resolution level. The numbers increase to 50% and 60% at the 100 resolution level.

Chapter 3

MATCHING GROUND AND AERIAL VIEWS

In the previous chapter, I presented an approach of reconstructing two of the first relight-able city scale models using both aerial imagery and ground level images. One of the challenges raised in that approach is registering these two sources of imagery in the SfM stage. We used SIFT feature matching [48]. However, the experiments were limited to two landmarks, the Colosseum in Rome and San Marco Square in Venice, which appear relatively large even in the aerial views. In particular, we employ imagery from low-altitude helicopters and use overlook views from the tall towers (i.e., semi-aerial views) to bridge the gap between aerial and ground viewpoints. Unfortunately, such semi-aerial views are not typically available for most landmarks. To address this challenge and extend the visual Turing test on a large number of real-world landmarks, in this chapter, I present a novel ground-to-aerial geo-registration approach that works for a much broader range of landmarks and does not rely on semi-aerial views.

So why registering ground and aerial imagery is important? If we take a closer look the problem, Internet photos in general are mostly taken from the ground, therefore the reconstructed multi-view stereo (MVS) models are highly detailed, but are often disconnected due to the lack of photo coverage in less popular areas. At the same time, these ground-level models are usually not accurately geo-located in a global coordinate system, making it difficult for them to support applications such as digital mapping and autonomous navigation. In contrast, commercial products like Google Earth, Apple's 3D Maps, and Bing maps use geo-located aerial imagery for more uniform 3D reconstruction. The aerial 3D models are complete but much less detailed than the ground-level models. We want achieve the best of both worlds by using the aerial and ground imagery together in 3D reconstructions.

However, it is difficult to directly match ground and aerial images together, due to the large

(a) (b) (c)

Figure 3.1: A typical scenario for the ground-to-aerial image registration problem. (a) An aerial image shows part of the city of Rome. The red rectangle highlights the Sant'Andrea della Valle. Even for human vision, it is difficult to find the target geometry from the aerial view. (b) A close-up view of (a). (c) A ground image of Sant'Andrea della Valle.

differences in their camera viewpoints and imaging conditions. Figure 3.1 illustrates the challenges. First, in the case of aerial images, the scene is observed from much greater distances and at very different angles than in the case of ground images. Typically, landmarks roughly corresponds to 400×400 pixels in high resolution aerial images. Second, depending on the direction of the sunlight, certain facades appear very dark in the aerial images, making standard feature detection and matching difficult. In addition, most previous wide-baseline feature-matching methods rely on dominant planar structures [52, 80], but the actual 3D geometry can be more complicated, an assumption that fails for many landmarks (e.g., Figure 1). In fact, we rarely see compelling 3D reconstructions obtained from the mix of aerial and ground images, especially across a large number of scenes.

In this chapter, we address the problem of registering ground-level models to aerial imagery. To this end, we introduce a new viewpoint-*dependent* matching technique to establish pixel accurate feature correspondences between aerial and ground imagery. Our approach helps mitigate the problems caused by the large discrepancies in viewing angles and image resolutions that have frustrated prior efforts. As a result, we can now achieve pixel-level accuracy in geo-registration

(within a few centimeters in many cases). This is a significant improvement over the ~ 5.5-meter accuracy attainable using GPS or text labels in prior approaches [47, 84].

Our contributions are: 1) a novel viewpoint-dependent matching method that handles large viewpoint changes; 2) a fully automated geo-registration pipeline for matching ground-level photos to aerial imagery; and 3) a large-scale geo-registration evaluation which consists of the most popular outdoor landmarks in Rome, demonstrating an approximately 70% success rate with the proposed system. Aligning ground-based models enables adding dramatically more details to aerial reconstructions.

The remainder of the chapter is organized as follows. I first give a briefly introduction to relate work on image geo-registration in Section 3.1, and describe an overview of our system in Section 3.2. The proposed viewpoint-dependent feature matching is presented in Section 3.3, and evaluated with a large scale evaluation in Section 3.5. We discuss the work and show its limitations in Section 3.6.

Note that the work in this chapter was published in the 2014 International Conference on 3D Vision [66].

3.1 Related Work

Ground-to-aerial image matching for geo-registration is difficult, and standard feature matching techniques often fail (Figure 3.3a), necessitating manual intervention [16, 71]. Very few approaches have demonstrated fully automatic matching and reconstruction of aerial and ground imagery.

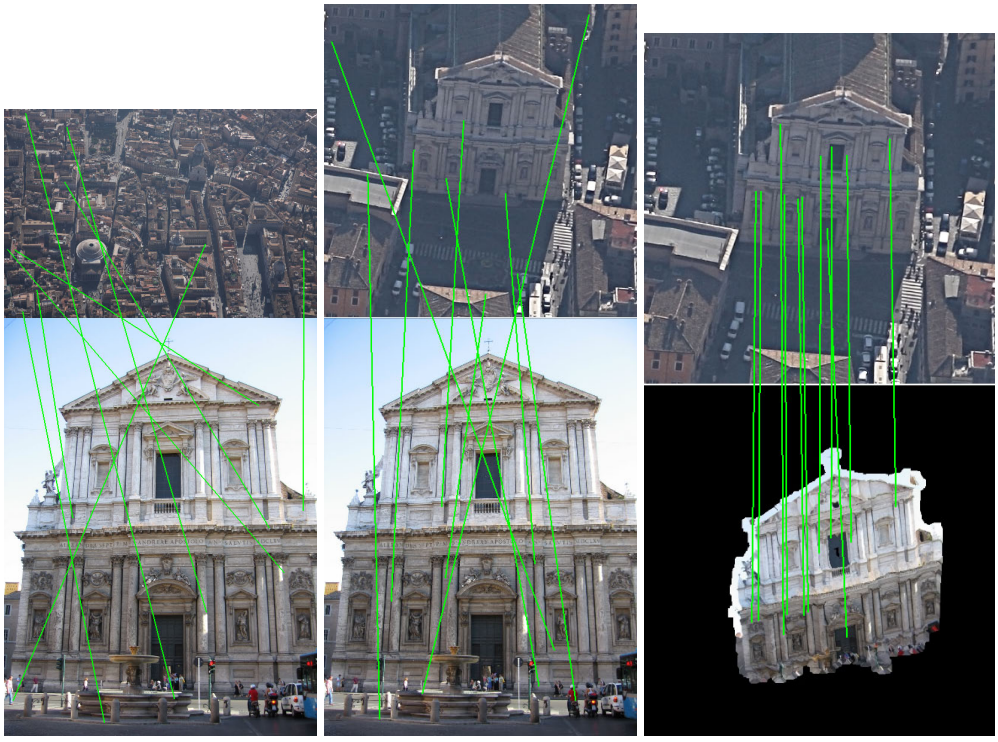
Coarse geo-registration of ground-level models is possible using photo meta-data. GPS and text labels are commonly used to estimate rough geographic location. For more accurate registration, early attempts focused on matching aerial image edges (or map edges) to 2D projections of ground models (projection along the "up" vector) [24, 39, 84]. However, reliable matching by these methods requires multiple facades (or multiple map edges) in the ground-level models, which is not always the case. Furthermore, their altitude estimation is often less accurate. Recently, researchers have looked into using other sources of geo-location proxies, for example, matching ground im-

Figure 3.2: Warping the ground-level image into target view using depth maps and corresponding camera poses.

ages to geo-located Google Street View photos [23, 78], or looking for GPS tagged images with similar appearance [32, 42], matching to geo-located ground-level 3D points [47]. Nevertheless, it is difficult for these methods to achieve high precision; for example, the average error is about 20 meters in [84], and 5.5 meters in [47]. In this chapter, we establish feature matches between aerial and ground imagery for geo-registration with pixel-level (centimeter) accuracy.

Invariant features (e.g., SIFT) are typically used to tackle viewpoint changes. Beyond the invariance to scale changes, there exists a rich body of work on affine or perspective invariance for improved robustness to large viewpoint changes [72, 52, 80, 81, 11]. These techniques usually assume dominant planar geometry for simulating different views with a homography or affine transformation, and as a result, their performance suffers when the scene geometry is complex or has many foreground occluders. Furthermore, the viewing angle variation between the aerial and ground imagery is so dramatic (45°) that it usually falls outside the operating range of most image matching methods. Our experiments with state-of-the-art methods ([52] and [80]) show that they are insufficient for our aerial to ground registration task.

An alternative to image matching is direct 3D model alignment using 3D feature points [12, 36, 83, 59]. These approaches assume meshes with similar resolutions and with a substantial



(a) (b) (c)

Figure 3.3: Two-view matching of ground and aerial images. (a) Matching the whole aerial photo with the ground image produces mostly useless feature pairs. (b) Matching an automatically cropped and sharpened aerial photo with the ground image also fails. (c) More reliable feature correspondence is obtained by matching the cropped aerial photo with the synthesized target view from the ground image.

amount of overlap. Unfortunately, aerial models are much coarser than ground models (meter versus centimeter resolution), and it is difficult to extract accurate mesh features for matching. Furthermore, the aerial and ground models usually do not overlap enough. Geometry that is visible in aerial views, e.g., rooftops, rarely appears in ground images, and vice versa. Therefore, it is difficult to achieve pixel-level accuracy via 3D feature based techniques.

3.2 Algorithm Overview

Given ground-level MVS reconstructions, our goal is to accurately and automatically align these MVS models to the aerial images, which have been geo-referenced already. The following is an overview of the proposed algorithm.

We first obtain an approximate geo-referenced ground-based MVS model by performing GPS-based geo-registration using the EXIF tags of ground images. The ground-level images are collected from Flickr [19], of which roughly 10% have GPS tags [22]. As many of the GPS tags are inaccurate (due to poor reception, user tagging, etc.), the RANSAC process typically can locate the 3D models only within a 20 meter range [84].

Based on the estimated geo-location of the ground models, we retrieve oblique aerial views from Google Maps [29].[1] The oblique images are captured from 4 different directions, east, south, west, and north. Our method finds feature matches between the ground and aerial images to geolocate the ground models to pixel-level accuracy.

Section 3.3 proposes a new viewpoint-dependent matching method, which effectively deals with the large viewpoint differences between the ground-level and the aerial images. Section 3.4 presents our aerial view-selection algorithm, which leads to efficient and robust matching. The final 3D transformation is recovered by applying RANSAC to the feature matches. In our experiments, 41 out of 59 outdoor landmarks in Rome were successfully registered, a 70% success rate.

[1] Note that we don't need aerial 3D geometry for geo-registering the ground models.

3.3 Viewpoint-Dependent Feature Matching

We consider the problem of finding accurate feature matches between two sets of images with large viewpoint changes. In ground-to-aerial matching, we have accurate geo-reference information for aerial images, while the location of the ground MVS reconstructions can be recovered only approximately from GPS tags.

SIFT is sufficient for small viewpoint changes, as the local transformations are close to similarity transforms. For larger viewpoint changes, affine invariance can be achieved on planar structures [52]. When accurate, dense 3D reconstructions of both models are available, improved invariance can be achieved with viewpoint-invariant patches extracted from synthesized local orthogonal views [80]. Unfortunately, the ground-to-aerial registration problem has (i) drastic viewpoint changes, (ii) very complicated geometry, and (iii) sparse and noisy reconstructions from aerial imagery. Therefore, none of the above techniques are applicable.

Instead of seeking invariant feature detections, we propose to match view-dependent features by exploiting approximate alignment information and underlying 3D geometry. Consider matching a ground MVS model (source) and an aerial image (target). We assume that dense depth maps exist for the source images, and that approximate alignment information is available, from which we can synthesize the source images rendered from the target viewpoint. Standard small-baseline features can then be applied to match the synthesized views with the target image. Note this is fundamentally different from VIP matching [80], which needs to synthesize invariant views for both source and target images. In fact, the proposed matching method only requires the dense 3D geometry of the source imagery.

For the target view synthesis, we compute MVS reconstructions of the source imagery and create dense depth maps with a bilateral filter-based interpolation process as described in [33]. The depth maps are first computed by back-projecting visible MVS points to each view, and then interpolated with a bilateral filter to fill in possible holes (window radius is 10 pixels and the regularization parameter is 0.16). We then further smooth the depth maps with a Gaussian filter of size 11 to reduce warping artifacts.

Given the recovered depth maps, we are able to synthesize the target view for each source (ground level) image by depth-based warping . After that, we use SIFT to match with the aerial views. See Fig. 3.3(c) for an example. Experiments show that the viewpoint-dependent feature matching works well for the large viewpoint and scale changes in the aerial and ground matching, where direct matching will typically fail.

A key advantage of our viewpoint-dependent feature matching over [80] is the ability to handle large scale changes and exploit the approximate alignment information. By warping the ground level images into the aerial views, and matching at the resolution of the aerial views, our algorithm naturally ignores the small 3D structures that are invisible in the aerial views. In fact, we found that the failures of [80] in our problem are often due to feature matching at the wrong scales.

3.4 Aerial View Selection and Matching

The initial alignment for the ground models is obtained by using a GPS tag based RANSAC, which typically has an accuracy of around 20 meters. This precision allows us to automatically select the proper aerial views for matching.

First, given an approximately aligned ground model, we identify the aerial images that contain the model in their viewports. Specifically, for each of the four aerial oblique viewing directions, we select the aerial image, whose center is the closest to the center of the ground model on the image.

Second, each ground model usually corresponds to a small fraction of an aerial image, due to the large scale changes. For the purpose of efficiency, our system automatically crops the aerial images to narrow down the search space (Figure 3.3b). Specifically, based on the 3D bounding box of the ground model, we extract the sub-images that cover the desired region of interest by projecting the bounding box into the aerial views. In our implementation, we use 5 pre-defined crop sizes: 401×401, 601×601, 801×801, 1001×1001, and 1201×1201. To account for the error in the initial registration, we choose one crop size that is approximately twice the size of the region of interest.

Third, we apply contrast adjustment to deal with landmarks in shadow. The aerial images are

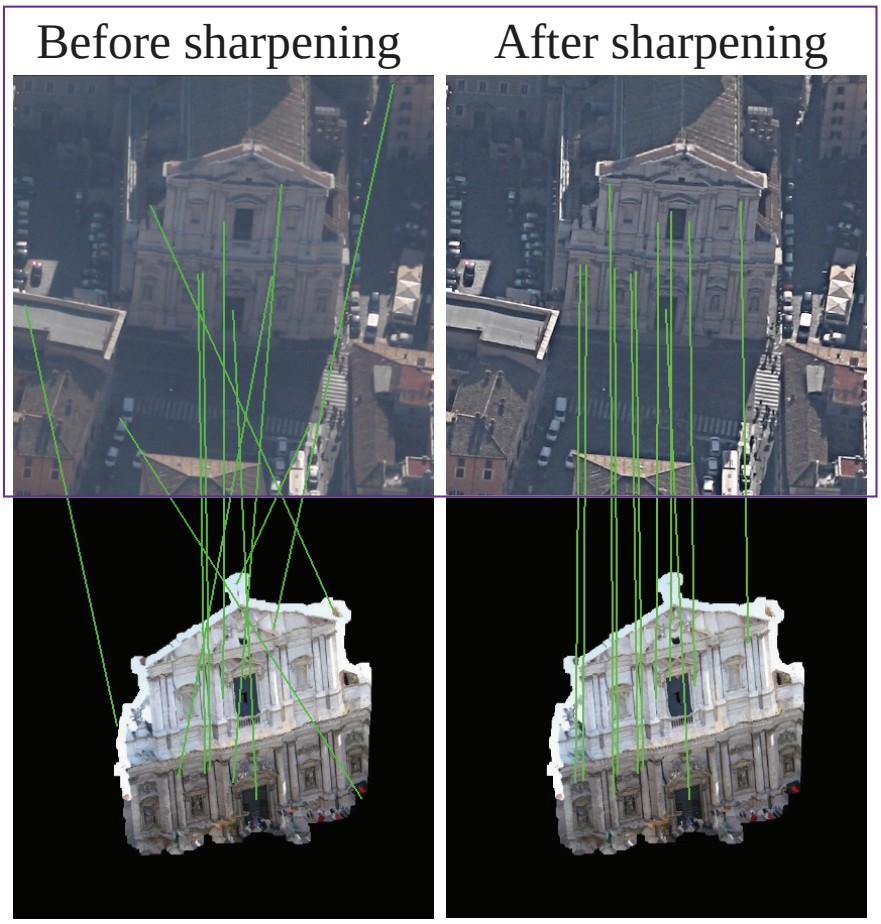
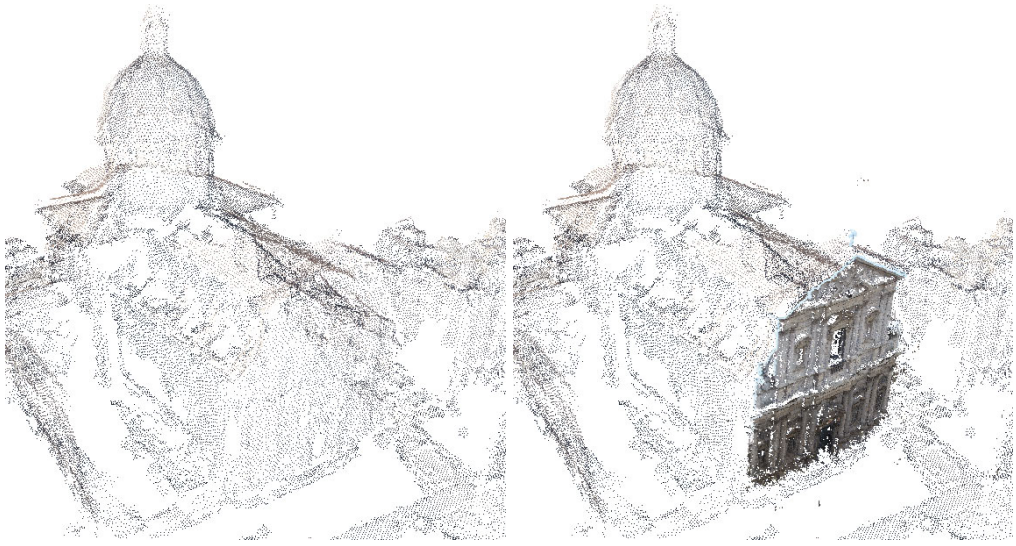


Figure 3.4: Image sharpening brings up the contrast level of aerial views in shadow, improving the feature matching. Note that the aerial crop we show in Figure 3.3 (b,c) is after sharpening.



(a) (b)

Figure 3.5: Applying the estimated similarity transform to the ground model. (a) The aerial MVS model. (b) The transformed ground model on top of the aerial model. Note that accurate geo-registration has been achieved.

all taken on sunny days in Rome, Italy. Therefore, north-facing facades are always in shadow, resulting in low contrast aerial views. The contrast mismatch has a significant effect in SIFT matching. To address it, we apply an unsharp mask to enhance these north facing views. That is, $I_s = (1 + a)I - aI * g$, where I_s is the sharpened image, I is the original cropped aerial image, g is a 7×7 Gaussian filter with standard deviation $\sigma = 1$, $*$ is the convolution operator, and a is the sharpening ratio (0.25 in our implementation). The aerial view enhancement produces better two-view matching (Figure 3.4), leading to more accurate geo-registration.

Feature matches between the ground and aerial images are then converted into a list of 3D point pairs, denoted $\{(P_i^a, P_i^g)\}$. P_i^a is one (feature) point in the aerial view, back-projected into 3D according to the underlying aerial geometry. P_i^g is the matched ground point, which is back-projected based on the interpolated depth map. The ground-to-air alignment is sought by finding the optimal 3D similarity transformation between the two sets of 3D points. The error to be minimized can be written as $\sum_i \|P_i^a - (s\mathbf{R}P_i^g + \mathbf{T})\|^2$, where s is the scale factor, $\mathbf{R}$ is the rotation matrix, and

$\mathbf{T}$ is the translation vector. The closed form solution for the similarity transform is given by [73]. For robustness to outliers, we use a RANSAC process to find the transformation with the largest number of inliers. We set the distance threshold to 5 centimeters. To account for possible low inlier ratios, we empirically let the RANSAC process take $100,000$ iterations, where each iteration randomly picks 3 pairs of points. This would guarantee a success probability of 0.999996 even if the inlier ratio is 5%. Finally, the estimated similarity transform is applied to the ground model for geo-registration (Figure 3.5).

3.5 Evaluation

The proposed system is evaluated on popular landmarks in the city of Rome. We downloaded ground images from Flickr, and ran a standard 3D reconstruction pipeline (VisualSFM [79] followed by PMVS [27]). After removing indoor scenes and small models of less than 20 images, we keep 59 datasets for the geo-registration experiment. The number of images in each dataset varies from 28 (Santa Croce in Gerusalemme) to 5000 (Colosseum). 12 of the 59 datasets have more than 1000 images. As the geo-registration target, we collected $31,891$ aerial images that cover the entire city of Rome. Most of the computation time is spent in the pre-processing steps computing SfM [79] and MVS [27]. In particular, it takes about a day for each of the biggest datasets with a distributed reconstruction system. The geo-registration process takes less than 20 minutes on a single machine with 8 threads.

The proposed method successfully registers 41 out of the 59 landmarks in Rome, which gives a 69.5% success rate. Qualitative results can be found in Figure 3.6, 3.7, 3.8, and 3.9, which demonstrate the accuracy of the proposed registration method. Our method shows clear improvements over the initial GPS-based geo-registration.

Our view-dependent feature matching approach is critical to handling large viewpoint and scale changes. For comparison, we also run the geo-registration pipeline using standard SIFT [48], ASIFT [52] and VIP matching [80]. As expected, SIFT and ASIFT are not capable of handling the drastic viewpoint changes (See Fig. 3.7). One may suspect that a possible reason of the failure is the ratio-test. We experimented with this hypothesis by disabling the ratio-test. It indeed increases

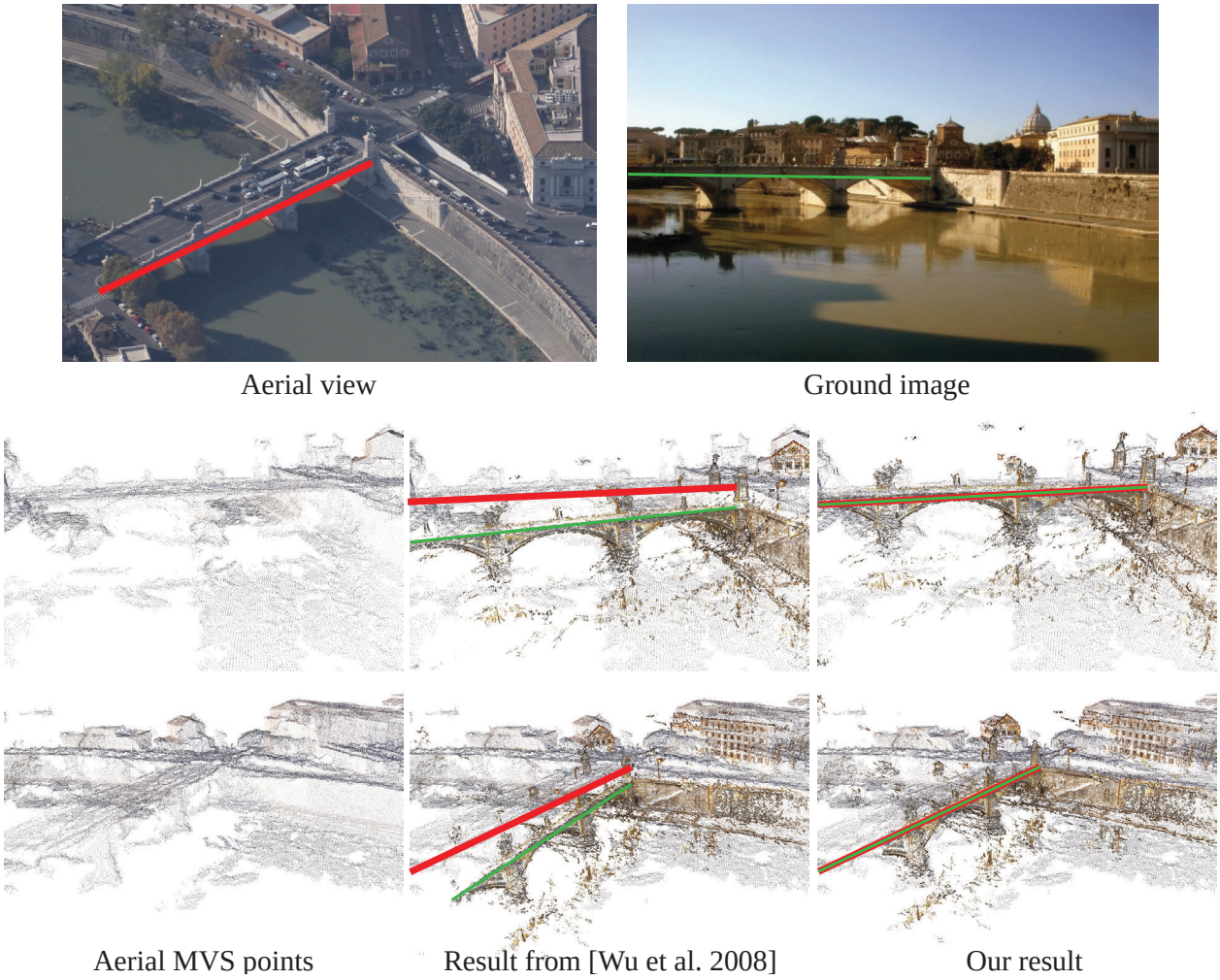


Figure 3.6: Comparing against the VIP matching in [80]. For better visualization, we manually place red and green lines to highlight the bridge beams in aerial and ground models, respectively. Note that the geo-registration from the proposed method is more accurate as the bridge beams from the aerial and ground model overlap. Landmark: Ponte Vittorio Emanuele II.

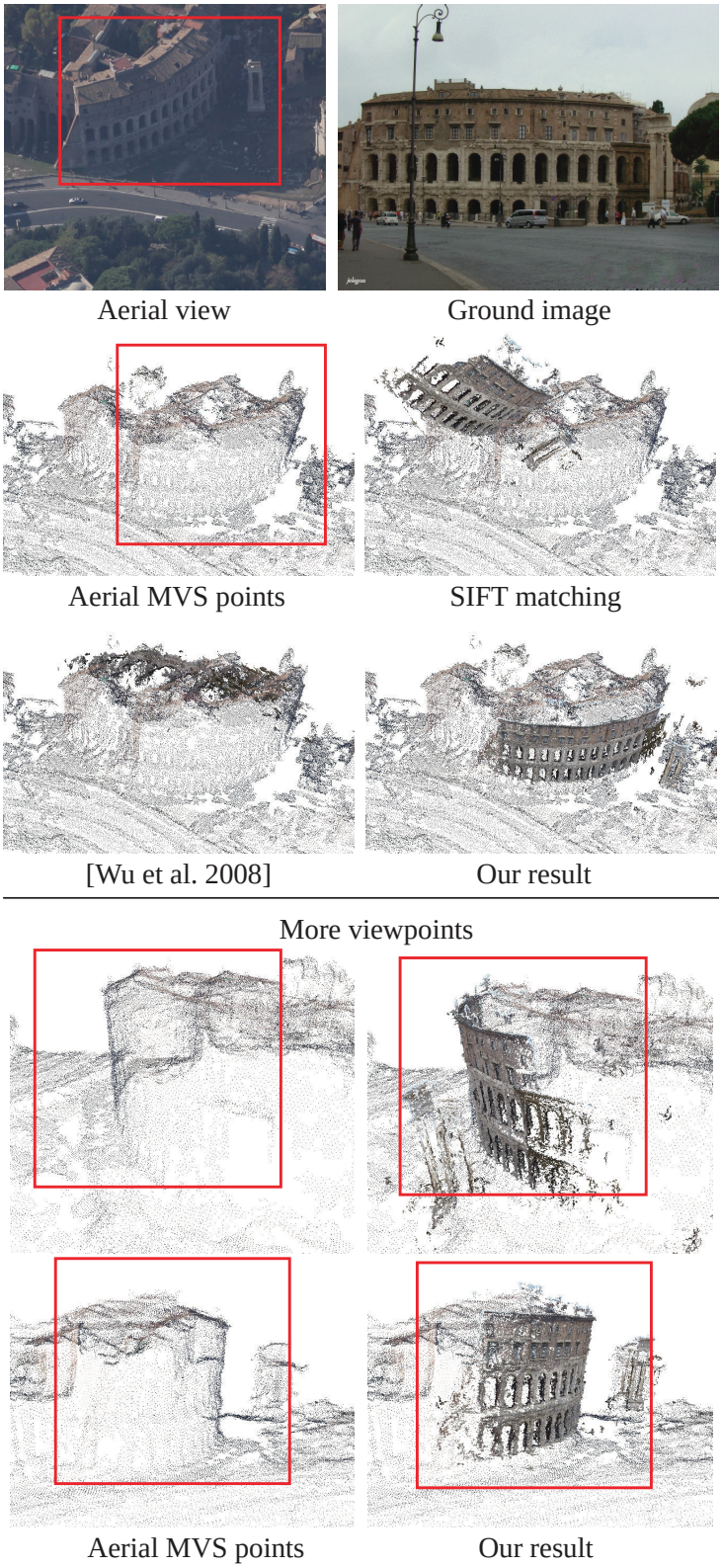


Figure 3.7: Comparing against the baseline method and the VIP matching in [80]. Landmark: Theatre of Marcellus.

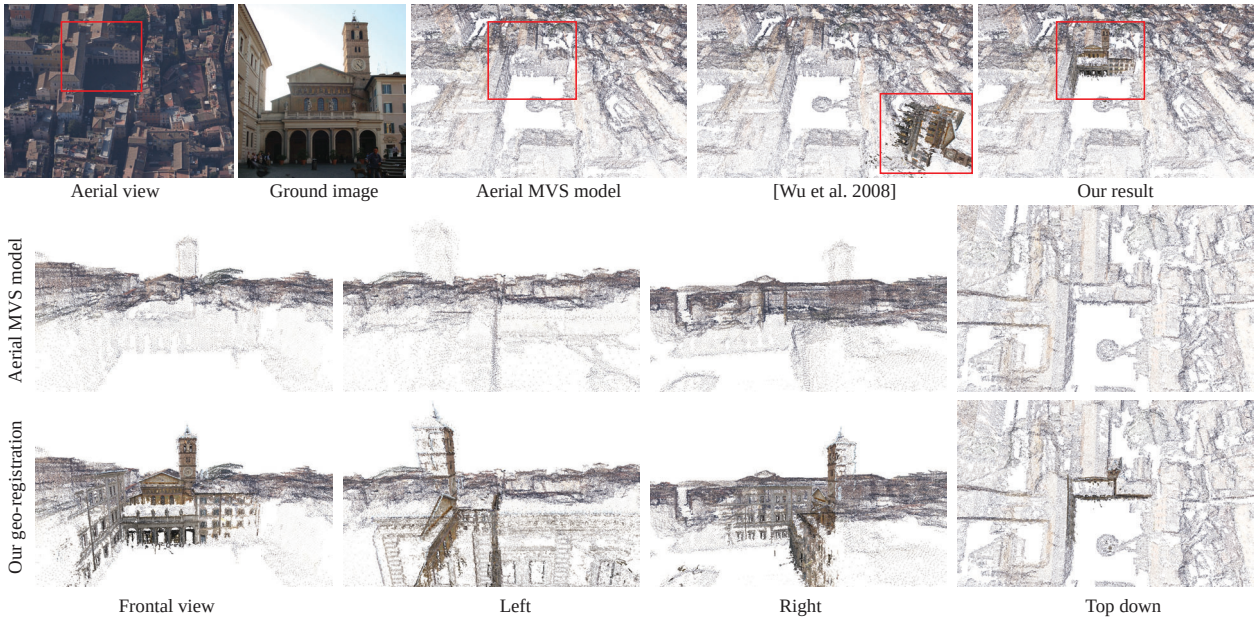


Figure 3.8: Comparing against the VIP matching in [80]. Landmark: Santa Maria in Trastevere.

the number of putative feature matches between the ground and aerial views. However, the feature matches become much noisier. In the end, disabling the ratio-test does not increase the number of successfully registered landmarks. VIP matching works reasonably well for a few datasets where the scene geometry is relatively simple, but fails in most cases. One of the successful examples of VIP matching is shown in Figure 3.6, where the ground MVS model of the bridge has been registered to the aerial model. However, the registration error from VIP matching is significantly larger. Two failure examples are shown in Figure 3.7 and 3.8. VIP detection and matching relies heavily on correctly parsing local scene geometries, e.g., the plane detection process. The performance varies depending on various thresholds which need to be tuned for each landmark. Moreover, the matching tends to get confused by the large number of small 3D features in the ground-level reconstruction.

We tried to conduct quantitative evaluations of the geo-registration accuracy, but it is not clear how to define a good metric. One option is to define a recall score by setting a distance threshold and measuring the percentage of ground MVS points that have aerial reconstructions within the

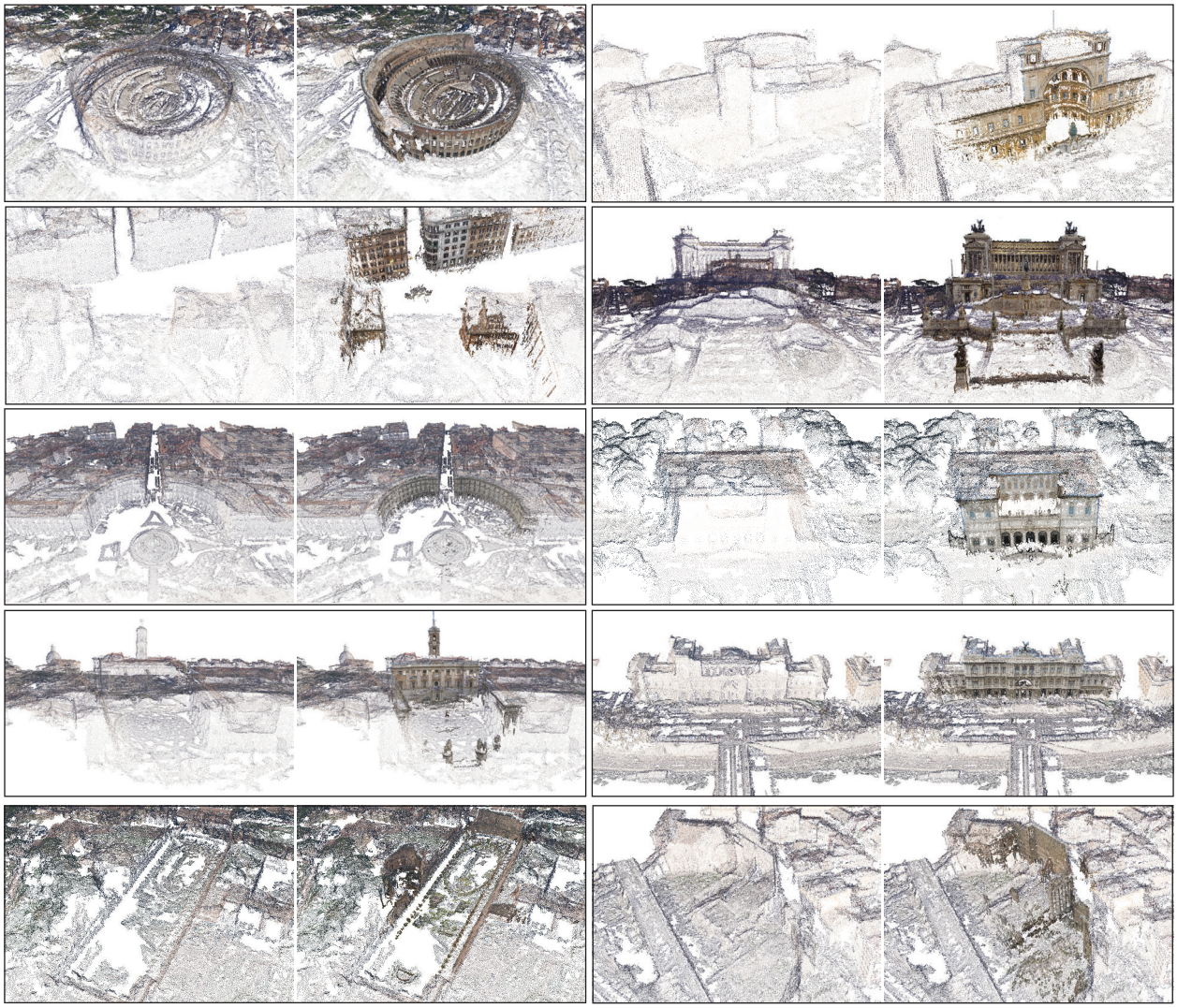
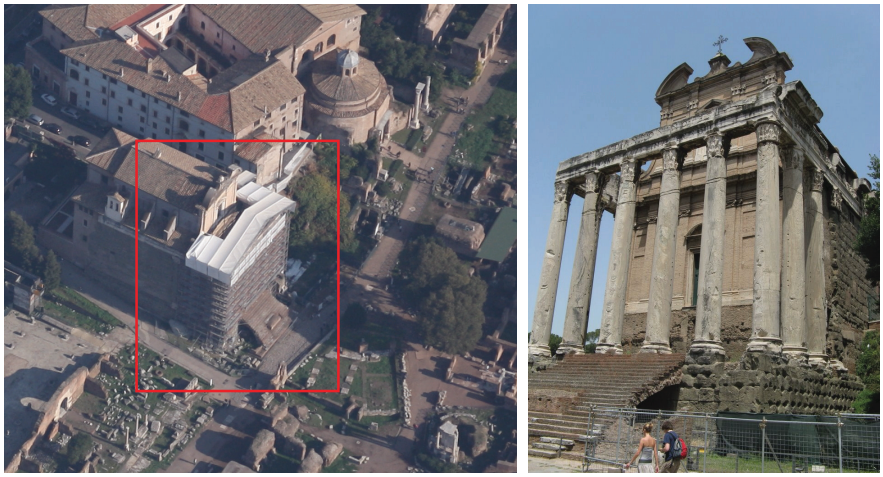


Figure 3.9: More results on matching the aerial view to the ground image. Left: the point cloud from aerial MVS models. Right: our geo-registered models. Please zoom into the original resolution for best visual quality.



Failure case 1: inconsistent aerial/ground appearance

Failure case 2: noisy ground MVS reconstruction

Figure 3.10: Failure cases. The proposed method fails when (i) the aerial/ground geometry is not consistent, and (ii) the ground geometry is noisy.

threshold. However, due to the large difference in resolution and coverage between the aerial and ground models, this metric does not necessarily favor the better geo-registration result. For example, a ground model of the frontal facade of a church may be mis-matched to an incorrect planar region in the aerial model, and produce a comparable or higher recall score. We hope to develop a better metric for the problem in future work.

The proposed approach does fail for some landmarks. Figure 3.10 shows such an example, where the building is under construction in the aerial model, but not in the ground model. When the aerial and ground s are not consistent, the proposed method is not able to find enough feature matches. Another cause of failure is due to noisy ground reconstructions. Since our viewpoint-dependent feature matching relies on the ground geometry for warping, it is vulnerable to noisy ground reconstructions, which result in severely distorted synthesized views.

3.6 Conclusion

This chapter presents a fully automatic system to geo-register ground MVS reconstructions. The system is capable of handling drastic viewpoint variations by adopting a novel view-dependent feature matching approach. We conducted a large scale experiment using 59 popular outdoor landmarks in Rome. Our results are significantly better than existing techniques.

Our approach does have some limitations. It relies on the quality of ground MVS reconstructions, and assumes consistent appearance in both aerial and ground imagery (Figure 3.10). The registration accuracy degrades at the presence of severe occlusions. Although the ultimate goal is to create high-resolution city-scale 3D models, currently we are only able to achieve this desired resolution at city landmarks where dense ground images are available. One topic of future work is to incorporate another source of ground-level imagery such as Google Street View images. Finally, the proposed approach simply estimates a similarity transformation between the aerial and ground models. Slight mis-alignments are observed in some of the datasets, which could be reduced by global bundle adjustment on all the ground and aerial imagery, incorporating the ground-to-aerial feature matches obtained by the proposed algorithm.

Chapter 4

OCCLUDING CONTOUR FOR MULTI-VIEW STEREO

One of the key challenge arises from our study to the Visual Turing Test (Chapter 2) is obtaining high quality geometry. The state-of-the-art pipelines typically produce meshes with noisy surface details and undesirable overly-inflated boundaries which can be easily spotted by human perception [63]. In this chapter, we propose a new approach that dramatic improves MVS quality both around object contours and in surface detail.

Object silhouettes provide remarkably strong shape cues; a single silhouette constrains the entire volume of 3D space that projects outside of it to be "empty." Combined with image intensity cues, silhouettes have been shown to greatly enhance the output of multi-view stereo (MVS) methods particularly for low-textured scene regions [34, 76]. However, as the focus of 3D reconstruction research has shifted out of the lab and into "the wild," [5, 22] silhouettes have become less relevant, as it is not clear how to define, much less separate, the "background." E.g., for applications like city modeling (Google Maps and Apple Maps), there is no concept of a silhouette as the goal is to reconstruct the entire scene.

Occluding contours, aka "internal silhouettes," provide similarly powerful shape cues but in a much more general setting, without the need to define a background. An occluding contour corresponds to a boundary in the image between an object surface (e.g., part of a statue) and another surface further away (e.g., wall) that it partially occludes. In principle, occluding contours could be leveraged similarly to silhouettes, to identify regions of empty space between closer and more distant surfaces. The main challenge, however, is that identifying such free-space regions requires accurate reconstructions of both the foreground and the background surface to start with, i.e., it's a chicken-and-egg problem. For example, both statues and walls are hard to reconstruct due to lack of texture, and give rise to incomplete or noisy models, complicating the inference of

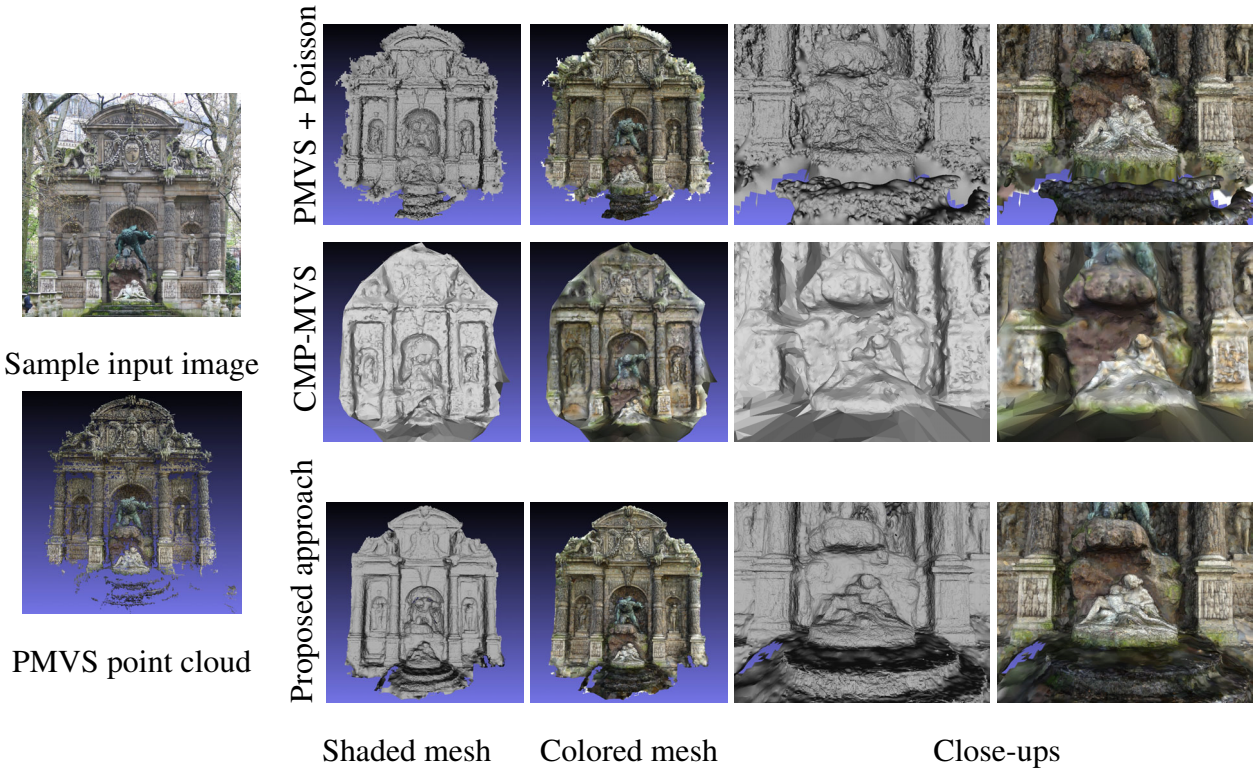


Figure 4.1: Luxembourg Gardens in Paris. The top row shows the baseline reconstruction by PMVS [27] and Poisson Surface Reconstruction software [40]. The middle row shows another baseline reconstruction by CMP-MVS software [37]. Our results are in the bottom row, illustrating much clearner geometry boundary and more accurate details. PMVS+Poisson and our meshes are colorized from the closest PMVS points. CMP-MVS mesh is colored by the reconstruction software.

free space and occluding contours.

Our contributions are 1) a new technique to identify free-space regions arising from occluding contours, and 2) an approach for incorporating the resulting free-space constraints into surface reconstruction. Our approach is based on *extrapolating free space* using the assumption that piecewise constant image regions have simple (quadric) surface geometry. While this type of assumption is commonly used in stereo methods to interpolate nearby surface geometry, e.g., [70], prior work has not considered the effects that the interpolated geometry has on free-space and how to globally propagate this free-space information across the scene in a globally consistent and noise-robust manner. To this end, we introduce *free-space voting* into the Poisson Surface Reconstruction [40] framework, and demonstrate how this novel extension allows for high quality, free-space aware MVS reconstructions.

We show that incorporating this occluding contour information into an MVS method yields considerably cleaner, more accurate, and more complete reconstructions, especially around object boundaries. Our focus is Internet imagery, which tends to be more challenging than lab-captured datasets. The greatest improvements are in areas of very fine-scale geometry, which tend to be lost using standard regularization approaches due to over smoothing; contour information is critical to retaining these structures. When available, our approach can also incorporate standard silhouettes, and we show results on using external (sky-based) and internal silhouettes together to achieve state-of-the-art reconstructions of large landmarks.

The remainder of the chapter is organized as follows. Section 4.1 describes an overview of our approach with intuitions on its design. The detailed algorithm is presented in Section 4.2. I show evaluation results in Section 4.3 and conclude in Section 4.4.

The work in this chapter was published in the 2014 IEEE Conference on Computer Vision and Pattern Recognition [64]

4.1 Overview

Here we present a high level view of our proposed framework. A detailed description appears in Section 4.2.

Starting from an internet photo collection for a given site, we recover the camera positions with freely available structure-from-motion software [67] followed by multi-view stereo using PMVS [27]. PMVS is well-suited to recovering geometry from photo collections, but only gives a semi-dense reconstruction, and can be very sparse in low texture regions. In a typical reconstruction pipeline (Figure 4.2 (a)), these PMVS points are then used directly for surface reconstruction. Instead, we modify the pipeline (Figure 4.2 (b)) to include construction of dense depth maps that respect occluding contours and provide additional shape and visibility information.

We construct an initial depth map for each input view, starting with the PMVS points that are visible in that view. Each initial depth map can be quite sparse, inadequate for providing dense visbility information. To address this, we interpolate the depth maps by optimizing for depths that are smooth between PMVS samples, with an important modification. Observing that depth discontinuities (occluding contours) tend to coincide with image discontinuities, we relax the smoothness near apparent image contours and edges during optimization.

Next, we prepare a free space volume and an augmented point set that will drive the surface reconstruction. We observe that the optimized depth maps are useful both for visibility constraints and for adding new surface points. Though not accurate everywhere, the depth maps tend to be good proxies for space carving. Thus, we create a free space volume directly from the depth maps. In addition, we use the same depth-map visibility constraints to cull spurious PMVS points, e.g., those that can often appear near occluding contours.

In some interpolated areas, the depth maps can also provide reasonable approximations to the true geometry. The intuition is that within regions that do not contain much image detail, including constant albedo regions with little shading variation, PMVS does not recover much, if any, geometry, but that geometry is likely fairly smooth. Our image-guided interpolation algorithm tends to propagate depths well within such textureless regions. In the end, we consider each depth map point as a candidate for surface reconstruction, while retaining only the points that are consistent with the all the depth maps. We combine the surviving depth map points with the PMVS points (that are not in visibility conflict) to construct an augmented point set.

Finally, we solve for a surface that fits to the augmented point set and performs fair hole-filling,

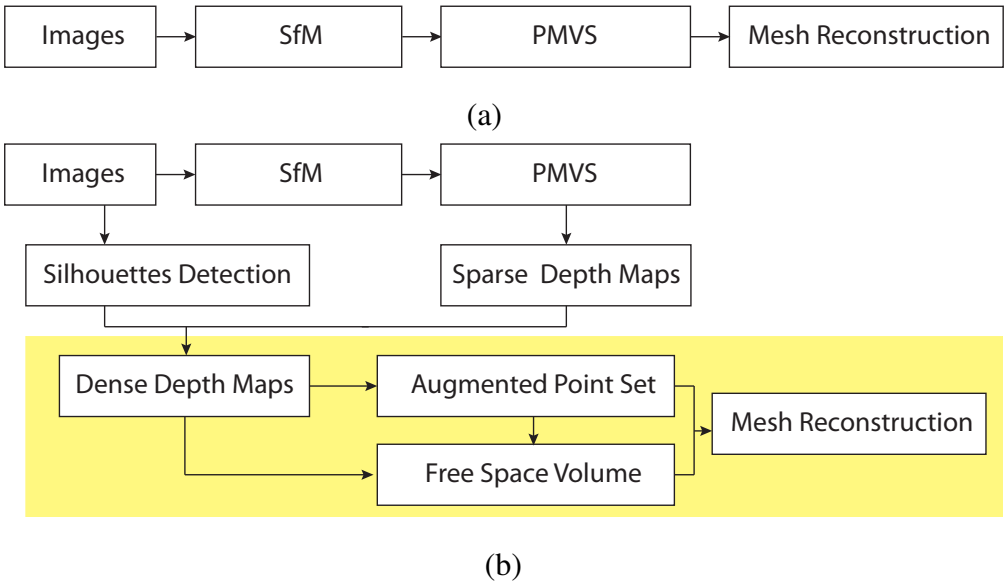
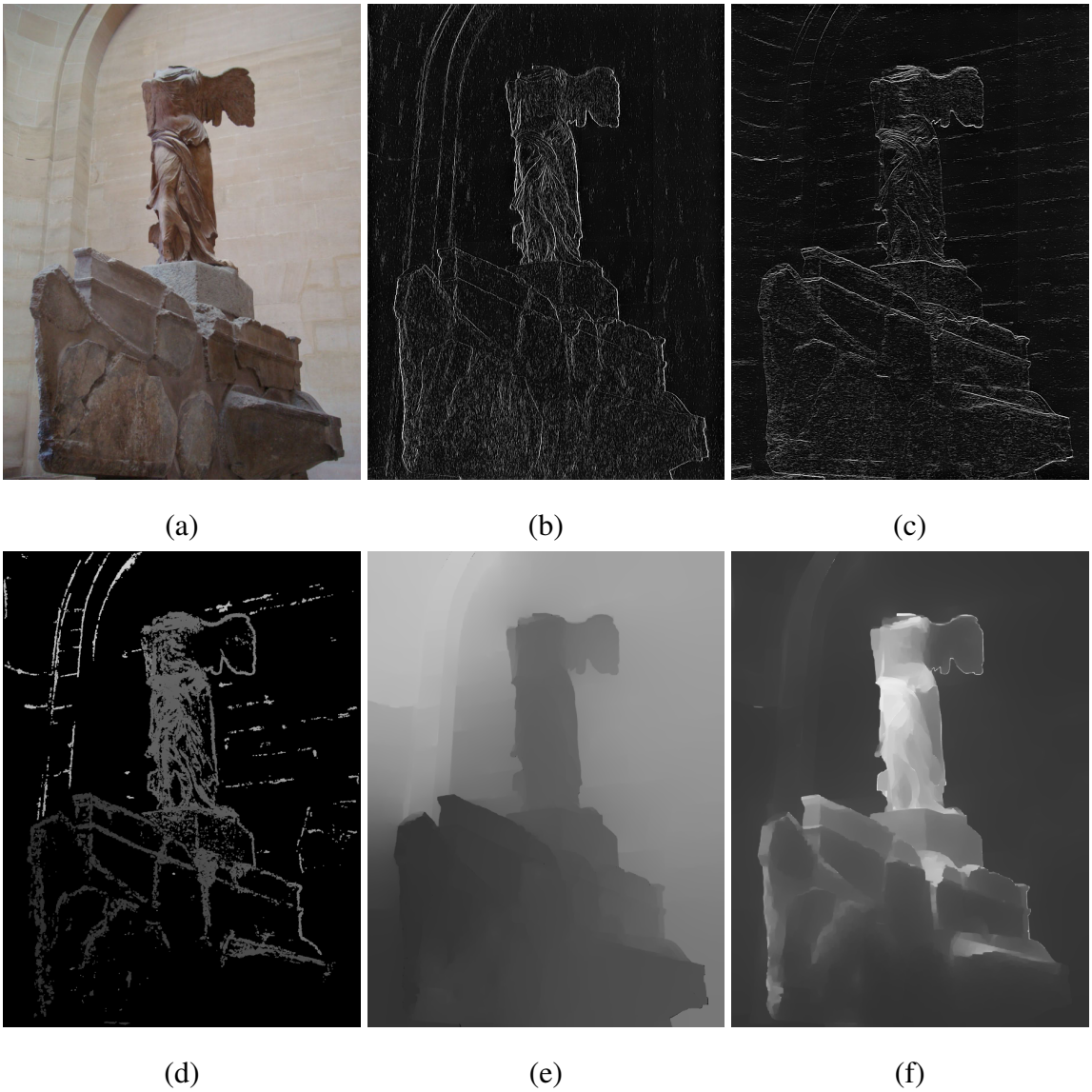


Figure 4.2: System overview. Our contributions are highlighted in the yellow rectangle. (a) A typical scene reconstruction system pipeline. (b) Our system pipeline.

while respecting the free space volume. To achieve this, we employ screened Poisson surface reconstruction [41] with a simple modification: the addition of a soft, free-space volume constraint. The resulting surface has smooth hole fills in under-sampled areas but does not "baloon" out into free space in the way standard Poisson reconstructions often do. In addition, the free space volume provides, in effect, a tighter boundary condition on the screened Poisson problem, which improves the overall quality of the result.

Our system uses occluding contour information to improve 3D geometry reconstruction from uncontrolled photo collections. It consists of two novel pieces, i.e., (i) computing occluding contour information and encoding it elegantly with dense depth maps, and (ii) a Poisson based approach that respects occluding contours for mesh reconstruction.



(a) (b) (c)

(d) (e) (f)

Figure 4.3: Densifying depth maps. (a) Input color image. (b) gPb contour response along horizonal direction. (c) gPb contour response along vertical direction. (d) Initial depth map based on visible PMVS points. (e) Our estimated dense depth map. (e) Our estimated confidence map.

Figure 4.4: Two single view depth point clouds. The depth interpolation does not create accurate background depth values, as there are not enough samples. But it does create a depth-discontinuity boundary that aligns well with images and recovers geometry where PMVS points are dense.

4.2 Algorithms

4.2.1 Densifying depth maps

PMVS recovers oriented points and provides a list of views used to reconstruct each point; we say a point is (conservatively) visible in these listed views. We can therefore construct an initial, conservative depth map for each view by projecting its visible PMVS points into a depth buffer for that view. As shown in Figure 4.3(d), these depth maps can be fairly sparse.

To estimate a dense depth map for a given view, we define an energy function that encourages depths to be (1) close to the PMVS points where available and (2) smooth between PMVS points in a spatially adaptive way, i.e., respecting image contours and color discontinuities. Let (x, y) be a pixel location in an image, $d_{x,y}$ be the unknown depth values, and Ω be a set of pixels with depth values $\hat{d}_{x,y}$ derived from projected PMVS points. Our energy definition is then:

$$E_d = \sum_{x,y \in \Omega} (d_{x,y} - \hat{d}_{x,y})^2 + \lambda \sum_{x,y} w_x(x,y) \left(\frac{\partial^2 d_{x,y}}{\partial x^2} \right)^2 + w_y(x,y) \left(\frac{\partial^2 d_{x,y}}{\partial y^2} \right)^2 \quad (4.1)$$

where λ is a global smoothness weight and $w_x(x, y)$ and $w_y(x, y)$ are spatially varying smoothness

weight functions along horizontal and vertical directions, respectively (described below). We set λ to a relatively large value (50) in our experiments to boost the regularization due to the fairly noisy PMVS points that arise when reconstructing from internet photo collections.

Note that we choose to use second order derivatives in order to encourage low curvature reconstructions. We approximate these derivatives as

$$\begin{aligned} \frac{\partial^2 d_{x,y}}{\partial x^2} &= 2d_{x,y} - d_{x-1,y} - d_{x+1,y}, \\ \frac{\partial^2 d_{x,y}}{\partial y^2} &= 2d_{x,y} - d_{x,y-1} - d_{x,y+1}. \end{aligned}$$

We construct the smoothness weighting functions $w_x(x, y)$ and $w_y(x, y)$ to have values close to 1 in visually smooth regions, to encourage depth propagation, and close to 0 on contours and color boundaries, to stop depth propagation and encourage depth discontinuities at visual boundaries.

To quantify visual smoothness and proximity to boundaries, we employ two measures. Recent work in computing image contours has shown significant progress. We leverage the work of gPb [9], which computes oriented contour strength at each pixel, measured in 8 directions. For our purposes, we use just the horizontal and vertical contour strengths, $g_x(x, y)$ and $g_y(x, y)$, which we show for one example in Figure 4.3(b) and (c). The second measure we use is simply the second derivative of image intensity in the x and y directions. Putting these together, we define the smoothness weighting functions to be:

$$\begin{aligned} w_x(x,y) &= exp\left(-\left\|\frac{\partial^2 I(x,y)\|}{\sigma_1 \partial x^2}\right\|\right) exp\left(-\frac{g_x(x,y)}{\sigma_2}\right), \\ w_y(x,y) &= exp\left(-\left\|\frac{\partial^2 I(x,y)}{\sigma_1 \partial y^2}\right\|\right) exp\left(-\frac{g_y(x,y)}{\sigma_2}\right), \end{aligned}$$

where $\sigma_1 = \sigma_2 = 0.1$ in our implementation.

Minimizing Eq. 4.1 is a linear least squares problem for which the global optimum is readily computed. Figure 4.3(e) illustrates an example of dense depth map estimation.

In later steps, having a confidence measure for each depth estimate is important. Depths near projected PMVS points should have relatively high confidence, whereas depths far from these points should have low confidence. Applying nearly the same framework we used for depth esti-

mation (Eq. 4.1), we estimate per-pixel confidence $c(x, y)$ by minimizing an objective:

$$E_c = \sum_{x,y}(c_{x,y} - \hat{c}_{x,y})^2 + \lambda \sum_{x,y} w_x(x, y) \left(\frac{\partial c_{x,y}}{\partial x}\right)^2 + w_y(x, y) \left(\frac{\partial c_{x,y}}{\partial y}\right)^2 \quad (4.2)$$

In this case, we define the data $\hat{c}(x, y)$ across all pixels: $\hat{c}(x, y) = 1$ at projected PMVS points, otherwise $\hat{c}(x, y) = 0$. Here, the smoothness weights allow confidences to "diffuse" without crossing color contours; thus, for example, a high confidence foreground does not raise the confidence of a low confidence background. Note that the smoothness term now uses first order instead of second order derivatives, since $c(x, y)$ does not have a geometric meaning that requires second order smoothness. We approximate these derivatives as:

$$\frac{\partial c_{x,y}}{\partial x} = c_{x+1,y} - c_{x,y}, \qquad \frac{\partial c_{x,y}}{\partial y} = c_{x,y+1} - c_{x,y}.$$

Again, we set $\lambda = 50$.

Figure 4.4 shows two depth maps visualized as 3D point sets. Note how these point sets capture the occluding contours of the foreground against the background as depth discontinuties. The point locations are accurate at regions with good PMVS point coverage. They are less accurate at areas with very sparse coverage, for example, the wall in the back. We use the confidence map (Figure 4.3(f)) to measure this accuracy and later to limit the effect of bad depth values on our final reconstructions.

4.2.2 Augmenting the PMVS point cloud

The reconstructed depth maps provide useful visibility information, as well as new points that can potentially fill in geometry in less textured regions that are not well-covered by PMVS. In this section, we describe a method for enhancing the PMVS point set using the dense depth maps.

First, we consider each depth map point with confidence greater than 0.2 to be a candidate for inclusion in the augmented point set $\mathcal{P}$, which is initially an empty set. Denote the location of this point in world coordinates as q. We will add q to $\mathcal{P}$ if q (1) is not in significant visibility conflict with the all the depth maps and (2) is near other depth map points and thus likely on the surface.

Let $\pi_j : R^3 \to R^2$ be the function that projects a 3D point into viewpoint j and let $\pi_j^d : R^3 \to R$ be a function that computes the projected depth of that point. $d_j(x, y)$ represents the depth stored

(a) (b)

Figure 4.5: Augmenting the PMVS point clouds. (a) The direct output from PMVS. (b) Our augmented point cloud.

in the j-th depth map at location (x, y). We compute two confidence-weighted visibility votes for each point q:

$$K_f(q) = \sum_j c(\pi_j(q)) \cdot \delta(\pi_j^d(q) \leq l_b d_j(\pi_j(q)))$$
$$K_s(q) = \sum_j c(\pi_j(q)) \cdot \delta(l_b d_j(\pi_j(q)) \leq \pi_j^d(q) \leq u_b d_j(\pi_j(q))),$$

where

$$\delta(x) = \begin{cases} 1, & x = \text{true} \\ 0, & x = \text{false} \end{cases}.$$

l_b and u_b are tolerance bounds to determine if a point lies near the surface; we set $l_b = 0.99$ and $u_b = 1.01$. $K_f(q)$ then measures the degree to which q lies in the free space of all of the depth maps. $K_s(q)$ measures the amount of support for q being with range of some set of depth map points. We only add q to $\mathcal{P}$ if $K_f(q) < \gamma$ and $K_s(q) > \gamma$, where we set $\gamma = 6$ in our experiments.

We additionally add the original PMVS points to $\mathcal{P}$, except those points that are in significant visibility conflict with the depth maps, i.e., for which $K_f(q) < \gamma$.

Figure 4.5 illustrates the greater coverage provided by the augmented point set.

4.2.3 Computing free space volume

We additionally compute a free space volume that will later constrain the surface reconstruction. First, we form a finely sampled grid of voxels. Given a voxel u, we project its center (which, with some abuse of notation, we will also denote as u) into each view j to compute the accumulated free-space vote for the voxel as:

$$K'_f(u) = \sum_j c(\pi_j(u)) \cdot \delta(\pi_j^d(u) \leq l'_b d_j(\pi_j(u))),$$

where l'_b determines how close to the surface to carve. Here we are more conservative and let $l'_b = 0.97$, effectively assuming most high confidence values on the depth map are within 3% error.

To prevent carving through well-supported regions of space, we compute the number of augmented points in $\mathcal{P}$ that lie within a voxel – call this number $K_n(u)$ – and finally define the free space volume V_f as:

$$V_f = \{u : (K'_f(u) > \gamma') \cap (K'_f(u) > 10 \cdot K_n(u))\} \tag{4.3}$$

In our experiments, we conservatively set $\gamma' = 15$.

4.2.4 Occluding contours against the sky

Sky regions in images provide strong occluding contour cues for bounding foreground structures. Suppose we have fairly conservatively detected some (but not all) sky pixels in a given view. We incorporate this information into our framework by assigning very large depth values (10^8) to those pixels, and then proceed with densifying the depth map for that view as before. In later steps, depth map pixels with very large depths are not considered as candidates for inclusion in the augmented point set $\mathcal{P}$ – we are not actually reconstructing the sky geometry – but they are used for free space computations.

Our procedure for conservatively identifying sky pixels is as follows. First we reconstruct a Poisson surface just based on the original PMVS points. We then project this surface into each view. Pixels are initially labeled sky if they are not covered by this surface. Assuming sky pixels tend to be blue or gray, we then narrow this set to pixels with (r, g, b) colors that satisfy $r + g \leq 2b$. Finally, we conservatively erode this set in each view with a disk of size 11. Note that this procedure is only to initialize a conservative sky seed. The accurate sky mask is computed through depth densifying using interior contours.

4.2.5 Poisson reconstruction with free space

We modify the screened Poisson surface reconstruction alrogithm [41] to handle free-space volumes. In particular, we minimize the following objective:

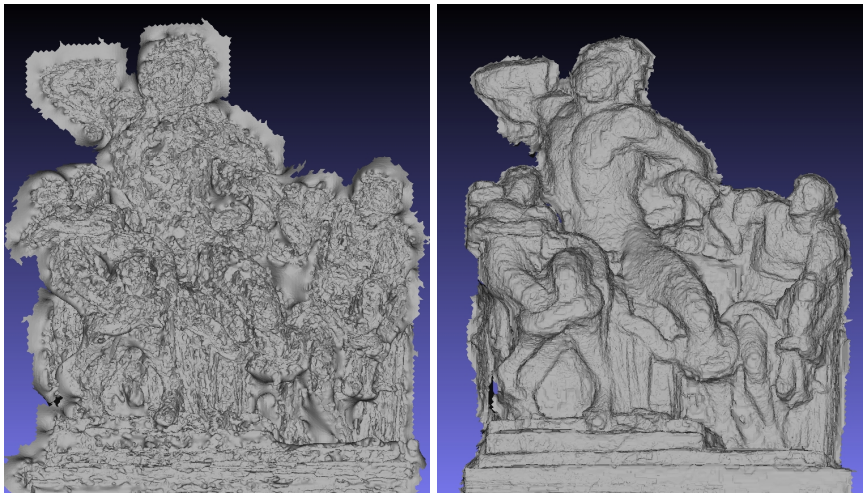
$$E(\chi) = \int \|\nabla\chi(u) - V(u)\|^2 du + \quad \alpha\frac{A}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} c(p)\chi^2(p) + \beta\int_{u\in V_f} \|\chi(u) - 1\|^2 du. \tag{4.4}$$

The first two terms give rise to the screened Poisson equation [41]. In the first term, u is a point in the volume, V is the vector field constructed from the point normals, and χ is an indicator function; $\chi(u) > 0$ means that u lies outside the surface, and $\chi(u) < 0$ means u is in the interior of the surface. In the second term, α trades off matching point normals and point locations over the point set $\mathcal{P}$ (of size $|\mathcal{P}|$), and A is an area term automatically computed in [41]. This second term

Figure 4.6: Reconstruction of Place Saint-Michel (583 images after SfM).

encourages values of χ to be zero near the input points. The third term is our contribution: a soft constraint that encourages the indicator function to take on values near 1 in free space. We build this modification into the released source code of [41], version 4.0. Finally, solving for the χ that minizes Eq. 4.4 results in an implicit function from which the surface can be extracted as the zero level set.

In our experiments, we set $\alpha \in [0.5, 2]$ and $\beta = 1$. We also supply normals for the vector field V. For PMVS points, we use the original PMVS normals. For depth map points, we compute the normals directly from the depth maps in the standard way.



PMVS + Poisson Proposed approach

Figure 4.7: Reconstruction of Laocoon and his Sons at Vatican Museums (303 images after SfM).

4.3 Experiments and Evaluations

We evaluate the proposed system on 6 datasets, consisting of images collected from the Internet. The computational bottleneck is in the pre-processing, namely, image contour calculation by gPb [9], SfM execution [79], and PMVS execution [27]. These pre-processing steps may take days for large datasets, in particular, nearly a week for our largest dataset *Colosseum*, which started from more than tens of thousands of images and ended up with an SfM model consisting of 3276 images. Conversely, the new steps we contribute – depth map densification, point set augmentation, free space construction, and constrained Poisson reconstruction – finish within an hour for all the datasets on a 40-core cluster, not including file I/O time.

To evaluate the effectiveness of our approach, we have made comparative evaluations. In Figures 4.1 and 4.6, our reconstructions are compared against two other state-of-the-art methods: 1) screened Poisson surface reconstruction [41] operating on the original PMVS point-cloud [27]; and 2) CMP-MVS [37], which improves reconstruction near silhouettes and compares favorably to other state-of-the-art techniques such as [77]. Also note that the screened Poisson surface reconstruction software "hallucinates" large pieces of geometry in an attempt to produce a water tight

model. Such geometry consist of large triangles; thus, we remove triangles whose edge length is longer than 10 times the average edge length of the entire mesh for Poisson reconstructions. The figures show that our results have much cleaner geometry boundaries showing the effectiveness of the enhanced depthmaps and their carving power along the occluding boundaries. Notice, in particular, the precise outline of the winged dragon in our 3D model at Place Saint-Michel (Figure 4.6) as well as sample depthmaps produced by our system.

The top two rows of Figure 4.8 further illustrate improvements over the standard PMVS+Poisson approach, with tighter bounds on the geometry. We can also see the effect of various components of the system in the bottom row. The bottom left sub-figure shows that augmenting the point set does a better job of completing one of the wings, but the reconstruction is still noisy and does not have a tight silhouette. Alternatively, as shown in the bottom-middle sub-figure, not adding these points, but performing the carving on the original PMVS points does tighten the silhouette, but leaves Poisson to just smoothly fill in missing pieces. We note that the reconstruction is much less noisy, due to the free space volume providing tigher constraints on the Poisson reconstruction. Finally, the combination of augmenting the point set and using the free space constraint gives the best of both worlds as shown in the bottom-right sub-figure.

Figure 4.7 provides another example where, in particular, PMVS creates a noisy point set and leads to poor surface reconstruction using the standard Poisson approach. Our method significantly improves the mesh quality.

Finally, we highlight the results of incorporating the occluding contours against the sky in depth map densification and free-space volume construction, as shown in Figure 4.9. The top row (San Marco Basilica) shows how PMVS and Poisson tend to inflate the boundary into sky regions. After carving out sky regions in the free space volume using the improved depth maps, we are much better able to resolve the outlines of statues and even the fine cross structure atop the dome. In the middle and bottom rows (Colloseum), we can see how sky carving has both removed spurious geometry along the top of the Colloseum and also enabled carving through some of the portals around the structure.

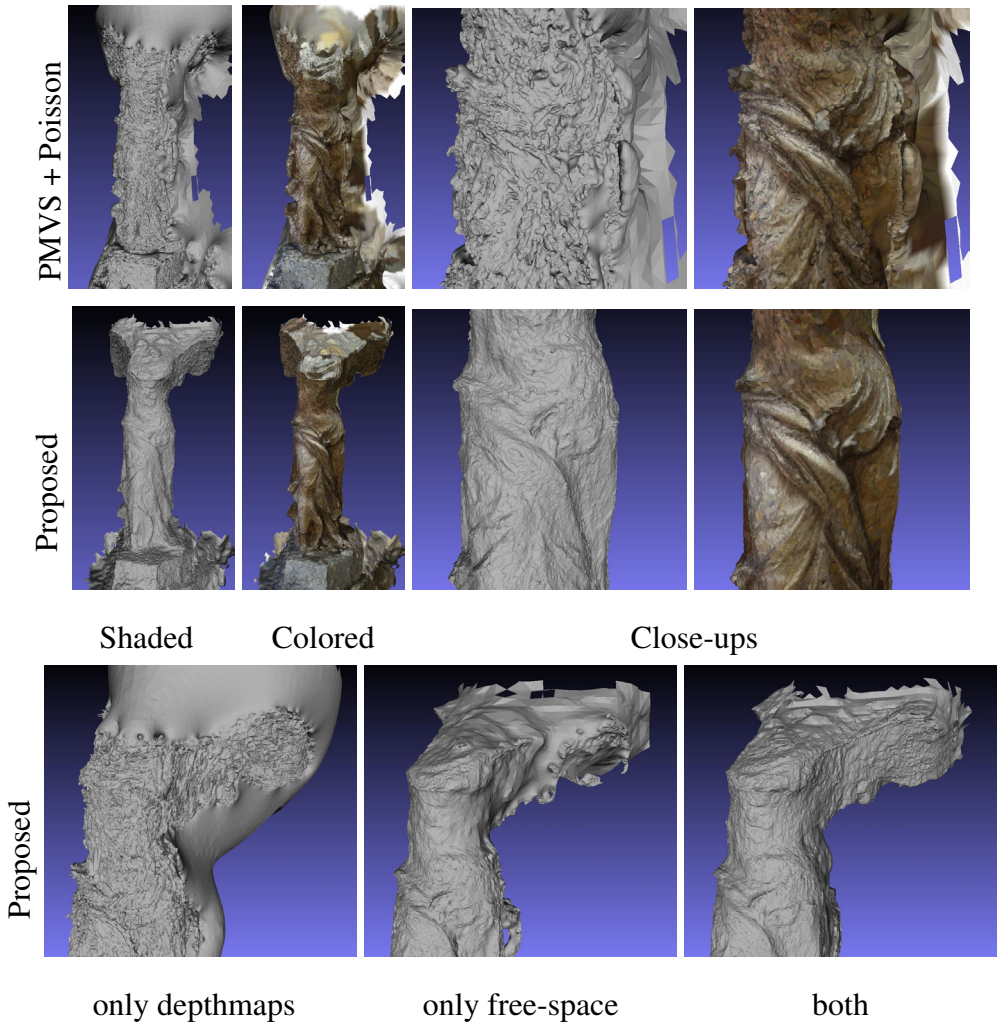
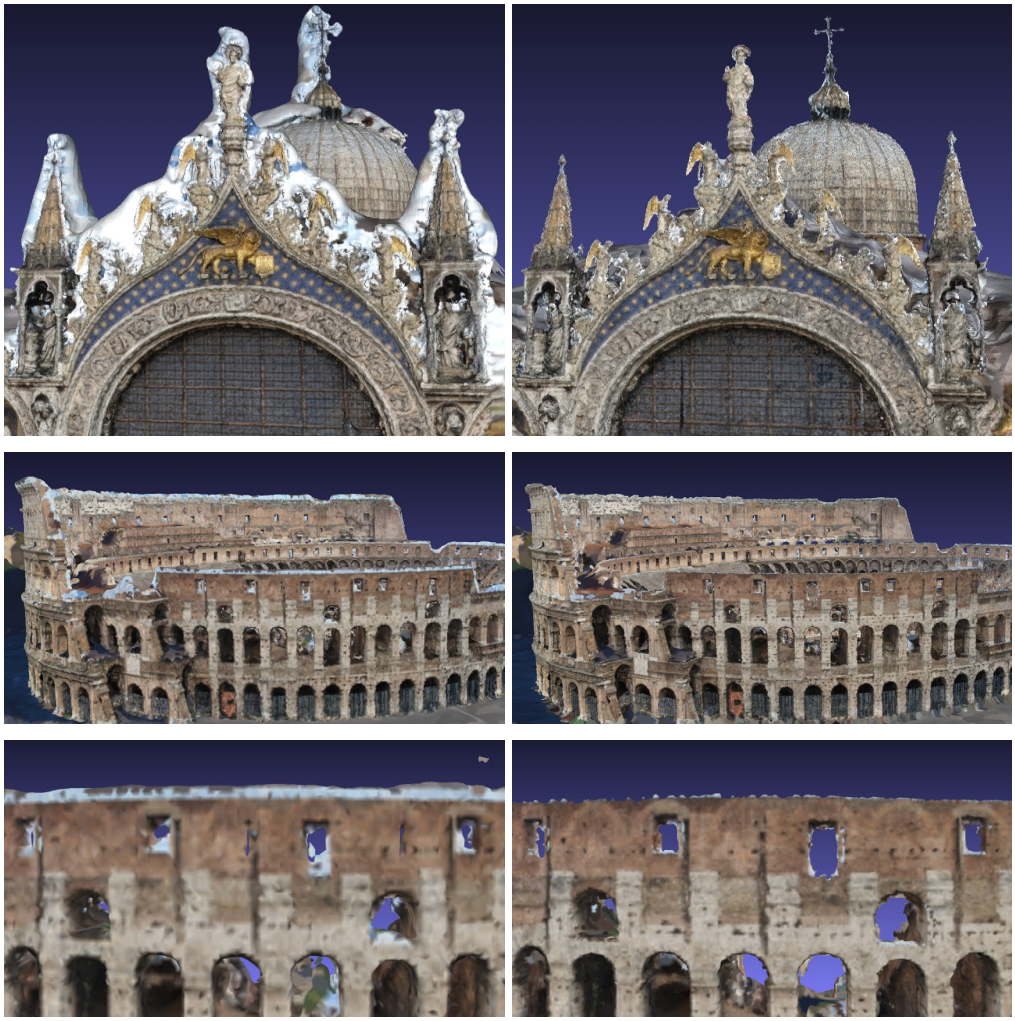


Figure 4.8: Reconstruction of Winged Victory of Samothrace at Louvre (274 images after SfM).



[63] Proposed approach

Figure 4.9: Our reconstructions for San Marco Square (top) and Colloseum (middle and bottom) datasets showing the improved geometry boundaries over [63]. The numbers of images after SfM are 2687 and 3276, respectively.

4.4 Summary

This chapter presents two contributions to the problem of MVS from photo collections: 1) a new technique to identify free-space regions arising from occluding contours, and 2) a new approach for incorporating the resulting free-space constraints into surface reconstruction. We propose a new dense depth map interpolation from 3D point clouds and occlusion boundaries, and a Poisson formulation that incorporates free space constraints. The free space constraints effectively modify the space of solutions with a tigther bound around the initial point cloud. This new formulation retains the regularization/hole filling properties of Poisson [41] while improving its accuracy with noisy data. It outperforms state-of-the art MVS techniques on Internet photo collections, and results show dramatic quality improvements.

The method we have described does have limitations. First, it depends on time-consuming pre-processing steps, though these can be difficult to avoid (e.g., SfM and PMVS). Faster implementations of gPb are now available, which should accelerate this part of the process. Second, errors in depth map interpolation have the potential to do too much carving. Third, occluding contours can give better bounds on parts of a scene, but screened Poisson surface reconstruction is still left to fill in parts that were not observed, e.g., the backs of objects. In some cases, the filled-in regions behave unexpectedly, such as the wings of Winged Victory, which are merged across the top, rather than being filled around the back. Finally, 3D volumetric contouring artifacts are visible in some reconstructions, likely due to the influence of the binary free space volume on the Poisson reconstruction. We believe the results could further be improved with a soft voting scheme or perhaps signed distances from depth maps.

Chapter 5

PHOTO UNCROP

As we have seen in Chapter 2, creating photo-realistic visualization by directly rendering from a reconstructed 3D model is a grand challenge, and may take decades of research effort. But is it true that "directly rendering" is the only solution of visualizing a 3D scene? I believe the answer is no. Innovating image based rendering to work with inaccurate geometry (created from Internet photo collections) is a promising area of research to create "photo-realistic" visualizations, and thus is able to enable a wide variety of applications. In this chapter, I present an application we called "photo uncrop". The advantage of our approach is twofold. First, it tolerates geometry errors from MVS approaches. Second, it renders salient scene elements (transient objects, hard-to-model geometry) and thus dramatically improves the photo-realism of the results.

Travel photos often fail to create the experience of re-visiting the scene, as most consumer cameras have limited field of view (FOV). Indeed, mobile phone cameras (which far outnumber any other photography device) typically have a FOV around 50-65 degrees, significantly narrower than the human eye [8]. Capturing large scenes is therefore tricky. Modern cell phones are equipped with camera apps providing a panorama mode, which allows you to take multiple pictures and stitch them into a bigger image. However, the process is often tedious. Furthermore, you cannot operate on your past photos. As a result, your photos are often more tightly *cropped* than desired (See Fig. 5.1).

We address the problem of extending the FOV of a photo—an operation we call *uncrop*. The goal is to produce a larger FOV image of the scene captured in your photo, leveraging other photos of the same scene from the Internet (captured at different times by other people). We make an important distinction between producing a *plausible* extended image using a technique such as texture synthesis [57], vs. producing an extended rendering of the *true scene* which is intended

Figure 5.1: Capturing family photos with the desired background in the image frame can be tricky. Our approach expands the FOV of a user photo thus enables better spatial context. Landmark: Stravinsky Fountain in Paris.

to be accurate. The latter case is more challenging and potentially more useful, as it gives you information about the real world, allowing you to *zoom out* of any photo to get better spatial context.

For almost any photo you take at a tourist site, there exist many other photos from nearby viewpoints, collectively capturing the scene across a potentially large FOV. Our approach is to automatically select, reproject, and composite a subset of this imagery into a large image screen centered on your photo. This problem is challenging for several reasons. First, the photos are not captured from the same optical center, resulting in too much parallax for existing state-of-the-art panorama stitchers (which produce severe artifacts as we will show). Second, the appearance (color, exposure, and illumination) varies dramatically between photos, making it difficult to produce a coherent composite. And finally, the presence of people, cars, trees, windows, and other transitory or hard-to-match objects make the alignment problem especially challenging.

This problem represents a compelling application that sits between traditional panorama stitching, which requires capturing many images and is thus labor intensive, and full 3D scene reconstruction, which has too many failure modes. Indeed, our experiments with state-of-the-art 3D reconstruction techniques [26, 37, 63] rarely produce hole-free geometry, omitting ground, people, trees, windows, and many other salient scene aspects. Our approach therefore assumes *incomplete* geometry in the form of depth maps, and leverages a novel Markov Random Field (MRF) based

compositing technique to generate compelling full-scene composites complete with people, trees, etc.

Our contributions are two-fold: (i) the first system to produce compelling uncroping results with Internet photos; (ii) a novel MRF based formulation adapted to handle significant geometry errors.

We show convincing results on a wide range of scenes. Like existing panorama stitchers, our results are not entirely free of artifacts, and stitching seams and misregistration artifacts are occasionally noticable. However, we argue that for the intended application (giving you spatial context for your photo), small artifacts are quite tolerable. I.e., it's less important that every pixel is right than being able to zoom out and see that the building behind you is the Uffizi, or that you're standing in the middle of a large town square.

The remainder of the chapter is organized as follows. I first define the input data of our system in Section 5.1. The detailed uncrop algorithm and implementation details are presented in Section 5.2 and 5.3, respectively. I show our experimental results in Section 3.5 and conclude in Section 5.5.

The work in this chapter was published in the 2014 European Conference on Computer Vision [65].

5.1 Input Data

We download images from Flickr (`http://www.flickr.com`) for a variety of sites, and use existing structure from motion (SfM) software [79] to compute camera poses. Uncropping is performed on images selected from the SfM model to show the capability of our system, though it would be straightforward to apply our system to an arbitrary new photograph by simply adding it to the relevant image set and performing incremental SfM. Publicly available multi-view stereo software is used to reconstruct per-view depthmaps [25]. Then, we warp each image by reprojecting its depth map and colors to the viewpoint of the image to be uncropped. More details on these preprocessing steps are found in Section 5.3.

5.2 Uncrop Algorithm

We propose an MRF-based compositing algorithm to construct a wide FOV target image around a reference image. We assign a label l to each source image, such that $l \in \{-1, 0, 1, \cdots, N-1\}$, where N is the number of images that survived the view selection process (including the reference image itself), and -1 is the null label. After re-projecting each source image, we have a set of partial, warped images $C_l(p)$ that each cover parts of the target image. We seek to solve for the label map $l(p)$ over target pixels p that will yield a high quality composite when copying warped image colors to the target image. We include the null label $l = -1$ to allow for a small number of pixels not covered by any of the images. After computing the composite, we perform a Poisson blend to give the final result.

We formulate the MRF problem as the sum of a unary term, a binary term, and a label cost term:

$$E(l) = \sum_{p} E_{\text{unary}}(p, l(p)) + \sum_{\{p,q\} \in \mathcal{N}(p,q)} E_{\text{binary}}(p, l(p), q, l(q)) + E_{\text{label}}(l). \tag{5.1}$$

where $\mathcal{N}(p, q)$ denotes pairs of neighboring pixels in a standard 4-connected neighborhood. With abuse of notation, l here denotes the set of all the labels in the image. What is novel is the actual formulation of the unary and binary terms. We first describe their principles, where detailed formulation will be discussed in the following sections.

5.2.1 Principles

$E_{\textbf{unary}}$**:** It is nearly impossible to reconstruct perfect geometry for a complicated scene like ours, and a warped image may not be exactly aligned with the reference image. Therefore, the unary term incorporates the confidence of estimated depth information. Appearance mismatch is another source of artifacts. For example, compositing a daytime photo with a nighttime shot is challenging. We assign each image a score that measures the appearance similarity to the reference. Furthermore, appearance variation within an image due to shadows, over-saturation, and flash photography can result in spatially varying pixel quality. Thus, we assign lower cost to high contrast pixels.

$E_{\textbf{binary}}$**:** Traditional image stitching uses E_{binary} to minimize seams by looking for cuts on image edges. We follow a similar path, but also introduce a new measure to encourage any given reconstructed patch in the composite to resemble at least one warped source image at the same location. This helps to avoid making abrupt transitions in the composite that can arise from geometric misalignments, because noticeable artifacts at such transitions do not resemble corresponding regions in any of the input images.

$E_{\textbf{label}}$**:** Building a composite out of many images can lead to a quiltwork of stitched patches that can stray from the desired result. It is natural instead to encourage the stitcher to take pixel examples from a sparse set of warped views. In our approach, we achieve this by assigning a constant cost to each unique label used in the compositing.

5.2.2 Unary term

We construct the unary term from several components:

$$E_{\text{unary}}(p,l) = E_{\text{geometry}}(p,l) + \alpha_1 E_{\text{appearance}}(l) + \alpha_2 E_{\text{contrast}}(p,l) + \alpha_3 E_{\text{reference}}(p,l), \tag{5.2}$$

where $\alpha_1 = 10$, $\alpha_2 = 5$, $\alpha_3 = 1$ are used in all of our experiments. Note that each warped source image $C_l(p)$ only partially covers the target image; if warped image l does not have a color at pixel p, the unary term is automatically set to infinity.

Geometry: We define the geometry term $E_{\text{geometry}}(p,l)$ as the possible error in the position of a reprojected pixel. It is determined by two factors: the accuracy of the original depth value and the baseline between the reference view and the source view. First, we model the accuracy using the range of depths in a local neighborhood in the source image l. More concretely, let u denote a source pixel in image l, and U to be the corresponding 3D point on the depthmap, which is reprojected to p in the reference. We look at a local neighborhood of size 11×11 pixels centered at u, and compute the minimum and the maximum depth values in the window. We have assumed a 1% depth error, and subtracts from the minimum and add to the maximum depth values by $0.01D_u$, where D_u is the depth value at u. We take the 3D point U and shift its location to the minimum and

the maximum depth locations, and project it to the reference image. Let us call the two projected location $p_{\text{near}}(p, l)$ and $p_{\text{far}}(p, l)$, respectively. Then, the geometry term is defined as follows:

$$E_{\text{geometry}}(p, l) = \max(|p - p_{\text{near}}(p, l)|, |p - p_{\text{far}}(p, l)|). \tag{5.3}$$

By minimizing this term, the optimization will favor pixels from images that have a smaller baseline relative to the reference view (less room for parallax errors) and images that sample surface regions more densely in close-ups and thus are more likely to cover a smaller range of depths. It is possible that multiple pixels u may warp to pixel p (see Section 5.3), in which case, we simply take the average projected location.

Appearance: Internet photos exhibit a wide range of illumination conditions. It is important to encourage the use of images with similar appearance. To do this, we assign an appearance cost to each source image. Specifically, we take the color histogram of each image, and score it by its KL divergence from the histogram of the reference image. Then the images are sorted in ascending order. Let k_l be the index of image l in this sorted list. We now define the overall image appearance cost as:

$$E_{\text{appearance}}(l) = k_l/N, \tag{5.4}$$

where N is the number of images in the set. Smaller cost in this case means less divergent from (more similar to) the reference image. Note that this unary term is constant for image l, regardless of which target pixel is being considered.

Contrast: Undesirable appearance variations such as shadows and over-saturation can be penalized based on the contrast. We address this by defining a local contrast cost. Let (G_x^l, G_y^l) be the finite difference gradient of image l after mapping image l to grayscale (intensity values $\in [0, 1]$). We use the following formula to measure the lack of contrast over 11×11 window Ω centered at u in image l, which corresponds to p after the warping:

$$E_{\text{contrast}}(p, l) = \frac{1}{|\Omega|} \sum_{v \in \Omega} \sqrt{(1 - |G_x^l(v)|)^2 + (1 - |G_y^l(v)|)^2}. \tag{5.5}$$

If multiple pixels from source image l map to p after warping, we again simply take the average of their scores.

Reference: Finally, it is important to respect the reference image. Let us define the core region of the image Ω_{core} to be a set of pixels inside the reference image and more than 11 pixels in distance from its boundary. The reference cost is defined by applying the following four rules from top to bottom:

$$E_{\text{reference}}(p,l) = \begin{cases} 0, & l = l_{\text{ref}} \\ 10000, & l = -1 \\ 100, & p \notin \Omega_{\text{core}} \\ \infty, & p \in \Omega_{\text{core}} \end{cases} \tag{5.6}$$

where l_{ref} is the label of the reference image. It is possible that some of the pixels in the target image are not covered by any of the images, thus we allow the $l = -1$ label, with high cost.

5.2.3 Binary term

Similar to previous work [6], we encourage label switches in regions with edges, where seams will be less noticeable. Further, we use a novel compatibility term to encourage constructing regions in the target image that resemble warped source image regions. Our binary term can then be written:

$$E_{\text{binary}} = E_{\text{edge}} + \beta E_{\text{compatibility}}. \tag{5.7}$$

where β trades off the relative contribution of the compatibility term. (We set $\beta = 10$ in all of our experiments.)

Edge: We first define a Sobel filter cost for a single pixel u and in (unwarped) source image l:

$$E_S(u,l) = \left(6 - \frac{||S(u,l)||_1}{4}\right)^2. \tag{5.8}$$

$S(u,l)$ is the concatenation of the Sobel filter responses in the x and y directions for each of the r, g, and b color channels, where we take the L_1 norm of this 6-dimensional vector. Now, for neighboring target pixels p and q with labels l and m, respectively, the binary edge cost is:

$$E_{\text{edge}}(p,l,q,m) = \begin{cases} 0, & l = m \\ E_S(u,l) + E_S(u,m), & l \neq m. \end{cases} \tag{5.9}$$

If multiple pixels correspond to p after warping, we take their average over u.

Compatibility: To encourage regions in the target image to resemble regions in the source image, we introduce a novel label compatibility term. Consider a pixel p and one of its neighbors q in the target image, and an image l. We define an 11×11 window around the two pixels and collect the pixels of $C_l(p)$ (corresponding to the warped version of image l) in the overlap into a vector $W_{p,q}(l)$. If there will be a transition between labels l and m in going from p to q, respectively, then the resulting window in the final result will likely resemble the average of the windows $W_{p,q}(l)$ and $W_{p,q}(m)$. This average in turn should resemble at least one of the (warped) source images. Thus, we define the following compatibility cost:

$$E_{\text{compatibility}}(p, l, q, m) = 1 - \max_n \text{NCC}\left[\frac{1}{2}\left(W_{p,q}(l) + W_{p,q}(m)\right), W_{p,q}(n)\right] \tag{5.10}$$

where $NCC[\cdot, \cdot] \in [-1, 1]$ is the normalized cross-correlation between two vectors, and n ranges over all of the labels. Note that, by this definition, this term becomes 0 when $l = m$. In addition, we set the term to ∞ if either $W_{p,q}(l)$ or $W_{p,q}(m)$ includes pixels where $C_l(p)$ or $C_m(p)$ are undefined.

5.2.4 Label cost

We encourage the image stitcher to take color from a small number of images by assigning a constant cost for each additional label. If K is the number of unique labels in the composite, we set $E_{\text{label}}(l) = 500000 \cdot K$.

5.2.5 Optimizations and Accelerations

The energy definition in Eq. (5.1) falls naturally in the category of multi-label optimization with label cost. We optimize it with an iterative alpha-expansion solver [18].

Directly solving the problem is impractical due to the image resolution (millions of pixels) and the large label space (thousands of labels). Therefore, we apply (i) a simple up-sampling scheme and (2) a pre-filtering process to limit the solution space. The computational time varies from 10 seconds to a few minutes for solving the graph cut problem with a single thread on a 3.4Hz CPU.

Up-sampling a lower resolution label map: The iterative alpha-expansion solver is performed on a target image that is $1/8$ the resolution (in each dimension) of the desired result. After optimization, the label values are upsampled as follows. Each pixel in the original high resolution target image has four possible label candidates at the 4 nearest pixels in the low-resolution label image. We simply pick the label with the lowest appearance penalty (Eq. 5.4).

Pre-filtering: First, we reduce the label set by discarding input images that are far from the COP of the reference view or cover only a small portion of the target image (see Section 5.3 for more details of this process). Next, we observe that the optimization process tends to reject pixels that (i) have large geometry cost, (ii) have poor patch compatibilities, or (iii) are too dark or over-saturated (essentially, pixels in solid black or white regions). Removing some obviously low quality pixels before performing the optimization limits the solution space and can thus greatly improve the computational efficiency. Specifically, we remove a label l at pixel p from the solution space, that is, assigning infinity cost, when (i) $E_{\text{geometry}}(p, l) > 20$, $E_{compatibility}(p, l, p, l) > 0.6$, or $E_{\text{contrast}}(p, l) > \sqrt{2} - 0.01$.

5.2.6 Poisson image blending

The final, blended composite is computed from the MRF composite by solving a Poisson equation (Fig 5.2). We first compute the x, y gradient from the MRF composite, and set the values to be 0 at places where the label changes or where the label is -1. The blended composite should keep the color from the reference images; thus, we set a large weight (1000) to penalize differences from the reference image colors at the locations where reference pixels are available.

5.3 Implementation Details

Depth map reconstruction: We use publicly available multi-view stereo software [25] to reconstruct per-view depth maps, then apply cross bilateral filtering [33] for smoothing, as noise and high frequency geometric details often cause artifacts during image warping. The local window radius is 50 and the regularization parameter is 0.16 (suggested by the code of [33]). Note that

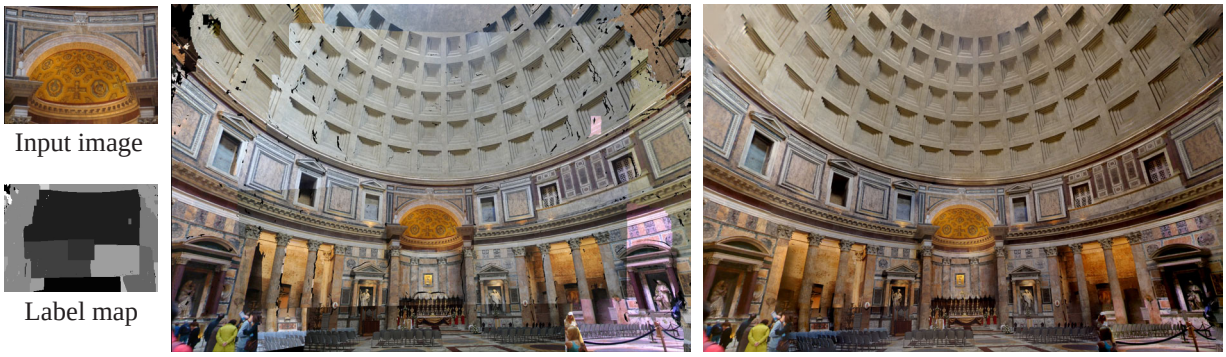


Figure 5.2: Landmark: Pantheon in Rome. Typically 10-20 unique labels are present in the label map after the graph-cut optimization. It is used to create an MRF composite.

we use the corresponding color image as the reference for the bilateral filtering. This process also helps in filling in missing depth values, where kernel weights are simply set to 0 for holes in an initial depth map. Finally, we compute a normal per pixel based on the depths.

Image Selection and Warping: Given a reference photograph and the SfM reconstruction, we first remove each source image with an optical center that is more than a distance τ_{COP} from the reference; we set $\tau_{\text{COP}} = 50$[1] in our experiments.

Next, we forward-warp the remaining source images into the target image using splatting and a soft Z-buffer algorithm. We project each source image pixel into the target view, eliminating source pixels that are backfacing to the target view. In general, re-projected source pixels land between target pixels; furthermore, due to occlusions, foreshortening, and differences in image resolution, it is possible for multiple source pixels to land between the same set of target pixels. We associate each source pixel with the four nearest target pixels, storing at each target pixel p a sample $\{u, l, C, w, d\}$ comprised of the position u, image identifier l, color C, bilinear weight w, and re-projected depth d of the source pixel. We project all source images in this manner, storing a list of samples at each pixel. We then eliminate all samples that are behind the reference viewer

[1]The length of 1 unit in our 3D models is the distance between the first pair of images selected by VisualSfM. The pair is selected to have a large number of features in common while having a sufficiently large triangulation angle (greater than 4 degrees between their optical axes).

($d < 0$) or occluded by other samples based on a soft Z-buffer; i.e., for each target pixel p, we find the closest positive depth d_{closest} and consider a given sample with depth d at p to be occluded if $d > d_{\text{closest}} + \tau_{\text{depth}}$. (We set $\tau_{\text{depth}} = 20$ in our experiments.) For each target pixel p, we then collect all the samples from the same image l, compute a weighted average color $C_l(p)$ and a source pixel list $U_l(p)$, which will be used in computing label costs in the MRF formulation. Note that $C_l(p)$ only covers part of the target image and is "invalid" elsewhere; further, it is possible for source samples to land apart from each other due to grazing angle surfaces or if the source image is low resolution, leaving gaps between the projected samples.

Finally, we perform one last image selection step: for each image l, if the valid portion of $C_l(p)$ which lies outside of the reference image region covers less than 5% of the target image, then image l is eliminated from further consideration. This step tends to remove images that are: not looking in the direction of the scene of interest, are much lower resolution than the target image, or are close-ups of only a small portion of the scene of interest.

5.4 Results and Evaluations

We evaluated our system on 10 datasets from the city of Rome and Paris. The number of images in each dataset (i.e., SfM model) ranges from 262 (Stravinsky Fountain) to 2397 (Piazza Navona), where the largest two datasets contain more than 2000 images. For each example, we generated results for several target image sizes and kept the largest image that looked plausible after manually cropping to discard image boundaries with significant artifacts (Fig. 5.3). Automatically selecting the target image sizes and cropping is an area for future work.

5.4.1 Ground truth experiment

Figure 5.4 illustrates an experiment which allows us to compare our result against the ground truth. We take a relatively wide FOV image (one from San Peter Cathedral dataset), crop to $1/9$ of the image in the center, then run our system to uncrop. Note that the ground truth image is not used for stitching. Despite minor intensity differences, our result faithfully reconstructs the original

(a) (b)

Figure 5.3: Manual cropping to discard image boundary with significant artifacts. (a) The immediate output from the Poisson color blending step. (b) We manually crop the image borders. The boundary artifacts are caused by Poisson blending at regions without source image coverage.

image using other photographs. In fact, our result has better contrast and reveals more details, in particular, in the bottom half of the image. To take this one step further, we can expand the FOV even more than the original image and generate a convincing composite with much wider field of view than the input.

5.4.2 Evaluation of the geometry and compatibility terms

Here we evaluate the effectiveness of two novel components of our MRF formulation: E_{geometry} and $E_{\text{compatibility}}$. The E_{geometry} term prefers source pixels from smaller baseline views with more accurate depth estimates. These views typically produce fewer distortions. Fig. 5.5(a) shows the MRF composite and its Poisson blend when E_{geometry} is set to 0. The optimizer picks a patch with large geometry distortion, causing misalignment artifacts. On the other hand, $E_{\text{compatibility}}$ is designed to discourage switching labels to a misaligned image. We show the result of setting

(a) (b) (c) (d)

Figure 5.4: Ground truth experiment (San Peter Cathedral). (a) The ground truth image. (b) We only keep $1/9$ of the image in the center, which is the input to our system. (c) Uncropped to the ground truth image size. The ground truth image in (a) was not used in creating this composite. (d) Uncropped to even wider FOV than the original.

$E_{\text{compatibility}} = 0$ in Fig. 5.5(b). Severe misalignment is visible at the boundaries between image patches in the MRF composite. By incorporating both terms (Fig. 5.5(c)), the optimizer creates a better composite with fewer visible artifacts.

Comparitive evaluation against baseline methods

To the best of our knowledge, there does not exist a system that can address the same uncropping problem with community photo-collections. The closest ones are the Photoshop CS6 PhotoMerge tool [3] and Scene Collage [55] (with executables released). Here we treat these two as baseline methods. Neither of them is capable of handling the large amount of images in our datasets (processes crash with our 64-bit Windows machine with 48 GB memory).

A common problem of the baseline methods is the inability to handle non-planar geometry and reason about visibility, as shown in Fig. 5.6. Both PhotoMerge and Scene Collage copy pixels from a bridge that is *behind* the camera. The baseline methods usually prefer wider FOV source images, thus tend to use images containing occluders, the bridge and the bus in this case, in the

Figure 5.5: Evaluating the effectiveness of E_{geometry} and $E_{\text{binary compatibility}}$ (San Peter Cathedral). We show close-up views of the image mosaic and the Poisson blended results for better visualization. (a) E_{geometry} is turned off. (b) $E_{\text{binary compatibility}}$ is turned off. (c) Both terms are turned on.

composite.

The presence of large parallax is also a challenge for the baseline methods. Most 2D image transformations used for image stitching, such as a planar homography, are not sufficient to correctly warp images, unless the underlying geometry is near planar. This problem is well illustrated at the top portion of Institut de France in Fig. 5.6. Results in Fig. 5.8 show similar misalignment artifacts with the baseline methods, where our composites are significantly better.

More experimental results are provided in Fig. 5.8, which clearly illustrates that the uncropped images with extended FOV provides better spatial context of the scenes.

5.5 Summary

This chapter presents the work on utilizing Internet imagery to extend the field of view of a user photo. We employ multi-view stereo to warp images into a target, wide FOV image and propose

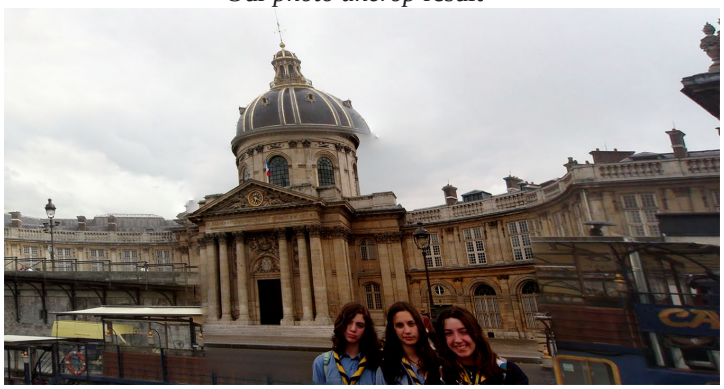
a novel MRF-based formulation designed to handle inevitable geometric inaccuracies. It creates results with image content that resembles the *real scene*. The evaluations on a wide range of real world datasets demonstrate the effectiveness of our approach. The results, while not perfect, are convincing and provide real spatial and visual context not available in the original user photo.

Our approach does have limitations. First, it only works for photos taken at sites where a sufficient number of Internet photos are available (e.g., tourist sites). It requires a certain density of source photos, and would fail to reconstruct regions where there is no coverage. The ground is often a problem area, as people seldom photograph the ground (examples in Fig. 5.8). Although our approach takes geometry error into account and minimizes potential image misalignment risk, severe geometry noise would still cause problems 5.9. As with most panorama stitchers, transient objects in the source images – e.g., people and cars – can be problematic, and seams through them may occur (Fig. 5.10). Recognition and segmentation algorithms could help address this problem. If the user photo itself contains transient objects that are not entirely in frame, then they will remain clipped in the final composite if the new field of view extends beyond them; automatically and realistically extending such objects (people, cars, etc.) out of frame would be interesting if quite challenging.

User input

A subset of Internet photos
from the same scene

Our *photo uncrop* result



Photoshop CS6 PhotoMerge with manual color blending

[Nomura et al. 2007]

Figure 5.6: Institut de France in Paris. We don't show the color blend result of [Nomura et al. 2007] since it is not straight forward from their output.

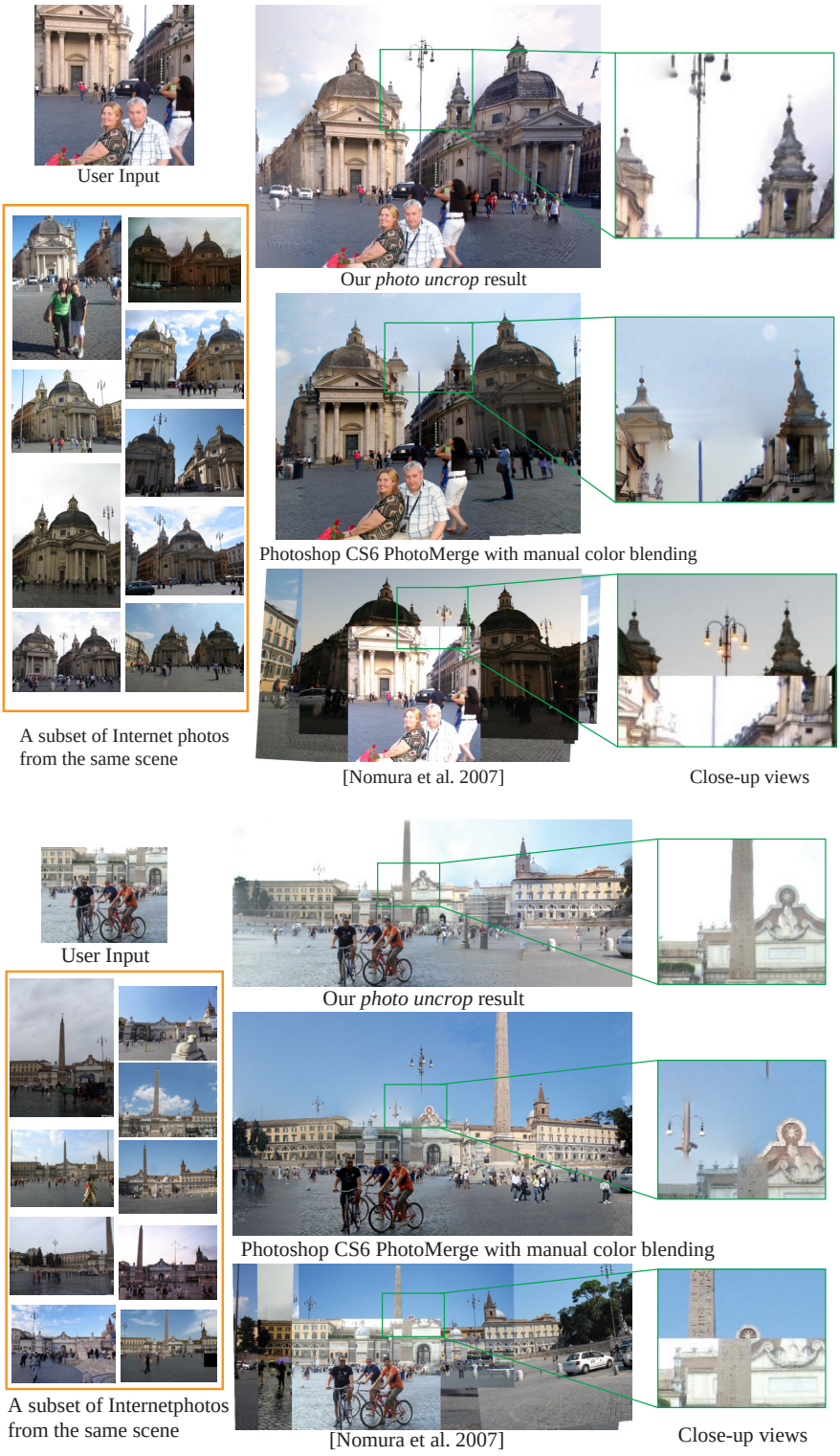


Figure 5.7: Two datasets from Piazza del Popolo. Notice the geometry misalignment in results from PhotoMerge and [55]

User input results

Figure 5.8: More results.

User input

Our *photo uncrop* results

Figure 5.9: A failure case with artifacts caused by large transient objects. Our future work includes exploring recognition and segmentation algorithms to help address this problem. Landmark: Place de la Concorde in Paris.

User input

Our *photo uncrop* results

Figure 5.10: A failure case. Large geometry errors can cause visual artifacts. The depth map estimation process performed poorly on the fountain sculptures in this example. When the source images are taken from viewpoints that are substantially different from the user photo, then warping and stitching these images can lead to significant artifacts in the composite. Landmark: Piazza Navona in Rome.

Chapter 6

ADDING PARALLAX TO YOUR PHOTOS BY MODELING PEOPLE IN 3D RECONSTRUCTION

In this chapter, I present our work on extending "photo uncrop" by adding parallax to an input photo. The key contributions include explicitly modeling people in photos and being able to simulate parallax and defocus blur effects from a single input photo. Our approach handles both transient and hard-to-reconstruct scene elements.

The world is dynamic. Yet, existing image-based reconstruction algorithms assume a static scene and cannot reconstruct objects that change or move between images. Take a typical tourist photograph in Figure 6.1 for example. Thanks to the sophistication of Internet 3D computer vision techniques, there exist many systems and algorithms that are able to recover precise geometry of the most of the scene by harnessing Internet photo collections [4, 22, 64, 63, 67]. These approaches are typically incapable of handling hard-to-model scene elements including texture-less regions and thin objects. Moreover, none of the reconstructions from these methods would contain the people, which are in fact the most important subjects in this photograph. This chapter presents work that reconstructs a depth map of this photograph including the people, more generally transient objects in this scene, or otherwise un-modeled objects. Our approach enables immersive 3D visualization effects such as adding parallax or controlling the depth of field.

The reconstruction of transient objects exposes new challenges for 3D modeling techniques. Existing Multi-View Stereo (MVS) methods rely on the *photometric consistency metric* for inferring scene depths. This metric assumes a static scene across multiple images and doesn't work for transient scene elements. For example, a person does not usually appear at the same location with exactly the same posture in multiple images, unless synchronizing the acquisition on purpose, and never in Internet photo collections. Besides reconstruction, visualization of transient objects

Input image | Photo Popup [Hoiem et al. 2005] | Our depth map rendering

Figure 6.1: Comparing with single image modeling.

also exposes challenges for existing image-based rendering techniques. Even if we assume we can recover rough geometry, rendering parallax typically results in severe visual artifacts. Therefore, reconstructing people is only part of the problem, and it is crucial to develop a visualization method that deals with geometry errors and complicated occlusions.

The approach presented in this chapter takes advantage of recent developments in MVS [64], human pose detection and estimation [82] and image compositing [65] techniques. It makes a first step towards photo-realistic visualization of 3D reconstructions with people. The contributions of this work are (i) a fully automatic system that creates compelling 3D visualizations from a single image (with the help of additional Internet images of the same scene); (ii) a novel 3D reconstruction approach to model people and other hard-to-model objects in the scene; (iii) a visualization method that reduces a number of artifacts.

In the remainder of this chapter, I give a brief introduction of related work in Section 6.1. I give an overview of our system in Section 6.2, and present our 3D modeling and visualization approaches in Section 6.3 and Section 6.4. I show results in Section 6.5 and conclude in Section 6.6.

6.1 Related Work

A rich literature exists on various aspects of this work: MVS [62, 37], dynamic texture reconstruction [38], human detection and pose estimation [82], and image based rendering [88, 14, 65]. But

none has been proposed to solve the whole problem of adding and rendering parallax in Internet images with people. To the best of our knowledge, this is the first work that addresses the problem with a fully automatic system. We here limit our description to closely related fields, in particular, people reconstruction in photographs and image-based visualization techniques.

6.1.1 Reconstructing People in Photos

There are few existing works that explicitly model people from multiple (unsynchronized) photographs. Zheng et al. [87] estimate the location of people by assuming that pedestrians uniformly distribute along a 2D path. This assumption does not necessarily hold for general scenes, especially popular sightseeing spots in open areas (See Fig. 6.1). Foreground/background segmentation is an alternative for modeling people in photographs. However, these methods typically require user interaction [85].

Single-view modeling methods have the potential to properly reconstruct transient objects. However, modeling people in a single-view fashion is rarely explored, and there has been limited progress on the problem. The problem of single-view modeling is highly ill-posed and extremely challenging, and typical approaches rely on user interaction [15]. Automatic methods, on the other hand, impose strong assumptions on the scene layout, appearance, and geometry, e.g. assuming planar or a special class of curved surfaces [56, 86, 43]. These methods are incapable of handling Internet photo collections with complicated geometry and foreground occluders. The Photo Popup [35] approach, for example, segments a scene into a small set of image regions and models them with planar proxies. This approach cannot handle the case where people and more complicated geometry are present. In Fig. 6.1, we show the result of applying Photo Pop-up to one of our images. Our approach is to leverage other views found n Internet photos. This approach is not as general as existing methods that do not require multiple photos of the same scene, however, it will work at popular sites and give significantly better results.

Recent approaches adopt large scale RGBD datasets for estimating object shapes from a single image [60, 69, 21]. These approaches are only evaluated on a small set of objects and are not easily scalable to Internet photo collections. The reason is multi-fold (i) their performance relies

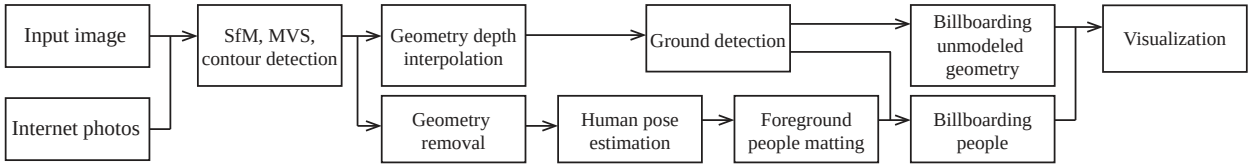


Figure 6.2: System overview.

on object detection and finding good matches from the dataset, but building such a detector or extended RGBD database for Internet photos is very difficult; (ii) these methods only estimate local object shapes instead of global layout.

6.1.2 Visualization

Image based rendering (IBR) techniques have been used for high-fidelity scene visualization [17, 13, 14, 58]. However, these systems require careful capturing of input images. The viewpoints need to be densely sampled on a contiguos path, and without foreground occluders to prevent visual artifacts. A full dynamic scene reconstruction is possible from multiple calibrated and synchronized videos [10]. Such data is not typically available in Internet photo collections. Photo Tourism is the first work on visualizing 3D scenes from Internet photo collections [67]. However, they model scene geometry for each image as a planar proxy, and are not able to add more dramatic 3D effects such as parallax or depth of field. Photo Tours [44] does model some parallax, but does not handle people. Photo uncrop [65], described in the previous chapter of this dissertation, is another IBR technique that works with Internet photos. It creates a panorama of a scene starting from a single user input, by minimizing the pixel misalignment when compositing together images warped from nearby viewpoints. The output is only a still image without 3D effects. Extending this technique for rendering camera motion video is not straightforward.

6.2 System Overview

Our system takes a single photograph (reference image), together with a dataset of photographs roughly at the same location (source images). These source images are downloaded from Flickr (`http://www.flickr.com`). We use existing structure-from-motion (SfM) software [79] to compute camera poses and the MVS system in [25] to generate sparse per-view depth maps. We obtain over-segmentations of the reference and source images and the per-pixel contour response by gPB [9]. In this section, we focus on explaining intuitions and high-level ideas of various components in the pipeline (Fig. 6.2). Detailed algorithms are explained in Sec. 6.3.

Our system consists of two major components, namely, 3D reconstruction and visualization.

Our 3D reconstruction approach recovers much of the consistent geometry across all the images, by counting MVS points that are reconstructed. This consistent geometry is usually static scene elements in the scene. Other image regions, whose appearance cannot be well explained by MVS reconstruction, are likely to be transient objects, and difficult-to-reconstruct static geometry and will be further modeled as billboards.

We use the off-the-shelf human pose detector from [82] to detect people in the images. Note that there is a sensitivity parameter in the detector which strongly affects the detection response. Automatically determining the value of this parameter is a non-trivial research problem. Using the default value leads to unreliable performance on general Internet photos (note the false positives in Fig. 6.3 and detection error on the head of the person in Fig. 6.4). To improve results, we apply the detector on images after removing elements well-reconstructed by MVS, and the performance is significantly improved. Note that we use a human pose detector, instead of just a human detector, since we need the semantic meaning of parts for people matting (reconstructing skeleton) and depth assignment (putting feet on the ground).

We estimate the depth of each human billboard by assuming feet touch the ground, thus naturally rise the question of how to reconstruct the ground. Ground reconstruction is usually not trivial for MVS reconstruction, due to the grazing angle between camera rays and the ground (people do not tend to point their cameras at the ground). Here we assume that the ground can be approxi-

mated as a plane, and that most cameras are hand-held and about 5 feet above the ground. Though simple, the assumptions work surprisingly well in practice. We adopt a RANSAC process to fit a plane that agrees the largest set of camera centers, rejecting the remaining cameras as outliers. We then move the plane five feet down. Note that we geo-locate the models using GPS data in photos [66]. Thus we know the metric (physical distance in feet) and the gravity direction.

General transient objects and other hard-to-reconstruct geometry are more difficult to recover. Ideally we can build a large set of object detectors that covers all categories of objects that appear in outdoor scenes. This process would be a major undertaking in itself. Instead, we simply use planar proxies for these transient objects, and assume that they need ground support or are attached to other scene geometry. Therefore, we iteratively propagate the depth information by assigning the depth of each unassigned pixel region (super pixel) with the smallest depth value that is around that region. Although imperfect, we find this approach works well for enabling camera dolly-in motion and defocus effects. Furthermore, we identify sky in images similarly to the process in Section 4.2.4.

The transient aware 3D modeling produces dense depth maps for the reference image (user input) and the source images (Internet photos from nearby viewpoints). These depth maps are inaccurate, and in general don't agree with each other. Our visualization process takes this inaccurate per-view geometry, and renders a video that minimizes the artifacts. Here we adopt a similar idea as in Section 5.2 that the energy definition is to minimizes pixel misplacement when compositing from multiple images.

6.3 3D Modeling

In this section, we describe the key components in detail.

[Geometry depth interpolation] We generate initial per-view depth maps using the method of [25]. These depth maps only contain sparse depth values from consistent geometry across multiple images. These depth maps are far from being useful for our visualization, as they are full of holes and omit any transient scene elements. Depth hallucination is then necessary. We describe depth interpolation for consistent geometry in this subsection. Billboarding other hard-to-model objects

is explained later.

To interpolate depth values of consistent geometry, we adopt a super-pixel based approach using over-segmentations from gPb [9]. Given one super-pixel, our approach counts the number of sparse depth values in it. If there are more than 3 sparse values, it approximates the depth within that particular super-pixel by fitting a smooth surface. In particular, we minimize the L2 norm of the curvature as solving

$$\arg\min_{d} \sum_{x,y\in\Omega} \left(d_{x,y} - \hat{d}_{x,y}\right)^2 + w \sum_{x,y} \left\| \frac{\partial^2 d_{x,y}}{\partial x^2}, \frac{\partial^2 d_{x,y}}{\partial y^2} \right\|_2^2,$$

where (x, y) denotes a location in a super-pixel, $d_{x,y}$ is the unknown dense depth, Ω is a set of locations with sparse depth values $\hat{d}_{x,y}$, and w is a weight parameter. Note that this approach assumes the geometry within each super-pixel is of a simple form, i.e., quadratic or planar. This assumption works well in practice. More complicated geometry tends to have non-flat shading and is likely to be segmented into multiple super-pixels, each of which has a relative simple shape.

[Geometry Removal] Next, to improve the human pose detection, we remove pixels corresponding to geometry recovered in the previous step. One super-pixel from the over-segmentation is considered to be a geometry segment if there are 3 or more sparse depth values. Fig. 6.3 illustrates the resulting image after geometry removal.

[People detection and matting] We use the human pose detector proposed in [82], which demonstrated the state-of-the-art performance on various benchmarks. The human pose detection, in general, is affected by a threshold parameter. Lowering the threshold leads to more detection response as well as more false positives (Fig. 6.3), for example, the false positive on the wall at the background and on the sculptures. However, working with community photo collections gives us the luxury of reasoning about the scene geometry. Hence we can compute which parts in the images represent consistent scene elements (at least those reconstructed well enough by MVS), and filter them out in the pose detection process. Note that a significant number of false positives are filtered out. We also evaluated the pose estimation on images with more crowded scenes (Fig. 6.4). Note the improvement in detecting the foreground person after filtering out much of the scene geometry.

Input image

Pose detection on the input image

Removing geometry

Pose detection on the geometry removed image

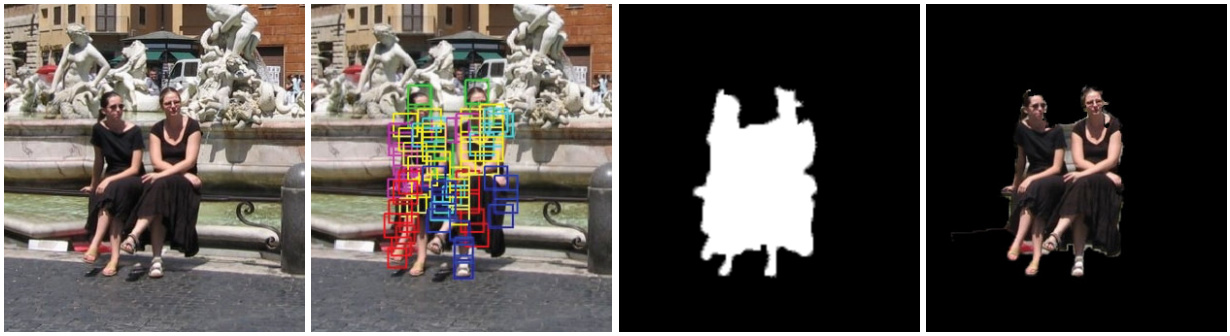
Figure 6.3: Removing consistent background geometry helps to filter out false positives.

(a) (b) (c)

Figure 6.4: Removing background leads to more accurate human pose estimation. (a) Human pose estimation on the original image. (b) Removing geometry reconstructed by MVS. (c) Human pose estimation on geometry removed image. Note that the head of the foreground person is correctly detected on the geometry removed image.

In the current implementation of our system, we compute a whole-body bounding box for each human pose response, and only keep the pose responses whose body bounding boxes are no smaller than half the size the largest one. Therefore, in most cases, we only keep the people in the foreground while treating the background people as other transient objects.

A person typically consists of multiple gPb segments and multiple part bounding boxes. We consider one super-pixel is on a human body if is contains the center pixel of one part bounding box. This process gives a binary mask of the foreground people. We then compute a color based matte from the binary mask, to reduce aliasing artifacts in the final blending, using the method proposed in [46]. In particular, we erode the foreground binary mask by 5 pixels and use it as foreground strokes for the matting algorithm and then erode the inverse of the foreground mask by 5 pixels to create background strokes. The output matte has a smooth transparency transition at the boundary and therefore creates fewer aliasing artifacts when compositing during visualization.



(a) (b) (c) (d)

Figure 6.5: Using human pose estimation results to segment foreground people. This is a closed-up view of the example in Figure 6.3. (a) The original image patch. (b) The human pose detection result. (c) The estimated alpha matte. (d) The color segmentation.

[Billboarding people] We use billboard proxies for people. And the problem here is to figure out the depth values on the billboard. From the previous step, we have a binary mask for the foreground people, and we know that the feet is detected in the pose. The depth of the billboard is simply the ground depth at the bottom pixel of the mask.

6.4 Visualization

6.4.1 Simulating camera dolly-in motion

To impart the 3D sense of the scene in a photo, the visualization process generates a video with camera dolly-in motion, where the camera translates gradually forward into the scene. Camera motion reveals occluded regions in the user input image. Therefore it is important to fill these regions with textures from source images. The problem of generating a single novel view and filling the missing image regions is similar to Photo Uncrop [65] described in Section 5.3. Here we adopt a simplified version of Photo Uncrop to achieve better computational efficiency. In particular, we optimize the objective of Equation 5.1, with the exception of the compatibility term, on a frame-by-frame basis. Poisson blending is applied in a frame-by-frame fashion to blend the images patches and compensate for the appearance difference in source images.

The per-frame novel view generating process causes flickering artifacts, as there is no temporal consistency on either the label or the color space. Per-frame Poisson blending is another source of flickering, since it is highly sensitive to the boundary conditions which vary from frame to frame. To address this, we adopt a deflickering process. The pixels in the Poisson blended frames are from two sources, the reference image and the source images to fill the texture of occluded geometry. The pixels from the source images are the cause of flickering. We want these pixels to be consistent over the frames. In order to do this, we back project these pixels (from all the frames in the video sequence) to the 3D world coordinate system. To improve the rendering efficiency, we merge points (averaging colors) that sit on top of each other (distance less than 0.1 of a pixel width in the 3D space). We use these merged points and the reference image (with depth) to render the final video with point splatting.

Note that we don't render the foreground people in the previous rendering process. The foreground people are rendered separately and then composited into the final frame in the end.

6.4.2 Simulating depth-of-field blur

We use Google's LensBlur code [30] to render depth-of-field blur. Basically, given a focal depth, and a depth map, one can compute the radius of lens blur at each pixel. The next problem is to perform a depth-guided convolution with a disk kernel. We use a layered approach by dividing the image into layers according to the pixel depth, blur each layer independently, then blend the layers at the end.

6.5 Results and Evaluations

We evaluate our system by rendering 10 video sequences using landmarks in Rome. We collected about 2400 Internet photos from Flickr that are taken near the input images. For more informative results, we refer the readers to the supplementary videos on YouTube [1].

We show a comparison between naive photo zoom-in and dollying in with single image depth

[1] `http://www.youtube.com/playlist?list=PLCOBz9xYbBdCWKsr_Zhdv_RwRLuv0NIRz`

Figure 6.6: Comparing against naive photo zoom-in and single image depth map rendering.

map rendering (Fig. 6.6). Naive photo zoom-in (cropping and scaling the image) gives the illusion of dollying in but the visual experience is not convincing as there is no parallax. Note however that single depth map rendering does have artifacts in regions where there is not enough information to fill in regions previously occluded in the reference image, leaving holes or flat Poisson filling.

Please see the supplemented videos on YouTube at `http://www.youtube.com/playlist?list=PLCOBz9xYbBdCWKsr_Zhdv_RwRLuv0NIRz`.

6.6 Conclusion and Future Work

This chapter describes a fully automatic system that can create a 3D visualization from a single user input, with the help of Internet photos. It adds parallax to a camera dolly-in motion and defocus blur effects to provide a controlled, 3D experience of the scene. This work introduces transient object aware 3D reconstruction, and explicitly models people in the images. Planar geometry

Figure 6.7: Synthesizing defocus blur.

proxies are used for people and other hard-to-model scene elements.

Our approach does have limitations. First, it only has ground texture at the regions captured by input images. Simulating zoom-out camera motion would often be difficult due to the lack of ground coverage, texture synthesis could be helpful in this case, a potential area of future work. Our method only explicitly models people, while common transient objects in the scene include cars, trees, chairs, etc. Specialized object detectors and single image car/tree modeling would be interesting future work. Our current implementation only models foreground people. It is not trivial to extend the work to handle people in the background who are much more likely to be partially occluded. Our results have noticeable artifacts, which arguably is tolerable given that it is more appealing to have 3D visualization with parallax and defocus effects. Currently our system only simulates camera dolly-in motion, automatic planning of more complicated motion path would be quite interesting and would likely to lead new rendering challenges as disocclusions become more prevalent.

Chapter 7

CONCLUSION

This thesis presented my work on advancing the state-of-the-art in "photo-realistic" 3D scene modeling and immersive visualization using online photo collections. This work made the following specific contributions to the computer vision community.[1]

- **The Visual Turing Test**, described in Chapter 2, is a novel quantitative evaluation scheme for scene reconstruction work.

- **Building city scale relightable models**. Our work advances the state-of-the-art by creating two of the first large scale re-lightable models. It proposes to combine multiple sources of images that complement each other. Lighting and scene albedo are estimated by optimizing a non-linear energy formulation, bootstrapped by using cloudy day photos. The experimental results show that although we are still far from achieving the goal, the proposed reconstruction process is a substantial step towards fully passing the Visual Turing Test.

- **Ground-to-aerial geo-registration with pixel level accuracy**, described in Chapter 3, is a first work showing ground-to-aerial geo-registration that performs consistently on a challenging large scale dataset. It solves an important yet difficult problem, that is aligning a ground-based MVS model to its aerial counterpart, resulting in a 3D model with complete scene coverage from aerial images and high-res details from ground level photos. This work introduces a novel viewpoint-dependent feature matching approach which renders ground-level images+geometry into aerial views to bridge the gap between wide viewing angle changes. We conduct large scale experiments using hundreds of thousand of images from the most

[1] Some of the code is open sourced at http://github.com/shanqi/research-code.git.

popular outdoor landmarks in Rome. 41 of the total 59 landmarks were successfully geo-located, a 69.5% success rate.

- **High quality geometry reconstruction using occluding contours**, described in Chapter 4, proposes a novel MVS system that effectively suppresses over-inflated boundary artifacts. It proposes a novel way of interpolating visibility and per-view depth maps in a contour-aware fashion. This visibility information is then used to compute a free-space volume which serves as a tight envelop to the mesh reconstruction process. The technical contribution includes a new technique to identify free-space arising from occluding contours and a new free-space-aware surface reconstruction approach. Experimental results show that the new formulation retains the regularization property of Poisson surface reconstruction [41], while significantly improving its accuracy by suppressing boundary artifacts and surface noise.

- **Photo Uncrop**, described in Chapter 5, extends the field of view of a single input image using Internet photos taken at nearby viewpoints. It introduces a novel approach to automatically select, warp, and stitch tourist photos using an MRF formulation that minimizes potential patch mismatch risk.

- **"Adding parallax to your photos"**, described in Chapter 6, extends "Photo Uncrop" by adding parallax and defocus blur effects to create a more compelling visual experience. It completes the scene geometry with ground estimation, and it explicitly models people, as well as other hard-to-model objects, using billboard proxies. We consider this work a step towards the grand goal of visualizing a scene with transient elements.

7.1 Future work

In this dissertation, I presented a collection of results showing substantial progress on large scale scene reconstruction from online photo collections. However, they are still early attempts of pushing 3D reconstruction research towards being able to render a scene "photo-realistically." As the technology keeps progressing, I envision a fully automatic, cloud-based system where all available

images reside. Compelling 3D models and visualization effects of any place in the world at any time will be easily accessible. These 3D models will both capture public spaces (e.g., city scenes and world landmarks) and private spaces (e.g., houses, offices), and will be updated progressively with new data flows in. In this section, I describe potential future projects that are stepping stones for achieving this grand goal. Some of the projects, I believe, I shall be exploring actively in the near future.

7.1.1 Enabling real applications

As 3D technology keeps progressing, one important question to ask is: "what is the purpose of 3D reconstruction research?" Although it is fulfilling to study the problem just for the sake of science, it is unfortunate that we haven't seen many real applications that are built upon these techniques. In my opinion, the following two areas are among the most promising. And I believe we will see a boom in the applications in these two areas with great impact in the near future.

Content creation in virtual reality and augmented reality

Virtual and augmented reality has been extremely popular and attracted a great amount of capital recently, following Facebook's acquisition of Occulus Rift. Currently people in the VR/AR field are focusing on building better hardware and improving head/eye tracking. The content creation technology is still uncharted. Almost nobody has tried it. The standard process now is mostly manual and unsurprisingly is extremely expensive. More importantly, the resulting models lack fine details, necessary to provide convincing real-world visual experiences.

One potential solution is automatically generating content from the scenes that are physically present in the real world. Unfortunately VR/AR has a very high requirement for model quality and a low tolerance to visual artifacts. So far we don't see a clear path toward automatically creating artifacts free models, thus the current generation of 3D reconstruction techniques are still far from being suitable in this domain.

Having a system that automatically creates perfect 3D models from images is a fundamental

problem. Note that in this case it is not necessary, and often impossible, to rely on Internet photo collections, as content creators can capture the images on their own. Developing such a fully automatic system will require a long term research effort and there is no sign of being close to the goal. Fortunately, there is little reason to visualize a scene by directly rendering from 3D models. In my opinion, a compelling solution that is achievable in short term is first creating a (imperfect) 3D model from the input images, then visualizing the scene by warping and compositing input images using the underlying geometry. We consider this technique in the category of image based rendering.

Apparently there are quite a few major challenges to be addressed. Geometry holes is one of the most apparent artifacts. Image-based rendering with inaccurate geometry often suffers from temporal inconsistency when switching between source image, as well as artifacts from geometry distortion. Moreover, it is difficult to obtain consistent texture when illumination and camera parameters change during the capturing process.

To summary, content creation for VR/AR is a rarely explored, extremely interesting and important research topic with many unsolved problems. It is undoubtedly the next hot research area in computer vision and graphics.

3D indoor modeling

3D indoor modeling is one of the most interesting applications for 3D reconstruction research. It enables museum virtual tours, video games, interior design, etc. Online real estate shopping might well be the most promising among these potential applications. Currently, consumers can typically only look through a handful of images of each house or apartment. The experience is far from satisfactory as the consumers get very little knowledge about the layout and the 3D space of the house. There are a few start-up companies exploring the space with Kinect-style depth sensing and/or interactive solutions. MatterPort [49] uses a specially designed depth camera array. The data capturing process involves carrying the hardware to the premises and place it at multiple locations in a house to take scans. Consecutive scans must be close by and have a lot of overlap, in order to match the scans. This process is quite time consuming and the scan matching often fails.

(a) (b)

Figure 7.1: Initial experiments on indoor reconstruction using occluding contours. Top two rows: a Tokyo hotel room reconstruction. Buttom two row: a living room reconstruction. We show mesh rendering in (a), and corresponding texture mapped mesh in (b).

Floored [20], on the other hand, uses a CAD approach where the 3D models are usually created by artists. DIADRIT [74] and StreetEasy [75] are dealing with different real estate markets and assume floor plans are available. Their systems rely on artists interactions, and only able to produce 3D walled up models from 2D floor plans. They also use room templates (e.g., kitchen, living room, bedroom) to synthesize furniture layout. The created 3D models are not faithful reconstructions of real houses or apartments. Note that none of these companies' approach is capable of scaling to hundreds of millions of houses in the nation's real estate market.

Even the best-of-the-breed 3D models created from state-of-the-art techniques are not close to clearing the quality bar for this application. Although results from initial experiments on indoor reconstruction with our MVS pipeline using occluding contours are promising (Figure 7.1), they still suffer from low quality and incomplete geometry. The difficulty is multi-fold. First of all, computing camera poses in an indoor setting is still an unsolved problem. The images typically lack fine details for image matching to work reliably. Therefore the reconstructed SfM model often consists of multiple disconnected components. Second, in a typical indoor setting, the illumination often has a high dynamic range. The natural light from windows can be 100,000 brighter than the areas in shadow, beyond the operating range of 14-bit camera sensors. Moreover, it is non-trivial to propose a data capturing process for houses so that non-tech-savoy users (real estate agents) can follow.

7.1.2 Event reconstruction

In this dissertation, I presented "Photo Uncrop" (Chapter 5) and adding parallax to a photo (Chapter 6). They are early steps towards a grander goal: being able to reconstruct 3D visualizations of a scene, with convincing rendering of people and other transient scene objects. The success of event reconstruction technology could enable applications including forensics, sports 3D visualization, life event recording (e.g. weddings, ceremonies), etc.

BIBLIOGRAPHY

[1] Austin Abrams, Christopher Hawley, and Robert Pless. Heliometric stereo: Shape from sun position. In *ECCV*, 2012.

[2] Jens Ackermann, Fabian Langguth, Simon Fuhrmann, and Michael Goesele. Photometric stereo for outdoor webcams. In *CVPR*, 2012.

[3] Adobe. PhotoShop CS6 PhotoMerge. `http://helpx.adobe.com/en/photoshop/using/create-panoramic-images-photomerge.html`.

[4] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(14):105–112, 2011.

[5] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *ICCV*, 2009.

[6] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *SIGGRAPH*, 2004.

[7] Apple. Apple maps. `https://www.apple.com/ios/maps/`.

[8] Apple. iPhone 5 Specifications. `http://support.apple.com/kb/sp655`.

[9] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011.

[10] Luca Ballan, Gabriel J Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. *TOG*, 29(4):87, 2010.

[11] Mayank Bansal, Kostas Daniilidis, and Harpreet Sawhney. Ultra-wide baseline facade matching for geo-localization. *ECCV 2012. Workshops and Demonstrations, Lecture Notes in Computer Science*, 7583:175–186, 2012.

[12] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. *TPAMI*, 14(2):239 – 256, 1992.

[13] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001.

[14] Gaurav Chaurasia, Olga Sorkine-Hornung, and George Drettakis. Silhouette-aware warping for image-based rendering. *Computer Graphics Forum*, 30(4), 2011.

[15] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: extracting editable objects from a single photo. *TOG*, 32(6):195, 2013.

[16] Peter Cho, Noah Snavely, and Ross Anderson. 3D exploitation of large urban photo archives. In *SPIE Defense, Security, and Sensing*, pages 769714–769714. International Society for Optics and Photonics, 2010.

[17] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[18] Andrew Delong, Anton Osokin, Hossam N. Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *IJCV*, 96(1):1–27, 2012.

[19] Flickr. Flickr. `http://www.flickr.com`.

[20] Floored. Floored. http://www.floored.com/.

[21] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013.

[22] Jan-Michael Frahm, Pierre Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building rome on a cloudless day. In *ECCV*, 2010.

[23] Jan-Michael Frahm, Jared Heinly, Enliang Zheng, Enrique Dunn, Pierre Fite-Georgel, and Marc Pollefeys. Geo-registered 3D models from crowdsourced image collections. *Geospatial Information Science*, 16(1):55–60, 2013.

[24] Christian Frueh and Avideh Zakhor. Constructing 3D city models by merging ground-based and airborne views. In *CVPR*, 2003.

[25] Simon Fuhrmann and Michael Goesele. Fusion of depth maps with multiple scales. *SIGGRAPH Asia*, 30(6):148, 2011.

[26] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

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

[28] Google. Google image search. `http://images.google.com/`.

[29] Google. Google maps. `http://maps.google.com/`.

[30] Google. iGoogle LensBlur. `http://googleresearch.blogspot.com/2014/04/lens-blur-in-new-google-camera-app.html`.

[31] T. Haber, C. Fuchs, P. Bekaer, H.P. Seidel, M. Goesele, and H.P.A. Lensch. Relighting objects from image collections. In *CVPR*, 2009.

[32] James Hays and Alexei A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008.

[33] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *ECCV*, 2010.

[34] Carlos Hernández and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. In *Computer Vision and Image Understanding (CVIU)*, 2004.

[35] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *SIGGRAPH*, 2005.

[36] Arnold Irschara, Christopher Zach, J-M Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.

[37] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, 2011.

[38] Dinghuang Ji, Enrique Dunn, and Jan-Michael Frahm. 3d reconstruction of dynamic textures in crowd sourced data. In *ECCV*, 2014.

[39] Ryan S Kaminsky, Noah Snavely, Steven M Seitz, and Richard Szeliski. Alignment of 3D point clouds to overhead images. In *Second IEEE Workshop on Internet Vision*, 2009.

[40] M. Kazhdan, M. Bolitho, , and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, 2006.

[41] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), July 2013.

[42] Jan Knopp, Josef Sivic, and Tomas Pajdla. Avoiding confusing features in place recognition. In *ECCV*, 2010.

[43] Avanish Kushal and Steven M. Seitz. Single view reconstruciton of piecewise swept surfaces. In *3DV*, 2013.

[44] Avanish Kushal, Ben Self, Yasutaka Furukawa, David Gallup, Carloz Hernandez, Brian Curless, and Steven Seitz. Photo tours. In *3DImPVT*, 2012.

[45] P. Laffont, A. Bousseau, and G. Drettakis. Coherent intrinsic images from photo collections. *SIGGRAPH Asia*, 2012.

[46] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *TPAMI*, 30(2):228–242, 2008.

[47] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3D point clouds. In *ECCV*, 2012.

[48] David Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, volume 20, pages 91–110, 2003.

[49] MatterPort. Matterport. http://matterport.com/.

[50] Kevin Matzen and Noah Snavely. Scene chronology. In *ECCV*, 2014.

[51] Microsoft. Bing maps. `http://www.bing.com/maps/`.

[52] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2), 2009.

[53] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, pages 331–340, 2009.

[54] BBC News. James cameron: All entertainment inevitably 3d'. http://www.bbc.com/news/entertainment-arts-23790877.

[55] Y. Nomura, L. Zhang, and S.K. Nayar. Scene collages and flexible camera arrays. In *Eurographics Symposium on Rendering*, 2007.

[56] Mukta Prasad, Andrew Zisserman, and Andrew Fitzgibbon. Single view reconstruction of curved surfaces. In *CVPR*, 2006.

[57] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV*, 2009.

[58] Christian Richardt, Yael Pritch, Henning Zimmer, and Alexander Sorkine-Hornung. Megastereo: Constructing high resolution stereo panoramas. In *CVPR*, 2013.

[59] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3D registration. In *ICRA*, pages 3212–3217, 2009.

[60] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Depth perception from a single still image. In *AAAI*, 2008.

[61] Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. Multi-view stereo evaluation. `http://vision.middlebury.edu/mview/eval/`.

[62] Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006.

[63] Qi Shan, Riley Adams, Brian Curless, Yasutaka Furukawa, and Steven M. Seitz. The visual Turing test for scene reconstruction. In *International Conference on 3D Vision (3DV)*, 2013.

[64] Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M. Seitz. Occluding contours for multi-view stereo. In *CVPR*, 2014.

[65] Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M. Seitz. Photouncrop. In *ECCV*, 2014.

[66] Qi Shan, Changchang Wu, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M. Seitz. Accurate geo-registration by ground-to-aerial image matching,. In *International Conference on 3D Vision (3DV)*, 2014.

[67] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring image collections in 3D. In *SIGGRAPH*, 2006.

[68] John M. Snyder. Area light sources for real-time graphics. Technical Report MSR-TR-96-11, Microsoft Research, 1996.

[69] Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *ECCV*, 2010.

[70] Hai Tao, H.S. Sawhney, and Rakesh Kumar. A global matching framework for stereo computation. In *ICCV*, volume 1, pages 532–539, 2001.

[71] Kathleen Tuite, Noah Snavely, Dun-yu Hsiao, Nadine Tabing, and Zoran Popovic. Photocity: training experts at large-scale image acquisition through a competitive game. In *CHI*, 2011.

[72] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.

[73] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *TPAMI*, 13(4):376–380, 1991.

[74] DIAKRIT. DIAKRIT. http://www.diakrit.com/en.

[75] Sreet.

[76] George Vogiatzis, Carlos Hernández, Philip H. S. Torr, and Roberto Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *TPAMI*, 29(12):2241–2246, 2007.

[77] H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards highresolution large-scale multi-view stereo. In *CVPR*, 2009.

[78] Chun-Po Wang, Kyle Wilson, and Noah Snavely. Accurate georegistration of point clouds using geographic data. In *3DV*, pages 33–40, 2013.

[79] Changchang Wu. VisualSFM. `http://homes.cs.washington.edu/~ccwu/vsfm`.

[80] Changchang Wu, Brian Clipp, Xiaowei Li, J-M Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *CVPR*, 2008.

[81] Changchang Wu, Friedrich Fraundorfer, J-M Frahm, and Marc Pollefeys. 3D model search and pose estimation from single images using vip features. In *CVPR Workshops*, pages 1–8, 2008.

[82] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures-of-parts. In *ICCV*, 2011.

[83] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *CVPR*, 2009.

[84] AR Zamir and M Shah. Image geo-localization based on multiple nearest neighbor feature matching using generalized graphs. *TPAMI*, 36(8):1546–1558, 2014.

[85] Chenxi Zhang, Jizhou Gao, Oliver Wang, Pierre Georgel, Ruigang Yang, James Davis, Jan-Michael Frahm, and Marc Pollefeys. Personal photo enhancement using internet photo collections. *TVCG*, 2014.

[86] Li Zhang, Guillaume Dugas-Phocion, Jean-Sebastien Samson, and Steven M. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001.

[87] Enliang Zheng, Ke Wang, Enrique Dunn, and Jan-Michael Frahm. Joint object class sequencing and trajectory triangulation (JOST). In *ECCV*, 2014.

[88] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image oversegmentation. *IJCV*, 75:49–65, 2007.