The Visual Turing Test for Scene Reconstruction

Qi Shan[†], Riley Adams[†], Brian Curless[†], Yasutaka Furukawa[*], and Steven M. Seitz[*†]
[†]University of Washington [*]Google

Abstract

We present the first large scale system for capturing and rendering relightable *scene reconstructions from massive unstructured photo collections taken under different illumination conditions and viewpoints. We combine photos taken from many sources, Flickr-based ground-level imagery, oblique aerial views, and streetview, to recover models that are significantly more complete and detailed than previously demonstrated. We demonstrate the ability to match both the viewpoint and illumination of arbitrary input photos, enabling a* Visual Turing Test *in which photo and rendering are viewed side-by-side and the observer has to guess which is which. While we cannot yet fool human perception, the gap is closing.*

1. Introduction

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, I 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 as the lighting in that particular photo. We present the first large scale results on reflectance estimation and lighting matching for Internet Photo Collections. To our best 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.

2. Related Work

In the past few years, researchers succeeded in developing very large-scale 3D reconstruction pipelines for Internet photo collections [4, 5] 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 depthmaps from time-lapse webcam videos of outdoor scenes [2, 3], 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

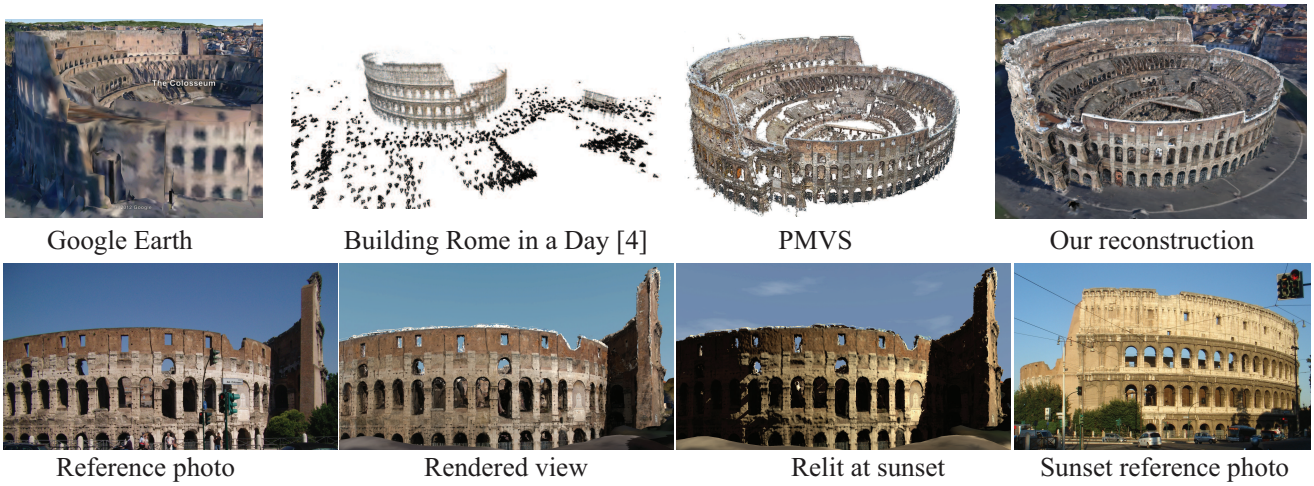


Figure 1. Comparison with state of the art (top row) and results for rendering and relighting of one viewpoint (bottom row)

in each photo. Indeed, the only previous work to attempt relighting for Internet photo collections is [8] 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 [10]. 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.

3. 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 [1] 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 capture from the in-browser Streetview rendering. We also invert the sRGB function

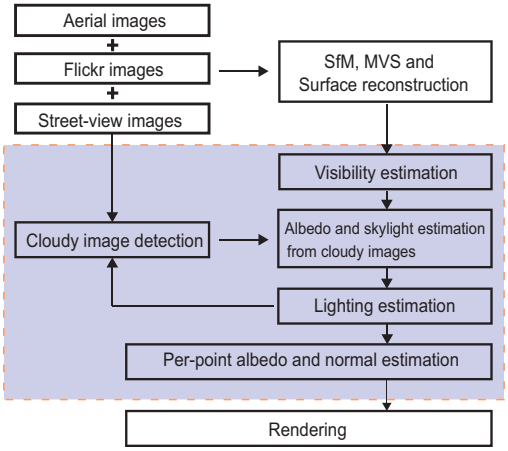
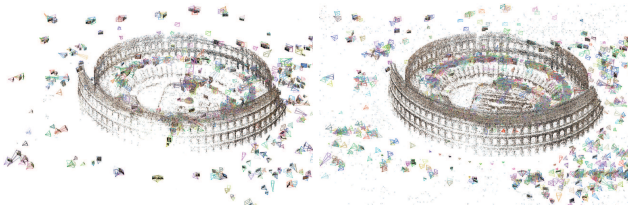


Figure 2. Workflow overview. We highlight the key technical contributions of the proposed system.



(a) (b)

Figure 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.

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 [13] to estimate camera poses, PMVS/CMVS [6, 7] to recover a dense, oriented point cloud, and Poisson Surface Reconstruction [9] 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.

3.1. 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 re-run 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 3 illustrates the results on the Colosseum after the two SfM steps.

3.2. 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 visibility 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 [11] (set for exact neighbor-finding), and we use pre-computed z-buffers instead of ray casting for occlusion testing.

4. 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 present 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.

4.1. Shading model and objective function

For the outdoor scenes we are reconstructing, we adopt a simple but 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:

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

$$\Theta = \{N_i, a_i, L_j, k_j^{sky}, k_j^{sun}, \delta_{i,j}\}. \quad (2)$$

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. Please refer to the supplementary file for additional constraints on these variables. 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 [8]. 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 [12]) 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 well in practice.

We assume, for the moment, that our images, albedos, and lighting are grayscale; we discuss color in Section 4.4.

Based on our shading equation (1), the lighting and reflectance estimation problem can be formulated as follows:

$$\underset{\Theta}{\operatorname{argmin}} \sum_i \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \| R_{i,j}(\Theta) - \tilde{R}_{i,j} \|_2^2. \tag{3}$$

$\tilde{R}_{i,j}$ is the observed pixel intensity 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.

4.2. Identifying Cloudy Images

Solving (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 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. [3]. 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. 4.3, 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.

4.3. 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 for the entire algorithm flow).

Skylight and partial albedo from cloudy images

For cloudy images, the shading equation (1) does not have a sunlight component and is simplified to

$$R_{i,j}^C(N_i, a_i, k_j^{sky}) = a_i f(N_i) k_j^{sky}. \tag{4}$$

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 (3) can similarly be reduced as follows:

$$\underset{\{a_i, k_j^{sky}\}}{\operatorname{argmin}} = \sum_{i \in \mathbf{P^C}} \sum_{j \in V_i \cap \mathbf{I^C}} \sqrt{\tilde{R}_{i,j}} \| R_{i,j}^C(a_i, k_j^{sky}) - \tilde{R}_{i,j} \|_2^2. \tag{5}$$

Note that the surface normal N_i is technically a variable in (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 (3) by using all the points, which could number in the tens of millions. 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 (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 (5) and the normal $\tilde{N}_i$ in the input mesh, giving a new objective:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{i \in \mathbf{P^L}} \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \|R_{i,j}(\Theta) - \tilde{R}_{i,j}\|_2^2 + \lambda_1 \sum_{i \in \mathbf{P^C}} \|a_i f(N_i) - \tilde{a}_i f(\tilde{N}_i)\|_2^2. \quad (6)$$

$\lambda_1 = 1$ 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 (5).

After solving (6), we update the cloudy image set $\mathbf{P^C}$ by using the estimated lighting parameters as in Section 4.2. Then, we solve (5) and (6) in exactly the same way.

Per-point albedo and normal 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:

$$\underset{\{N_i, a_i, \delta_{i,j}\}}{\operatorname{argmin}} \sum_{j \in V_i} \sqrt{\tilde{R}_{i,j}} \|R_{i,j}(\Theta) - \tilde{R}_{i,j}\|_2^2 + \lambda_1 \sum_{i \in \mathbf{P^C}} \|a_i f(N_i) - \tilde{a}_i f(\tilde{N}_i)\|_2^2 + w_i \|N_i - \tilde{N}_i\|_2^2. \quad (7)$$

The last 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. The definition of the per-point weight w_i is given in the supplementary material. Note that our system optimizes surface normals. It contributes a modest improvement in capturing the lighting variation in the input images.

4.4. Implementation details

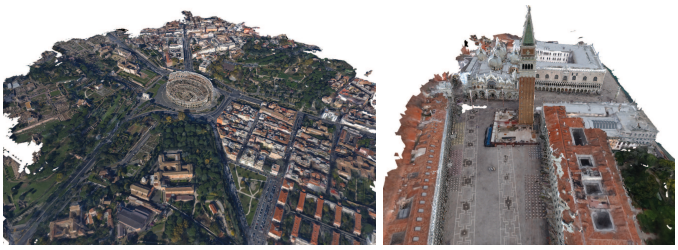
Here we describe several implementation details. First, we employ the Matlab function *lsqnonlin* to solve all of the the optimization problems.

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 [9] 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 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.



Colosseum San Marco Square

Figure 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.

5. Experimental Results

This section presents the first large scale 3D reconstructions with lighting and reflectance models from community

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 1. Statistics of our datasets.

photo collections mixed with aerial and streetview imagery. Figure 4 shows the two datasets used in our experiments, where some statistics are given in Table 1. The computational time is collected by running the system on a cluster of 80 cores. The visibility and the per-point albedo estimation processes distribute workload to all 80 cores while the lighting estimation process runs on a single core. Figure 4 shows 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.

5.1. 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 one hundred randomly selected Flickr photos as reference views. Each view is presented in four resolution levels; hence, there are in total four hundred 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 subjects to get right) pollute the low-res tests. Some of the image pairs are shown in Figure 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 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 probability of picking a rendered image is higher 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.

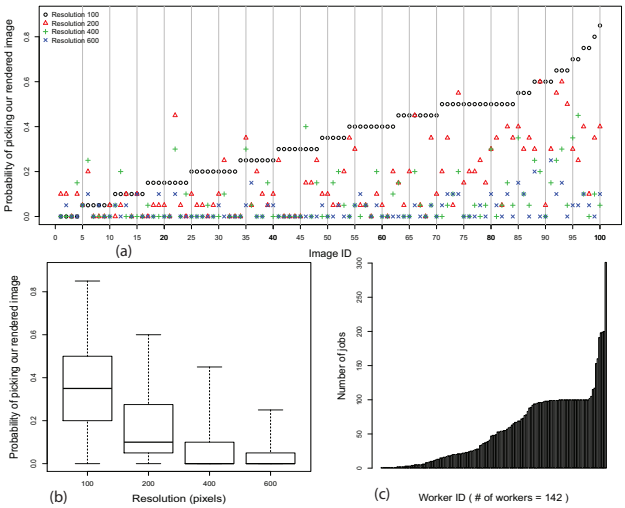


Figure 6. Statistics of the Visual Turing Test. Please zoom in for a better visualization of the plot. (a) Per-image average. (b) Per resolution level statistics. (c) Worker plots.

Figure 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 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 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 7, the viewpoint is located right behind some geometry fragments which occlude the object of interest and cause severe artifacts.

5.2. More Evaluations

To further validate the accuracy of our lighting estimation, we render images with and without shadow effects

Figure 5. Visual Turing test. In each image pair, the ground truth image is on the left and our result is on the right.

Figure 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 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.

An alternative solution for reproducing lighting effects without estimating lighting and albedo is to compute average/median colors over the mesh from visible images and applying a histogram matching to ground truth images. However, as illustrated in Figure 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.

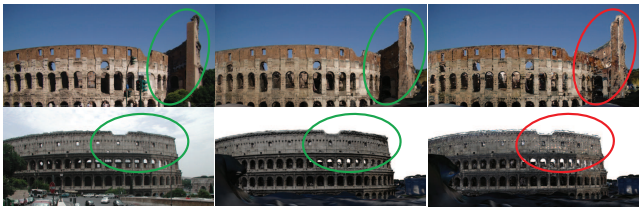
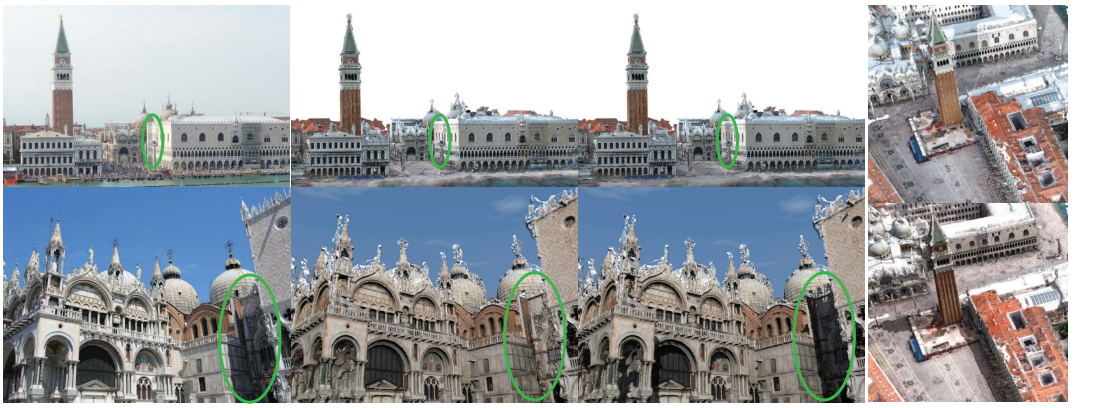


Figure 9. Rendering with median color and histogram matching. Left: input ground truth images; middle: our results; right: images rendered from average pixel color and applying histogram matching to the ground truth image. Note the lack of directional lighting effects and color noise in red ellipses.

Figure 10 illustrates the importance of the visibility test in our system, which removes the influences of foreground



Reference image | Our rendering without shadow | Our rendering with shadow | Aerial rendering with shadow

Figure 8. Validating the accuracy of our lighting estimation.

(a) (b) (c)

Figure 10. Our rendered image removes fore-ground objects. (a) A reference photo. (b) Rendered result. (c) A close-up view of (b).

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. Figure 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.

Please see the supplementary material for more results and details on the Visual Turing Test.

6. Conclusions and Limitations

We present 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 more complicated illumination models, e.g., night-time shots.

Acknowledgment

This work was supported by funding from National Science Foundation grant IIS-0963657, Google, Intel, Microsoft, and the UW Animation Research Labs.

References

[1] Flickr. http://www.flickr.com. 2

[2] A. Abrams, C. Hawley, and R. Pless. Heliometric stereo: Shape from sun position. In *ECCV*, 2012. 1

[3] J. Ackermann, F. Langguth, S. Fuhrmann, and M. Goesele. Photometric stereo for outdoor webcams. In *CVPR*, 2012. 1, 4

[4] S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(14):105–112, October 2011. 1, 3

[5] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *ECCV*, 2010. 1

[6] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010. 2

[7] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 32(8):1362–1376, 2010. 2

[8] T. Haber, C. Fuchs, P. Bekaer, H. Seidel, M. Goesele, and H. Lensch. Relighting objects from image collections. In *CVPR*, 2009. 2, 4

[9] M. Kazhdan, M. Bolitho, , and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, 2006. 2, 5

[10] P. Laffont, A. Bousseau, and G. Drettakis. Coherent intrinsic images from photo collections. In *SIGGRAPH Asia*, 2012. 2

[11] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, pages 331–340, 2009. 3

[12] J. M. Snyder. Area light sources for real-time graphics. Technical Report MSR-TR-96-11, Microsoft Research, 1996. 4

[13] C. Wu. VisualSFM : A visual structure from motion system. http://homes.cs.washington.edu/~ccwu/vsfm. 2